A Survey of Motion Planning for Self-Reconfigurable Robots

Sam Slee

December 8, 2005

Abstract

Self-Reconfigurable robots are complex distributed systems composed of multiple modular robotic units. These units can rearrange to form different structures that fit the specific tasks that face the robot. The unique and complex nature of self-reconfigurable robots has lead to a challenging subset of motion-planning problems. In this paper several types of reconfigurable robots are described and the current algorithmic designs for those robots are surveyed.

1 Introduction

The concept of a collection of simple structures combining to form something much more complex and versatile is utilized throughout nature. Modular self-reconfigurable robots implement this concept by having a collection of simple robotic modules that can attach and detach from each other to form structures that best meet the challenges of the robot's current environment. Over the past decade the intrigue of such a versatile design has caused the field of self-reconfiguring robotics to grow in popularity.

A number of research groups have now developed competing physical implementations and algorithmic control designs. The common scenario where such robots are cited as being useful is in search-and-rescue missions. A versatile self-configuring robot would seem to be best-suited to traversing the unknown environment of collapsed buildings or other possible rescue environments. Some researchers also see wide-ranging use for these robots in sea or space exploration, or even in more common every-day tasks.

However, to fulfill the potential for truly versatile robotics, the complicated challenge of intelligently managing a large collection of independent modules must be overcome. Control algorithms for self-reconfigurable robots require a unique perspective on motion planning algorithms. While basic robot motion planning focuses only on finding collision-free paths through a robot's environ-

ment, now a multitude of independent units must be handled. The motion planning problems encountered fall into 3 main categories:

- **Basic path planning:** The common task of computing collision-free paths for a robot's movement through its environment.
- **Reconfiguration:** Algorithms that control how a collection of independent modules can rearrange to form different larger structures.
- **Locomotion:** How the collection of modules can generate movement through their environment.

In addition to handling each of these 3 main types of problems, complications can also arise as the different problem types interact. Locomotion is obviously restricted by the robotic structures that are possible by self-reconfiguration and the abilities of those structures. Self-reconfiguration is potentially limited by the space of the environment in which the robot modules are placed. Altogether, self-reconfigurable robots present a distinct and challenging set of motion planning problems.

In the following sections a variety of different forms of self-reconfigurable robots and the problems facing them will be detailed. Section 2 notes some of the categories of reconfigurable robots and the characteristics that set them apart. Section 3 more carefully defines the main problems of reconfiguration and locomotion for these systems. Sections 4, 5, and 6 go into depth about some of the notable reconfiguration planning strategies that have been employed to date. Section 7 describes locomotion planning algorithms, encompassing solutions for each of the major categories of robotic systems covered. Section 8 then addresses some of the additional challenges facing this unusual robotic field, covering those that do not strictly fit into the categories of locomotion or reconfiguration motion planning. Finally, Section 9 concludes with a summary of what has been covered and some final thoughts about the field of self-reconfigurable robots.

2 Reconfiguration Classes

For reconfiguration planning, a "configuration" refers to a particular arrangement of connectivity between the independent modules [1]. The "reconfiguration space" is then the set of such configurations that a given collection of modules may form. Self-reconfigurable robots may transition between such configurations by a series of atomic movements which Yim et. al. denote as *rmoves* [1]. Exactly what comprises such an atomic move depends on the type of the robot implementation.

In addition to the unique motion planning challenges presented by self-reconfigurable robots the problem is further complicated by the varying hardware implementations used for the independent modules. Depending on the

hardware design used, planning algorithms largely fall into 3 different reconfiguration classes: mobile reconfiguration, substrate reconfiguration, and closed-chain reconfiguration. Only self-reconfigurable robot designs are covered, as systems requiring outside intervention for reconfiguration lose many of the unique abilities and design challenges that accompany independent systems.

Substrate Reconfiguration Substrate reconfiguration designs seem to be the most popular choice in the field. Robots in this category are composed of modules that can only attach at discrete locations on a lattice formed by other modules in the collection. Exact paths are then given for a module to transition from one location on the lattice to another. Lattices with modules "in transition" are not labeled as configurations [1]. Thus, an *rmove* for this category involves a module detaching from the lattice, traveling along the surface of the lattice, then reattaching at a new location. The methods by which this is achieved varies with different implementations. Designs range from hexagonal 2D modules that seem to roll around the lattice [3], to rhombic dodecahedron 3D modules [2], to "molecule" designs with two joined atoms that walk along the lattice structure [4], to expanding cube designs that push or pull other cubes through the lattice [5,6].

