Self-Reconfiguring Robots in the USA

Robert Fitch* and Daniela Rus* *Department of Computer Science, Dartmouth College

1. Robots that Change Shape

A robot designed for a single purpose can perform some specific task very well, but it will perform poorly on a different task, in a different environment. This is acceptable if the environment is structured; however if the task is in an unknown environment, then a robot with the ability to change its shape to suit the environment and the required functionality will be more likely to succeed than a fixed-architecture robot. Self-reconfiguring (SR) robotics addresses these issues: in an SR robot, hundreds of small modules autonomously organize and reorganize as geometric structures to best fit the terrain on which the robot has to move, the shape of the object the robot has to manipulate, or the sensing needs for the given task. For example, the robot could synthesize a snake shape to travel through a narrow tunnel, and then morph into a six-legged insect to navigate on rough terrain upon exit. This vision was introduced by Fukuda and Kawauchi [1] with CEBOT (cell structured robot). The first proposed SR robot, CEBOT was described as an implementation of the 1987 idea of a Dynamically Reconfigurable Robotic System (DRRS) [2]. The definition of DRRS parallels our current conception of self-reconfiguring robots - the system is made up of robotic modules (cells) which can attach and detach from each other autonomously to optimize their structure for a given task.

Self-reconfiguration provides a paradigm shift for studying the fundamental principles of organization and reorganization in physical systems. In robotics, self-reconfiguration defines a rich class of questions about designing, controlling, and using massively distributed systems of robots. Self-reconfiguration also offers fertile ground for applying existing concepts in novel ways. For example, in this domain there is a need for two kinds of planning algorithms: (1) to achieve a desired geometric shape and (2) to globally move the resulting shape. In the next two sections we examine some of the hardware and algorithmic challenges for achieving task versatility with SR robots, and we review key advances in hardware and algorithm design for modular robots, with an emphasis on research results from labs in the U.S.A.

2. Hardware Design

SR robot systems employ physical mechanisms that allow modules to dynamically and automatically configure themselves into a more complex robot. Creating an SR robot system poses many engineering challenges. These challenges are centered around designing the basic SR module and the inter-module connections, and aggregating distributed systems from these modules.

In general, the basic module shape should be as small and simple as possible, to enable the flexibility of creating a large range of geometric shapes. The modules should also be able to act independently. A simple module is easier to design and build, but simplicity may constrain functionality. Simplicity is also a key consideration in designing the inter-module connection mechanism. The modules will make and break connections frequently, so the connections should be simple and reliable. Other important design issues include the communication between modules, the actuator power, and the method used to supply electrical power to the system. Inter-module communication is necessary for cooperation and distributed control. A good connection mechanism could also be used as part of the communication system. Actuator power is the amount of force actuators have to exert to move the modules around. At a minimum, modules must be able to move their
own weight. In a 3D system, modules must be able to move their own weight against the force of gravity. Supplying electrical power to modules is difficult. If the modules supply their own power using batteries, their weight and size increases, which requires more power to move them around. One possibility is to use the connection mechanism to also transmit power. Design tradeoffs relating to these parameters are discussed by Castano and Will [3].

In [4], Yim, Zhang and Duff introduce the following nomenclature for types of SR robots. Lattice systems are systems whose modules are confined to a "virtual grid," whereas chain systems are composed of chains of modules attached to each other. See Fig. 1 for examples. We use this classification to describe existing SR hardware in this section, and also include modular systems that reconfigure manually.

2.1 Lattice-Based Robots

In lattice-based systems, modules are constrained to occupy positions in a virtual grid, or lattice. One of the simplest module shapes in a 2D lattice-based system is a square, but more complex polygons have also been proposed [5] [6]. Because of the discrete, regular nature of their structure, developing algorithms for lattice-based systems is often easier than for other systems. However, the grid constraint makes implementing certain rolling motions, such as the tank-tread, more challenging since module attachment and detachment is required.

The Metamorphic robot [7] is one of the first hardware prototypes for modular SR robots in the U.S.A. It was developed in the Robot and Protein Kinematics Lab at Johns Hopkins University and consists of modules designed as deformable hexagons. The modules are aggregated in a planar lattice by attaching neighbors on an adjacent hexagonal edge. Magnetic fields are used to swap connections, thus causing units to roll around each other as shown in Fig. 2. The basic unit of this robot is a six-bar linkage forming a hexagon. The kinematics of this shape were investigated when the design was proposed [8], and hardware prototypes were constructed later [7]. A unique characteristic of this system is that it can directly implement a convex transition: a given module can move around its neighbor with no supporting structure. The hexagon deforms and translates in a flowing motion. A square shape with this same property was also proposed. This motion primitive is important since it is required by many general reconfiguration algorithms, but many systems can only implement it using a group of basic units working together.

