

## MIT Open Access Articles

*Modular Robot Systems From Self-Assembly to Self-Disassembly*

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

**Citation:** Gilpin, Kyle, and Daniela Rus. "Modular Robot Systems." IEEE Robotics & Automation Magazine 17.3 (2010): 38-55. Web. 3 Feb. 2012. Gilpin, Kyle, and Daniela Rus. "Modular Robot Systems." IEEE Robotics & Automation Magazine 17.3 (2010): 38-55. Web. 3 Feb. 2012. © 2011 Institute of Electrical and Electronics Engineers

**As Published:** <http://dx.doi.org/10.1109/mra.2010.937859>

**Publisher:** Institute of Electrical and Electronics Engineers (IEEE)

**Persistent URL:** <http://hdl.handle.net/1721.1/69029>

**Version:** Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

**Terms of Use:** Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.




# Modular Robot Systems

*From Self-Assembly to Self-Disassembly*

BY KYLE GILPIN AND DANIELA RUS

**Celebrating  
50 Years  
of Robotics**



Our long-term goal of creating programmable matter will be achieved when we have the ability to build objects whose physical properties, such as shape, stiffness, optical characteristics, acoustics, or viscosity, can be programmed on demand. In this article, we survey the history of modular robotics and their connections to programmable matter systems capable of realizing arbitrarily shapes on demand. The goal shape may be a robot built for a specific task (a snake to pass through a tunnel or a rolling belt to quickly cover open ground) or an object designed for a particular job (such as a wrench, hammer, or bridge). When the task is complete, the modules in the structure can disconnect and be reused to create a different object. This type of self-reconfiguration leads to versatile robots that can support multiple modalities of locomotion, manipulation, and perception. Such variable architecture robots have been studied in the context of self-assembling systems, self-reconfiguring systems, self-repairing systems, and self-organizing systems.

We examine in detail several self-assembling robot systems and propose self-disassembly as an alternative for creating task-specific robots. In self-assembling robot systems, the individual robotic modules aggregate in a highly constrained way to form a specific shape. In self-disassembling robot systems, a large block of robot modules peels off extra modules to form the desired shape, temporarily abandoning the modules not necessary for the task at hand. We discuss the trade-offs between shape formation by self-assembly and by self-disassembly and report on recent algorithmic

© LUSHPIX ILLUSTRATION

Digital Object Identifier 10.1109/MRA.2010.937859

and hardware results for creating a hybrid, self-assembling/disassembling robot system.

Creating machines capable of changing shape has been our long-running dream. This dream has been inspired and fueled by the science fiction community and the movie industry who have created characters such as the “Barbapapa” family [1], the Changelings in “Star Trek” [2], the “Terminator” [3], and “Transformers” [4]. The Barbapapa creatures are colorful pear-shaped blobs that can take on any shape: They become tools for gardening, containers for shopping, toys for playtime, or boats for sailing.

Programmable matter aims to bring machines and materials closer together by creating machines that become more like materials and materials that behave more like machines. This is a considerable challenge with exciting potential for future payoffs: desktop rapid prototyping of electrically and mechanically active devices, paper computers, on-demand objects and tools, machines that can actively change their optical properties to become invisible or reflective, and machines with programmable acoustic properties for effective localization or acoustic camouflage.

To achieve programmable matter capabilities, the following are the important questions that must be addressed.

- 1) How do we create hardware capable of programmability with respect to one or more physical properties?
- 2) What is the algorithmic base for achieving the desired machine property?
- 3) How do we go from theory to practice to build and deploy systems that achieve the goals of programmable matter?

The robotics community has been addressing some of these questions in the context of modular, self-assembling, and self-reconfiguring machines. A sampling of these approaches is illustrated in Figure 1 that shows four separate approaches to creating a modular humanoid robot using different hardware and algorithmic approaches. In Figure 1(d), the Smart Pebble system can be formed from a loose collection of modules that self-assemble into a close-packed lattice and then self-disassembly to remove the extra modules not needed by the humanoid structure. Even though the scale of each system is not apparent from the photos, each Superbot module fits within a  $168 \times 84 \times 84 \text{ mm}^3$

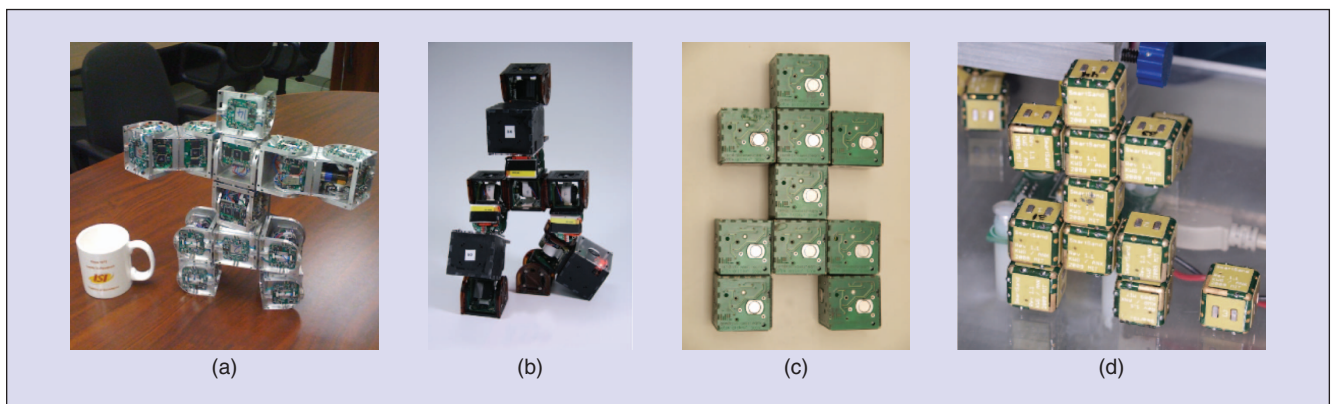
rectangular box, the CKBot modules fit inside a 60-mm cube, the Miche modules are 45-mm cubes, and the Smart Pebbles are 12-mm cubes.

In this article, we survey the history of modular and self-assembling robotics and present recent results in creating shapes using a two-stage process: 1) self-assembly to create a conglomerate block of modules without internal gaps and 2) self-disassembly to sculpt a desired shape from this block. We describe and analyze an efficient self-assembly algorithm and show data from experiments with a robot system consisting of 1-cm cubic modules.

The field of modular robotics started with a paper presented by Toshio Fukuda et al. at the IEEE International Conference on Robotics and Automation (ICRA) in the Spring of 1988 [5], in which he describes the abstract concept of a dynamically reconfigurable robotic system that can assume different shapes. In that article, Fukuda and Nakagawa envisioned a robot system composed different types of modules that can combine to accomplish a variety of tasks. Fukuda et al. refined this concept in a second paper [6], presented at the IEEE International Workshop on Intelligent Robots later that Fall, in which they gave their robot a name, cellular robot (CEBOT). The CEBOT was composed of wheeled modules with infrared photodiodes and ultrasonic transducers. The modules also contained an active latching mechanism that could be used to join two modules together, and the first algorithm Fukuda et al. proposed was aimed at autonomously mating two CEBOT modules.

Over the past 20 years, modular robotics research developed many facets: hardware design, planning and control algorithms, the trade-off between hardware and algorithmic complexity, efficient simulation, and system integration.

Our aim in this article is two-fold. First, we take a long and detailed look to the past to frame the development of programmable matter in the history of modular robotics. Then, in the second half of this article, we present a new system (hardware, algorithms, and experiments) that is capable of realizing programmable matter through a process called self-disassembly. Self-disassembly aims to achieve shape in a way analogous to



**Figure 1.** (a) Six Superbot modules assembled by hand to form a humanoid. (b) CKBot modules forming a similar structure that is able to self-repair itself after being damaged. (c) The Miche system lacks internal degrees of freedom, but it was produced through the self-disassembly of a  $3 \times 5$  block of modules. (d) The humanoid formed by the Smart Pebbles system. (SuperBot picture courtesy of Polymorphic Robotics Laboratory, University of Southern California, Dr. Wei-Min Shen. CKBot picture courtesy of Prof. Mark Yim, University of Pennsylvania.)

sculpting. The modular system starts as a contiguous block. Modules subtract or peel themselves off to reveal the desired shape.

## A Modular Robot Perspective

Modular robotic systems can be described and classified using several axes and properties. In what follows, we choose the traditional route of classifying these systems by the geometry of the system. In particular, we organize the prior work in four main categories: chains, lattices, trusses, and variable shape systems. Chain-type systems are composed of modules that are arranged to form single- or multibranch linkages. Both snakes and multilegged walkers are possible configurations of a chain-type system. Unlike chain systems that generally allow modules to be positioned arbitrarily in space, lattice systems generally require that modules occupy discrete locations. To allow for reconfiguration, this constraint does not preclude a module from transitioning from one location to another. Additionally, the lattice system can be made to look like a chain-based system if the only modules present form a chain. We define a third class of truss-based systems that have some similarities to both chain and lattice systems. The primary identifying characteristic of truss systems is that they create scaffold-type objects and use the linkages or struts between rigid nodes to reconfigure the system's shape. Finally, to account for a few unique systems that elude the above categorization, we define a class of free-form systems. This article is not the first to attempt to summarize past and ongoing work in the modular robotics field, and for details see [7].

The general approach for creating all existing modular robot systems has been to design the unit module of the system and develop algorithms that enable a group of unit modules to coordinate their degrees of freedom to control their motion for the goal of creating a shape, generating a locomotion gait, or interacting with the environment.

### Chain Systems

There have been a number of chain-like modular robot systems that combine many modules, each with a low degree of freedom, to form complex structures with significantly more flexibility. One of the first was the Polypod system developed by Yim [8], [9]. This system was composed of two types of modules: segments and nodes. The nodes were passive cubic modules with one connector on each face. The segments were two degree of freedom linkages able to expand or contract in length as well as angle to the left or right. Each segment contained an 8-b microprocessor in addition to angle and force sensors. The nodes contained batteries to power the system. Segments could be connected to other segments or nodes. The connectors were four-way symmetric, allowing the system to operate in three dimensions. These two features allowed the system to assume a wide variety of forms including rolling loops and hexapods. The Polypod was remarkably advanced for its time, and it went on to inspire many future chain-type systems.

Castano et al. developed a chain-type modular robotic system called CONRO that is detailed in [10]–[12]. Each module in the CONRO system was composed of two orthogonal servomotors that control the module's pitch and yaw. The modules had gendered connectors that required neighboring

modules to lie in the same plane. Additionally, there were only three female connectors and one male connector integrated into each module. The CONRO system was able to assume forms that resemble snakes and multijointed walkers. In [11], Shen and Will used the CONRO modules to attack the problem of autonomously docking chain-type modular robots in two-dimensional (2-D)—an unsolved problem at the time.

Murata et al. developed the M-TRAN modular robotic system [13]–[16], Murata-IROS06, which has undergone multiple revisions and improvements. The modules contain processing and battery power, along with two parallel rotational degrees of freedom. Multiple modules can be mated together in a variety of ways: end to end with 0 or 90° rotations between modules, side by side so that their actuators operate in parallel, or side to end. The system has been used to perform a wide variety of experiments. Kamimura et al. [14] employ a set of interconnected, out-of-phase oscillators (central pattern generators) to achieve walking gaits in the M-TRAN system. By optimizing the phase relationships between the oscillators, they can be used to drive the modules' motors in a coordinated fashion that leads to forward locomotion. The M-TRAN robot relies on evolutionary algorithms to perform this optimization process. The optimization process can occur in simulation before being implemented in hardware or it can be performed in real time to allow the robot to adapt to a changing environment. Murata et al. [15] developed a simulator based on the M-TRAN system to experiment with the system's ability to self-reconfigure. Murata et al. [16] expanded on this by adding cameras to the system so that a set of M-TRAN modules could separate, perform independent tasks, and then rejoin into a larger structure.

Marbach and Ijspeert improved upon the ability of systems such as M-TRAN to generate gaits in real time by using their modular system, YaMoR [17]. YaMoR is composed of single degree of freedom chain-type modules. By using Powell's method for function optimization instead of genetic algorithms, they claim YaMoR will converge to an optimal gait much more quickly. The risk is that the resulting gait may not be globally optimal, but, in practice, this only happens infrequently.

