

Control Algorithms for Self-Reconfigurable Robots

Sam Slee

January 31, 2006

Abstract

The abilities of modern computers has provided immense power and flexibility for modeling things in the virtual realm. Yet the robotic systems through which computers can interact with the physical world are far more limited. Robots are often designed for limited, specific tasks with little chance of extending to different abilities. Self-reconfigurable robots are systems composed of many small modules that can autonomously reconfigure the shape and structure of their system. The versatility of such systems gives a way to extend the flexible power of computers from the virtual realm to the physical world. While promising, development of self-reconfigurable robotics is still in the early stages and large challenges remain in designing the physical modules and especially in designing the algorithms that will control them. This project will seek to develop control algorithms for self-reconfigurable robots that extend and improve upon those currently used in the field.

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 environment, 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 unique and challenging set of motion planning problems.

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. Furthermore, while these different categories of reconfigurable robots are important to mention, the work of this project will focus mainly on the substrate or lattice style designs described next.

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 *remove* 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 *remove* 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 *remove* 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.

3 Direction Of Research

Self-reconfigurable robotics is still a new and growing field. Algorithmic work to date has largely focused on reconfiguration algorithms. The work for this project largely focuses on designing improved reconfiguration algorithms as well. In addition, this project will investigate exactly what “power” is gained by different physical implementations of modules and what is lost through certain restrictions on computation done by modules or communication between modules.

Reconfiguration Algorithms The earliest reconfiguration algorithms largely utilized central planners. This obviously does not scale well to large systems of hundreds or thousands of modules which is a long-term goal of the field. More recent reconfiguration algorithms have been distributed, allowing for concurrent movement of modules, but often required large amounts of communication between modules.

A current focus in the field is the design of local-rules algorithms. It may be possible for the entire system to behave intelligently when the individual modules follow only local rules based on their surrounding neighbors. Inspiration for these algorithms may come for various different fields such a cellular automata theory, the field of self-assembly, or even current work with controlling other types of distributed systems. To date only very simple local-rules reconfiguration algorithms seem to have been designed. It may be that the power of algorithms using only local rules is limited. Determining what these limitations may be as well as designing sophisticated algorithms to meet these limits is one research goal.

Parallel Performance Initial bounds developed for reconfiguration algorithms held a simplifying assumption that only one module was allowed to move at a time. This model allowed for efficient finding lower bounds on optimal reconfiguration performance given an initial and a goal configuration. However, the nature of self-reconfigurable robots is that large numbers of modules in a system may concurrently act and doing so significantly improves algorithmic running times. Establishing new bounds for reconfiguration algorithm performance that account for the possibility of concurrent actions would allow current algorithms to be more accurately judged. Such bounds would also require an improved sense of the difference between initial and goal configurations — a metric that measured the opportunity for concurrent movement as well as the geometric difference between configurations.

In addition, since self-reconfigurable robots act like moving, physical implementations of distributed systems, other valued attributes from distributed algorithms are of use. This includes the ability of reconfiguration algorithms to be redundant to module failure or delay. Recent reconfiguration algorithms have noted the possibility for module delay, but improvements could still be made in designing redundant algorithms and properly analyzing their performance in the face of module delay or failure.

A New Abstraction Model The most recent conceptual abstraction for self-reconfigurable robots is the sliding cube model [reference here]. In this model individual modules are represented as cubes which can slide along surfaces composed of other cubes. Modules are also able to make convex and concave transitions to other surfaces. Most physical designs for robot modules meet the requirements of this abstraction. Also, utilizing an abstraction allows for algorithms to be designed and analyzed free of implementation-specific details.

While the sliding cube model has been useful, simply adding one more small ability to the module abstraction may drastically improve algorithm running times. The ability for one module to torque or rotate a small number of neighbor modules is a reasonable addition within the capabilities of many current hardware implementations. However, with this addition we can use forces generated by the system as a whole to accelerate certain modules past a uniform speed that current models assume as a limit for all modules. Using this accelerating technique dramatic improvements can be made in reconfiguration algorithms. In addition, the ability for one module to push another while traversing a surface or to move through the interior of a lattice of other modules — rather than only along exterior surfaces — are capabilities of modern module hardware but are excluded by current abstraction models. Including these allows for improved algorithms in many cases, especially when concurrent action of modules is considered as noted in the ‘Parallel Performance’ subsection.

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. *Autonomous 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 Intl. 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.