The Metamorphic Robot module was a hexagon in two dimensions, but a similar 3D module, the Rhombic Dodecahedron, was later proposed by Yim et al. [9]. The Rhombic Dodecahedron has 12 faces, and each face is a rhombus. Rhombic Dodecahedron modules pack tightly
in a 3D grid and locomote by rolling around each other. This shape is an example of a class called Proteo modules [10].

The first module proposed by the Dartmouth group is the Molecule [11]. This 3D module consists of two atom units connected by a 90-degree bond, forming the overall shape of an elbow with a connection mechanism at each end (see Fig. 3). Each atom has five inter-Molecule connection points and two degrees of freedom. One degree of freedom allows the atom to rotate 180-degrees relative to its bond connection, and the other degree of freedom allows the atom (thus the entire Molecule) to rotate relative 180-degrees relative to one of the inter-Molecule connectors at a right angle to the bond connection. The current design uses R/C servomotors for the rotational degrees of freedom. A feature of our prototype is the use of a gripper-type connection mechanism.

The rotating connection points on each atom are the only connection points required for Molecule motion. The other connection points are used for attachment to other Molecules to create stable 3D structures. Each Molecule also contains a microprocessor and the circuitry needed to control the servomotors and connectors. The diameter of each atom is 10.2 [cm], making the atom-atom distance in the Molecule approximately 14.4 [cm]. The weight of the Molecule is 1.4 [kg]. An individual Molecule has the following basic motion capabilities: (1) linear motion in a plane on top of a lattice of identical Molecules, irrespective of the absolute orientation of the plane; (2) convex 90-degree transitions between two planar surfaces composed of Molecules; and (3) concave 90-degree transitions between two planar surfaces composed of Molecules. These primitive motions for a Molecule relative to a substrate can be combined and sequenced to achieve global motions for the entire robot. In [11] we show that the smallest robot that can move in general ways in the plane is a four-Molecule robot. The smallest robot that can climb stairs in addition is an eight-Molecule robot.

A variation on this design is the I(CES)-Cubes robot [12]. The link component on this module, however, is separated from the end components. Therefore, the system is termed bipartite. The end units, the cubes, are passive and connect to Link units, which are actuated. In this way, the I(CES)-cubes implement motion primitives similar to the Molecule robot [13], although there are fewer constraints since each cube can be placed independently.

Motion of the previously discussed modules is primarily achieved by movement over the surface of the robot. A separate class, unit-compressible systems, use modules which move through the volume of the robot. The actuation method of unit-compressible modules is termed scaling-based because the modules expand and contract in multiple dimensions. The basic idea of using extendable arms for actuation to construct a reconfigurable robot was patented by Tanie and Maekawa in 1993 [14]. The Crystal robot (Fig. 1) was constructed by the Dartmouth group as a 2D physical realization of a unit-compressible system [15]. Crystal units are squares that attach to each other at each of the four faces, and expand and contract in two dimensions. Communication in the Crystal robot is between neighbor units only, and is implemented using infrared devices mounted in the faces. Both power and computation for the Crystal are onboard each unit, distinguishing this robot as one of the few untethered SR systems.

The Crystal Robot has some of the motive properties of muscles and can be closely packed in 3D space by attaching units to each other. By expanding and contracting the neighbors in a connected structure, an individual module can be moved in general ways relative
to the entire structure. Crystal atoms never rotate relative to each other; their relative movement is actuated by sliding via expansion/contraction. This basic operation is unique in that it supports module relocation through the volume of the structure, and thus leads to new algorithms for global self-reconfiguration planning. Each module has on-board processing, sensing, communication, and power. When fully contracted, the Atom is a square with a 6.6 [cm] side. When fully expanded, the Atom is a square with a 13.2 [cm] side. The height of the Atom is 18.8 [cm] and its weight is 500 [g].

A unit-compressible system extended to three dimensions was developed at Xerox PARC [16]. This implementation, the Telecube, is similar to the Crystal but has an added degree of expansion and contraction for the third dimension [17]. The Telecube module is 6 [cm] across when contracted, and weighs 300 [g]. An external power source is required but the Telecube avoids a large number of tethers by routing power between modules.

2.2 Chain-Based Robots

In chain-based systems, modules aggregate as connected 1D strings of units. This class of robots easily implements rolling or undulating motions as in snake robots or legged robots. However, control is much more difficult for chain-based systems than for lattice-based systems because of the continuous nature of the actuation: modules can move to any arbitrary position as opposed to a fixed number of neighbor positions in a lattice.

