Modular Reconfigurable Robotics

Jungwon Seo,¹ Jamie Paik,² and Mark Yim³

¹Mechanical & Aerospace / Electronic & Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, junseo@ust.hk
²Reconfigurable Robotics Laboratory, EPFL, Lausanne, Switzerland, jamie.paik@epfl.ch
³General Robotics, Automation, Sensing, and Perception (GRASP) Laboratory, University of Pennsylvania, Philadelphia, USA, yim@seas.upenn.edu

Keywords
cellular and modular robots, distributed robot systems, multi-robot systems, reconfigurable robots

Abstract
This article reviews the current state of the art in the development of modular reconfigurable robot (MRR) systems and suggests promising future research directions. A wide variety of MRR systems have been presented to date, and these robots promise to be versatile, robust, and low-cost compared with other conventional robot systems. Reconfigurable robot systems thus have the potential to outperform the traditional systems with fixed morphology when it comes to performing tasks that require a high level of flexibility. We begin by introducing the taxonomy of MRRs based on their hardware architecture. We then examine recent progress in the hardware and the software technologies for MRR along with remaining technical issues. We conclude with a discussion of open challenges and future research directions.
1. INTRODUCTION

Modular reconfigurable robot (MRR) systems are made up of many repeated modules (or units) that can be rearranged or can rearrange themselves into different configurations depending on the task the robot is to solve at the time. Note that the meaning of the term configuration in reconfigurable robotics (and this article) is commonly generalized to incorporate the connectivity of the modules (which module is connected to which, represented as an adjacency matrix, a linked list, and the like) into the conventional robotics definition of the term that refers to just the pose of the robot (the full set of the joint angles of the robot). The term reconfiguration thus also refers to the process of changing connectivity.

MRR systems have three promises (1).

- **Versatility** The systems typically have many redundant degrees of freedom (DOF) and can adapt their configurations to suit a wide range of tasks.
- **Robustness** The redundancy and self-reconfiguration can be used for self-repair increasing robustness.
- **Low cost** Repeated modules mean that economies of scale can be used to reduce the cost of those modules.

Unfortunately, only one of the three promises, that is, versatility, has been proven to date. Existing literature shows that MRR systems demonstrating hundreds of locomotion and manipulation tasks (2, 3). However, these research prototypes are not very robust. The costs of these research prototypes are also large, typically ranging from several hundred to several thousand dollars (USD) for each module.

This article is intended to provide a review of reconfigurable robotics research for those who are interested in performing research. Rather than surveying many existing systems and demonstrations, this paper will focus on design insights and important research directions. The reader is expected to have a background in engineering and robotics research. This article will focus on the differences between MRR and traditional robot systems.

MRR systems can be classified into several architectural groups based on the geometrical arrangement of the units.
Lattice Reconfiguration Architectures: Lattice reconfiguration architectures have units that are arranged in a regular, three-dimensional pattern, such as a cubic crystal lattice or cannonball packing. These systems exploit this regularity to ease the computational aspects of reconfiguration.

Chain Architectures: Chain architectures are characterized by units that form serial chains. These chains are often connected to form a tree or closed chain loops. Through articulation, chain architectures can potentially reach any point or orientation in space, and are therefore more versatile. But generally they are more demanding to represent and analyze computationally, and more difficult to control.

Mobile Architectures: Mobile architectures have units that use the environment to maneuver around and can either hook up to form complex chains, lattices, or a number of secondary robots that can perform swarm-like behaviors.

A fourth MRR class has started to emerge, the reconfigurable truss. These systems do not fit into the lattice, nor mobile, nor chain style (serial chains are not an inherent part of the system). Instead, prismatic members form parallel truss structures that can reconfigure, changing the topology of the network. In addition, hybrid systems can often exploit the best properties of multiple different architectures.

These architectural groups will be elaborated with examples in Section 2, along with trade-offs in design. Sections 3 and 4 discuss recent advances and challenges in the hardware and the software technologies for MRRs, respectively. Section 5 presents a discussion of open problems in modular reconfigurable robotics.

2. OVERVIEW

As a field, there are dozens of groups who have constructed many versions of MRRs with many approaches for programming them. Over 1800 papers and a book (2) have been written on reconfigurable robotics at the time of this writing. Many of the early papers are covered in a survey (3).

2.1. Architecture

Here we will give examples of MRR systems in each architectural class. The intent is not to be exhaustive, but to find historically early representative examples.

2.1.1. Mobile Architecture. The first work towards an MRR system can be traced to Fukuda and his CEBOT robot (4), which can be considered a mobile reconfiguration system. The primary contribution of this work was a probe and drogue docking mechanism tested on a wheeled robot driving to and docking with a mating module.

Mobile reconfiguration architectures also take advantage of air and water mobility. One example can be seen in the reconfigurable floating structure system called the Tactically Expandable Maritime Platform—TEMP (5). This work emphasized the systemic issues of dozens of modules attaching in the open seas, featuring large amounts of uncertainty in sensing and control. A recent work called Modquad (6) has shown mid-air self-assembly (Figure 1). In this case, mechanical docking is enabled by magnets on the corners of a cubic frame around small quad-rotor UAVs. Because of the extreme weight budget requirements, the frame and docking mechanisms must be made as minimal as possible. Demonstrations of mid-air assembly of arbitrary planar shapes have been shown, but an undocking mechanism,
which will enable disassembly, has yet to be presented.

