Reconfiguration Planning for a Self-Assembling Modular Robot

Eiichi Yoshida, Satoshi Murata, Akiya Kamimura,
Kohji Tomita, Haruhisa Kurokawa and Shigeru Kokaji
National Institute of Advanced Industrial Science and Technology,
1-2 Namiki, Tsukuba-shi, Ibaraki 305-8564 Japan
eiichi@mel.go.jp

Abstract

This paper addresses motion planning of a homogeneous modular robotic system. The modules have self-reconfiguration capability so that a group of the modules can construct various robotic structures. Motion planning for self-reconfiguration is a kind of computationally difficult problem because of many combinatorial possibilities of modular configuration against the restricted degrees of freedom of the module; only two rotation axes per module. We will show a motion planning method for a class of a multi-module structure, based on global planning and local motion scheme selection that is effective to solve the complicated planning problem. The fundamental motion will also be demonstrated through hardware experiments.

1 Introduction

Reconfigurable robotic systems have been attracting more interest, as their feasibility has been examined through hardware and software experiments in recent years [1]–[8]. This paper focuses on self-reconfigurable and homogeneous modular robotic systems that can adapt themselves to the external environment by changing their configuration. They can also repair themselves by using spare modules without external help owing to homogeneous modular structure. They have various potential applications, especially for structures or robots that should operate in extreme environments inaccessible to humans, for instance, in space or deep sea, or in nuclear plants.

Hardware of reconfigurable modular robotic system is classified into two types, lattice type [1]–[3] and linear type [4]–[8]. The former corresponds to a system where each module has several fixed connection directions, and a group of them can construct various static structures like jungle-gym. However, it is difficult for such a system to generate some dynamic robotic motions. In contrast, the latter has snake-like shape that can generate various dynamic motions, nevertheless self-reconfiguration is difficult.

As to software of reconfigurable modular robots, there have been a number of studies on lattice-type modular robots. We have developed a series of distributed self-reconfiguration methods for two-dimensional and three-dimensional homogeneous modular robots [9, 10]. These methods enabled homogeneous modular robots to self-assemble and self-repair in a distributed manner based on local inter-module interactions. In contrast, most of other methods are based on centralized planning. Kotay et. al [11] developed robotic modules described a global motion synthesis method for a class of module group to move in arbitrary directions. Ünsal et. al [12] reported two-level motion planners for a bipartite module composed of cubes and links, based on heuristic graph search between module configurations. These methods are dedicated to modules that have sufficient degrees of freedom to move to desired neighboring lattice position.

We have recently developed a new type of modular robotic system that can realize both static structure and dynamic robotic motion [13]. Although this recent module can form various shapes such as a legged walking robot or a crawler-type robot, its motion planning is not straightforward because of restricted degrees of freedom and non-isotropic spatial property of movability of a module unlike lattice-type modules in other previous research. When a module moves from one position to another, some combined motions of other modules are usually required. The necessary motion combination should be duly planned for each particular local configuration.

We propose a two-layered motion planning method, a global and local planners to transfer a class of module cluster along a desired trajectory. The former part of the planner provides the *flow* of the cluster, which corresponds to a global movement. The latter generates local coordinated motions called *motion schemes* based on a rule database. The rules consider the non-isotropic spatial property of module movability modules by associating appropriate pre-planned motion schemes with various local configurations. This method is classified into a centralized method. In the experiment section of the paper, fundamental reconfiguration motions are demonstrated.

2 Hardware Design and Module Model

2.1 Hardware Overview

The developed module consists of two semi-cylindrical parts connected by a link (Fig. 1). Servomotors are embedded in the link so that each of parts can rotate by 180°. Figure 2a shows a hardware prototype of the module. Each module has six connecting surfaces (three for each part)

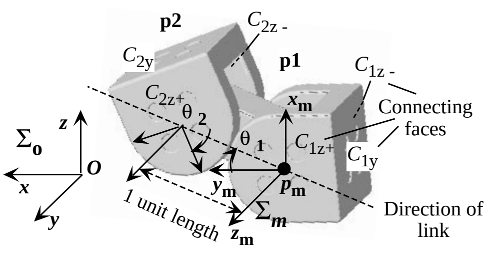
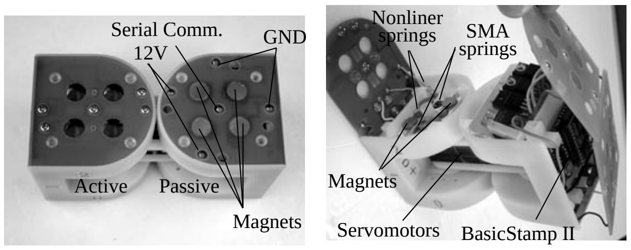


Fig. 1: A robotic module.