The first prominent chain-based system was Polypod, proposed by Yim in 1993 [18] [19] and developed at the Palo Alto Research Center (see Fig. 1). Polypod is made up of Segments, which are actuated 2-DOF 10-bar linkages, and Nodes, which are rigid cubes housing batteries. Polypod modules are about 5 [cm] in size, and consist of two square parts that can rotate by 90-degrees relative to each other by a standard hobby servo. The modules have the ability to to automatically attach and detach from each other, making the system self-reconfigurable. The connection mechanism is provided by two connection plates on either side of the module, which are identical and have a four-way rotational symmetry at 90-degree increments. Four grooved pins enter four holes and are grabbed by a latching mechanism that is released by a shape-memory alloy actuator.

Multiple gaits for locomotion, including rolling, legged, and even Moonwalk gaits, were demonstrated with Polypod. Polybot succeeds Polypod, sharing the same bipartite structure. Segments in Polybot abandon the 10-bar linkage in favor of a 1-DOF rotational actuator. The latest generation of Polybot prototypes has on-board processing and CANbus (controller area network) hardware for communication. Two CANbuses on each module allows the chaining of multiple module groups to communicate without running into bus address space limitations.

A system that uses a similar actuation design is CONRO (CONfigurable RObot) [20], shown in Fig. 4. The CONRO module has two rotational degrees of freedom, one for pitch and one for yaw, and was designed with particular size and weight considerations [21]. Considerable attention has been paid to the connection mechanism, which is a peg-in-hole connector with SMA (shape-memory alloy) latching mechanism that can be disconnected by either face. Computation is on-board each module, so unlike Polypod, CONRO has only one module type. Power can be provided externally or via batteries on later prototypes [3]. Examples of manually configured shapes are the snake and the hexapod, and the current CONRO system is designed for self-reconfiguration.

2.3 Manually Reconfigurable Robots

Manually reconfigurable modular systems share many design issues with self-reconfigurable systems. Many groups have developed modular robotics [22] [23] and manipulators, such as the Reconfigurable Modular Ma-
3. Planning and Control

SR robots are dynamic structures: they can move using sequences of reconfigurations to implement locomotion gaits, and they can undergo shape metamorphosis. In addition, these robots are also capable of self-repair. The ability to move and change shape preserve the number of units in the robot invariant. Self-repair often involves removing a bad unit from the robot, which decreases the size of the system. SR robots are also capable of self-division, an operation in which a large robot is split into a collection of smaller robots with the same behavior as the large robot, and merging, an operation in which two small robots aggregate as one. These capabilities can be formulated as motion planning problems, although initial theoretical work in planning and control for modular systems comes from the perspective of cellular automata theory. For additional information on early cellular and cooperative robotics research, see survey papers by Sandini [28] and Cao, Fukunaga and Kahng [29]. In this section, we survey approaches to the reconfiguration problem, locomotion, self-repair and division.

3.1 Reconfiguration Planning

The task of transforming a modular system from one configuration into another is called the Reconfiguration Planning problem. Solving this problem is fundamental to any SR system. In some approaches explicit start and goal configurations are given, and in others the goal shape is defined by desired properties. Centralized algorithms require global system knowledge and compute reconfiguration plans directly, whereas decentralized algorithms compute solutions in a distributed fashion without the use of a central controller. Reconfiguration algorithms can be designed for specific robots, or for classes of modules. Often a centralized solution is more obvious and is developed first, followed by a distributed version, although not always. Not all decentralized algorithms are guaranteed to converge to a solution, or are correct for arbitrary goal shapes. We review reconfiguration algorithms in this section.

A common technique used in reconfiguration algorithms for lattice-based systems is to build a graph representation of the robot configuration, and then to use standard graph techniques such as search to compute motion plans. Planning for the Molecule robot developed by the Dartmouth group is one example [11]. Another example from the Dartmouth group is planning for unit-compressible systems such as the Crystal [15]. This planner, named MeltGrow, uses the concept of a metamodule, where a group of modules are treated as a single unit with additional motion capabilities. The Crystal robot implements convex transitions using metamodules called Grains.

Centralized planners can also store pre-computed data structures such as gait-control tables. Once a gait is selected by the central controller, it is executed by local controllers on the individual modules. This type of algorithm is used by Polypod [19]. The division between central and local controllers is also used in by RMMS [24], and I-Cubes [12].

Distributed planners are often based on the message-passing approach. Shen, Salemi and Will [30] and Salemi, Shen and Will [31] propose a control system for CONRO using a message-passing scheme called Digital Hormones. The problem of distributed reconfiguration for unit-compressible modules was solved, using message-passing, by a combination of the Pacman algorithm developed by the Dartmouth group [32] and later modifications and analysis by Vassilvitskii, Yim and Suh [33]. This distributed algorithm is correct and complete for arbitrary shape reconfigurations of 3D unit-compressible cubic systems using metamodules.