Mobile Reconfiguration This type of hardware implementation consists of modules that can move independently through their environment. An *rmove* for this category would then involve a module detaching from the collection, moving through the environment to a new location outside the collection, then reattaching at that new location. These movements are in addition to any walking or climbing along the structure formed by the other modules as was done for substrate reconfiguration. This allows for a greater range of possible reconfiguration algorithms as modules are not restricted to always being fully connected. However this design does place a greater burden on the hardware implementation as each module must be self-reliant for mobility through the environment and for power. As a result some work has been done on this concept [13] but other designs have been more popular.

Closed-chain Reconfiguration This category of self-reconfigurable robots deviates from the lattice-like structures that were common to the previous two types. Here robot systems are composed of open or closed kinematic chains of modules. These chains then attach at common points to form complex chains or loops. When making a single chain, the visual image of the robot would be similar to a snake. The bending ability of individual modules can result in snake-like movements such as making sinusoidal waves for locomotion. One of the most notable implementations of this design [7] has had several demonstrations of locomotion.

An *rmove* for this class of self-reconfigurable robot would involve a single attach or detach operation as well as possible motion within any of the modules' degrees of freedom. This design category is unique because it places more

emphasis on the motion of attached modules swinging entire chains of other modules while the other categories largely focused on more local movements involving a couple of modules. This shift in design creates opportunities for different motion planning and reconfiguration algorithms, but also necessitates stricter hardware requirements. Individual modules must now be strong enough to perform motions while lifting the weight of chains of other modules.

Homogenous vs. Heterogenous Most of the systems in the categories above make the common assumption that the modules are identical, making for a homogenous system. Homogenous systems of modules have also been called *metamorphic robotic systems* [3]. Such systems have the obvious advantage that modules are interchangeable, greatly simplifying algorithm design. When computing a solution to a reconfiguration problem, it is not necessary to get specific modules to specific locations. Conditions are relaxed so that it is sufficient for any module to fill the specified locations.

There are, however, strong arguments for a heterogenous style of reconfigurable robot. The arguments given in [18] note that certain necessary sensing and processing parts for robots are likely to be too costly to include in every individual module. In addressing their concerns it is possible that all homogenous designs will at least have to make special allowances for expensive, special-purpose parts that sometimes may need to be included. However, the majority of hardware designs and algorithmic planning in the field is for homogenous designs or close approximations (very few module types or one uniform type and few non-uniform additional pieces). So, this survey focuses on homogenous systems.

3 Planning Problem Types

The Reconfiguration Planning Problem The general case reconfiguration problem is that of taking modules in one configuration and rearranging them into another configuration using atomic module movements within the restrictions of the physical module implementation.

For simulations or for the creation of generic reconfiguration algorithms real-world concerns are often abstracted away. When addressed, physical implementation considerations include gravity - will certain movements cause the robot to tip over, or to fall off a ledge - and collision avoidance. There is concern for collisions both with the environment and with other modules within the robot collection itself. Finally, when considering this problem there is also a question of what constitutes optimal behavior.

Bounds on the number of atomic movements are sought for most algorithms that are designed, but the proper context for such bounds is still unclear. Restrictions on what is possible differ depending on whether algorithms are cen-

tralized or distributed, and which category of modular design is being evaluated. Finally, physical concerns such as energy exerted by modules and time elapsed for all operations should be considered for system implementations. The algorithms described later in this paper will focus largely on abstractions without real-world concerns.

The Locomotion Planning Problem Generating motion of the collection of modules through its environment using atomic module movements within the restrictions of the physical implementation.

While certainly a necessary goal for any real-world robot, success for this question is still harder to define. While algorithmic solutions and physical implementations have been produced for this problem, the success of such designs in real-world settings is largely untested.

4 Substrate Reconfiguration Motion Planning

The substrate category of modular robots has seen the most work on reconfiguration planning. Initially, planning algorithms focused on centralized algorithms such as those seen in [12]. However, realizing the promise of very versatile robotic systems seems to require a large number of individual modules to comprise the reconfigurable system. Since distributed algorithms are better-suited to control such systems more recent work has focused on these algorithms.

Due to the complex nature of the problem, many heuristic algorithms and even some Artificial Intelligence techniques such as simulated annealing have been proposed. The difficulty is that the number of possible configurations grows exponentially with the number of atomic module units included in the robot. These configurations can be represented by a graph according to their connectivity arrangement. Correspondingly, optimal reconfiguration planning has a relation to the shortest path problem on a graph with exponential nodes.