2.1.2. Lattice Reconfiguration Architecture. One of the first important works in lattice reconfiguration was the crystalline robots shown in Figure 2 (7). It helped to start the local-rule based algorithmic approaches grounded in a physical robot with the concomitant constraints that arise from physical systems.

A critical shared aspect of the lattice reconfiguration architectures is that reconfiguration is a local process. When planning for and determining whether a module can move from one position to another, lattice systems need only check a fixed set of locally neighboring positions for the existence or absence of modules. All other architectures have global constraints. This implies that the computational time complexity for the reconfiguration of the lattice systems can be independent of the number of modules, not so for all other architectures. In addition, the control for each module need only be concerned with providing local motion of one module relative to another, which can simplify module design.

While Modquad in Figure 1 might be another example of a system that forms modules into a lattice, it is different than lattice reconfiguration architecture in that Modquad modules become rigidly attached and maneuver through the environment whereas lattice modules maneuver on other modules.

2.1.3. Chain Architecture. The PolyBot system (1) shown in Figure 3 is one of the earliest chain architecture systems. It is composed of two types of modules, one that has one articulated DOF with two connection plates and the other that has zero DOF with six connection plates as the faces of a cube. The one-DOF modules can be attached end-to-end to form snake-like articulations. The zero-DOF is used as structural nodes in tree like configurations.

Whereas the lattice and the mobile architectures do not form serial chains, this architecture can form traditional robot arms with full six-DOF control of the distal end of the arm. As a result these types of systems have demonstrated a variety of traditional robot arm tasks, whereas the mobile and lattice architectures have focused more on forming different shapes.
2.1.4. Hybrid Architectures. One of the most impactful designs in hybrid architectures was the MTRAN series of systems shown in Figure 4 (8). Its module is composed of two linked cubes. These linked cubes can form serial chains for manipulation or locomotion tasks and reconfigure in the way the lattice systems do. The design was also elegant in the attempt to minimize the required DOF to achieve both chain tasks as well as lattice reconfiguration.

Another important hybrid system is the SMORES system (9) shown in Figure 5, which has modules at a lower granular level than MTRAN: individual modules are one cube instead of two attached cubes. One important contribution in the SMORES design is the ability to emulate the function of MTRAN and most other MRR systems. One important aspect of the design that enables SMORES to emulate many other systems builds on the observation that many reconfigurable systems are based on a cube shape, can be arranged in a cubic lattice, and are able to rotate about a major axis through the center of the cube.
2.1.5. **Truss Architecture.** A truss reconfiguration system whose nodes are reconfigurable, called Variable Topology Truss is shown in Figure 6 (10). The system can metamorphose by changing the length of the truss members and the degree of each node (the number of the truss members incident to it). The advantage of this architecture is that the systems can be made very large and strong while being efficient in the material used, just as the trusses used in constructing buildings or bridges.

Representing the truss architecture as a graph will enable a graph-theoretic analysis computationally. However, the physical constraints that arise from having elements with finite, non-negligible sizes complicate the analysis of these systems. For instance, when the lengths of the members are changed, the angle between the nodes will also change; however, the physical size (e.g., diameter) of the member will prevent the angle between two members sharing a node from becoming too small. This limitation must be accounted for in any motion planning algorithm. Although the truss architecture has not been considered one of the major classes of reconfigurable systems in previous surveys (11), some relevant early examples include the variable geometry truss systems (12, 13, 14) in which truss members are variable, but not automatically reconfigurable.

2.2. **Trade-offs in Design**

There are a variety of aspects to consider when it comes to designing MRR systems. The hardware and the software design issues in MRR systems are highly intertwined, while in conventional robotic systems these are often examined separately. Interestingly, many of the difficult implementation issues occur on the software side. Given that the refinement of hardware design is extremely expensive after the construction, it is economical to put more effort into optimize the trade-offs before the software implementation.

**Manual vs Self-reconfiguration** Automatic self-reconfiguration is the main focus of this paper, though many of the aspects of manual (human assisted) reconfiguration systems are also shared. Even when considering manually reconfigured systems, Many can be made self-reconfigurable or robot-assisted, by replacing the human with a robot arm.

**Generality vs Specificity** When thinking about the goals of many autonomous robot systems from an academic (versus a purely industrial application) viewpoint, interesting questions arise from the goal of making the systems as general as possible. Academically, the more general or broadly applicable the technology, contributions to science, engineering, and impact on world tend to be greater. However, industrial applications require proving the ability to achieve specific performance levels. From an MRR point of view, this generality applies, but to an even larger degree. Since inherently the approach is focused at the building block level, the promise of versatility overlaps with generality.

**Multiplying Effect of \( n \) Modules** Determining which features to include in a module has an interesting effect on the design of modules. Robot designers realize that any change to one module gets repeated \( n \) times and thus can have major impact on a full system. In some cases, this can lead to “feature creep” where adding extra features leads to complex modules with high cost and often low reliability.

To counter these trends, the designers of the Claytronics system (15) decided to adopt a minimalistic approach they call the *ensemble axiom*: a robot should include only enough
functionality to contribute to the desired functionality of the ensemble. Their end goal was to create and program very large numbers of modules (millions) at very small size (less than 1mm on a side).

**Compactness** Another consequence of the multiplying effect of module design results in an increased emphasis on compactness. Tasks often have a characteristic workspace that often sets required robot parameters such as link lengths. Whereas modules can often be added to satisfy larger workspaces, the smallest workspaces are directly correlated with module size. Another way to view this for a given workspace, and a set of modules that span that workspace, halving the module size (length of a linear side) means being able to fit eight times as many modules in the same volume. This increases the number of DOF and thus the space of solutions for any given task.