(a) Hardware overview (b) Inside structure.

Fig. 2: The hardware module.

that can attach and detach other modules by using magnets and shape memory alloy (SMA) actuator [13]. Figure 2b shows the inside structure of the module. The connecting surfaces have also electrodes for power supply and serial communication. All the connected modules can be supplied power from one module connecting to the power source. This eliminates the tether entanglement that becomes significant in three-dimensional configuration.

Each module is equipped with a PIC microprocessor that drives servomotors and SMA actuators. In the current development, all the modules are controlled from a host PC that provides motion commands through serial communication lines. The size of one semi-cylindrical part is 6cm cube and a module weighs approximately 400g. The SMA actuator is controlled by 200Hz PWM with 60% duty ratio, 12V voltage and average current 2A.

2.2 Model description

Each semi-cylindrical part of a module is distinguished as p1 and p2 in the model description. The position and orientation of the module m are uniquely determined by specifying the position and orientation of one part and the rotation angles of the servomotors. Let Σ_m be the local coordinate system fixed on the part p1 of module m as shown in Fig. 1, where z_m is the rotation axis and y_m is the direction from the part center to the arc top. The axis x_m is uniquely determined by z_m and y_m. Let Σ_0 be the absolute coordinate system here. We describe the position and orientation of module m as follows:

- the central position $p_m(x, y, z)$ of p1 in terms of Σ_0,
- the orientation of basis vectors z_m and y_m of Σ_m in terms of Σ_0,
- the rotation angles of each part (θ_1, θ_2).

In the following, we assume that both parts of modules move only on orthogonal-lattice grid and that the rotation angles (θ_1, θ_2) are limited to 0° or ±90° for simplicity. A unit length of the lattice grid is defined as the length between the two rotational axes of a module. A module therefore occupies two adjacent points in the grid.

We also denote the connection faces as C_{iz+}, C_{iz-} and C_{iy} (i = 1, 2 for p1, p2) according to Σ_m. The state of a connecting face, S(face), takes either of the following:

`T(ID)`	Connecting to module ID
`T(*)`	Connecting to a module but ID not specified
`F`	No module connected

The connection state of a module is written as $[S(C_{1z+}), S(C_{1z-}), S(C_{1y})], [S(C_{2z+}), S(C_{2z-}), S(C_{2y})]$.

For example, Fig. 3 shows a configuration of two modules (initial state of two-module motion explained later in Fig. 7) which is described as follows.

ID 1 $p_m(-1, 0, 0)$ $z_m(0, 1, 0)$ $y_m(0, 0, 1)$
$(\theta_1, \theta_2) = (-90°, 0°)$,
connection state: $[\texttt{F}, \texttt{F}, \texttt{T(*)}], [\texttt{T(2)}, \texttt{F}, \texttt{F}]$

ID 2 $p_m(-2, 1, 0)$ $z_m(0, 0, 1)$ $y_m(0, 1, 0)$
$(\theta_1, \theta_2) = (0°, 0°)$
connection state: $[\texttt{F}, \texttt{T(*)}, \texttt{T(1)}], [\texttt{F}, \texttt{T(*)}, \texttt{F}]$

2.3 Motion description

When a module makes a motion, one of the parts should be attached to another module to keep the connectivity. We call this fixed part a *base part*. Module motion is described using module IDs, base parts, rotation angles and the number of carried modules and their IDs if any. A *motion sequence* is a series of these motions.

3 Reconfiguration Motion Planning

The goal of planning is to let a class of a module cluster trace a certain given three-dimensional trajectory in the lattice grid (Fig. 4). This allows the module cluster to move into narrow space or to go over the obstacle. The planner should generate appropriate motion sequence so that the cluster motion can be guided along the desired trajectory.