Hexegonal metamorphic robots One group, Walter et. al. [9] has mostly concentrated on algorithmic designs for two dimensional hexagonal modules. In this system the modules have the ability to "roll" along the lattice structure formed by other modules. A module is aware of neighboring modules on any of 6 sides by touch sensors as well as being aware of its own orientation (which sides are facing which directions). Since the hexagonal structure of the modules forms a coordinate system, modules are also assumed to know their coordinate location in the arrangement.

The main reconfiguration problem addressed is to take an initial straight chain of hexagonal modules or cells and reconfigure those modules into a goal configuration that meets a stated admissibility condition. The researchers address this with a two stage algorithm. The first step is a centralized planning

stage to determine the feasibility of the reconfiguration task and — passing that test — the final target coordinates for all modules in the system. This is decided by creating a directed graph towards and through the target cells in the goal configuration area. Since modules need to traverse along backbone structures created by other modules, the algorithm focuses on first finding a "substrate path" that will first be filled and evenly divide the goal area. When completed modules will be able to traverse both sides of this substrate path, thus reducing the time to complete the reconfiguration. Using the directed graph construction all possible substrate paths are considered and weighted according to a heuristic function. The substrate path with lowest weight is deemed best and chosen. All coordinate positions in the target configuration are then assigned rankings such that locations along the substrate path will be filled first, then positions deepest into the target configuration (hardest to reach), etc. These locations are then carefully assigned to module cells in the initial straight line configuration.

The second phase of the algorithm involved all modules independently executing algorithms for reaching their final target coordinates. The algorithm specified by the other relies only on the coordinated target locations given to all module cells and on touch sensors for each of 6 sides for each module. It is shown that the distributed algorithm is complete in being able to fill the target configuration while avoiding collisions, deadlock states, or movements that would disconnect any part of the system.

The algorithm is also shown to be optimal for certain assumptions about the model and the module capabilities. Later papers [11] also extended this work to fitting reconfigurations surrounding obstacles. The authors emphasize that their distributed algorithm is able to perform using only touch sensors and without message passing between modules as almost all other solutions do. Their solution is also deterministic, which potentially has reliability advantages over other proposed probabilistic proposals. However, the distributed algorithm relies heavily on the centralized initial phase and modules are required to move in synchronous rounds. Simulated results are given to show the effectiveness of the algorithm, but no physical implementations are known.

The Molecule robot This substrate category design by Rus et. al. consists of modules called Molecules that each consist of two apple-sized atom units linked by rigid bond connection [4]. Each atom can rotate 180 degrees relative to the bond. The bond can rotate one atom 180 degrees in relation to the other atom. This combines to give four degrees of freedom for the entire Molecule module. An individual module would start moving by attaching one atom to another Molecule module contained within a lattice substrate structure. The bond of the moving Molecule module would then move the second atom through space so that atom could come to rest upon and connect with another lattice cell in the direction that the moving module is traveling. Once this connection is established the first atom can now detach and be moved through space by the bond to continue movement of the Molecule module through an alternating

sequence of detachment, turn, and reattachment.

The flexibility of the Molecule bond permits it to traverse not only level lattice structures, but also make convex or concave transitions onto surfaces on perpendicular axes. For movement of a given Molecule module along the exterior of a lattice surface a distributed implementation of a graph search algorithm is used to plan a path, then local control algorithms in the module execute it. This is not complete, however, as it does not fully account for the topology of the lattice surface nor the possibility that the environment containing the lattice does not have enough free space for another Molecule module to traverse past.

The reconfiguration algorithms devised specifically for the Molecule module are mostly just a combination of simple movements. The most basic involves a pair of modules "leap-frogging". The pair starts with one module in front of the other. The back module then climbs over the first to become the new front module. In this way translation is possible with just two modules, even in the absence of a larger structure. More advanced, global reconfiguration algorithms for the Molecule module are given in the form of generic algorithms for a physical abstraction that the Molecule module satisfies. These generic reconfiguration algorithms are discussed later in this paper.

5 Expanding Cube Style Robot Reconfiguration Planning

Although they still fall in the category of substrate reconfiguration designs, crystalline style robots are themselves a unique design. Rather than bendable modules that can roll or walk along the exterior of a lattice structure, crystalline modules are expandable cubes that can push or pull each other into place. For two dimensional solutions this means four expanding and contracting faces, for three dimensions six faces of a cube have this ability. Through this method squares or cubes of modules can combine at uniform coordinates to form a rigid crystal-like structure. A set range for contracting and expanding will then allow two neighboring modules to contract to half their normal size and fill a single grid space in the lattice. A single grid space hole is then formed for other modules to be pushed into. This concept of expanding cubes seems to have first been created by Rus et. al. [5]. Although it is possible to apply some generic reconfiguration algorithms to both molecule and crystalline style modular robots, these expanding cube modules provide a unique set of abilities and difficulties.