### 3. HARDWARE AND DESIGN: ADVANCES AND CHALLENGES

One key characteristic of MRRs is the ability to adapt their hardware structure to suit a given task or environment. In this section, we will discuss how this poses unique challenges and has affected the hardware design of MRR systems on three major aspects:

- **Actuation** While off-the-shelf actuators are prevalent for MRR systems, there are an increasing number of customized and smart material based actuators to overcome geometric design restrictions and address the compactness design goal. For self-reconfiguration, latches can employ dedicated actuators for additional versatility in assembly of each module.

- **Sensing** The ultimate goal of MRR systems is to adapt and react to different environments and tasks. This requires not only having self-awareness of individual modules but also their assembled state and its interaction with the perceived environment.

- **Structure and Connection** Enabling modules to attach and detach in the presence of uncertainty is one of the unique aspects of MRR systems over traditional robots. Additionally, in order to make a system modular and self-reconfigurable, there is an inherent loss in strength with the increase in the number of the assembled modules (stiffness, ultimate tensile strength, etc.).

#### 3.1. Actuation

MRR modules typically have two types of actuators, a large main actuator which moves modules relative to other modules or to the environment, and smaller latch actuators that enable or disable the bond between modules.

A dimensionless measure of the main actuator is the *characteristic strength* which is defined to be the number of modules that can be supported in a cantilever fashion under gravity. This number is important as it describes the largest serial chain that a system can support without breaking. Note, this ultimately depends also on the bonding strength (important for lattice systems) and the module mass. Typically this number is on the order of 2 to 5.

MRR systems gain task space and capability by increasing the number of modules in a system. Therefore, the power density of the main actuator is critical in achieving a higher characteristic strength, and the capability of the module to support its own weight and to apply forces on the environment. In MRR systems, several actuation technologies are
used depending on the number and type of joints per module. Off-the-shelf actuators are
used in a variety of MRR systems: DC motors (16, 17, 18, 19), stepper motors (20, 21),
and servo motors (17, 22). There are customized actuators for achieving unconventional
movements such as a three-DOF pneumatic actuator (23, 24) or an electromagnetic inertial
actuator (25). These actuators provide mobility to an independent module or to the rest
of the structure for changing the configuration (19, 21, 25).

Not all motors need to be embedded on the structure: remote actuation sources have
demonstrated to be effective to launch reconfigurations. Platform setups that induce vibra-
tions (26, 27), magnetic field (28), or thermal gradient (29) can also initiate singular and/or
multiple sequences of motion. While such global actuation induces reconfigurations without
limiting the number of controllable modules, controllable DOF of individual modules in a
compact packaging is crucial for practical assembly tasks involving multiple modules. When
the power and torque density of the motor are critical for the overall reconfiguration space,
customized actuators using functional smart materials could provide new design freedom
to MRR systems. In robotics, there are examples of such actuators using electroactive
polymer (30), piezoelectric crystals (31, 32), shape memory polymer (33, 34), and shape
memory alloy (SMA) for their high force, torque, or strain to weight ratio characteristics
(for example, 1N force output from a 20mg actuator (35)). Among them, various forms of
SMA actuators have been demonstrating their application in lightweight MRR systems.

**SMA Actuators** SMA actuators (most often Ni-Ti alloy) enable some of the most compact
actuation mechanisms important for MRR systems. SMA actuator’s memory effect comes
from the crystal shape transition between the Martensite and Austenite state where each
state is defined by a predefined temperature threshold. To activate this phase transition,
Joule heating is often used to directly apply the current to the SMA actuators. This is
feasible for spring or thin ribbon actuators but is not efficient for other forms of SMA
actuators that produces higher forces. Otherwise, additional customized heater layer that
targets a specific zone of SMA is used (36, 37). MRR systems use various types of SMA
actuators such as torsional (38) or folding (39) actuators in addition to the most prevalent
spring/string actuators (40). Ribbon-type linear SMA actuators can also be composed of
modular blocks called unit cells (41) that can extend, rotate, and bend.

Most linear actuators are realized by winding SMA wires to achieve a larger range of
motion beyond the typical maximum 5% strain rate. This winding process requires them
to maintain the equal tension of the wire throughout including the curing process. An
alternative actuator type uses SMA sheets. Due to their planar geometry, they can be
fabricated by laser-cut patterning (42) or by etching techniques (43, 44). This process is
repeatable, fast, flexible, and creates actuators that can be directly annealed without need-
ing to secure them onto another element. Furthermore, as these actuators are planar, they
can easily be integrated into prototypes by adding an actuation layer in combination with a
heating layer. This type of SMA can pave the way for an assembly-free fabrication process.
The manufacturability of SMA actuators in various forms brings advantages to modular
design. Laser cutting technology enables flexible, plug-and-play and “no gearbox” designs
of sheet-type SMA actuator modules (39, 35) for multi-crease self-folding robotic origami
sheets (45) with minimal mechanical assembly effort. The choice of actuation dictates the
mechanical performance of the MRR, while the customizability, compactness, light weight,
and high force output make SMA actuators highly attractive for robotic applications. These
features are especially beneficial for MRRs, as each module requires the self-contained and
integrated design of several mechanisms for motion, coupling, and reconfiguration.