In addition to algorithmic solutions, other theoretical issues related to the reconfiguration problem have been addressed. Chirikjian and Pamecha [34] discuss upper and lower bounds for self-reconfiguration. Metrics for reconfiguration planning have also been studied [35] [36], and Vassilvitskii, Yim and Suh provide complexity analysis for their distributed reconfiguration algorithm [33].

3.2 Locomotion

SR robots are well-suited for locomotion in unknown environments. For example, SR robots can climb stairs even in the absence of models of the height, width, and length of the stairway. The robot will be given the command to move forward. The robot will proceed with a translation motion until the front sensors mounted on the front modules in the structure will detect an obstacle (e.g., the first step). At this point the robot
will change the locomotion modality from translation on the ground to stacking. When the top modules detect free space again (that is, after the first step has been cleared), the robot will change locomotion modality again to unstacking and translation. This operation requires sufficient modules to make a tower at least as large as the height of a step. Alternatively, suction cups or electromagnets can be added to each module to enable them to climb, much like our climbing robot [11].

Generally, chain-based systems locomote without module detachment, while lattice-based systems require reconfiguration to perform locomotion. Yim[19] demonstrates control for rolling and legged locomotion gaits for Polypod. Stay, Shen and Will[37] present distributed locomotion algorithms for CONRO. In our previous work, we studied distributed inchworm-style locomotion for unit-compressible systems [38].

The ability of SR robots to bend and flex around obstacles enables them to use very simple on-line planning strategies to navigate around complicated 3D objects. On-line algorithms for navigation take advantage of SR capabilities to create a “water-flow”-like locomotion [39].

3.3 Other Tasks

The built-in redundancy of SR systems leads to interesting fault-tolerance and self-repair properties. If an arbitrary part of a fixed-architecture robot fails, the robot cannot usually repair itself; a human or a different robot must perform the task. An SR robot carrying some additional modules may excise the failed part and replace it with the spare units. Because modules are identical and they can move in general ways relative to one another, it is possible to detect and eliminate defective modules, while replacing their functionality in the system. This problem has been addressed by defining algorithms for detecting failure, for ejecting failed modules from the system (using a version of the module relocation algorithm), and then inserting working modules in the holes (also using module relocation gaits) [40] ~ [42].

Another useful capability of SR robots is self-replication, the ability of a large robot to divide itself into several independent smaller robots with the same basic functionality (but not identical size). For example, a system consisting of 100 modules could function as one large robot or 10 smaller robots each consisting of 10 modules, or any number of other configurations. Self-replicating robots are useful in tasks where the overall effectiveness and task completion time is improved by parallelism, such as distributed surveillance or exploration. Consider, for example, the distributed surveillance and monitoring task. A single robot can only survey the portion of the environment currently in its field of view; a robot that can self-replicate will enlarge the overall field of view. Similarly, a single robot requires longer to traverse a path in the environment. A robot that can split itself into several pieces can take advantage of parallelism to complete the exploration task faster. Solutions to this problem are in their infancy, but some first results are described in [43].

4. Conclusion

Creating SR robots is a considerable challenge, which can be met through (1) new designs for reconfigurable systems and (2) new ideas on algorithmic planning and control. The robots developed so far are very encouraging first steps towards creating new robotics applications.

Acknowledgements Support for this work was provided through NSF awards EIA-9901589, IIS-9818299, IIS-9912193, EIA-0202789 and0225446, and ONR award N00014-01-1-0675.

References

Robert Fitch

Robert Fitch received the B.A. degree in psychology with honors from Oberlin College, Oberlin, Ohio in 1996. He is currently pursuing the Ph.D. Degree in the Department of Computer Science at Dartmouth College, Hanover, NH. He has been a member of the Dartmouth Robotics Laboratory since 1998. His research interests include distributed control and reconfiguration planning for self-reconfiguring robots.

Daniela Rus

Daniela Rus is a professor in the Computer Science Department at Dartmouth, where she founded and directs the Dartmouth Robotics Laboratory. She also co-founded and co-directs the Transportable Agents Laboratory and the Dartmouth Center for Mobile Computing. She holds a PhD degree in computer science from Cornell University. Her research interests include distributed robotics, self-reconfiguring robotics, mobile computing, and information organization. She was the recipient of an NSF Career award and an Alfred P. Sloan Foundation Fellowship. She is a class of 2002 MacArthur Fellow.