By having modular units that can expand and contract, it is now possible for movement of modules through the interior of a robot structure rather than only along the exterior surface. Recent algorithmic designs for reconfiguration focus on the fact that it is a homogenous system and the final shape of a goal configuration is important, not the exact location of specific modules. This in-

sight is available to all homogenous systems to reduce the daunting complexity of large-scale reconfiguration problems. Since expanding cube style modules move by being pushed or pulled rather than walking individually, this design is particularly well-suited to take advantage. However, one drawback of this design is that certain configurations of atomic modules are "rigid" [10]. Other configurations can neither transform into them, nor can they transform into others. A simple example is a straight line of cubic modules. In this case all of the modules in the system can only push or pull in a single axis direction. Thus it is impossible to move modules in other axis directions. Algorithmic designs by two different research groups have found ways around this restriction.

Melt-Grow Algorithm The Melt-Grow algorithm was one of the first proposed for reconfiguration of expanding-cube style modular robots. As described above, modules are still moved through the structure by the inch-worm method of scrunching two neighboring modules into a single grid space leaving a free space behind. A third module can then expand to push a fourth one into the open space. In this way modules can also be pushed one level outside of the current surface of a structure. This begins a process of advancing the entire surface one grid length further in that direction.

For reconfiguring to occur, out of place modules in the starting state must be moved into vacant spaces in the desired final state. Due to the physical limitations of modules and of the entire "crystal" structure, it is not enough to simply select any out of place module and begin moving it. Certain modules in certain configurations may have no immediate way of being moved, or may need to remain in place to maintain the stability of the entire structure. Thus, *mobile* modules, by criteria defined by the authors, are found to be moved and in this manner all out-of-place modules may begin their transition towards places in the final configuration.

In the global reconfiguration process, the melt-grow algorithm first "melts" the starting configuration into some reachable, well-known intermediate structure. Then, from this intermediate form the final configuration may be "grown". The utility of an intermediate structure is twofold. First, it is commonly easier to reconfigure from a more complicated arrangement to one that is simpler and well-known. This facilitates making a plan for moving from the start state to the intermediate state. By finding a similar plan from the final state to the intermediate state we can simply reverse the steps of that plan to complete the reconfiguration process. Secondly, the use of an intermediate state is useful in ensuring that the algorithm does not become stuck in some undesirable state during the reconfiguration process and have to backtrack to undo some of its moves. Such an undesirable state would likely include an absence of useful mobile modules.

One more issue addressed by the designers of this reconfiguration process is the unreachability of certain configurations. Structures such as the single line

of modules still cannot be dealt with. However, if meta-modules each composed of several atomic modules are used as the basic units this single axis restriction can be avoided. The authors describe a 4x4 meta-module of 8 atomic cube modules for the two dimensional case as being sufficient for complete reconfiguration abilities. For the three dimensional case a 4x4x4 meta-module is used. Through this design the troublesome unreachable configurations are avoided. It is also noted that arbitrary precision for representing structures is still possible as the size of atomic modules is decreased.

Despite the innovative design of the expanding cube modules and the melt-grow algorithm, there is still the drawback that it is a centralized algorithm for reconfiguration control. The promise of robotic versatility come from systems involving large numbers of atomic modules, and the performance of centralized algorithms degrades as the number of modules it has to command increases. A distributed algorithm breakthrough came with the PacMan algorithm.

The PacMan Algorithm The PacMan algorithm appears to be the first distributed reconfiguration algorithm for expanding cube style modular robots. The same basic principles of inchworm propagation of cube modules is utilized. An initial method - started by an outside source - is still used for giving the final target configuration to the robot. This may be done by giving the target configuration to a single module, as well as that module's coordinate location in the starting and final configurations. The structure of the final configuration may then be propagated throughout the modular structure. This procedure is still reasonable as it runs in a distributed fashion. Also, until sensor and processing technology is such that the robot can decide on its own what final shape to take, descriptions for final configurations can only come from an outside source. After this target configuration information dissemination step, the remainder of the algorithm is run in a distributed fashion by the individual modules in two phases.

The first phase requires a planning step of constructing a path beginning with out of place modules in the initial configuration and going to the locations where they are needed in the final configuration. This comes in the form of *target* atoms and *spare* atoms. Target atoms are those in the initial configuration that lie next to unfilled locations in the final configuration. Spare atoms are those in the initial configuration that are no longer needed in the final configuration. Modules can recognize if they are one of these two types when they are given the plan for the target configuration in the initial step.