**Bonding Actuation** Ideally the connection plates transfer information and power, but at a minimum, it must physically anchor one module to the next. These mechanisms are often hooks that are activated by DC motors (16, 17, 19, 46, 47) or use permanent magnets (9), electromagnets (48), and/or electro-permanent magnets (49). While SMA actuators' maximum torque or force output may be too small to be used as the main actuator on heavier MRR modules, for activating latches it is an attractive choice for its high force density and direct drive that requires no additional transmission. Earlier work on spring SMA-based connection mechanisms demonstrate compact female-male retractable pin connectors to lock-unlock several modules (50) arranged in lattice or chain reconfigurations employing magnetic connectors (51). A similar approach with spring SMA actuators is proposed for automatic coupling of modular origami robots (21) by retracting-releasing a sliding shaft with rotary teeth that, when coupled with stationery teeth, allows rotation of connected modules relative to one another. For self-assembly inside a fluid, the fluid flow can be used (much like magnets) to apply forces on the modules as well as form pressure bonds to hold structures together (52). More recently, a robot that melts (or remelts) plastic around the rim of the robot to join two modules has resulted in bonds that could support over 5kg load in 100 repeated trials (53).

### 3.2. Sensing and Communication

The sensing and communication in MRRs are often homogeneous over distributed modules. Because the systems are expandable, the reach of their sensory resources need to be scalable. We can separate sensing into two categories: internal and external sensing. The internal sensing modalities are for monitoring the states of each module locally. The external sensing modalities are involved with observing the environment outside the robot system.

**Internal Sensing** As MRR systems are characterized by large numbers of independent modules and their DOF, the principal sensing requirements concern the control of active/passive DOF. The majority of MRR systems use rotary joints, although some also have linear joints. The states of these joints are monitored by mixing and matching rotary encoders (17, 19, 20), potentiometers (16, 9, 54), Hall-effect sensors (18), or various strain sensors. One example that maximizes compactness—one of the important MRR design goals—is PaintPots (55). PaintPots utilize conductive paint to build potentiometers into the structure of a robot module at a relatively low cost. Some MRR systems are also equipped with sensors such as accelerometers that can measure the orientation of modules, which can be critical for startup and initialization procedures (16, 56, 54).

Another critical sensing function for MRR is to help guide docking processes, especially for chain and mobile reconfiguration. Chain systems utilize inverse kinematics with the internal sensors mentioned above, but often errors are too large. On-board cameras (57) or arrangements of LEDs and photosensors (58) can be integrated for better state estimation.

**External Sensing and Communication** External sensing can be used to detect other modules as well as the environment. This is crucial for reconfiguration and global tasks, in which modules need to recognize each other and communicate directly or indirectly with their neighbors. More specifically, this involves identifying who and where a neighboring module
is (16), as well as detecting any interaction between them with, for example, a touch switch (19) or spring-loaded contact (59, 60). Once a physical contact is made, the coupling is commonly used for direct communication amongst modules through electrical contacts (18, 61, 17) or induction (49). A bus system can simplify global communication, but is limited to a group of modules in direct contact. Serial communication between neighboring modules can overcome the communication bus limits for scalability, but increases the complexity of an individual module and the system may still require some form of global communication. Wireless communication has been implemented both globally and locally using infrared (16, 19, 25, 56, 54), Bluetooth (16) and Wi-Fi (62), greatly improving adaptability and autonomy of the overall system. Although adding a considerable amount of overhead, it opens the possibility of sharing sensory information and enabling a more distributed approach to various functions. Broadcast messages are particularly useful as they are not limited to an address space. Initial studies have used distributed sensing of the environment using bump switches (20) as well as mapping using onboard sensors (63), while combining with an external sensing system seems to be effective for achieving dedicated tasks, either through mobile (64) or global systems (5). In heterogeneous systems, some of the sensing tasks can be off-loaded to dedicated sensing modules (65, 64).

Historically, most homogeneous systems have had limited on-board computation and sensing. While rigorous evaluation of the advantages of the homogeneous approach has not been done, a hierarchical approach with a centralized sensor and environment understanding seems to be more effective. Due to the complexity of realizing a MRR system, a considerable amount of research has focused on functionality, system integration, and control, whether physically or in a simulation. However, in order for a system to realize tasks in real-world scenarios and fulfill the the promise of versatility, environmental sensing and awareness are crucial to success.

3.3. Distributed Control Architecture

Typically, the distribution of computational resources tends to complicate the programming and control of a system versus a single centralized resource. However, including computational resources on each module means available memory and computational cycles scales as the number of modules increases. There are a variety of control architectures that could implement the software functions expected from MRRs. Homogeneously distributed approaches are elegant and interesting from a computer science perspective, which range from bioinspired control (66) to control of millions (67) in simulation and nearly a thousand (68) physically. The challenge tasks demonstrated include self-repair (69), locomotion (70), and manipulation (71), all of which are still at a relatively low level.

From a practical point of view, an approach that takes advantage of the inherent hierarchical physical arrangements of computational elements might work better for higher level tasks that involve combining tasks of manipulation, locomotion, and environmental sensing. The hierarchy can include centralizing activities such as sensing and interpreting the environment, as well as reasoning about what is the appropriate configuration for the conglomerate, while distributing local activities, controlling individual DOF or motions of subgroups whose actions are tightly coupled (e.g., a robot arm). Section 4.1 shows an example of such divide between the centralized and distributed control which was more efficient than the fully central or fully distributed control (64).
3.4. Structural Strength

The utility of the conglomerate shapes that MRRs can form is an important issue. The DARPA Programmable Matter program (72) (not to be confused with the CMU Programmable Matter effort (73)) focused on developing useful structures often using a wrench as an example shape: could an MRR system form a wrench that has enough strength to turn a bolt?