The Superbot system [18] improves on the mechanical design of M-TRAN by adding an additional degree of rotational freedom between the two existing rotation axes. The Superbot system was designed to be a more robust modular robot capable of operating in real-world situations—specifically, planetary exploration. The Uni-Rover [19] developed by Hirose et al. was another modular robot developed for planetary exploration. The Uni-Rover consists of a number of identical wheels connected to a larger body through a four-degree of freedom linkage. Each of these linkages can detach from the body. As a separate unit, the wheel/linkage combination could use the wheel as a base and the linkage as a manipulator. Alternatively, a single wheel/linkage unit could roll independently or join with other modules to roll over rougher terrain (Figure 2).

The PolyBot, developed by Yim et al. [20], [21], is a chain-type reconfigurable modular robot: chains of modules, each

with a single degree of freedom, connect to form structures such as loops, legs, or tendrils. The active modules consist of two connectors arranged on opposite sides of a cube. The module itself has a single rotational degree of freedom around an axis perpendicular to two of the faces without connectors. Passive modules that include six mating faces but lack the ability to rotate may also be integrated into the system to form a wider variety of structures. The PolyBot system has the ability to achieve locomotion through any number of configurations. For example, a number of segment and node modules can form a loop that rolls across smooth terrain. If confronted with more challenging terrain, the robot can reconfigure itself as a multilegged walking robot.

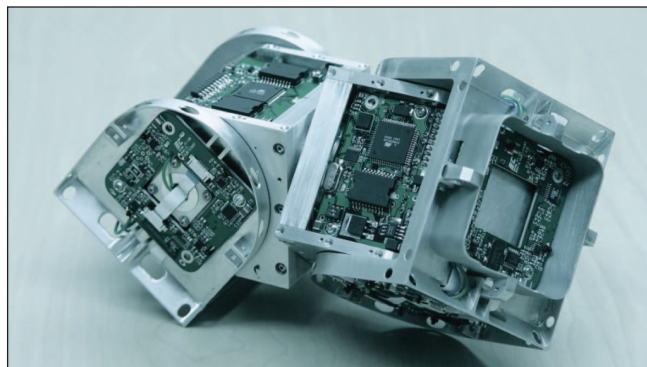
PolyBot evolved into CKBot that has been used more recently in many interesting ways. Yim et al. [22] show that a robot built from CKBot modules is able to reassemble itself after being accidentally or intentionally destroyed. The modules cannot repair their internal workings, but groups of modules are able to locate other modules, crawl toward them, align, and connect to restore and initial shape and functionality of the system before it was broken. Additionally, Yim et al. studied the ability of a closed chain of CKBot modules to efficiently roll across a surface [23]. They found that more elliptical shape achieves higher velocities, while circular shapes are more energy efficient (Figure 3).

The Molecube system [24], developed by Lipson et al., is another example of a chain-type modular system with only one degree of freedom but still able to achieve interesting three-dimensional (3-D) configurations. The single rotational joint in each Molecube module is located on the cube's longest diagonal between opposite corners. Given a one-dimensional (1-D) chain of Molecubes, rotating one joint will transform the chain into a 2-D L-shaped structure. Rotating a second joint will create an additional L-shape that extends in the third dimension, making the entire structure 3-D. Lipson et al. have shown that a short chain of Molecube modules, along with some free modules, can self-replicate. After executing a preprogrammed sequence of moves, the chain of modules can assemble the free modules into another identical chain that could, in turn, replicate itself again.

### Lattice Systems

Chirikjian et al. developed one of the first lattice-based modular robotic systems [25], [27] in which the modules are deformable hexagons capable of bonding with their neighbors. Each joint in the hexagonal modules is driven by a motor that can change the joint angle by at least 120°. By deforming and employing latches on each of its faces, a single module may traverse around the perimeter of a neighboring hexagon without ever losing contact. Chirikjian et al. also analyzed their system to produce bounds on the minimum and maximum number of single module traversals required to reconfigure from an initial shape to a goal formation. Others such as Walter et al. [28] have further analyzed these hexagonal type systems to create distributed motion planners capable of reconfiguring the system from one state to another.

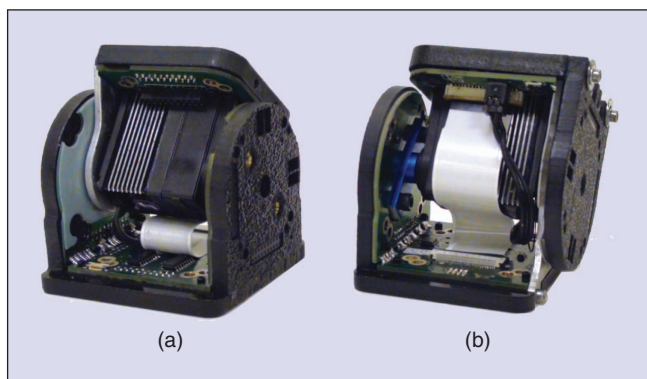
Murata et al. were also early contributors to the development of lattice-based modular robotic systems, with their



**Figure 2.** The Superbot modules have three rotational degrees of freedom and are designed to operate in real-world scenarios. Each Superbot module fits within a  $168 \times 84 \times 84 \text{ mm}^3$  rectangular box. (Picture courtesy of Polymorphic Robotics Laboratory, University of Southern California, Dr. Wei-Min Shen.)

development of a roughly hexagonal module capable of rolling around its neighbors in 2-D [29], [30]. Kurokawa et al. expanded on this 2-D system with a 3-D version [31] composed of cubes with six protruding arms capable of rotation. The six arms were driven by a common timing belt, but each could be disconnected from the belt using a clutch mechanism. The end of each arm was fitted with a connector capable of mating with matching connector on a neighboring module. By latching to a neighbor, rotating one or more arms, and then unlatching, the system could reconfigure itself in 3-D.

Yoshida et al. improved on this system with a new design that used shape-memory alloy actuators to rotate one robot module around the perimeter of a neighbor [32]. The smallest of these modules fits inside a 2-cm cube and weighs only 15 g. Yoshida et al.'s system, which is currently confined to 2-D, consists of square modules that include two male connectors (on opposing vertices) and two female connectors (on the opposite set of opposing vertices). The male connectors can swing through an 180° arc and bond with their female counterparts. This design allows one module to traverse around the exterior of a number of other connected modules. One drawback to the system is the amount of power it consumed. The



**Figure 3.** The CKBot modules each contain one rotational degree of freedom and are capable of mating with other modules in a variety of configurations. Each module fits within a 60-mm cube. (Picture courtesy of Prof. Mark Yim, University of Pennsylvania.)

system's connectors do not dissipate power in their static state, whereas actuation requires 1 A. Consequently, the modules do not contain their own power supplies. Furthermore, all of the processing required for the system's motion planning is performed on a separate computer, after which control commands are transmitted to the individual modules.

In [32] and a separate article [33], Yoshida et al. describe a 3-D adaptation of this system. Additionally, [33] presents a recursive method for describing the structures formed by modular robots. This recursive representation uses multiple layers of abstraction to separate the low-level details of the structure from the structure's general shape. For example, eight modules could be arranged to form a cube, which is then viewed as a single, indivisible node by higher level descriptions of the system as a whole. The authors state that using such a recursive representation enables one to describe sizable, complicated shapes that would be impossible to characterize otherwise.

Rus et al. also explored the idea of 3-D modules capable of reconfiguration through a series of latching, rotations, and unlatching with the molecule system [34], [36]. Each molecule module, two of which are shown connected in Figure 4, was composed of two identical halves called atoms. These atoms were attached to each other with a rigid 90° connector. Each atom had two rotational degrees of freedom so that a molecule had a total of four—two less than Kurokawa et al.'s system addressed in the previous paragraph. The molecules initially attached to their neighbors using magnetic forces, but the system was later redesigned to use mechanical latches. Kotay et al. proved that the molecule system, despite its unique motion constraints, was capable of performing a wide array of reconfiguration tasks.

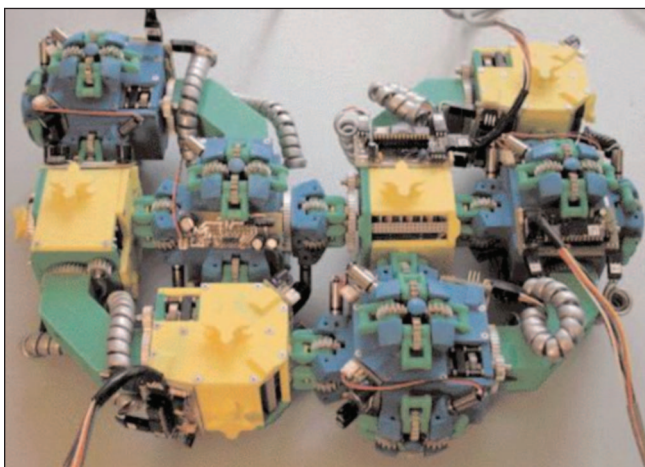
In [37] and [38], Vona and Rus describe a different type of deformable lattice system. The crystal system, as shown in Figure 5, is composed of square modules able to expand and contract by a factor of two in the  $x$ - $y$  plane. The crystal modules are composed of four movable faces arranged in a square. Two

of these faces contain active connectors, and two contain passive connectors. The active connectors can mate with passive connectors. By selectively latching and unlatching from their neighbors, a collection of crystals is able to arbitrarily modify its structure in 2-D. The authors also present algorithms to accompany the crystal hardware. These algorithms prove that composite crystal structures can assume arbitrary configurations and that any individual module can relocate to any position in the structure. Suh et al. expanded on the crystal concept with the telecubes [39]. Like the crystal system, the telecubes were able to expand their dimensions by a factor of two. Additionally, the telecubes could move in 3-D by expanding all six of their faces.

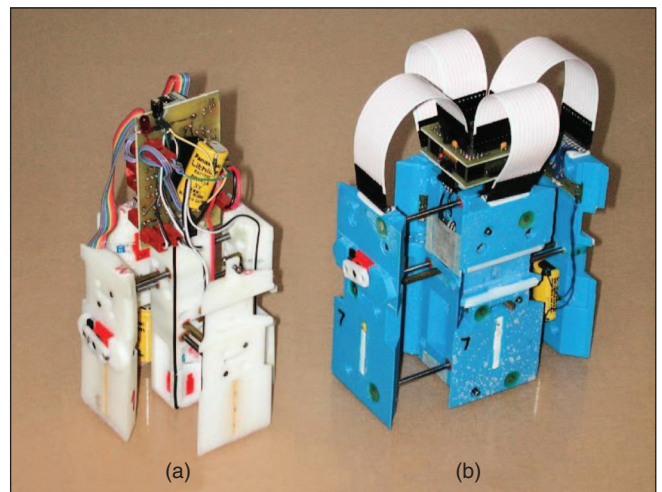
Chiang and Chirikjian described how to perform motion planning in a lattice of rigid cubic modules able to slide past each other [40]. Their approach, which begins with a description of the initial and final configurations of the system, consists of repeatedly finding the bisecting configuration (i.e., midpoint) between the start and end configuration. By recursively bisecting each new pair of configurations, they are able to produce an ordered set of simple, single-module motions that transform the initial configuration to the final configuration in an efficient manner.

The CHOBIE robot developed by Koseki [41] is able to actually perform the sliding motion assumed by Chiang and Chirikjian in [40]. The modules in the CHOBIE system, which are rectangular, are able to move by sliding in two planes relative to one another. A module cannot only slide horizontally across the top of another module but vertically down a module's side as well. This ability allows one robot to climb up and over another. Although the CHOBIE robots contain only basic processing power, they are self-contained and able to operate untethered. The system is confined to 2-D.

More recently, An developed the EM-Cube system [42] that is also capable of sliding motion. The 60-mm cubic modules in An's system rely on permanent magnets to keep neighboring modules in contact (without consuming power) and



**Figure 4.** The Molecule system, developed by Kotay and Rus [35], uses a series of latching, rotation, and unlatching to self-reconfigure. Each module has four degrees of rotational freedom and mates with its neighbors using gendered connectors. (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT.)