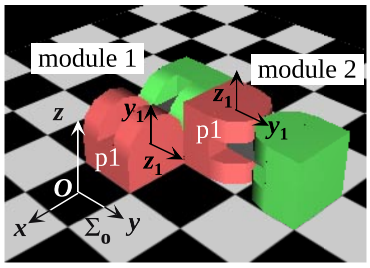


Fig. 3: Description of initial configuration in Fig.7.

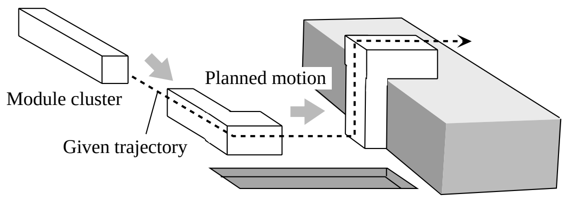


Fig. 4: Planning of cluster motion.

3.1 Planner architecture

The module's non-isotropic geometrical property makes it difficult to obtain the motion sequence straightforwardly. Since a module has only two parallel rotation axes, its three-dimensional motion usually requires a combined coordinated motion sequence of other surrounding modules. This kind of coordinated motion sequence must be carefully chosen in each case of particular local configuration.

We deal with the motion planning to a particular class of module cluster as a basic case. To cope with the complexity of the planning problem, we take a two-layered approach, namely *global flow planner* and *local motion scheme selector*. The global flow planner searches possible module paths to provide the global cluster movement, called *flow*, according to the desired trajectory. This is realized as a motion of a module group, a *block*, such that the tail block is transferred toward the given heading direction. The local motion scheme selector verifies if the paths generated by the global planner are valid for each *member* module of the block based on rule database. If a given path from the global planner turns out to be valid, the selector updates the motion plan by adding a set of local motion sequences called *motion schemes*. Otherwise it tries another possible module path generated by the global planner. Note that this is a centralized planning method assuming that all the information of modules in the cluster is available.

In the following, we suppose that only one motion scheme is allowed during the cluster flow for simplicity. Another assumption is that one module can lift only one other module in the planning, which comes from the limited torque capacity of the hardware.

3.2 Atomic Motion

There are mainly three types of atomic motion, *pivot mode*, *forward-roll mode* and *mode conversion*. Figures 5 and 6 show two different atomic motions on a plane, forward-roll and pivot, whose orientation of rotational axes are in different direction. Mode conversion is a two-module motion to convert from one mode to the other, where a helper module is required as illustrated in Fig. 7. By combining modules in both forward-roll mode and pivot mode, a variety of three-dimensional structures are possible.

3.3 Cluster Flow and Global Planner

As a basic case of motion planning, we consider a class of module cluster mainly composed of two layer of pivot mode modules (Fig. 8). The cluster also includes a couple of forward-roll mode modules called converters to change the orientation of the rotation axes of the pivot mode modules. The connectivity condition of the whole cluster is satisfied by placing the modules so that the directions of y-axis are different in each layer.

The input to the global planner is the desired trajectory of

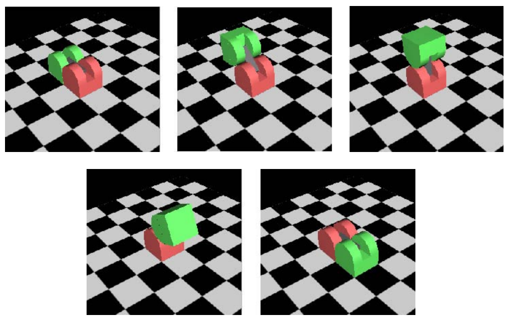


Fig. 5: Forward-roll mode.

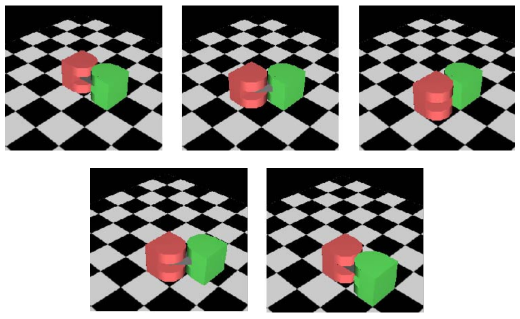


Fig. 6: Pivot mode.