Next, target atoms begin by searching for spare atoms that could fill their neighboring gap. This is done in a depth-first manner as target atoms pass specific "plan pellets" to their neighbors in a depth-first manner. In this way a distributed search is begun to match up spare atoms with the places they should occupy in the final configuration. When such a plan pellet reaches a spare atom, it sends a signal back towards the start of the path turning the plan pellets into "path pellets" along the way. These pellets then mark the path the spare atom

must travel to reach its place in the final configuration.

Finally, modules at the end of confirmed paths begin "eating" the pellets as they traverse those paths. Again, specific modules do not travel the entire distance of a path. They are merely inched one step along the path, then swap identities with the next module. This module will then inch one more step along the path, repeating the process. This continues until a modules completes the last step of the path, being pushed into the proper place in the target configuration.

The dissemination of the target configuration, the search for spare atoms, and the movement of those atoms toward target locations in the final configuration all require only local interaction between modules. Thus, local checks are sufficient for avoiding collisions and maintaining stability of the overall crystal robot structure. Rus et. al. have produced a large number of simulations showing their crystal design in action as well as physically implementing working expanding cubes. The extent of physical experimentation with those cubes and these algorithms is not known.

Telecube modules A second research group of Yim et. al. has also worked on designs for expanding cube style modular robots [10]. In addition to algorithms in the style of Melt-Grow and PacMan, contributions are made in other ways. The authors focus on design issues for three dimensional expanding cube modules and design them as Telecube modules. Similar to Rus et. al. an approach is taken to use meta-modules composed of several atomic Telecube modules as the basic building block. However, they are able to reduce the needed size to 8 basic modules for a 2x2x2 construction rather than the 4x4x4 construction. This is gained by a creative use of "free blocks".

Free blocks are simply Telecube modules that are completely contracted and rest between partially contracted Telecube modules in the normal lattice structure. However, these added modules can be used to facilitate the reduction in meta-module size as well as uses in affecting the weight distribution, density, and structural strength of sections of the robot system as required. In addition, the authors are able to show that the amount of space needed for reconfiguration is bounded by the space needed by the initial configuration and the final configuration between which the transition occurs.

A worst-case bound of $O(N^2)$ meta-module moves is also proven for optimal reconfigurations between configurations involving N meta-modules. Yet, this bound does ignore the possibilities of concurrent movement of meta-modules. As robotic systems advance to large numbers of modules — which would in turn comprise large numbers of meta-modules — the utilization of concurrent movements would seem to become of critical importance. Like the concurrent PacMan algorithm described for the crystal robot, Scent-based systems have been developed for Telecube modules.

6 Closed-Chain Reconfiguration Planning

The past two sections have detailed implementations and algorithm designs for the substrate or lattice category of reconfigurable robots. Modules in these designs make discrete movements along a substrate formed by other modules, thus simplifying movement and analysis. In contrast, closed-chain reconfigurable robots are not restricted to lattice cell positions but are instead movable chains and loops of modules. These are capable of turning extended lines of modules (similar to the movement of a snake, though with far fewer degrees of freedom). Completing connections between modules is no longer the simple task assumed for lattice-bound modules. Matching up swinging loops and chains of modules in fact presents an array of engineering challenges. This greatly complicates atomic movements for the modules, but opens up new algorithmic possibilities for reconfiguration planning.

Generic Closed-Chain Reconfiguration Planning Yim et. al. are one of the few groups to explicitly tackle the problem of reconfiguration planning for closed-chain modular robots [1]. They present a divide-and-conquer strategy as a general framework for dealing with closed-chain robots. This strategy is based on defining the uniqueness of configurations by the module connectivity it contains, which can be represented as a "modular graph". Here vertices correspond to individual modules and edges to connections between those modules. Once in this form the total structure may be divided into a hierarchy of subgraphs based on sets of commonly found substructures.

Sets of substructures are chosen such that they are topologically distinct or non-homeomorphic, they appear often in a large number of configurations, and reconfiguring between structures in the set are simple [1]. If proper choices for subsets are made then many reconfiguration operations may be decomposed into a well-known set of pre-computed (and pre-stored) sub-reconfigurations. When considering a given robot configuration, the system is partitioned into instances of the known substructures. Once these have been identified the system can be regarded as a connected collection of substructures rather than individual modules. Given these substructures their connection arrangement may then be represented by the hierarchical structure of a tree. Though, special allowances must still be made for breaking or ignoring remaining cycles.