**Figure 5.** The Crystal module, developed by Rus and Vona [38], is a 2-D lattice system that uses dimensional compression to reconfigure. Each module can expand its linear dimensions by a factor of two. (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT.)

electromagnets to induce both linear and rotational motion. The modules themselves are tethered to a power source and rely on a centralized controller, but their lack of moving parts is noteworthy. The system is capable of movement sequences that allow a module to move through both concave and convex configurations. Additionally, by introducing the ability to rotate one module with respect to its neighbor, the system allows for transitions from one plane to another. The magnets in the system are not quite strong enough to lift modules against gravity, so existing experiments have been confined to the  $x$ - $y$  plane.

Another unique lattice is the I-Cube developed by Khosla et al. [43], [44]. The 3-D I-Cube system consists of passive cubes that are connected by active links with three rotational degrees of freedom that are able to grab, reposition, and release the cubes. By passing the cubes from one link to another and by directing the links to traverse over a stationary lattice of cubes, the system can reconfigure. To complement their hardware, Khosla et al. developed a hierarchical motion planner that prescribes how to move from an initial configuration of cubes and links to a desired configuration. They do so by defining and planning over larger groups of cubes and links termed metacubes. Each metacube consists of eight modules and 16 links. The planner first generates a motion plan for the metacubes. Once this is complete, it generates a plan for each individual I-Cube. Finally, using the motion plan for each I-Cube as a guide, it generates a motion plan for each link in the system. The I-Cubes may also be thought of as a special case of a truss-based system in which cubes (nodes) are connected by links or struts. In the I-Cube system, all nodes reside on a cubic lattice.

The 3-D I-Cube system was an improvement of the 2-D system [45] developed by Hosokawa et al. for rearranging cubic modules in a vertical plane. The authors relied on robots with attached arms that were able to pivot as well as extend and mate with the arms of a neighboring module. When two modules linked arms, one module could lift the other above and overhead. The stationary module could deposit the moving module either on top of a neighbor or itself.

The ATRON system [46], [47] was developed to improve upon the M-TRAN system. Lund et al., who developed the ATRON system, wanted to keep M-TRAN's ability to form dense lattices from inherently anisotropic modules using connectors in addition to those at the head and tail of each module. Additionally, they wanted to take advantage of the two orthogonal degrees of freedom (pitch and yaw) found in the CONRO system. Unlike M-TRAN, the CONRO system could not form tightly packed lattices. To accomplish these goals, Lund et al. developed the ATRON module that is spherical in shape, driven by a single rotational degree of freedom about its equator, and employs eight-gendered connectors. In an ensemble, the ATRON modules are arranged so that the rotational axes of neighboring modules are perpendicular. Christensen et al. [48], [49] have worked to combine collections of ATRON modules into virtual metamodules that can be used to simplify the planning and execution of self-reconfiguration much like Khosla et al. did for the I-Cube system.

As shown in Figure 6, Rus et al. developed the Miche system [50], [51] that is capable of 3-D shape formation through



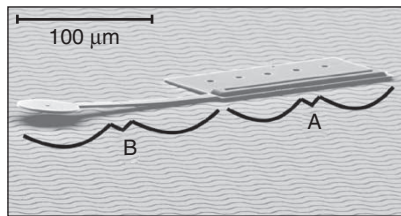
**Figure 6.** The Miche system consists of 45-cm cubic modules capable of forming 3-D structures. The modules contain batteries, two microprocessors, IR communication, and switchable permanent magnets for bonding with neighboring modules [51]. (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT.)

a process of self-disassembly, a process that removes excess modules from an initial structure to reveal the desired shape. Like a sculptor that chips extra stone from a block of marble to form a figure, the Miche system (short for Michelangelo) removes extra modules from an initial collection to realize a goal shape. Each module in the system is a completely autonomous 45-cm cube with three active and three passive faces. The intermodule latching mechanism only consumes power when connecting or disconnecting. The algorithms controlling the shape-formation process only transmit the minimal amount of information needed to each module in the system. At no point is the entire shape to be formed transmitted to any single module, much less all modules.

One of the newest lattice-type modular robotics is an aerial system composed of identical, hexagonal, single-rotor modules [52]. A group of modules may connect to form a flying platform with an arbitrary arrangement of multiple rotors. In addition to the ability to fly, each module contains wheels so that the system may self-reconfigure on the ground for the specific task at hand.

To date, almost all modular robotic systems have been limited to a few dozen hardware units. Hardware is expensive and time consuming to build, difficult to program and debug, and challenging to maintain. These realities have not prevented researchers from developing algorithms and software systems capable of simulating enormous number of modular robots cooperating to accomplish meaningful tasks. Fitch and Butler present one solution [53], [54] for reconfiguring hundreds of thousands of modules in a distributed manner. They

parallelize the process of planning a path for each module from its current position to a goal location by framing the task of planning each module's path as a Markov decision process. By including randomness in the planning model, each module may act using only local information without regard for the actions of other modules in the system. To find an optimal set of paths, Fitch and Butler use dynamic programming combined with a local cycle search. The search for a cycle in the modules ensures that by moving a module serving as an articulation point does not break connectivity of the system as a whole.



**Figure 7.** Donald et al. [60] developed this scratch-drive robot that is driven forward when voltage pulses are applied to the electrodes embedded in the substrate on which it operates. (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT. The Scratch Drive Actuator was developed at Dartmouth College.)

### Truss Systems

Unlike the lattice-based systems just described, the truss-based system does not need to operate on a cubic or any regular lattice. Most truss-based systems under development employ struts that expand or contract to achieve structural deformation. One of the first such systems to employ these telescoping links was Tetrobot [55]. Because all the links in the system may change their length, the resulting structure can easily deform its shape in a variety of ways.

The Odin system, conceived by Lyder et al. [56], [57], consists of three physically different types of modules: active strut modules capable of changing their length, passive strut modules of fixed length, and joint modules with 12 connection points. The joints are designed such that they may form the basis of a cubic close-packed structure. They are responsible for transferring power and communication signals between the strut modules. The strut modules may be reconfigured to perform a variety of tasks. Some are configured as telescoping modules capable of changing their length. Others may be configured as battery modules and others as camera modules. The two male connectors integrated into each strut module are spring-loaded and able to deflect  $23^\circ$  in any direction from their neutral position. The ability to deflect imbues the entire structure with its ability to deform.

The biologically inspired Morpho system [58] developed by Nagpal et al. is similar to Odin. It also uses active links, passive links, and connector cubes. The Morpho system adds the concept of membranes that can cover 2-D and 3-D curves formed by the links in the system. The authors present two interesting applications of these membranes. First, they show that a series of deformations of the membrane can transport an object from one side of the membrane to the other. Second, they show that the system could be configured as bridge (the membrane as the bridge surface) that is able to adapt to rough terrain to keep the surface of the bridge level.

Not all truss-based systems have relied on changing the length of the linkages that compose the system. One alternative system created by Amend and Lipson [59] employs linkages that are capable of changing their rigidity. The linkages are formed from tubular elastic membranes filled with granular material that jams when compacted but flows smoothly when

pressure is removed. By applying a vacuum to the elastic membrane, the authors were able to transition each strut from a limp, easily deformable shape to a more rigid cylinder capable of supporting weight. The authors demonstrate that that six such deformable links can be arranged to form a tetrahedron capable of changing its shape, but they make no attempt to automate or systematically control the structure. The authors also present the larger vision of creating programmable matter using the phenomenon of jamming. They envision macroscale objects (e.g., furniture) that are able to modify their shape or material properties on demand.

### Free-Form Systems

Several research groups have developed modular robotic systems that defy characterization as either a chain or a lattice. These free-form systems have the ability to aggregate modules in at least semiarbitrary positions. One such system is the micro-electromechanical systems (MEMS) robot developed by Donald et al. [60], [61]. The system, as shown in Figure 7, consists of thin ( $7\text{--}20\ \mu\text{m}$ ), rectangular (approximately  $260 \times 60\ \mu\text{m}^2$ ), scratch-drive robots capable of moving on an insulating substrate embedded with electrodes. By pulsing the voltage applied across the substrate electrodes, the rectangular body of the robot is driven up and down creating forward motion. Additionally, the authors designed the robots with elevated tails that, with sufficient electrode voltage, are attracted to the substrate and serve as pivot points that transform what would be forward motion into curved trajectories. The authors have used four of these robots to build larger composite structures [61]. By varying the design parameters of the four robots so that each responds to a different set of actuation voltages, the authors are able to drive each module independently of the others so that multiple modules, if initially separate, can be driven together to form an assembly. Remarkably, Donald et al. are able to achieve an accuracy of  $5\ \mu\text{m}$  when docking two modules.

Another example of a system without a regular lattice structure is the Slimebot [62], [63] created by Shimizu et al. The system consists of identical vertical cylindrical modules that move on a horizontal plane. The perimeter of each module is covered by six genderless hook and loop patches used to bond with neighboring modules. These patches are not rigidly fixed to the module body, but each diametric pair oscillates radially in and out from the center of the body. Each module has one additional degree of freedom, allowing it to increase or decrease its friction with the surface on which it sits. Each module in the system acts as an oscillator that is lightly coupled to its neighboring oscillators. While oscillating, the module alternates between an active and passive state. In the active state, the module attempts to move itself by decreasing its friction and moving its connectors. In the passive state, the module acts as an anchor against which active modules may push. It does so by increasing its friction while leaving its connectors stationary. External



stimuli, such as light, may be used to create a phase gradient across the oscillators in the system. As the result of such a gradient, the system as a whole will move toward the light source. The Slimebot system is unique, because it does not rely on traditional algorithms or careful planning to achieve coordinated movement. Instead, it relies on analog processes that mimic nature.

Around 2005, Goldstein et al. [64] and Goldstein and Mowry [65] published several articles describing what they termed claytronic atoms or catoms. These vertically oriented cylindrical robots, which were incapable of independent motion, used 24 electromagnets around their perimeters to achieve rolling locomotion about their neighbors. While the Catom system could be considered a lattice-type system, we have categorized it as a free-form system because its modules do not need to form a regular lattice structure to function.

Goldstein et al. [64] envisioned a system in which millions of smaller catoms could form arbitrary shapes using a randomized algorithm that avoided conveying a complete description of the shape to each module in the system. Instead, the algorithm only distributed shape information to the modules at the boundary of the collection. By creating or absorbing void pockets (areas without any modules), the edge catoms could expand or contract the edge in their own proximity. Although not explicitly demonstrated in their article, the authors claimed that their shape-formation algorithm could be distributed across all modules in the system and that each module only required local information to execute it successfully.

One of the newest catoms systems is one envisioned by Karagozler et al. in [66]. The system that still appears to be under heavy development employs hollow cylinders rolled from SiO<sub>2</sub> rectangles patterned with aluminum electrodes. The authors hope that two of these cylinders, termed Catoms, when placed in close proximity with their axes aligned, will be able to rotate with respect to one another using electrostatic forces. Specifically, the electrodes (which reside on the inside of each cylinder and are electrically isolated by the SiO<sub>2</sub>) will be charged, so that they attract and repel mirror charges on the neighboring cylinder in a way that causes rotation. For the purposes of our classification, it appears that single Catoms are able to move independently of their neighbors if they are placed on an insulating, unbroken conducting substrate. In its current instantiation, the Karagozler's system appears to be constrained to form 2-D structures. The authors claim the completed system will have a yield strength similar to that of plastic and that the modules will be able to transfer power and communication signals capacitively from neighbor to neighbor.

## Self-Assembling Systems

In an attempt to simplify the process of creating intricate modular robotic systems, researchers have attempted to mimic and improve upon natural self-assembling systems. Self-assembling systems are common in nature: Geologic forces crystallize polygonal columnar basalts with remarkable regularity, and DNA in all living organisms uses a soup of free nucleotides to self-replicate during cell division. Scientists and engineers have long been interested in these types of self-assembling

systems because they display the ability to spontaneously create complex structures from simple components.

Whitesides et al. have investigated a wide variety of engineered self-assembling systems [67], [68]. In one system, they employed truncated octahedra covered in electrical contacts to form 3-D electrical networks [69]. They found that if these 5-mm octahedra were placed in a liquid at a temperature above the melting point of the solder covering the electrical pads and gently agitated, the modules would self-align to form structures as large as 12 units. The main drawback of this system is that one cannot control the final shape of the assembled structure: the modules may form a chain, a cube, or a more irregular shape.