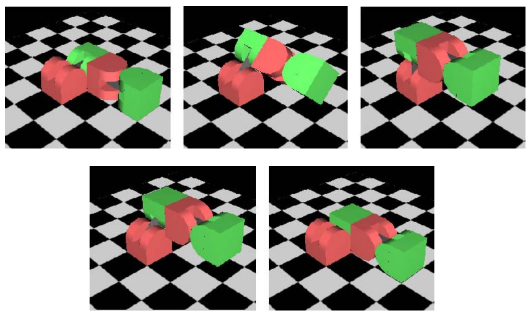


Fig. 7: Mode conversion from pivot to forward-roll.

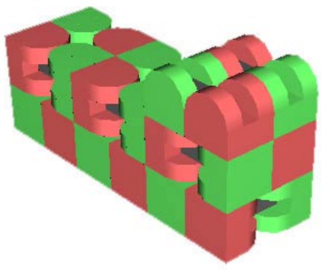


Fig. 8: A cluster composed of two layers of pivot mode modules with two converter modules.

the cluster. The cluster *flow* is defined as the trace of block motion, where the *tail* block is removed and put at the other end as the new *head* (Fig. 9). A *path* denotes a rough routing of a member module of a block to move from the head position to the tail. The *global flow planner* is in charge of generating possible paths of modules to realize the desired flow. While there are several ways of generating reconfiguration motion of this kind of cluster, we adopt a simple conveyer-like motion to realize the desired flow. The tail modules move toward the heads by using forward-roll and some coordinated atomic motions on the side of the cluster (Fig. 9). They become new heads when they reach the other end of the cluster. The next tails will be sent to the heads, and so forth. The paths are derived by tracing lattice positions on the side of the cluster, starting from the initial position until the module reaches one of target positions next to the current head block.

3.4 Motion Scheme Selector

After the global planner outputs possible module paths, appropriate *motion schemes* should be selected to achieve the paths, considering connectivity condition and collision avoidance. The *motion scheme selector* does this job based on a database of rules for local coordinated motion.

The selector verifies the validity of possible paths given by the global planner for each member module by applying the rules in the database. This verification is done in increasing order of traveling distance. Namely, the path with the shortest length is first tried, next the second shortest, and so on. Each rule includes a motion scheme associated with an initial configuration that is described as a connectivity graph (Fig. 10a).

More precisely, a rule in the database is composed of a `if`-condition part and a `then`-action part. The former is a connectivity graph that describes a local connection state to

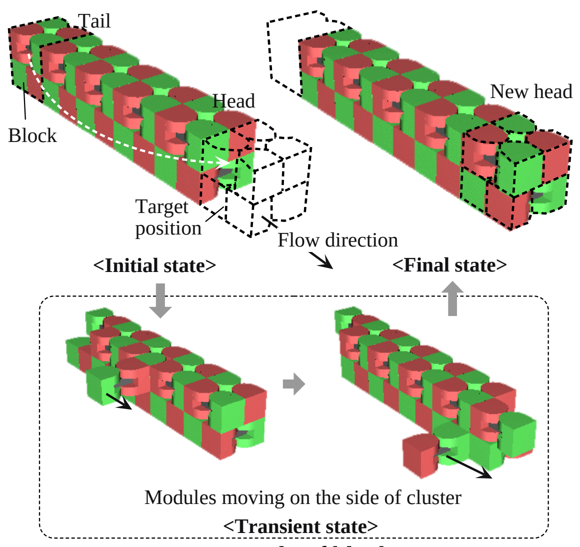


Fig. 9: Example of block motion.