Given these general decomposition techniques, the authors present two algorithms for closed-chain reconfiguration. The first uses a tactic mentioned earlier for the Melt-Grown crystalline algorithm: that of a reliable intermediate form. Given that we are dealing with chain-style modules, the first possible intermediate stage suggested is that of a single chain. Deconstructing any complex form into a simple chain of modules should be a relatively easy reconfiguration task. Since decomposition moves from the goal state to the intermediate state can be reversed, a straightforward algorithm for the entire reconfiguration process is developed. While a simple chain is the first choice, more intelligent intermedi-

ate choices would depend on the tasks commonly faced or expected to be faced by the robot.

A second algorithmic design presented by Yim et. al. is to match the initial configuration to the goal configuration — layer by layer, step by step. In the hierarchical decomposition characteristics of the number of levels, number of substructures per level, type of substructures per level, size of substructures, and connectivity arrangement of each substructure were identified. Thus it would be possible to morph the initial configuration to match, in order, each level of complexity of the goal configuration. Though an interesting and unique algorithm design, the authors do not elaborate on many details of implementation. It would seem that such an algorithm is unlikely to ever be as simple as the intermediate stage algorithm, yet might be a more efficient choice for some cases.

7 Reconfigurable Robot Locomotion Planning

For Self-Reconfigurable Robots, the task of locomotion through an environment may be seen as a subset of the general reconfiguration problem. Specifying certain target configurations in relation to the current state of the system may cause the robotic system to shift in desired directions. Repeating a series of such reconfigurations would then generate movement through the environment. While this is a correct and complete description for locomotion, treating it as a subset of reconfiguration planning ignores the obvious differences between the goals of the two problems. General reconfiguration is interested in only the starting and ending configurations of the system. Abstracting or even ignoring the specifics of how that is achieved can lead to simplified and more efficient algorithms. As long as the system is guaranteed to reach the final configuration without collisions between modules or with its environment, there is little concern for the intermediate configurations that the modules form.

Conversely successful locomotion planning focuses on the intermediate configurations used nearly as much as the final configuration itself. Good choices for intermediate configurations are at least responsible for maintaining traction in the robot's environment. In some designs, locomotion is entirely generated by a continuous series of small configuration changes rather than achieving a set of goal configurations. A key reason for this is that in most cases reconfiguration planning is merely concerned with the configuration of attachments and geometric shape formed by the system. Little if any regard is given for the exact location of the system of its environment. Of the designs described above, only the hexagonal metamorphic robots specified a coordinate system for locations much farther than the exterior surface of the current system [3]. Still, as with reconfiguration planning, locomotion planning is still largely dependent upon the type of the physical implementation for modules in the system.

Locomotion with Expanding Cube Modules The Crystal and Telecube modules [5,6] provided some of the most interesting and possibly most advanced distributed algorithms for reconfiguration planning. In contrast, the options for locomotion planning are much more limited. As currently implemented, expanding cube modules are useful only for extending or contracting in coordinate axis directions. There is not the same bending or rotating aspect as found with most other reconfigurable robot designs. Thus, locomotion planning is mostly restricted to a "water flow" style of movement. In this type of algorithm modules at the back of the configuration would flow over or possibly move virtually "through" the initial configuration to result in a module being placed one lattice unit further towards the desired direction. Another visualization of this technique might be as a train-like object that continually lays its own tracks ahead of it. Finally, in addition to the expanding cube modules the hexagonal metamorphic designs [3] would also seem to be restricted to this type of locomotion planning.

Still, this is not to say the method is without future promise. The conditions that require the water-flow technique are easily abstracted to create a general framework which has been referred to as the sliding cube model. In this model cubes are used to represent modules. It is assumed that modules have the ability to slide along the exterior of lattice surfaces formed by other modules, and be able to make concave and convex transitions to other axes of attached lattice surfaces. This allows for the designing of generic algorithms suitable for substrate style reconfigurable modules and beyond.

A major approach taken by Rus et. al. towards finding such generic algorithms is to apply the basic cellular automata paradigm to robotic systems. This work deals with the an abstract level of self-reconfiguration modules so as to produce algorithms that are architecture independent. This abstraction allows correctness of algorithms to be proven more easily as well as giving multiple implementation choices for comparing algorithms, as well as comparing different physical module architectures. More specifically, a cellular automata describe basic rule sets for each module in the self-reconfiguration system to implement. By having each module implement a rule set based solely on its local configuration, an efficient distributed algorithm is created. Locomotion over uneven terrain has been the initial focus of these algorithms. The cellular automata approach is well-suited to noting the terrain and acting in accordance to generate water-flow locomotion. Also, since locomotion places less restrictions on the exact shapes and configurations taken by the system as it moves, this problem is more natural to start with than the more restrictive general reconfiguration problem.