Miyashita et al. performed a more theoretical analysis of self-assembly using pie-shaped pieces to form complete circles [70]. The authors performed analysis, simulations, and experiments to better quantify how angular size of the pie-shaped pieces affected the yield rate of completely assembled circles. In the process, they followed Hosokawa et al.'s lead [71] and modeled the system as chemical reaction. For their experiments, the authors used floating modules with permanent magnets to bond with their neighbors and vibrating pager motors to induce stochastic motion. By varying the voltage applied to all motors, the authors could affect what types of structures were formed even though the modules lacked any intelligence or communication capabilities. Other researchers have proposed equally simple system in which the modules do not have any innate actuation ability. Shimizu and Suzuki have developed a system of passive modules capable of self-repair when placed on a vibrating table [72]. The vibrations of the table cause rotational motion in the modules that wind in a string attaching each to the other modules in the system. When all the strings are wound in and taut, the system assumes an ordered configuration.

Some computer scientists have also investigated theoretical aspects of self-assembly in the context of 2-D tiles that selectively bond with their neighbors to form simple well-defined shapes, such as squares [73]–[75]. Typically, these tiles are allowed to translate but not rotate when moving randomly in the plane. Each side of every tile in the system has an associated bonding strength (different edges may have different strengths). When two tiles collide, they remain attached only if their cumulate bond strength exceeds a globally defined system entropy. The shape formed by this type of tile system is dictated by both the system entropy (which can be adjusted dynamically) and the types of tiles involved. To form a specific shape, one needs to undertake the relatively complicated task of designing a set of tiles with appropriate bonding strengths. Once this design step is complete, the tiles themselves do not need to display more than the minimal amount of intelligence necessary to determine when to bond.

Klavins et al. have worked to develop a more intelligent self-assembling system that employs triangular modules driven by oscillating fans on an air table to self-assemble different shapes [76]. The modules in the system can communicate and selectively bond using mechanically driven magnets. In addition to developing this hardware platform, the authors employ

knowledge of the module's local topology and internal module state information to execute a set of rules, (called a graph grammar) to form arbitrary shapes. The practical result of this approach is that each module can decide, in a distributed fashion, when to maintain or break a connection with its immediate neighbors. Global knowledge of the complete structure is not required. In additional work [77], Klavins et al. show that one can define a continuous time Markov process describing how a specific graph grammar will cause a particular system to behave. Using this model, they argue that one can quickly evaluate the performance of the grammar without running physical simulations or experiments. Griffith et al. have also worked with intelligent modules capable of selective bonding to show that self-assembling systems may self-replicate [78] if given an original configuration of modules to be duplicated, an excess of free modules, and a simple set of local rules.

Mataric et al. [79] have also presented rule-based approach to self-assembly termed transition rule sets. In particular, they present a method that, given a goal structure, produces a set of rules shared among all modules that govern when and where new modules are allowed to attach to the growing structure. Zhang et al. [80] have expanded on this work by optimizing the size of the rule sets used to form a specific shape. Werfel [81] also applied the idea of a transition rule set when studying the use of swarms to assemble complex structures from passive materials.

White et al. have developed hardware and algorithms for several 2-D stochastically driven self-assembling systems [82]. One of the hardware instantiations uses triangular modules with mechanically driven permanent magnets. The other uses square modules with electromagnets for connectors. Both systems lack batteries, and the modules only receive power after they connect to the structure being self-assembled. To form specific shapes, each module is provided with a representation of the desired shape and decides, based on its location in the structure, whether to allow other modules to bond to its faces. As the authors highlight, this approach leads to holes or concavities in the finished structure. These imperfections arise when free modules are blocked from attaching to potential bonding sites by other, already attached modules. To rectify this problem, the authors propose assembly of the structure in a layered fashion—one layer must be completed before the next can begin. The problem with this approach is that it requires global communication so that all modules in the outer layer know when the layer is complete.

Lipson et al. extended their 2-D system to 3-D [83], [84] by using cubic modules suspended in turbulent fluid to achieve self-assembly and reconfiguration. The modules themselves are unable to move on their own. As the free modules circulate in the fluid, they pass by a growing structure of assembled modules. When they come close enough, they are accreted onto the structure. The authors present two different systems: one that latches using electromagnets and the other that relies on the suction forces that result from fluid flow through a restrictive orifice. By activating valves on specific faces of each module, the authors can control the flow of the liquid and hence where unattached modules are allowed to bond to the growing structure. The main advantage of the fluidic system is that it could be scaled down to produce microscale modules.

## Simplifying Self-Assembly

The majority of existing self-assembly systems aim to form structures in one of two ways. Some systems such as [70], [72]–[75] use a collection of application-specific differentiated modules that are only capable of assembling in a particular fashion to form a specific shape. To self-assemble interesting structures, this type of system requires considerable initial design effort, may be time consuming to fabricate given the different types of modules required, and is challenging to reuse if one wants to form a different goal shape. The advantage to this type of system is that the modules may be less complex or completely passive.

In contrast, other systems such as [76], [77], [79]–[84] use completely generic modules with more computation and communication ability embedded in each module. As a result, these generic modules are likely to cost more and be more prone to failure. The advantage is their universality that allows for the same system to form a wide variety of structures by simply changing the software rules that govern when and where modules are allowed to bond. Both types of systems aim to form complex shapes in a relatively direct manner. As these structures grow from a single module, new modules are only allowed to attach to the structure in specific locations. By carefully controlling these locations and waiting for a sufficiently lengthy period of time, the desired structure is formed.

We propose a new approach that eliminates many of the complexities of shape formation by active assembly. Our Smart Pebble system employs a set of distributed algorithms to perform two discrete steps: 1) rely on stochastic forces to self-assemble a close-packed crystalline lattice of modules and 2) use the process of self-disassembly to remove the extra material from this block leaving behind the goal structure. By approaching shape formation in this manner, we hope to speed up the entire process, eliminate any global information that must be distributed throughout the system, and simplify the computing requirements of each module.

As the individual modules in self-reconfiguring and programmable matter systems continue to shrink in size, it will become increasingly difficult to actuate and precisely control the assembly process. In particular, designing modules capable of exerting the forces necessary to attract their neighbors from significant distances will be challenging. Instead, these systems may find assembly and disassembly much simpler when driven by stochastic environmental forces. The Pebble modules presented in the “Hardware for Self-Assembly and Disassembly” section, which are able to latch together from distances approximately 20–35% of the module dimensions, could easily take advantage of these stochastic assembly mechanisms to form an initial structures. Our particular system also relies on external forces to carry the unused modules away from the final shape. In our system, this force is often gravity, but it could also be vibration, fluid flow, or the user reaching into the bag of smart sand particles to extract the finished object.

Another advantage of our Smart Pebbles system is the mechanical simplicity. The Pebbles contain no moving parts, making them simpler to manufacture, less expensive, more reliable, and easier to miniaturize than more traditional modular robotic systems which often rely on complex, gendered

mechanical latches to connect neighboring modules. The simplicity of their shape also ensures that the modules passively self-align eliminating the need for precision sensing and motion control.

Figure 8 illustrates the process the system must undergo to form a shape. Starting from a collection of loose modules, the system self-assembles to form an initial block of material. Once this initial close-packed block is large enough to enclose the goal shape, the system undergoes self-disassembly, leaving the finished structure behind. To form another shape, the finished structure disintegrates and the unbonded modules can again self-assemble into a close-packed lattice.

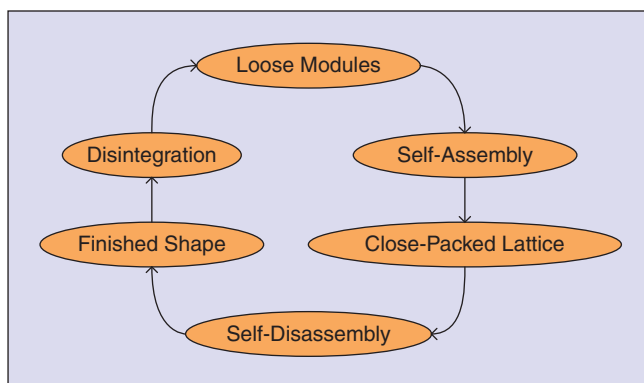
During the initial formation of the close-packed lattice, we make only limited attempts to restrict which modules or faces are allowed to bond with the growing structure. Unlike the graph grammar and transition rule set approaches of [76], [77], [79], [80], we make no attempt to encode the final shape to be formed in a set of local rules. The only rules enforced by our self-assembly algorithm are designed to ensure that we achieve a close-packed structure to serve as the basis of the self-disassembly process. As illustrated in Figure 9, after we form this initial block of material, we complete the shape-formation process through self-disassembly and subtraction of the unwanted modules.

For a more concrete application of our system, consider an isolated situation, like interplanetary spaceflight or a research station at the South Pole, where weight and space are severely limited, a bag of Smart Sand could serve as a universal toolkit capable of forming task-specific tools with a high degree of precision on demand. An astronaut would convey the shape of the desired tool to a bag full of loose modules and then begin to gently shake the bag. As the modules inside came into contact and naturally aligned with their neighbors, they would selectively bond and unbond to form the desired tool. The astronaut would then open the bag, reach inside, grab the finished form, brush off the extra grains, remove and use the tool. Once finished with the tool, the astronaut could drop it back into the bag where it would immediately disintegrate so that the modules could be reused to form a different structure.

In particular, our Smart Pebbles are a modular system in which each module is an identical cube with planar faces. In our model of the system, we make two basic assumptions: 1) modules are able to communicate with their immediate neighbors and 2) modules can mechanically bond with and release their immediate neighbors. Additionally, the modules each contain some limited processing power and memory. The modules themselves lack any mechanical degrees of freedom, and the structures that the system forms are rigid.

### Hardware for Self-Assembly and Disassembly

To realize a self-disassembling system in hardware, we need unit modules capable of performing two basic tasks: selectively latching and unlatching with their neighbors and communicating with their immediate neighbors. These two tasks also imply that each module contain some limited degree of intelligence that may be realized in a microprocessor or hard-coded in an ASIC. One additional requirement on the latching mechanism is that, when deactivated, it does not protrude beyond the face of the

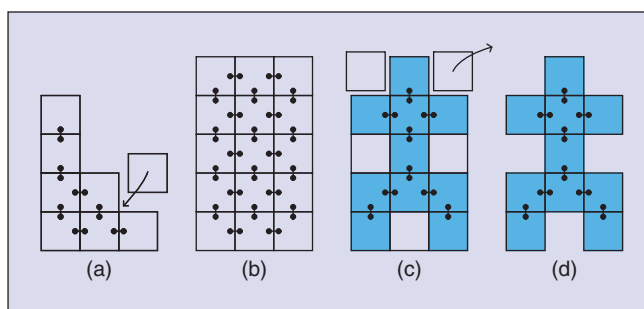


**Figure 8.** The Smart Sand system. This can be reused to form wide variety of solid shapes.

module. This ensures that modules can be removed from the structure to create concavities even if they are surrounded by neighbors on all but one side. We have realized these requirements in two different hardware systems: first Miche [50] and then the Pebbles [85]. Both types of module are shown in Figure 10.

The Miche modules are 45-cm cubes fabricated from a set of six PCBs that are joined together through a combination of interdigitated fingers and dual in line pluggable connectors. Each module in the Miche system is powered by rechargeable batteries powered and relies on an ARM7 processor to execute all of the self-disassembly algorithms. The modules communicate with their six immediate neighbors at 9,600 b/s using an infrared (IR) light-emitting diode (LED)/photodiode pair.