The strength (e.g., stiffness) of a conglomerate is a function of the module materials as well as the bonding stiffness between the modules. Typically, the bonding mechanism is significantly weaker than the module materials. (74) presents a strength analysis using the 6×6 stiffness matrices of the systems composed of both soft and rigid components.

An alternative approach to creating mechanically stronger MRR systems is to change the architecture of the system from attaching elements serially to inherently parallel mechanisms such as the truss architectures. Parallel structures are naturally stronger than serial ones, but still the strength of the full system is dependent on the strength of the constituent truss members. In the VTT case (10), the spiral zipper mechanism used to form the tubes of each member has optimal strength to weight ratio for beams under compression (75).

While structural weakness is typically the main issue with MRR conglomerates, sometimes structural flexibility is required. For example, in the TEMP project (5) with an assemblage of large floating structures in the open ocean, the waves from the sea state combined with the inertia of the modules can induce very large forces and moments. Complying with the waves can reduce these forces and moments so the structure can actually survive.

An important aspect of robot arm systems coupled with stiffness is precision. In order to obtain highly precise positioning, the structure of robot arms need to be stiff. While there are some examples of modulating the individual joint stiffness for achieving higher precision in grasping, the lack of stiffness in MRR systems leads to difficulties in docking with itself, imposing design constraints on the connection mechanism.

3.5. Connection Mechanisms

Although there are a wide variety of MRR systems ranging from rotating cube-shaped modules in lattice systems to extending prismatic beam in truss systems, there are a few common aspects to all self-reconfiguring systems. One of them is the ability to physically attach and detach from other elements of the system. We will call that interface the connection plate. Two modules can join and become rigidly attached together by having two connection plates physically lock (mate) together. There are several important characteristics for mating. These are explained below:

- **Gender** A connection plate can have four different types of gender, *male*, *female*, *genderless*, or *hermaphroditic*. Male plates usually have a physical feature that protrudes and mating female plates have the negative feature to that protrusion. Note that male/female primarily denotes a polarity, as magnetic connectors (as in SMORES shown in Figure 5) can be considered gendered male or female even without protrusions - labeling north or south arbitrarily as male or female. Hermaphroditic connectors have both male and female features and tend to be the most common as all connection plates can be the same, so any two connection plates can mate.

- **Approach Direction** The docking process between two connection plates assumes that there is a specific approach direction. Typically, this will be a translation parallel
to the normal of the plane that characterizes the mating face; for example, perpendicular to the faces of the Modquad, Crystalline, and SMORES modules. Note while the vast majority of approach directions are simple translations, it is possible to design a connection plate with a complex approach direction, such as the motion of a screw (still one-DOF motion, but coupled translation and rotation).

- **Latching** Every docking connector needs to form a rigid connection between the modules. This includes preventing motions backing out along the approach direction. This can be done with a latch which can be passive or active. The difference is whether actuators are added to the connection plate specifically to latch or unlatch. Passive latches utilize the main actuators to cause some form of unidirectional holding action (e.g., magnets or bistable snap-through latches). Undocking for these latches must be either through the main actuation again or through some other external mechanism (e.g., a human hand or a robot gripper).

- **Compactness** As with modules in general, compactness of the docking mechanism is an important design trade-off. High reliability or the ability to tolerate large errors often come at the cost of space. For example, the docking mechanism of the Atron system (76) consumes more than half of the space inside the module (77), although it is considered arguably one of the most reliable docking mechanisms.

3.5.1. **Area of Acceptance.** Given the difficulties with precision positioning (Section 3.4), many systems aim to maximize the ability to dock in the presence of uncertainty. Thus, a metric to measure how much uncertainty a docking system can tolerate yet still dock can be used as a figure of merit.

The area of acceptance (AA) is defined as “the range of possible starting conditions for which mating will be successful” (78). In other words, AA is the full set of possible errors (for translation, rotation, and the two in combination) that two docking connectors can have positionally relative to each other yet still successfully dock together.

**Zero Rotation Area of Acceptance** When all rotational DOF are constrained, we call the set of positions that align successfully the Zero Rotation Area of Acceptance (ZRAA). In case two modules approach in the z-direction translationally, ZRAA can be represented as a finite region in the xy-plane. This gives us a relatively simple, quick picture of the acceptance potential of the given connector. A similar idea can be seen in (79). Some analytically determined ZRAA values are summarized in Table 1.

**Full Area of Acceptance** The Full Area of Acceptance (full AA) is obtained by taking all the possible DOF into account. For two-dimensional connectors the full AA is represented with two parameters for one translational and one rotational DOF. For three-dimensional connectors five parameters are needed to represent the two translational and three rotational DOF. In the most general case, it is difficult to develop an analytical model to estimate AA; sampling-based methods can be used instead.

4. **SOFTWARE AND CONTROL: ADVANCES AND CHALLENGES**

Ultimately, the vision of MRR systems includes fully autonomous self-reconfiguration. Not only all the conventional AI/robotic issues ranging from semantic understanding to motion control, but also MRR-specific concerns need to be addressed. From a software control point of view we can break the MRR-specific issues down into two parts:

- **Task-Shape Matching Problem**
Table 1  Table of ZRAA metrics, normalized to characteristic length of the face.