Locomotion with the Molecule module Since it fits in the substrate module category, the Molecule module is also capable of implementing the generic "sliding cube" algorithms as described above. These work for global reconfiguration algorithms as well as for locomotion algorithms. In addition, the bendable

bond and rotating attachment degrees of freedom permit other kinds of locomotion for the Module module and other similar substrate style designs. Rus et. al. have been able to show that locomotion is possible with only 2 Molecule modules in an uneven rolling gait. For stair-climbing, a minimum of 8 conjoined Molecules were required. These constructs largely relied on just a few modules continually leapfrogging ahead of each other.

For more global approach to reconfiguration and locomotion planning, an idea of scaffold planning is introduced. This idea is implemented for the Molecule module, but seems to have been first designed for hexagonal metamorphic robots by Yim et. al. Instead of completely filling the interior of a module lattice structure, this design instead leaves some of the space unfilled to create tunnels through the interior of the structure. This permits more surface area for modules to move across and allows the surface area to be scaled with the size and volume of the system as it grows. Again, this idea of interior tunnels may work for any of the substrate type of modules, but it particularly useful for hexagonal and Molecule modules that can only traverse along the surface of a lattice structure.

Locomotion with Closed-Chain Modules The PolyBot system is a heterogenous closed-chain style modular robot. Here a node module supplies power to actuated segment modules that permit movement. This permits chains and loops of modules to be moved about as described earlier in section 2. Because attachment of modules to new locations is no longer a trivial operation of moving with respect to a touching neighbor module, photo-diodes and light-emitting diodes are used to determine the position and orientation of modules.

To go along with their design, an impressive array of locomotion gaits are described. Control for these operations comes from precomputed gait tables, although a demonstration of dynamic downloading of such tables was given. Locomotion gaits ranged from a rolling loop, to a snake-like chain with sinusoidal movement, to a multi-legged spider or centipede construction. For real-world implementations and locomotion, at this time these close-chain designs seem to be the farthest along. Still, it is uncertain if they hold the same abilities to scale well to very large numbers of nodes to reach the full versatility potential of reconfigurable robots.

8 Additional Self-Reconfiguration Robot Challenges

The most basic challenge for self-reconfigurable robots is the designing of physical modules with the strength, versatility, and cost-effectiveness to implement large scale systems. Along with that, the basic algorithms for path planning, self-reconfiguration planning, and locomotion planning are all important motion

planning algorithms subsets that need intelligent solutions to control those systems. The unique nature of self-reconfiguration planning and motion planning for these modular systems, as well as the need for distributed management of very large systems of modules, makes for a particularly challenging combination. In addition, once these systems have the ability to effectively move and reconfigure, there is still the question of how those abilities should be utilized. Some techniques from the field of Artificial Intelligence have been proposed.

One possible question is what constitutes and optimal or near-optimal configuration for acting out a specific task. In [14] a proposal is given for finding "optimal" module assembly configurations given certain joint and link modules. Genetic algorithms were used for finding such assemblies. Another more specific task involves multiple robots coordinating together for tasks such as box pushing. In [15] a reinforcement learning strategy focuses on task allocation for this problem.

For self-reconfigurable robots, the question of being able to divide up and recombine systems of modules is only starting to be addressed. While a single system of modules provides a great degree of versatility, those abilities are amplified when the modules are able to divide into smaller sub-systems. In the commonly given potential uses such as search-and-rescue or exploration in unknown environments, the ability to subdivide (and later recombine) is of great use. Rus et. al. are one group that addresses this question in [16].

9 Conclusion

The growing field of self-reconfigurable robots is one with bright future promise as well as a unique set of difficult challenges. To achieve the potential of such systems, very large numbers of independent modules must be coordinated together. For the physical implementation, this presents problems with material design for strength, actuator motors for movement power, and connection designs that can operate in the vast array of unforeseen environments in which self-reconfigurable robots would be most useful. Reliable functionality for communication between modules and processors for computation are also needed. On top of those challenges, solutions must be cost-effective so that it is possible to build the multitudes of modules that are necessary to compose truly versatile system. In addition, many uses for these robots will require that the modules be very small so that the entire system can fit through small spaces or perform fine precision operations.