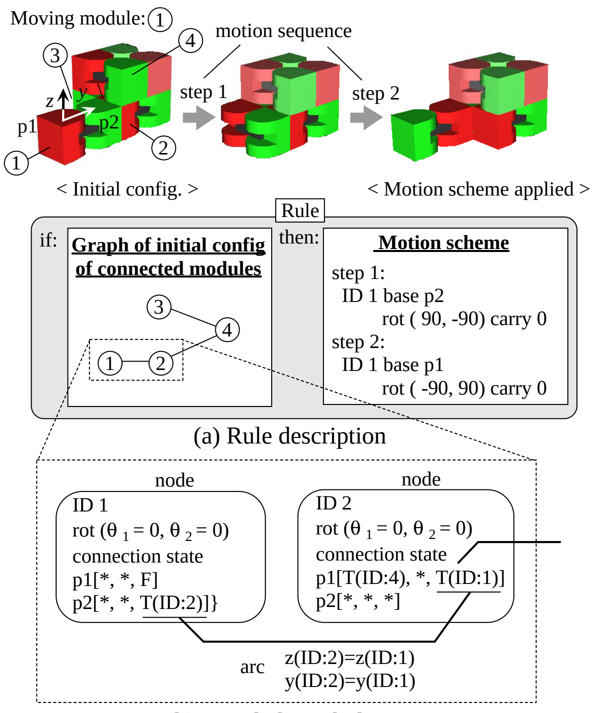


Fig. 10: Example of a rule for a rolling motion scheme

be matched to the current local configuration of the moving module. The latter corresponds to a motion scheme written in the form of motion sequence. Figure 10b illustrates the graph description of local configuration. In the connectivity graph a node is assigned to each module. The node includes such data as a temporary ID number, rotation angles and the states of the six connecting faces. To make the rules applicable to various cases, we introduce a wild card state "*" (don't care) that matches all the states. An arc in the connectivity graph denotes the connection to other modules and specifies the relative direction of z and y axes of connecting module m, such as $[(z(m), y(m)]$.

Among the rules that matches the current local configuration, a motion scheme that is valid and gives the largest forward movement is selected. The validity check is performed from two aspects, collision avoidance and connectivity of total cluster. By applying the motion scheme to the moving module, collision can be detected by calculating the sweeping area of its motions. Similarly, the connectivity is examined during the motion by tracing the current connectivity graph from the moving module down to connected modules. The motion scheme of the selected rule is stored in a temporary motion sequence. If all the motions of the member modules are correctly determined, the planner updates the motion plan by appending the temporary output sequence to it. Otherwise, the selector tries next possibilities of paths.

In order to implement the motion scheme selector, we

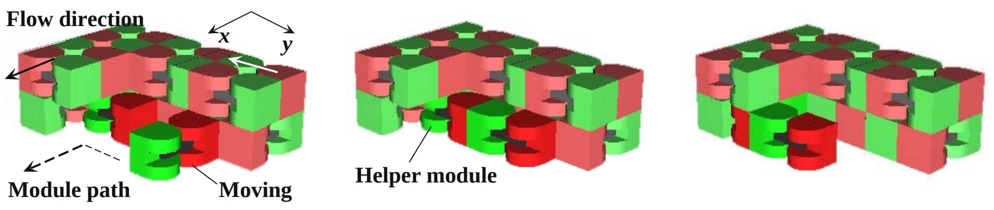


Fig. 11: Direction change of cluster on a plane.

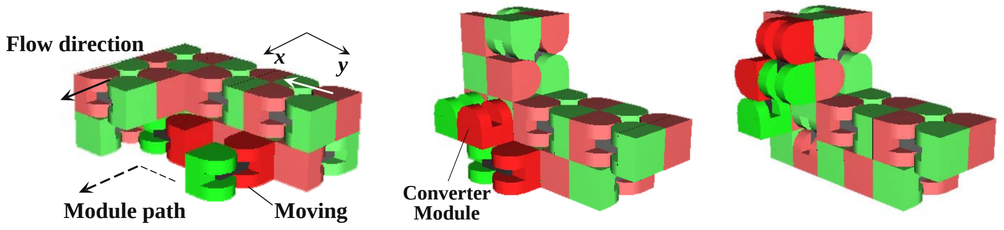


Fig. 12: Direction change to vertical direction on a plane.

extracted several fundamental motion schemes as follows.