The Miche modules have three active faces and three passive faces that are composed of ferromagnetic steel plates. Active and passive faces latch together using switchable permanent magnets embedded in the active faces. These magnetic connectors consist of two permanent magnets. One is fixed, and the other can rotate to align or oppose the magnetic field of its fixed counterpart. This rotation is affected by a small dc motor driving a worm gear that turns a spur gear attached to the magnet. When the two magnets are aligned north to north, their fluxes add and the two act as a single magnet to attract the passive steel face of the neighboring module. When the two magnets are arranged north to south, the fluxes effectively cancel each

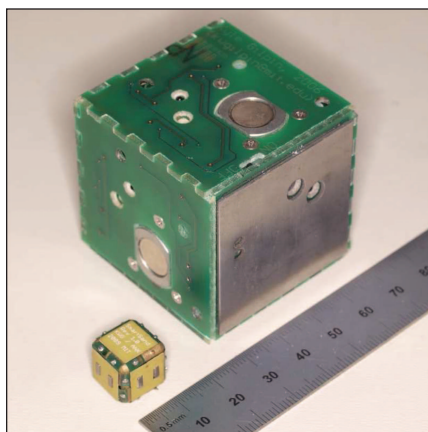


**Figure 9.** (a) To form shapes through subtraction, modules initially assembly into a regular block of material. (b) Once this initial structure is complete and all modules are fully latched to their neighbors, (c) the modules not needed in the final structure detach from the neighbors. (d) Once, these extra modules are removed, we are left with the final shape.

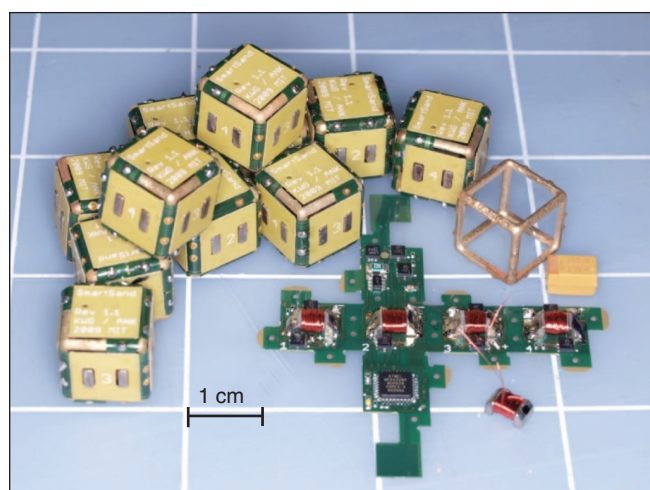
other, very little flux passes through the neighboring steel plate, and the two modules are not attracted to each other. When a connector is on, it can support 20 N—the weight of 17 other modules. One disadvantage of the system of active and passive faces is that all modules in an ensemble must be oriented in the same way. If they are not, some passive faces will be adjacent to other passive faces, and these modules will be unable to bond.

The Smart Pebbles are a much improved version of the Mische hardware. The Smart Pebbles are 50 times smaller by volume (as shown in Figure 10), about five times stronger when normalized by module weight, use gender-less connectors, and do not require recharging. The only significant drawback to the Pebble system is that it can only be used to form 2-D structures. More specifically, each module in the Pebble system is a 12-mm cube capable of autonomously communicating with and latching to four neighboring modules in the same plane to form 2-D structures. Each completed module weighs 4.0 g and may be rotated any one of four ways on the assembly plane and still mate with its neighbors. The major functional components of each cube are power-regulation circuitry, a microprocessor, and four electropermanent (EP) magnets, which are responsible for latching, power transfer, and communication.

The Pebbles are formed by wrapping a two-layer flexible printed circuit around an investment-casted brass frame. The flexible circuit is reinforced with polyimide sections that stiffen the



**Figure 10.** The Pebble modules are 50 times smaller by volume (12 versus 45 mm per side) and five times stronger by weight. (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT.)



**Figure 11.** Each programmable matter Smart Pebble is a cube with 12-mm sides. A collection of Smart Pebbles are able to form complex 2-D shapes using four EP magnets that are able to hold 85 times the individual module weight. The Pebbles are formed by wrapping a flexible circuit around a brass frame. An energy storage capacitor hangs between two tabs occupies the center of the module. (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT.)

faces of the cube and serve as a solid mechanical foundation for the surface mount components mounted on the top of the circuit. All components are mounted on the top side of the flex circuit so that they end up on the inside of the cube. The four active faces contain cutouts so that the two poles of each EP magnet are able to protrude from the interior of the cube. The other two faces are occupied by the microprocessor responsible for controlling each module and the power conditioning and regulation circuitry. All the major components of a single Pebble, in addition to several fully assembled modules, are shown in Figure 11.

The Pebbles do not contain batteries. So, power is injected into the system

through contacts on the bottom of one-root module and then distributed through the system from one module to the next through the poles (which are electrically isolated from each another) of the EP magnets. As a result, the EP magnet poles of neighboring modules must be arranged north to south, the voltage seen from the North Pole referenced to the South Pole in a particular cube may be positive or negative. A full-wave bridge rectifier is used to convert this to a known polarity before it is used to charge a 150- $\mu$ F capacitor that provides local, low-impedance energy storage. The electrical resistance of each module is only 0.3  $\Omega$ . Given that each module consumes only 15 mA, more than 3,000 modules may be chained to a single root module supplied with 20 V before the resulting voltage drop at the end of the chain begins to approach the drop-out voltage of the 5 V regulator supplying power to the microprocessor in each module. In any real-world structure, a more reasonable aspect ratio would ensure many parallel electrical paths from the root to any other module, lowering the effective resistance.

In addition to transferring power, the EP magnets serve as the latching mechanism for neighboring modules. Each EP magnet consists of one rod of low-coercivity Alnico V and one rod of NeFeB, a high coercivity material. These two rods are placed side by side, capped with soft-iron poles, and wrapped with an 80-turn copper coil. The coil is energized with a 20-V, 300- $\mu$ s pulse that completely reverses the magnetic polarization of the Alnico while leaving the NeFeB unaffected. When the remnant fluxes of the Alnico and NeFeB are opposed (so that the two rods are aligned north to south), the flux circulates from one rod to the other and never leaves the greater EP structure. As a result, the EP does not attract ferromagnetic materials and the connector is effectively off. Alternatively, when the fluxes align (so that both North Poles are adjacent), the combined flux flows out the structure's nearer pole piece, through the EP magnet of any adjacent module, and back into the EP magnet structure through the other pole piece. In this configuration, the connector is on, and two modules will attract with an empirically measured force of

3.2 N—enough to support more than 80 other modules. Once the connector is on or off, no additional energy is required to maintain its state.

Each module also uses its EP magnets to communicate with its neighbors at 9,600 b/s. When connected, two adjacent EP magnets form an effective 1:1 isolation transformer. By sending a short pulse of current through the coil of one EP magnet, we induce a similar pulse in the coil of the neighboring module. These pulses are only 1- $\mu$ s long and polarized such that they always strengthen the bond between neighboring modules. To save space inside each module, the four EP magnets share some circuitry so that only a single EP magnet may latch, transmit, or receive at a given time. We overcome this limitation with algorithms for handshaking and multitasking. All of the software executes on an Atmel ATmega328, an 8-b processor with 2 KB RAM, and 32-KB flash running at 8 MHz. For more information about all aspects of the Pebble hardware, see [85].

### Algorithms for Self-Assembly and Disassembly

Self-disassembling systems require two high-level capabilities: 1) the ability to aggregate the initial block autonomously and 2) the ability to virtually sculpt the block into the desired shape. In this article, we present a solution for 1), and algorithms for 2) have been presented in [51]. The two capabilities are interrelated. When the utility of an object is exhausted, its modules are returned to the collective system. The self-assembly operation will create the new block, which, in turn, will be sculpted into the next object by self-disassembly. Thus, to ensure the creation of a wide range of objects, it is important that the result of self-assembly be a solid block without internal holes. The rest of this section describes a decentralized algorithm that guarantees the formation of a block without internal holes.

By initially aiming to form a close-packed lattice during the self-assembly phase, we eliminate the need to transmit a description of the goal shape, of any form, to every module in the structure. Previously, this blueprint has been necessary so that modules on the boundary of the structure know whether to allow new neighbors to bond. Without a need for this blueprint, the intermodule communication requirements are significantly reduced as are the storage and processing requirements of each module. Additionally, the fact that the self-assembly process is forming a close-packed lattice will help to guide free modules into alignment with their neighbors. Following the assembly process, we employ a set of self-disassembly algorithms that only transmit the minimal amount of shape-description information. These algorithms avoid transmitting the entire shape description to the structure as a whole.

### Self-Assembly of Initial Block

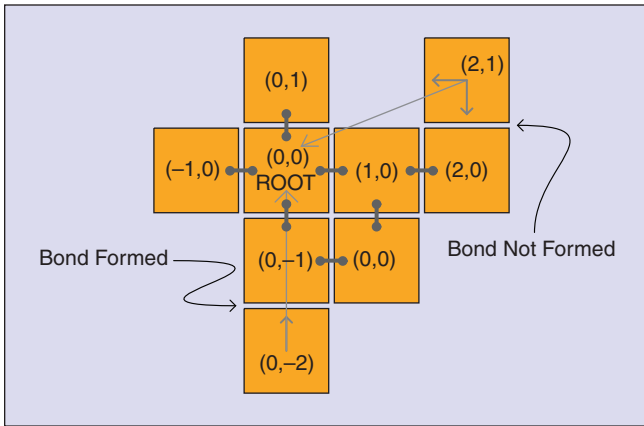
During the self-assembly process, we want to ensure that no gaps are formed in the growing structure. Gaps weaken the structure and reduce the available communication paths. If we allow new modules to be accreted at any location on the growing structure, it is easy to create gaps in the structure that are theoretically difficult and practically impossible to fill. To be more specific, a loose module will never fill a lattice position that is already surrounded on three sides. Therefore, our

goal is prevent the creation gaps surrounded by neighboring modules on more than two sides. Doing this also guarantees that we do not create holes in the structure.

To avoid holes in the self-assembled structure, we propose a simple distributed algorithm that only requires local information. Based on this information, each free module coming into contact with a potential bonding site on the solidified structure must decide whether to permanently bond with the structure or move on and look for another bonding site. The algorithm we describe is similar to the self-assembly rule set generated by Mataric et al. in [79] for forming a rectangular structure. Mataric et al.'s work focuses on the broader question of how to generate a set of rules to assemble arbitrary structures and, as a result, generates a larger, more complex set of rules that depends on each module knowing in which of eight potential sectors it resides. In contrast, our work focuses on developing a minimal complexity, easy to implement algorithm that guarantees the assembly of a close-packed lattice. By following the self-assembly process with self-disassembly, we eliminate the need for complex sets of rules that govern when and where modules may attach to the growing structure.

Our self-assembly algorithm makes two assumptions. First, all modules correctly assume the location of the root module. This is easy to hard code into each module's process as location (0,0). Second, once each module is added to the structure, it can determine its (x,y) position. This requirement is also easy to meet. The user informs the one module anchored to the assembly platform that it is the root and therefore at location (0,0) and that it is rotated 0°. Using this information, the root can inform the module added to its right that the new module's location is (1,0). Likewise, the module added below the root is at location (0,-1). Based on which of its faces the new module receives this message, it can determine its orientation. Now that the root's neighbors know their locations and orientations, they inform their newest neighbors of their locations. More details and a proof that this algorithm is correct are proved in [51]. Note that the algorithm only requires neighbor-to-neighbor communication, and it does not rely on any global information being communicated within the structure. All modules in the structure are able to determine their coordinates without any concept of the structure as a whole.

The entire self-assembly algorithm, shown as pseudocode in Algorithm 1, begins as the free module receiving power when it comes into contact with a module that is already a part of the crystallized structure. Immediately, the module queries its neighbor to determine its location. Based on this location, the module then constructs a root vector pointing back to the root module. The vector may have  $x$  and  $y$  components. The new module permanently bonds with the structure—by calling the `LatchAllFaces()` function—if it detects that it has neighbors in both the  $x$ - and  $y$ -directions of the root vector (if they exist). For example, consider a new module that determines its location (10,2). As shown in Figure 12, the root vector is then (-10,-2) that has both  $x$  and  $y$  components. As a result, the module only bonds with the structure if it has neighbors at (9,2) and (10,1). Instead, if the new module were located at (0,-5) and the root vector was (0,5), the module in question would only bond if it detected a neighbor at (0,4).



**Figure 12.** During self-assembly, modules only permanently attach to the already assembled structure if they detect immediate neighbors along a vector that points back to the root module.

If the new module does not detect neighbors in the appropriate locations, it informs whatever neighbors it is contacting, and they deactivate their connectors releasing the module. The module will lose power, so when it next contacts the structure, the self-assembly algorithm will restart.