<table>
<thead>
<tr>
<th>System</th>
<th>Normalized ZRAA Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENFA Connector(80)</td>
<td>0.00353</td>
</tr>
<tr>
<td>Polybot(70)</td>
<td>0.00503</td>
</tr>
<tr>
<td>M-TRAN III(16)</td>
<td>0.00592</td>
</tr>
<tr>
<td>JHU(81)</td>
<td>0.00592</td>
</tr>
<tr>
<td>I-Cubes*(82)</td>
<td>0.0187</td>
</tr>
<tr>
<td>CONRO*(83)</td>
<td>0.0425</td>
</tr>
<tr>
<td>Vacuubes(84)</td>
<td>0.0555</td>
</tr>
<tr>
<td>X-CLAW(85)</td>
<td>0.0649</td>
</tr>
<tr>
<td>ACOR(unpaired)(86)</td>
<td>0.0711</td>
</tr>
<tr>
<td>SINGO Connector(87)</td>
<td>0.306</td>
</tr>
<tr>
<td>DRAGON(88)</td>
<td>0.353</td>
</tr>
<tr>
<td>amour(89)</td>
<td>1.57</td>
</tr>
<tr>
<td>3D X-Face(78)†</td>
<td>2.00</td>
</tr>
</tbody>
</table>

* Estimated ZRAA values; † The 3D X-Face is not a full connector.

• Reconfiguration Planning and Control

The highest level activity in MRR scenarios is to understand the task or situation and then matching a configuration that can address the task (Sec. 4.1. Task-Shape Matching). The system must then determine how to metamorphose its current configuration into the desired one (Sec. 4.2 Reconfiguration Planning and Control). This can be broken into two parts, one that focuses on the connectivity arrangement and a second one that determines how to move the modules in a collision-free manner despite many DOF in a typical MRR system.

Many of the specific complications in developing software for reconfigurable robots are derived from the architecture of the hardware. For example, some latching systems require docking approaches from specific directions and most systems have a limited number of modules that can be suspended in a cantilever fashion under gravity. These types of constraints along with module geometry constraints can drastically change the implementation of software to control reconfiguration or to perform tasks. However, there are still a set of common classes of software tasks for MRR systems.

4.1. Task-Shape Matching

The task-shape matching problem with MRRs is concerned with determining which of the possible connectivity arrangements of a MRR system may be the most suitable for a given task, which can be any robotic locomotion or manipulation activity in the most general case. Developing a formalism to rigorously describe the tasks that we want robots to perform is an active research area. One approach to user-friendly, high-level task description is to use linear temporal logic (LTL) formulas (90). At a relatively low level, particularly in the robotic manipulation and locomotion literature (91), robotic tasks are given as a control objective for motion, force, hybrid motion-force, or impedance control.

In the MRR context, it is critical to understand the space of all the possible connectivities of a given robot system. This is nontrivial because the search space and the number of constraints on module connectivity will grow exponentially with the number of modules.
In many cases, such constraints may also be complex and global. For example, a docking event between two modules may be affected by the presence of another module that is not directly involved with that process. Constructing the database of robot connectivities becomes more complex if we additionally take into account what each connectivity is suitable for: its functions and behaviors. Here, problems worth further investigation include understanding what connectivities may function identically, due to structural symmetry for example. Such high-level information can help us figure out the structure of the search space better and query the database more efficiently.

Given a database of feasible robot connectivities and the descriptions of a desired task, a variety of approaches to task-shape matching can be considered. In (95), the problem is formulated as a discrete optimization procedure, with a task-oriented objective function. In (96), a sensorimotor primitive layer is proposed as a tool for bridging the robot hardware system and the tasks. The concept of a rapidly deployable system is presented in (97). It features the paradigm of software assembly that has the potential to facilitate the generation of control software, as part of a solution to task-shape matching. In (98), a hierarchical selection process is proposed as a way to systematically search for the space of robot connectivities. Although these approaches may be effective in case the search space is relatively small, the highly complex and nonlinear nature of the general case remains as a great challenge.

**Example: End-to-End System for Accomplishing Tasks with Modular Robots** In (99), an end-to-end approach to task-shape matching is presented. The paper features a system for accomplishing complex tasks with MRR. The system is composed of a high-level mission planner, a large robot connectivity-behavior library, a design and simulation tool for populating the library with new connectivity and behaviors, and MRR hardware (SMORES, see Figure 5). At the highest level, a target task is specified in an abstract manner, in terms of the task environment and desired behavioral properties: for example, “if the robot is moving in a tunnel, maintain a maximum height of 3 units.” The high-level mission planner of the system then selects robot connectivities and behaviors fulfilling all the requested functionalities, from the library. Finally, the specified mission is compiled and sent to the robot hardware to complete the task. Figure 7 shows a wide range of robot connectivities synthesized by the system, which were applied to the demonstrated challenge task, cleaning the top of a table. A high-fidelity physics simulator and a more exhaustive connectivity-behavior library (as presented in (100)) can further improve the toolchain.

![Figure 7](image_url)

*Figure 7*  
A set of robot configurations assembled with SMORES (9) needed to fulfill the user-specified task—cleaning the top of a table. For example, the leftmost panel shows a snake-shaped connectivity to climb up the table (99).
4.2. Reconfiguration Planning and Control

A key ability of MRR systems that differentiates them from normal robot systems is the determination and execution of a course of action that includes connectivity changes to switch from a starting configuration to another desired one. This problem is referred to as reconfiguration planning and control.