(1) rolling on a side of the cluster (Fig. 10)
(2) carrying a module by right-angle on a plane (Fig. 11)
(3) converting the rotational axis of a module (Fig. 12)

Figure 10 shows a rule corresponding to a simple motion scheme of the rolling on the side of the cluster. Figures 11 and 12 illustrate how module configuration changes in the latter two motion schemes. Supposing that the initial module cluster are located on x-y plane, the direction change between x-y axis is done by alternating the layers (Fig. 11). The converter modules are used when the desired cluster flow requires change of rotational axis of the module (Fig. 12). The number of converter modules can be augmented if necessary.

3.5 Planning Results

The motion planner can generate simple three-dimensional paths for various sizes of clusters. There are approximately thirty rules in the current development.

Figure 13 show some snapshots taken from planned motion of a cluster of 22 modules starting from a configuration on a plane. The cluster first changes its flow direction on the horizontal plane, then moves in a vertical direction.

Although the currently developed method applies to only a particular class, We believe that the basic framework of the two-layered approach is generally effective for other classes. For the global planner, we need to devise a method to narrow the search space according to the problem. On the other hand, the motion scheme selector is less problem-dependent owing to its locality. However, the rule database should be refined to be more complete by adding more rules since classes for the database are currently limited. We are thinking of extending the database based on some evolutionary methods, and also generating more complex rules including rule hierarchy.

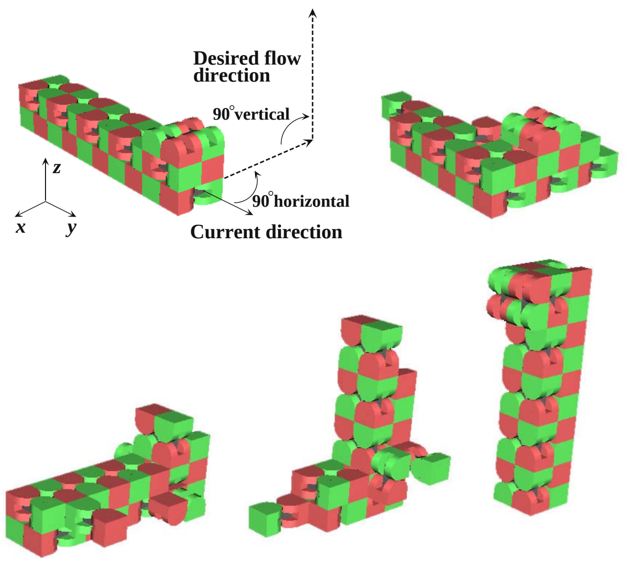


Fig. 13: Simulated plan of motions in different flow directions from initial configuration on a plane.

Simultaneous motion of several modules is another issue to be addressed. We intend to reduce the time required for reconfiguration by merging multiple module paths.

4 Hardware Experiments

We are building hardware prototype of robotic modules. Although the planned motion have not been fully implemented yet in the hardware, we will verify the fundamental motion capacity of hardware module.

Figure 14 and 15 show the experiment of forward-roll motion (Fig. 5) and mode conversion (Fig. 7). In these experiments, the connecting mechanism showed reliable performance; it has enough strength to hold the module against

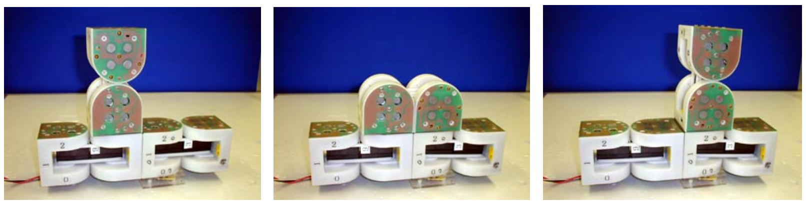


Fig. 14: Experiment of forward-roll motion.