Once a module decides that should permanently bond to the growing structure, it enters a loop in which it simply listens for disconnect request messages on its faces. When a new module decides that it cannot connect to the structure, it sends one of these disconnect request messages—using the `UnlatchAllFaces()` function—to all of its neighbors. When the previously solidified module receives one of these messages on a particular face, this module keeps the connector on that face deactivated for a fixed period of time to allow the rejected module to move out of range of its attractive force. This is the purpose of the `DisableFace()` function in the pseudocode. Eventually, the connector is reactivated in hopes that the bonding site will have become valid.

### Algorithm 1

The self-assembly algorithm uses the existence or absence of two of a module's neighbors to determine whether it is allowed to bond with its neighbors and become a part of the self-assembling structure.

```

1: procedure Self-Assemble ( )
2:    $\vec{myPos} \leftarrow \text{Localize} ( )$ 
3:
4:    $\vec{root} \leftarrow (0, 0) - \vec{myPos}$ 
5:
6:   if  $\vec{root}.x \neq 0$  then
7:      $\vec{neighPos} \leftarrow (\vec{myPos}.x$ 
8:        $+\text{sgn}(\vec{root}.x), \vec{myPos}.y)$ 
9:     if NeighExists( $\vec{neighPos}$ ) = FALSE then
10:      UnlatchAllFaces()
11:    return
12:   end if
13: end if
14: if  $\vec{root}.y \neq 0$  then

```

```

15:    $\vec{neighPos} \leftarrow (\vec{myPos}.x, \vec{myPos}.y + \text{sgn}(\vec{root}.y))$ 
16:   if NeighExists( $\vec{neighPos}$ ) = FALSE then
17:     UnlatchAllFaces()
18:   return
19:   end if
20: end if
21:
22: LatchAllFaces()
23:
24: loop
25:   for  $face \leftarrow 1$  to 4 do
26:     if DisconRqstd( $face$ ) = TRUE then
27:       DisableFace( $face, LockoutTime$ )
28:     end if
29:   end for
30: end loop
31: end procedure

```

### Theorem 1

The self-assembly algorithm (Algorithm 1) prevents the formation of gaps in the lattice structure that are surrounded by more than two neighbors.

### Proof

Guaranteeing that the algorithm never creates a gap that is surrounded on more than two sides is equivalent to ensuring that, on any vertical or horizontal line of the lattice, an unpopulated gap between two distant modules is not formed. Consider, for illustrative purposes, any unoccupied position on the lattice and the horizontal (or vertical) line extending to positive and negative infinity from this point. If this line intersects solidified modules (arbitrarily far away) in both the positive and negative directions, one could imagine working from the solidified modules inward to fill this gap. Eventually, enough modules will be attached so that the initial unoccupied position has immediate neighbors to its left and right. Once this occurs, the empty position will be impossible to fill. As a result, if the algorithm avoids creating a gap, no matter how wide, along any horizontal or vertical transect of the lattice, it will avoid creating gaps in the lattice that are surrounded by more than two immediate neighbors.

The self-assembly algorithm, if it does not detect immediate neighbors along both the  $x$  and  $y$  components of a vector pointing from the potential bonding site to the root module, assumes that other, more distance modules may exist along those transects. As a result, by not connecting a module to the structure, the algorithm does not risk creating gaps along these transects.

Finally, the algorithm is guaranteed not to create gaps along the  $x$  and  $y$  vectors originating at the potential bonding site but pointing away from the root module. For this type of gap to be created, a solidified module would have existed farther away from the root in either than  $x$ - or  $y$ -direction than the bonding site in question. Conveniently, this is impossible. As explained in the earlier paragraph, a module will never bond if there is any potential for an empty position in the lattice along either component of the module's root vector, which, in this scenario, there would have been. ■

## Theorem 2

The self-assembly algorithm prevents the formation of holes in the lattice.

### Proof

By Theorem 1, the self-assembly algorithm never creates gaps with more than two neighbors, so the algorithm can never create a gap with four neighbors—the definition of a hole. ■

While the algorithm presented here has pertained to a 2-D system, the extension to 3-D is straightforward. Instead of a 2-D vector pointing back to the root module, each module will have a 3-D vector and will need to check for neighbors in three directions. Likewise, the 3-D algorithm guarantees that a gap with more than three neighbors will never be created, which implies that holes will never be created.

### Self-Disassembly

Once the initial block of material has been assembled, self-disassembly by subtraction proceeds through four basic stages: model formation, virtual sculpting, shape distribution, and disassembly. After self-assembly is complete, each module knows its location within the initial structure. In the first stage, model formation, each module, as evidence of its existence in the initial structure, sends a reflection message containing its position back to the root module. The root does not store these messages but forwards them to a graphical-user interface (GUI) running on a PC. The GUI builds a virtual model representing the initial arrangement of modules in the physical structure. Using this GUI model, the user drives the virtual-sculpting stage by selecting which modules should be included in the final shape. During the entire shape-formation process, the PC is the only entity that stores the entire shape description. After this sculpting process is complete, the PC program generates a sequence of inclusion messages. During the shape-distribution stage, the GUI transmits these inclusion messages to the root module. The structure then propagates these inclusion messages to their proper destinations. As with the localization process, the messages only contain local information. During the disassembly phase, the modules not designated to be in the final structure disconnect from their neighbors to reveal the shape the user sculpted previously. Each of the self-disassembly phases is dependent on a distributed, localized message passing algorithms executing on each module. Figure 13 shows a dog

being formed through the self-disassembly of a block of 27 Miche modules. As shown by the progression of time in the frames, most modules disconnect quickly and fall away under gravity's influence in less than 5 s. Others get stuck or disconnect but still rest on top of other modules. These must be removed by vibrating the structure or by hand.

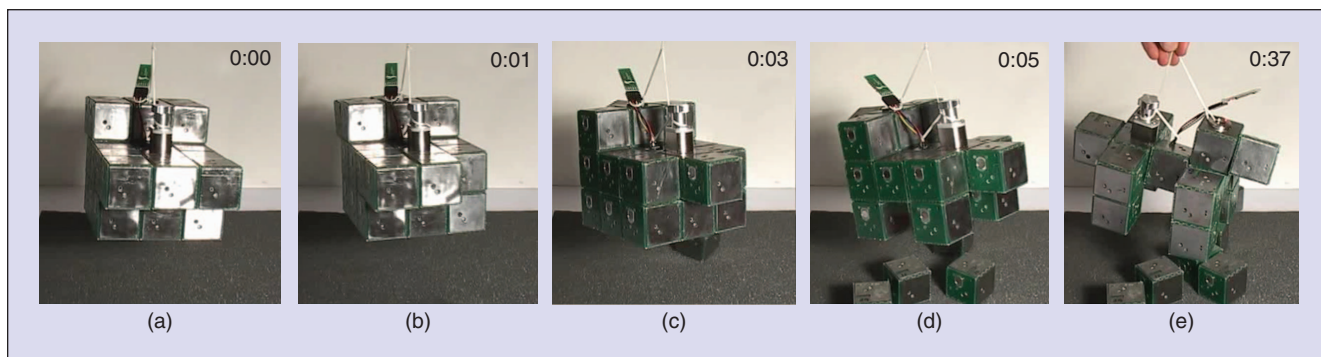
When self-assembly and self-disassembly are combined to form structures, the resulting shapes cannot contain internal cavities. Not only does the self-assembly algorithm, in its mission to form a close-packed lattice, avoid introducing holes, the self-disassembly process has no way to remove unnecessary modules from the interior of a structure if no exit hole exists. Additionally, as one may suspect, if the modules being removed from an initial block to form the final structure have a narrow or convoluted exit path, it is unlikely that they will be able to exit from the structure. In future work, we plan to examine how the self-assembly process can be modified to accommodate the formation of holes in the finished structure through a series of repeated self-assembly and self-disassembly iterations.

### Experiments

We have experimentally tested both the self-assembly and self-disassembly algorithms. We use a collection of 17 modules that was described in the “Hardware for Self-Assembly and Disassembly” section. In 3-D, we imagine shaking a bag full of modules to drive the self-assembly process. The 2-D analog is an inclined vibration table. As shown in Figure 14, we built a custom-vibration table that provides the stochastic forces necessary to move and align the modules. The frequency and amplitude of the vibration can be controlled, as can the tilt of the table. The perimeter of the table is surrounded by a low barrier that prevents modules from falling off.

In our experiments, we anchored one module, the root, in a corner of the vibration table at coordinates (0,0). The root module provides the power and communication link between the system and the user. Then, we tilted the table 4° in both the  $x$ - and  $y$ -directions to bias the movement of all free modules toward the root module located at (0,0).

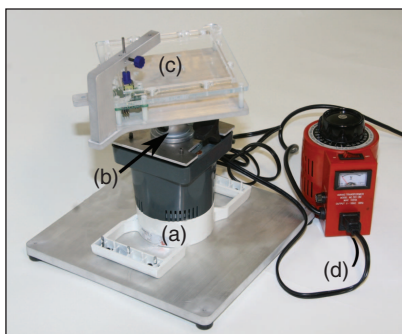
Using 16 randomly arranged modules (in addition to the fixed-root module), we first tested the self-assembly algorithms in a series of 13 trials. A progression of still images from one of these trials is shown in Figure 15. Initially, the connectors on each module are deactivated, and they are only turned on when



**Figure 13.** A dog-shaped structure can be self-disassembled from an initial configuration of 27 suspended Miche modules (each 45-mm per side). (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT.)

a module successfully communicates with the growing structure. The last frame shows that all modules bond together to form a solid shape that can then be used for self-disassembly. After the modules have coalesced and have been given sufficient time to latch with their neighbors, the structure can be removed from the test fixture without falling apart.

Figure 16 shows how the 17 modules tended to be distributed after all modules had settled into discrete grid positions. Not surprisingly, the experiments show that the modules tend to form an isosceles right-triangular configuration. In addition to determining the most likely distribution of initial modules, we wanted to ensure that all modules were able to bond with their neighbors and communicate with the system's PC-based user interface. In a series of 15 trials, each using 17 modules, we observed a total of only 22 instances in which a module failed to localize and send a message back to the user interface through the root module—a failure rate of 8.2% vector but not the other. In most of these cases, one or more modules were clearly not in contact with one more of its neighbors. In one particularly bad trial, one of the modules adjacent to the root was about 45° out of alignment, resulting in the 13 of the 17 modules in the system not localizing,



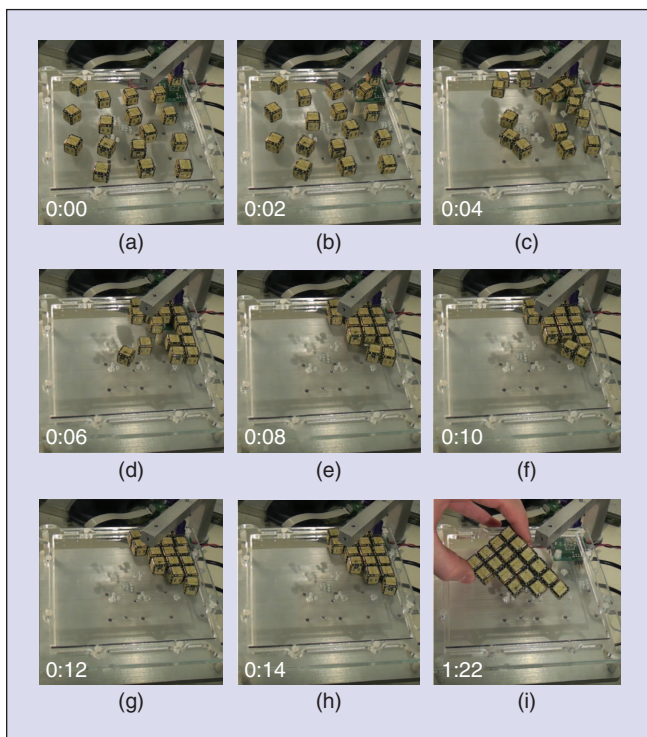
**Figure 14.** A vibration table is used to drive the self-assembly process. It consists of (a) a vibrating base, (b) a universal joint to control tilt, (c) the assembly surface, and (d) a variac to control the vibration frequency. (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT.)