Although it may be possible to regard the problem as a variant of the general robot motion planning and control problem, which has been studied in robotics for many decades (101, 102, 103, 104), the problem is distinct from the traditional approaches in that the connectivity of the modules (or the topology of the robotic structure) also changes as needed. Challenges here include the possibly massive number of modules and their DOF, which will necessitate analyzing data in high-dimensional spaces (often referred to as “the curse of dimensionality”). Standard motion planning techniques that work well with a handful of dimensions and have guarantees of finding a solution can fail with an excessive number of DOF whose configuration space can be very complex topologically.

The main advantage of the lattice reconfiguration system over other types of architectures is that planning and control can be simplified through discretization. Both the connectivity of the system and the control actions that each module can take can be represented as a discrete set. Reconfiguration planning is then reduced to searching for a sequence of discrete motions in which the modules move to the adjacent lattice cells on the surface of the robotic structure. Collision detection, which is typically the most time-consuming part of motion planning, is also facilitated with the simplification that modules nominally sit on lattice positions, and changes in module position occur along discretized paths, by successively moving into adjacent lattice cells. The local nature of collision detection also paves the way to formulate motion planning in a distributed fashion. For those who are interested more in the computational aspects of reconfigurable robotics, the lattice reconfiguration architectures thus tend to be the most popular platform.

The reconfiguration of the chain architecture systems features changes in the topology of the linkage, for example, between an open and closed kinematic chains. There is a wide range of fundamental issues regarding the process, for example, determining what types of linkages have connected connectivity spaces such that reconfiguration can be possible between any two feasible connectivities. The reader is referred to (105) for more history and details of the research area. If it turns out that two configurations can reach each other, the next task is to find a collision-free path connecting them. Such a query can be addressed by modern robot motion planning software (for example, MoveIt! (106)), but it can be computationally expensive when it comes to highly redundant kinematic chains with complex topology featuring multiple holes, which can commonly be imagined in realistic scenarios. The high computational cost stems from the fact that detecting collisions in linkages is a global issue in which all the DOF of the system needs to be taken into account. One approach to lowering the cost is to freeze the mobility of a subset of modules, for the reduction of the dimensionality of the search space.

In order for the computed reconfiguration plans to be physically executable, the static/dynamic constraints of the robot and the environment need to be taken into account (kinodynamic motion planning is the term referring to this approach). For example, in the chain architecture systems, the payload of a module is always upper bounded, so there is a limit on the number of modules that it can cantilever. What makes kinodynamic planning for the chain-type systems more challenging is that the inertial properties of the system keeps changing each time a reconfiguration occurs.
Generally, reconfiguration can be performed more efficiently via a common intermediate configuration, sometimes called a canonical configuration (107). One interesting research question here is to find an optimal canonical configuration in terms of, for example, minimizing the sum of the control efforts of all the modules. In the chain architecture systems, an obvious choice for the canonical configuration can be a simple open kinematic chain (107). It will be a good research direction to further investigate what advantages there can be and implement it with real hardware systems.

In the case studies below, we will take a look at the practices of reconfiguration in each architecture.

**Example: Reconfiguration of the Lattice-Type Systems** The problem studied in (108) is concerned with algorithmic issues regarding decentralized construction with a bipartite modular system composed of passive blocks to embody the structure and active robots to carry the blocks on the surface of the structure (Figure 8). Planning for connectivity changes is performed at block level (the blocks are assumed to have some computing power). The blocks share a common coordinate frame and the description of a desired structure. They then determine at which of their faces additional blocks should be allowed to attach according to the local-scale rules for growing the structure in a greedy manner and preventing any dead-end sites that cannot physically be reached by the blocks. Motion planning and control strategies for the mobile robots include random movement in which the robots move arbitrarily on the surface of the growing structure and a more systematic approach featuring, for example, gradient descent following a numerical gradient signaled by the blocks. In these approaches, there is a trade-off between the cost of communication and robot control effort.

Figure 9 shows a lattice reconfiguration system presented in (109), named 3D M-Blocks. The system is capable of dynamic reconfiguration. Each module is able to exert both forward and backward torques, generated by the momentum of the flywheel inside, about three orthogonal axes, which enables it to pivot around an adjacent module bonded magnetically.

---

**Figure 8**
Decentralized construction of an hourglass-shaped structure (left to right, top to bottom).

**Figure 9**
Two M-Block modules, which are able to move dynamically relative to each other (109).

### Example: Self-Assembly of Modular Robotic Boats
The self-assembly of large teams of autonomous modular robotic boats was presented in (5). See Figure 10. The cuboid-shaped boats are designed to dock only along their long sides and form a regular pattern.
that looks like the common brick wall. Each robot is capable of holonomic locomotion on water. The robots can thus be considered as a mobile reconfiguration system. The connectivity planning algorithms presented in (110, 111) parse a blueprint for a target structure and return an assembly sequence that specifies an order to assemble the structure in $O(m)$ time, where $m$ is the size of the target structure (that is, the number of the dock sites of the target structure). The resulting plan facilitates motion planning for the modules in the sense that an open dock site is not flanked by two other modules already assembled in the structure, which will form a narrow gap that can essentially block off the open site. In addition, if a target structure covers a simply connected area with no hole (as in Figure 10), it is possible to perform module assembly in a parallel, distributed manner.

Based on the outcome of the connectivity portion of reconfiguration planning, the motion planning scenario featured in (5) is concerned with a large pool of available modular robotic boats and open docking sites around the perimeter of the growing structure. This task necessitates a highly scalable motion planner that can handle the plurality of the mobile robots and the candidate docking positions, which may change frequently as the robots dock to the structure in a decentralized manner. The trajectory planning algorithm featured in (5) builds on Dijkstra’s algorithm (112) and runs in polynomial time—$O(n^3)$ time where $n$ is the number of the modules.