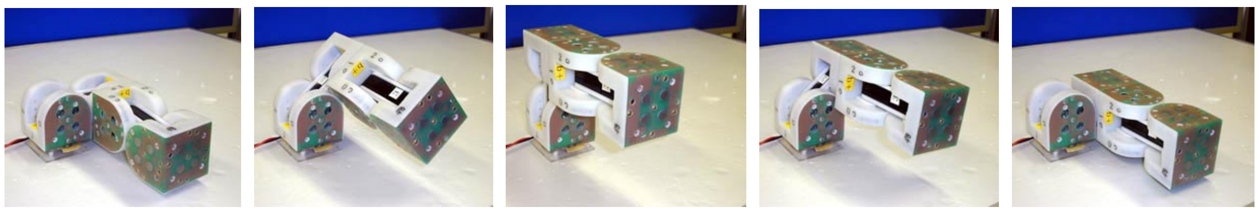


Fig. 15: Experiment of mode conversion.

gravity and the smooth detachment is realized as well. We can also verify the module has sufficient torque to conduct certain two-module motions from Fig. 15. By combining these basic module motions, various motions are possible.

5 Conclusions

This paper discussed motion planning of a self-reconfigurable modular robot. The module was designed to generate both static structure and dynamic robotic motions. This module can form various three-dimensional shapes in spite of simple design. On the other hand, it has non-isotropic property of module movability due to its limited degrees of freedom, which imposes a stronger constraint on motion planning compared to ordinary lattice-type modules. The motion planning should consider this constraint, and we adopt a planning method based on global flow planner and motion scheme selector. The former outputs global paths to realize overall cluster flow, and the latter selects valid motion schemes as combined atomic motions based on a rule database. The non-isotropic geometric property of module movability was properly reflected in the rules that associate appropriate pre-planned motions with corresponding local configurations. We also verified the basic functions of hardware modules through experiments.

The future work concerning the motion planner includes building a general global path-finding algorithm applicable to wider classes of configuration and investigating more complete rule description. We are also aiming to implement the motion planner to the hardware modules. By equipping modules with some external sensors, the module cluster can move around in unknown environments with bumps or walls, adapting its shape to the outside world.

References

[1] S. Murata, et al. : “A 3-D Self-Reconfigurable Structure,” *Proc. 1998 IEEE Int. Conf. on Robotics and Automation*, 432–439, 1998.

[2] K. Kotay, et al. : “The Self-Reconfiguring Robotic Molecule,” *Proc. 1998 IEEE Int. Conf. on Robotics and Automation*, 424–431, 1998.

[3] C. Ünsal, et al. : “I(CES)-cubes: a Modular Self-Reconfigurable Bipartite Robotic System,” *Proc. SPIE, Sensor Fusion and Decentralized Control in Robotic Systems II*, 246–257, 1999.

[4] T. Fukuda and S. Nakagawa: “Approach to the Dynamically Reconfigurable Robotic System,” *Journal of Intelligent and Robot Systems*, **1**, 55–72, 1988.

[5] E. Yoshida, et al. : “Miniaturization of Self-Reconfigurable Robotic System using Shape Memory Alloy,”, *J. of Robotics and Mechatronics*, **12**-2, 1579–1585, 2000.

[6] G. Hamlin and A. Sanderson: A Modular Approach to Reconfigurable Parallel Robotics, Kluwer Academic Publishers, Boston, 1998.

[7] A. Casal and M. Yim: “Self-Reconfiguration Planning for a Class of Modular Robots,” *Proc. SPIE, Sensor Fusion and Decentralized Control in Robotic Systems II*, 246–257, 1999.

[8] A. Castano, et al. : “Autonomous and Self-Sufficient CONRO Modules for Reconfigurable Robots,” *Distributed Autonomous Robotics 4*, Springer, 155–164, 2000.

[9] E. Yoshida, et al. : “A Distributed Method for Reconfiguration of 3-D homogeneous structure,” *Advanced Robotics*, **13**-4, 363–380, 1999.

[10] K. Tomita, et al. : “Self-assembly and Self-Repair Method for Distributed Mechanical System,” *IEEE Trans. on Robotics and Automation*, **15**-6, 1035–1045, 1999.

[11] K. Kotay and D. Rus: “Motion Synthesis for the Self-Reconfigurable Molecule,” *Proc. 1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 843–851, 1998.

[12] C. Ünsal, et al. : “Motion Planning for a Modular Self-Reconfiguring Robotic System,” *Distributed Autonomous Robotics 4*, Springer, 165–175, 1999.

[13] S. Murata, et al. : “Hardware Design of Modular Robotic System,” *Proc. 2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, F-AIII-3-5, 2000.