This was the only trial of the 15 in which the vibration table was unable to align all of the modules. The average time taken to self-assemble the 17 modules was 1 min, 47 s. The self-assembly process worked most efficiently when the table vibration was swept up and down several times through varying frequencies.

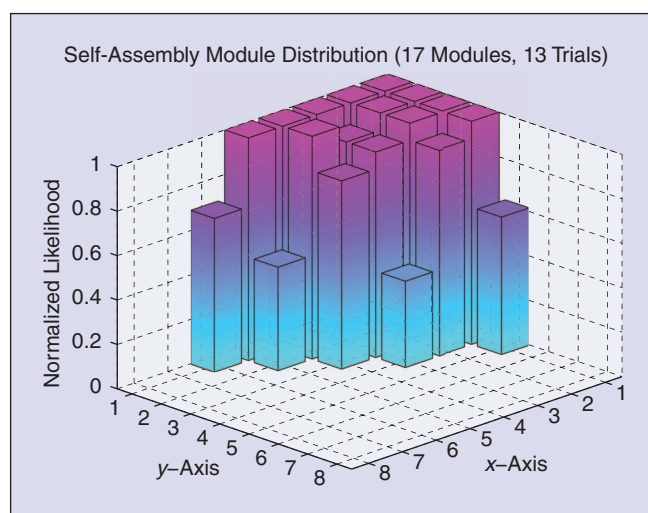
We also tested the system's ability to self-disassemble. We performed a set of 25 experiments in which we hand-assembled a  $3 \times 5$  block of modules. Our goal was to then self-disassemble the initial block to a humanoid robot as shown in Figures 9 and 17. The initial  $3 \times 5$  block contains 22 bonds between neighboring modules and the completed humanoid structure contains nine. Therefore, 13 bonds must be broken to form

the completed structure. To provide some measure of the self-disassembly algorithm's success, we kept track of how many of these bonds were correctly severed in each trial. There were only errors in four of the 25 trials, and each of these errors only affected a single module. In two of the four cases, a single module did not detach. In the third case, a module that was supposed to be in the final structure was detached. In the fourth case, a module that was supposed to be in the final structure detached, and it also remained bonded to a single neighbor that was supposed to detach. This amounts to a total of five bonding errors. Given there were a total of 330 ( $15 \times 22$ ) bonds that need to be maintained or broken over the course of all 25 experiments, the error rate is only 1.5%.

The self-assembly and self-disassembly experiments show that the system is capable starting with a random scattering of modules, forming an initial close-packed block of material,



**Figure 15.** A collection of 16 randomly distributed Smart Pebble modules (each a 12-mm cube), and one fixed root module (back right of each video frame), self-assemble when placed on an inclined vibration table. (Picture courtesy of Daniela Rus, Distributed Robotics Laboratory at MIT.)



**Figure 16.** When 17 modules are placed on a vibration table included so that the (0,0) location is the table's low point, the modules self-assemble into a close-packed lattice. The likelihood that a particular position in the lattice is filled is shown in this plot.



and then using self-disassembly to remove the extra modules to form a specific shape.

## Summary and Future Outlook

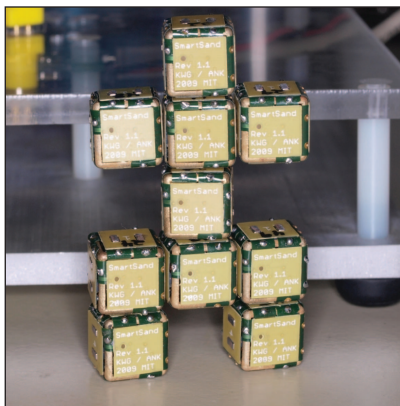
We have presented a detailed retrospective on modular robots and discussed connections between modular robots and programmable matter. This field has seen a great deal of creativity and innovation at the level of designing physical systems capable of matching shape to function and algorithms that achieve this capability. The success of these projects rests on the convergence of innovation in hardware design and materials for creating the basic building blocks, information distribution for programming the interaction between

the blocks, and control. Most current systems have dimensions on the order of centimeters, yet pack computation, communication, sensing, and power transfer capabilities into their form factors. Additionally, these modules operate using distributed algorithms that use a module's ability to observe its current neighborhood and local rules to decide what to do next.

Within this broad space, our own work spans the development of several modular self-reconfiguring robot systems. Building on this experience, we identified self-disassembly as a way of creating shapes out of smart components using a subtractive process. The key idea is to create a bag of smart components that can program their connections in an autonomous way to organize different shapes. This simplifies the mechanics of shape creation by eliminating the need for actively moving parts. The required actuation mechanism (disconnection) is generally easier, faster, and more robust than actively seeking and making connections. The trade-off is two-fold. First, self-disassembling systems must start from a preassembled structure of modules. Second, external forces must be employed to remove unwanted material from the system. Often, these forces can be found in the surrounding environment. For our first system prototypes, we used gravity to pull unnecessary modules away from the final structure.

A key innovation that enabled the miniaturization of the basic module for self-disassembly from a 4.5-cm cube to the 1-cm scale module has been the development of a small programmable connector capable of 1) holding state without power; 2) switching states using very short pulses; 3) encoding, transmitting, and decoding messages to neighbors in the structure; and 4) transmitting power. The functionality of this robot system is driven by two important capabilities: a) making shapes autonomously by disassembly and b) reassembling autonomously a building block. In this article, we summarized our solution for a) and discuss in detail our solution for b). We are currently working on completing a 50-module platform and on using this platform to evaluate the disassembly and re-assembly algorithms. In the future, we plan to extend resulting robot system with mobility, so that the objects formed by this method can function as mobile robots.

Our long-term hope is to create a self-disassembling system that can function analogous to a bag of Smart Sand that will be



**Figure 17.** A initial  $3 \times 5$  block of modules was used to form this 60-mm-tall humanoid through the self-disassembly process.

light and compact and that configures itself into a desired form at fine granularity. Our current system functions as a Smart Pebbles system. There is a suite of interesting challenges that have to be overcome to reduce the size of this system further from 1-cm scale to 1-mm scale and realize the dream of Smart Sand. New technology will have to be developed to package computation, sensing, actuation, communication, and power in a 1-mm scale module. New fabrication technology will have to be developed to fabricate such models rapidly and in cost-effective ways. New supporting algorithms that are scalable and matched to the properties of the hardware would have to be

put in place. Ultimately, these advances will lead to the creation of desktop-scale 3-D fabrication technology of electrically and mechanically active recyclable parts for everyday users.

## Acknowledgments

This work is supported by the Defense Advanced Research Projects Agency (DARPA) Programmable Matter and Chembots programs (Dr. Mitch Zakin, PM) and the U.S. Army Research Office under grant numbers W911NF-08-1-0228 and W911NF-08-C-0060, National Science Foundation (NSF) Emerging Frontiers in Research and Innovation (EFRI), Intel, and the National Defense Science and Engineering Graduate (NDSEG) fellowship program. We also thank Prof. Rob Wood and Dr. Ara Knaian.

## Keywords

Modular robots, self-assembling robots, self-disassembling robots, metamorphic robots.

## References

- [1] A. Tison and T. Taylor, *Barbapapa*, Les Livres du Dragon D'Or, 2003.
- [2] I. S. Behr, R. H. Wolfe, and R. D. Moore, "Star trek: Deep space nine," 1994, episode 47 & 48: The Search.
- [3] J. Cameron and W. Wisher, Jr., *Terminator 2: Judgment Day*, 1991.
- [4] B. Mando, B. Budiansky, J. Shooter, M. Higgins, R. Macchio, B. Sienkiewicz, F. Springer, K. DeMulder, N. Yomtov, and R. Parker, *The Transformers*. New York: Marvel Comics, 1984.
- [5] T. Fukuda and S. Nakagawa, "Dynamically reconfigurable robotic system," in *Proc. IEEE Int. Conf. Robotics and Automation*, Apr. 1988, pp. 1581–1586.
- [6] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss, "Self organizing robots based on cell structures—cebot," in *Proc. IEEE Int. Workshop on Intelligent Robots*, Oct. 1988, pp. 145–150.
- [7] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems: Challenges and opportunities for the future," *IEEE Robot. Automat. Mag.*, vol. 14, no. 1, pp. 43–52, Mar. 2007.
- [8] M. Yim, "A reconfigurable modular robot with many modes of locomotion," in *Proc. JSME Int. Conf. Advanced Mechatronics*, 1993, pp. 283–288.
- [9] M. Yim, "New locomotion gaits," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1994, pp. 2508–2514.
- [10] A. Castano and P. Will, "Mechanical design of a module for reconfigurable robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2000, pp. 2203–2209.