**Figure 10**
(a) A floating quadrotor landing pad assembled with six boats. (b) The target structure schematic. (c) Two different assembly plans for the target structure, represented as directed acyclic graphs. For example, the plan on the left begins by occupying site 3 (5).

**Example: Planning and Control for Chain Reconfiguration System** In (113), a planner for purely kinematic chain reconfiguration was presented. This work features a connectivity planner utilizing the canonical configuration approach (107) and a motion planner using a sampling-based motion planning approach.

There are other approaches that represent chain systems as trees (acyclic graphs) for connectivity planning without going through a canonical configurations (114, 115). These approaches can be distributed and make some assumptions about non-isomorphism of serial chains.

(116) presents a kinodynamic motion planning approach applicable to MRR systems that combines a sampling-based motion planner and a physics-based simulator. The method is demonstrated with the topology of an open kinematic chain. The problem of more general kinodynamic reconfiguration involved with the change of module connectivity still remains open.
Example: Reconfiguration of VTT  A unique aspect of the truss systems is the constraints on the types of topologies that can be obtained. The reconfigurability of a VTT can be described using the topology neighbor graph of the VTT (10). Each node in the topology neighbor graph represents a unique truss connectivity. Two nodes are connected if a single topological reconfiguration step takes a configuration in the first connectivity to a configuration in the second connectivity. Admissible connectivities and topological reconfiguration steps are determined by the constraints of the physical system. For example, the VTT system requires statically determinate structures (more precisely infinitesimally rigidity) (10). Merging and splitting of nodes is allowed as each joint between two members are partial; however, merging of edges in a graph of the connectivity is not allowed as they are made up of truss members that cannot share the same space. It is interesting to note that given the physical constraints for the VTT system a minimum of 18 members are required before topological reconfiguration can occur. In fact, Figure 11 shows the neighbor graph that indicates the 19 non-isomorphic connectivities and the reachability of each. As more members are added to a system the number of non-isomorphic connectivities grows exponentially, a 24 member system has over 10,000, and listing of these connectivities in a brute force manner becomes intractable.

5. FUTURE DIRECTIONS

While there are many examples and existence proof demonstrations that modular reconfigurable robots with many (dozens) of modules can perform simple locomotion and manipulation tasks and self-reconfigure, there are many directions for the near and far future.

Modular reconfigurable robot systems have promised to be versatile, robust and low cost. The state of the art is far from fulfilling those promises. When we say versatile the implication is that the variety of tasks performed by the robot will actually be useful tasks. To be useful, we may need to fulfill all of these promises simultaneously.

Scaling Numbers While early computational works focused on large numbers of modules in simulation, in practice there have rarely been physical systems with more than several dozen modules. Systems with a thousand modules have been shown (68) which showed that practical issues such as how do you program so many modules, and how do you recharge their batteries need to be addressed. However, those systems had loosely cou-
pled mobile elements moving in a plane as opposed to rigidly coupled serial chains or 3D lattice structures which might be more useful. At this stage, the important future question may not be what is the largest number of modules, but instead, what is the right number of modules to be useful for a set of tasks.

Referring back to the robustness element of the three promises, a larger number of modules increases the probability of failure of individual components. If the modules are tightly coupled and dependent on each other, the likelihood of failure drastically increases, so there is a stronger incentive to optimize this number.

**Scaling Size** The early work of Fukuda (4) established the dream of miniature robots that are injected into a human body to self-assemble and repair a human heart. While miniaturized robots moving through a body is being developed and even some self-assembly (117), the practical usefulness of miniature MRR systems is still in question. While self-assembly has been demonstrated, self-reconfiguration of repeated modular elements in the MRR style is less clear.

On the other side, scaling up the size again to fit a given task is an interesting question. This relates again to finding the optimal number of modules. The truss architectures seem promising in this respect as the size is adjustable without increasing the number of modules.

**Scalable Computation** Task-level computation is the highest level of computational problem. Ultimately, exploiting the self-reconfiguration capability in an autonomous system will require the understanding of tasks and the environment in which tasks are to be done. While work has begun on the task-shape matching problem, developing a better formalism for characterizing tasks at a low level (more general) will enable the researchers to develop algorithms to match capabilities to subtasks and mechanisms for compositing subtasks together to achieve complex behaviors.

Nearer term, reconfiguration planning is a lower level computational problem. While a variety of approaches for lattice systems have been shown to be effective, especially those that have ideal the characteristics of being complete as well as distributed we have yet to see demonstrations of these algorithms on a physical system doing something useful. This may require the use of hybrid MRR systems.

The truss reconfiguration robot systems promise to have a hardware structure that solves many of the usefulness problems of earlier systems (that of structural strength, number of modules and scalability of size) but presents many significant hardware and software design complexities. Future applications and requirements may drive the development of MRR systems as well, such as the needs for compact and multi-functional systems required for space travel. For extended stay extra-terrestrial habitation, self-repair is another requirement that will be important which is well suited for MRR systems but has yet to be proven.

**DISCLOSURE STATEMENT**
The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

**LITERATURE CITED**
514–520 pp.


60. Lyder A, Garcia RFM, Stoy K. 2010. Genderless connection mechanism for modular robots


80. Fu G, Menciassi A, Dario P. 2011. Development of a genderless and fail-safe connection sys-