While the physical implementation challenges are numerous, those for designing the control algorithms are possibly more daunting. These modular systems seem to require distributed algorithmic solutions that are among the most difficult of settings. The number of involved individual units needs to be large to allow the system to be a useful as possible, yet processor and communi-

cation abilities will likely be very limited by the nature and size of the modules. Work that began with global, centralized algorithms has given way to hardware-specific distributed solutions, and recently the development of architecture independent frameworks for abstracting the design of algorithms away from individual physical implementations. This evolution seems to lead toward a more structured science for the field and more rigorous methods for designing and evaluating new algorithms.

The research progress covered in this survey attempted to capture a snapshot of the major designs and trends in the self-reconfigurable robotic field. Although the field is still not large, various research designs were not included here. In most cases, it is because that research focused mainly on a physical hardware solution. This survey put the focus on the state of motion planning algorithms for these robots, while still providing enough of the hardware background so that the setting and motivation for these algorithms might be understood. A look at several hardware designs for reconfigurable robots is given in [18] and particularly in [17].

References

[1] A. Casal and M. Yim, Self-Reconfiguration Planning for a Class of Modular Robots. In *Proc. of the Society of Photo-Optical Instrumentation Engineers*, Conf. on Sensor Fusion and Decentralized Control in Robotic Systems II, 1999.

[2] M. Yim, J. Lamping, E. Mao, J.G. Chase, Rombic Dodecahedron Shape for Self-Assembling Robots. Xerox PARC, SPL TechReport P9710777, 1997.

[3] J. Walter, B. Tsai, N. Amato. Choosing good paths for fast distributed reconfiguration of hexagonal metamorphic robots. *Proc. of the IEEE Intl. Conf. on Robotics and Automation,* pages 102-109, May 2002.

[4] K. Kotay, D. Rus, M. Vona, C. McGray, The self-reconfiguring robotic Molecule: design and control algorithms. *Proc. of the Workshop on the Algorithmic Foundations of Robotics,* Houston, USA.

[5] D. Rus, M. Vona, Crystalline Robots: Self-reconfiguration with Unit-compressible Modules. *Autonomus Robots,* vol. 10, no. 1, pages 107-124, 2001.

[6] S. Vassilvitskii, J. Suh, M. Yim, A Complete, Local and Parallel Reconfiguration Algorithm for Cube Style Modular Robots. *Proc. of the IEEE Int. Conf. on Robotics and Automation,* 2002.

[7] M. Yim, C. Eldershaw, Y. Zhang, D. Duff, Limbless Conforming Gaits with Modular Robots. *9th Intl. Symposium on Experimental Robotics*, June 18-21, 2004.

[8] G. Chirikjian, A. Pamecha. Bounds for self-reconfiguration of metamorphic robots. *In Proc. of IEEE Intl. Conf. on Robotics and Automation,* pages 1452-1457, 1996.

[9] J. Walter, E. Tsai, N. Amato, Algorithms for Fast Concurrent Reconfiguration of Hexagonal Metamorphic Robots, *IEEE Transactions on Robotics,* Vol. 21, No. 4, pages 621-631, August 2005.

[10] S. Vassilvitskii, J. Kubica, E. Rieffel, J. Suh, and M. Yim, "On the General Reconfiguration Problem for Expanding Cube Style Modular Robots," In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 20002.

[11] J. Walter, B. Tsai, N. Amato, Enveloping Obstacles with Hexagonal Metamorphic Robots. *Proc. of IEEE Intl. Conf. on Robotics and Automation,* May 2003, pages 741-748, Taipei, Taiwan.

[12] G. Chirikjian, Kinematics of a metamorphic robotic system. *Proc. of IEEE Intl. Conf. on Robotics and Automation,* pages 449-455, 1994.

[13] T. Fukuda and S. Nakagawa, Dynamically Reconfigurable Robotic Systems. *Proc. of IEEE Intl. Conf. on Robotics and Automation,* 1988.

[14] I. Chen and J. Burdick. Determining task optimal modular robot assembly configurations. *Proc. of IEEE Intl. Conf. on Robotics and Automation,* pages 132-7, 1995.

[15] L. Parker. Heterogeneous Multi-Robot Cooperation. PhD thesis, MIT, 1994. EECS Department.

[16] Z. Butler, S. Murata, and D. Rus. Distributed replication algorithms for self-reconfiguring modular robots. *Distributed Autonomous Robotic Systems 5,* Springer-Verlag, 2002.

[17] K. Kotay. Self-Reconfiguring Robots: Designs, Algorithms, and Applications. PhD thesis, Dartmouth College, 2003. CS Department.

[18] P. Jantapremjit and D. Austin. Design of a Modular Self-Reconfigurable Robot. *Australian Conf. on Robotics and Automation,* Sydney, Australia, 2001.