- [11] W.-M. Shen and P. Will, "Docking in self-reconfigurable robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2001, pp. 1049–1054.
- [12] A. Castano, A. Behar, and P. Will, "The conro modules for reconfigurable robots," *IEEE Trans. Mechatron.*, vol. 7, no. 4, pp. 403–409, Dec. 2002.
- [13] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *IEEE/ASME Trans. Mechatron.*, vol. 7, no. 4, pp. 431–441, Dec. 2002.
- [14] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Automatic locomotion design and experiments for a modular robotic system," *IEEE/ASME Trans. Mechatron.*, vol. 10, no. 3, pp. 314–325, June 2005.
- [15] H. Kurokawa, K. Tomita, A. Kamimura, E. Yoshida, S. Kokaji, and S. Murata, "Distributed self-reconfiguration control of modular robot m-tran," in *Proc. IEEE Int. Conf. Mechatronics and Automation*, July 2005, pp. 254–259.
- [16] S. Murata, K. Kakomura, and H. Kurokawa, "Docking experiments of a modular robot by visual feedback," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2006, pp. 625–630.
- [17] D. Marbach and A. J. Ijspeert, "Online optimization of modular robot locomotion," in *Proc. IEEE Int. Conf. Mechatronics and Automation*, July 2005, pp. 248–253.
- [18] B. Salemi, M. Moll, and W.-M. Shen, "Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2006, pp. 3636–3641.
- [19] A. Kawakami, A. Torii, K. Motomura, and S. Hirose, "Smc rover: Planetary rover with transformable wheels," in *Proc. 41st Annu. Society of Instrument and Control Engineers Conf.*, Aug. 2002, pp. 157–162.
- [20] M. Yim, D. G. Duff, and K. D. Roufas, "Polybot: A modular reconfigurable robot," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Apr. 2000, pp. 514–520.
- [21] M. Yim, Y. Zhang, K. Roufas, D. Duff, and C. Eldershaw, "Connecting and disconnecting for self-reconfiguration with polybot," *IEEE/ASME Trans. Mechatron. (Special issue on Information Technology in Mechatronics)*, pp. 442–451, 2003.
- [22] M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, and C. Taylor, "Towards robotic self-reassembly after explosion," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Nov. 2007, pp. 2767–2772.
- [23] J. Sastra, S. Chitta, and M. Yim, "Dynamic rolling for a modular loop robot," *Int. J. Robot. Res.*, vol. 28, no. 6, pp. 758–773, 2009.
- [24] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson, "Evolved and designed self-reproducing modular robotics," *IEEE Trans. Robot.*, vol. 23, no. 2, pp. 308–319, Apr. 2007.
- [25] G. S. Chirikjian, "Kinematics of a metamorphic robotic system," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 1994, pp. 449–455.
- [26] G. Chirikjian, A. Pamecha, and I. Ebert-Uphoff, "Evaluating efficiency of self-reconfiguration in a class of modular robots," *J. Robot. Syst.*, vol. 13, no. 5, pp. 317–388, 1996.
- [27] A. Pamecha, I. Ebert-Uphoff, and G. S. Chirikjian, "Useful metrics for modular robot motion planning," *IEEE Trans. Robot. Automat.*, vol. 13, no. 4, pp. 531–545, 1997.
- [28] J. E. Walter, E. M. Tsai, and N. M. Amato, "Algorithms for fast concurrent reconfiguration of hexagonal metamorphic robots," *IEEE Trans. Robot.*, vol. 21, no. 4, pp. 621–631, Aug. 2005.
- [29] S. Murata, H. Kurokawa, and S. Kokaji, "Self-assembling machine," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1994, pp. 441–448.
- [30] E. Yoshida, S. Murata, K. Tomita, H. Kurokawa, and S. Kokaji, "Distributed formation control for a modular mechanical system," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, 1997, pp. 1090–1097.
- [31] H. Kurokawa, S. Murata, E. Yoshida, K. Tomita, and S. Kokaji, "A 3-d self-reconfigurable structure and experiments," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 1998, pp. 860–865.
- [32] E. Yoshida, S. Murata, S. Kokaji, A. Kamimura, K. Tomita, and H. Kurokawa, "Get back in shape! A hardware prototype self-reconfigurable modular microrobot that uses shape memory alloy," *IEEE Robot. Automat. Mag.*, vol. 9, no. 4, pp. 54–60, 2002.
- [33] E. Yoshida, S. Kokaji, S. Murata, K. Tomita, and H. Kurokawa, "Micro self-reconfigurable robot using shape memory alloy," *J. Robot. Mechatron.*, vol. 13, no. 2, pp. 212–219, 2001.
- [34] K. Kotay, D. Rus, M. Vona, and C. McGray, "The self-reconfiguring robotic molecule," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1998, pp. 424–431.
- [35] K. Kotay and D. Rus, "Motion synthesis for the self-reconfiguring robotic molecule," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Oct. 1998, pp. 843–851.
- [36] K. Kotay and D. Rus, "Algorithms for self-reconfiguring molecule motion planning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2000, pp. 2181–2193.
- [37] D. Rus and M. Vona, "A basis for self-reconfiguring robots using crystal modules," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2000, pp. 2194–2202.
- [38] D. Rus and M. Vona, "Crystalline robots: Self-reconfiguration with compressible unit modules," *Int. J. Robot. Res.*, vol. 22, no. 9, pp. 699–715, 2003.
- [39] J. W. Suh, S. B. Homans, and M. Yim, "Telecubes: Mechanical design of a module for self-reconfigurable robotics," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2002, pp. 4095–4101.
- [40] C.-J. Chiang and G. S. Chirikjian, "Modular robot motion planning using similarity metrics," *Auton. Robots*, vol. 10, no. 1, pp. 91–106, 2001.
- [41] M. Koseki, K. Minami, and N. Inou, "Cellular robots forming a mechanical structure (evaluation of structural formation and hardware design of "chobie ii")," in *Proc. 7th Int. Symp. Distributed Autonomous Robotic Systems (DARSO4)*, June 2004, pp. 131–140.
- [42] B. K. An, "Em-cube: Cube-shaped, self-reconfigurable robots sliding on structure surfaces," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2008, pp. 3149–3155.
- [43] C. Ünsal and P. K. Khosla, "Mechatronic design of a modular self-reconfiguring robotic system," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Apr. 2000, pp. 1742–1747.
- [44] K. C. Prevas, C. Ünsal, M. Önder Efe, and P. K. Khosla, "A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2002, pp. 787–792.
- [45] K. Hosokawa, T. Tsujimori, T. Fujii, H. Kaetsu, H. Asama, Y. Kuroda, and I. Endo, "Self-organizing collective robots with morphogenesis in a vertical plane," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 1998, pp. 2858–2863.
- [46] E. H. Østergaard and H. H. Lund, "Evolving control for modular robotic units," in *Proc. IEEE Int. Symp. Computational Intelligence in Robotics and Automation*, July 2003, pp. 886–892.
- [47] M. W. Jørgensen, E. H. Østergaard, and H. H. Lund, "Modular atron: Modules for a self-reconfigurable robot," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sept. 2004, pp. 2068–2073.
- [48] D. J. Christensen and K. Stoy, "Select a meta-module to shape-change the atron self-reconfigurable robot," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2006, pp. 2532–2538.
- [49] D. Brandt and D. J. Christensen, "A new meta-module for controlling large sheets of atron modules," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Oct. 2007, pp. 2375–2380.
- [50] K. Gilpin, K. Kotay, and D. Rus, "Miche: Modular shape formation by self-disassembly," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Apr. 2007, pp. 2241–2247.
- [51] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu, "Miche: Modular shape formation by self-disassembly," *Int. J. Robot. Res.*, vol. 27, no. 1, pp. 345–372, 2008.
- [52] R. Oung, F. Bourgault, M. Donovan, and R. D'Andrea, "The distributed flight array," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2010, pp. 601–607.
- [53] R. Fitch and Z. Butler, "Scalable locomotion for large self-reconfiguring robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, Apr. 2007, pp. 2248–2253.
- [54] R. Fitch and Z. Butler, "Million module march: Scalable locomotion for large self-reconfiguring robots," *Int. J. Robot. Res.*, vol. 27, no. 3–4, pp. 331–343, Mar./Apr. 2008.
- [55] G. J. Hamlin and A. C. Sanderson, "Tetrobot: A modular system for hyper-redundant parallel robotics," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 1995, pp. 154–159.

- [56] A. Lyder, R. F. M. Garcia, and K. Stoy, "Mechanical design of odin, an extendable heterogeneous deformable modular robot," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Sept. 2008, pp. 883–888.
- [57] A. Lyder, H. G. Peterson, and K. Stoy, "Representation and shape estimation of odin, a parallel under-actuated modular robot," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2009, pp. 5275–5280.
- [58] C.-H. Yu, K. Haller, D. Ingber, and R. Nagpal, "Morpho: A self-deformable modular robot inspired by cellular structure," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Sept. 2008, pp. 3571–3578.
- [59] J. John Amend and H. Lipson, "Shape-shifting materials for programmable structures," in *Proc. Int. Conf. Ubiquitous Computing: Workshop on Architectural Robotics*, Sept. 2009.
- [60] B. Donald, C. G. Levey, C. D. McGray, I. Paprotny, and D. Rus, "An unthered, electrostatic, globally controllable mems micro-robot," *J. Microelectromech. Syst.*, vol. 15, no. 1, pp. 1–15, Feb. 2006.
- [61] B. R. Donald, C. G. Levey, and I. Paprotny, "Planar microassembly by parallel actuator of mems microrobots," *J. Microelectromech. Syst.*, vol. 17, no. 4, pp. 789–808, Aug. 2008.
- [62] M. Shimizu, A. Ishiguro, and T. Kawakatsu, "A modular robot that exploits a spontaneous connectivity control mechanism," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Aug. 2005, pp. 1899–1904.
- [63] M. Shimizu, T. Mori, and A. Ishiguro, "A development of a modular robot that enables adaptive reconfiguration," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2006, pp. 174–179.
- [64] S. Goldstein, J. Campbell, and T. Mowry, "Programmable matter," *IEEE Comput.*, vol. 38, no. 6, pp. 99–101, 2005.
- [65] S. C. Goldstein and T. C. Mowry, "Claytronics: An instance of programmable matter," in *Proc. Wild and Crazy Ideas Session of ASPLOS*, Boston, MA, Oct. 2004.
- [66] M. E. Karagozler, S. C. Goldstein, and J. R. Reid, "Stress-driven mems assembly + electrostatic forces = 1mm diameter robot," in *Proc. IEEE Conf. Intelligent Robots and Systems (IROS)*, Oct. 2009, pp. 2763–2769.
- [67] G. Whitesides and B. Grzybowski, "Self-assembly at all scales," *Science*, vol. 295, pp. 2418–2421, Mar. 2002.
- [68] G. M. Whitesides and M. Boncheva, "Beyond molecules: Self-assembly of mesoscale and macroscopic components," *Proc. Natl. Acad. Sci.*, vol. 99, no. 8, pp. 4769–4774, Apr. 2002.
- [69] D. H. Garcias, J. Tien, T. L. Breen, C. Hsu, and G. M. Whitesides, "Forming electrical networks in three dimensions by self-assembly," *Science*, vol. 289, no. 5482, pp. 1170–1172, Aug. 2000.
- [70] S. Miyashita, M. Kessler, and M. Lungarella, "How morphology affects self-assembly in a stochastic modular robot," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2008, pp. 3533–3538.
- [71] K. Hosokawa, I. Shimoyama, and H. Miura, "Dynamics of self-assembling systems: Analogy with chemical kinematics," *Artif. Life*, vol. 1, no. 4, pp. 413–427, 1994.
- [72] M. Shimizu and K. Suzuki, "A self-repairing structure for modules and its control by vibrating actuation mechanisms," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2009, pp. 4281–4286.
- [73] P. W. K. Rothmund and E. Winfree, "The program-size complexity of self-assembled squares," in *Proc. 32rd Annu. ACM Symp. Theory of Computing*, 2000, pp. 459–468.
- [74] L. Adleman, Q. Cheng, A. Goel, and M.-D. Huang, "Running time and program size for self-assembled squares," in *Proc. 33rd Annu. ACM Symp. Theory of Computing*, 2001, pp. 740–748.
- [75] G. Aggarwal, M. H. Goldwasser, M.-Y. Kao, and R. T. Schweller, "Complexities for generalized models of self-assembly," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2004, pp. 880–889.
- [76] J. Bishop, S. Burden, E. Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen, "Programmable parts: A demonstration of the grammatical approach to self-organization," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Aug. 2005, pp. 3684–3691.
- [77] N. Napp, S. Burden, and E. Klavins, "The statistical dynamics of programmed self-assembly," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2006, pp. 1469–1476.
- [78] S. Griffith, D. Goldwater, and J. M. Jacobson, "Robotics: Self-replication from random parts," *Nature*, vol. 437, p. 636, Sept. 2005.
- [79] C. Jones and M. J. Matarić, "From local to global behavior in intelligent self-assembly," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2003, pp. 721–726.
- [80] J. Kelly and H. Zhang, "Combinatorial optimization of sensing for rule-based planar distributed assembly," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, 2006, pp. 3728–3734.
- [81] J. Werfel, "Anthills built to order: Automating construction with artificial swarms," *Ph.D. dissertation*, MIT 2006.
- [82] P. White, K. Kopanski, and H. Lipson, "Stochastic self-reconfigurable cellular robotics," in *Proc. IEEE Conf. Robotics and Automation*, Apr. 2004, pp. 2888–2893.
- [83] P. White, V. Zykov, J. Bongard, and H. Lipson, "Three dimensional stochastic reconfiguration of modular robots," in *Proc. Robotics Science and Systems*, June 2005, pp. 8–10.
- [84] M. Tolley, J. Hiller, and H. Lipson, "Evolutionary design and assembly planning for stochastic modular robots," in *Proc. IEEE Conf. Intelligent Robots and Systems (IROS)*, Oct. 2009, pp. 73–78.
- [85] K. Gilpin, A. Knaian, and D. Rus, "Robot pebbles: One centimeter robotic modules for programmable matter through self-disassembly," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2010, pp. 2485–2492.

**Kyle Gilpin** received his B.S. and M.Eng. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT). He is currently a Ph.D. student in the Distributed Robotics Laboratory at MIT. He works to improve communication and control in large distributed robotic systems. His past projects include developing ultrawideband radios, real-time image processing systems, reconfigurable sensor nodes, chain robots with compliant actuators, and a collection of 1.7-in cubes capable of shape formation through self-disassembly. Before beginning his Ph.D., he spent two years working as a senior electrical engineer at Proteus Biomedical, developing ultralow-power hardware and software for several implantable personalized medicine products. He is the recipient of both NSF and NDSEG fellowships.

**Daniela Rus** received her Ph.D. degree in computer science from Cornell University. She is a professor of electrical engineering and computer science, where she is associate director of MIT's Computer Science and Artificial Intelligence Lab (CSAIL) and codirects the MIT Center for Robotics at CSAIL. Her research interests include distributed robotics and mobile computing, and her application focus includes transportation, security, environmental modeling and monitoring, underwater exploration, and agriculture. She is the recipient of the NSF Career Award and an Alfred P. Sloan Foundation Fellow. She is a Fellow of the IEEE, a class of 2002 MacArthur Fellow, and a fellow of Association for the Advancement of Artificial Intelligence. Before receiving her appointment at MIT, she was a professor in the Computer Science Department at Dartmouth, where she founded and directed two laboratories in robotics and mobile computing.

**Address for Correspondence:** Kyle Gilpin, Computer Science and Artificial Intelligence Lab, MIT, 32 Vassar St., Cambridge, MA 02139, USA. E-mail: kwgilpin@csail.mit.edu.