Proceedings of 2001 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS2001), in press.

# A Motion Planning Method for a Self-Reconfigurable Modular Robot

Eiichi Yoshida, Satoshi Murata, Akiya Kamimura,
Kohji Tomita, Haruhisa Kurokawa and Shigeru Kokaji
Distributed Systems Design Research Group, Intelligent Systems Institute, AIST
Tsukuba East, 1-2-1 Namiki, Tsukuba-shi, Ibaraki 305-8564 Japan
e.yoshida@aist.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 a robotic structure. Motion planning for self-reconfiguration is a kind of computationally difficult problem because of many combinatorial possibilities of modular configuration and 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 multi-module structures. It is based on global planning and local motion scheme selection that is effective to solve the complicated planning problem.*

## 1 Introduction

In recent years, the feasibility of reconfigurable robotic systems has been examined through hardware and software experiments [1]–[8]. This paper focuses on a homogeneous self-reconfigurable modular robotic system that can adapt themselves to the external environment by changing their configuration. It can also repair itself by using spare modules without external help owing to homogeneity of the module. Its various potential applications include structures or robots that should operate in extreme environments inaccessible to humans, for instance, in space, deep sea, or 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 a fixed geometry of connections, and a group of them can construct various types of static crystalline lattices such as a jungle-gym. However, it is difficult for such a system to generate some dynamic robotic motions. On the other hand, the snake-like shape of the latter can generate dynamic motions, nevertheless self-reconfiguration is difficult for them.

On software side of reconfigurable modular robot, there have been a number of studies on lattice-type. We have also developed several distributed methods for two-dimensional and three-dimensional homogeneous modular robots [9, 10]. These methods enabled them to self-assemble and self-repair in a distributed manner using local inter-module communication. In contrast, most of other methods are based on centralized planning. For instance, Kotay et. al [11] developed a motion synthesis method for a class of module groups. 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 every neighboring lattice position.

Recently, we have developed a new type of modular robotic system that can realize both static structure and dynamic robotic motion [13]. This has been realized by simplified design of a module.

We have shown our recent module can form various shapes such as a legged walking robot or a crawler-type robot. However, its motion planning is not straightforward because of restricted degrees of freedom and non-isotropic spatial property of movability of a module. 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. Due to these restrictions of modules, it is difficult to find some generic law of motion planning unlike ordinary lattice-type modules.

In this paper, we propose a two-layered motion planning method for the purpose of generating a sequence of module motions that allows a class of module clusters to trace a desired trajectory. It consists of global flow planner and local motion scheme selector. The former outputs possible module paths to realize the overall cluster motion. The latter selects valid paths and comprises them by collecting appropriate locally coordinated module motions based on a rule database. These rules take account of restricted module movability by associating appropriate pre-planned motion schemes with various local configurations.

## 2 Hardware Overview and Module Model

Here, we give a brief overview of the hardware [13] and its model for motion planning.

### 2.1 Hardware Design

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 the parts can rotate by 180°. Each module has six connecting surfaces (three for

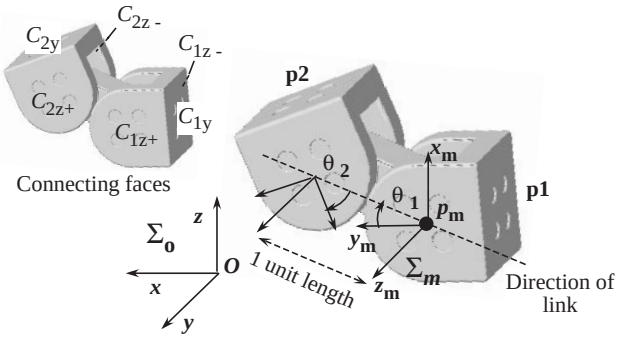



Fig. 1: A robotic module.

each part) that can attach and detach other modules by using magnets and shape memory alloy (SMA) actuator. 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. Each module is equipped with a PIC microprocessor that drives servomotors and SMA actuators.

## 2.2 Atomic Motion

Before introducing module model, it is helpful to explain the *atomic motions* that are the simplest module motions of one or two modules. There are three types of atomic motion, *pivot motion*, *forward-roll motion* and *mode conversion*. Figures 2 and 3 show two atomic motions on a floor

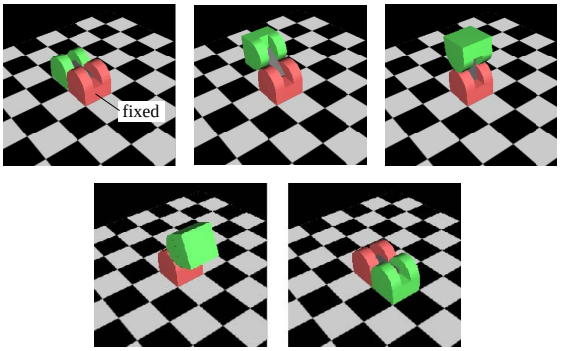



Fig. 2: Forward-roll motion.

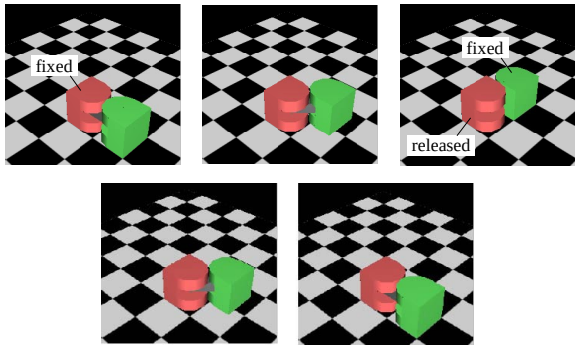



Fig. 3: Pivot motion.

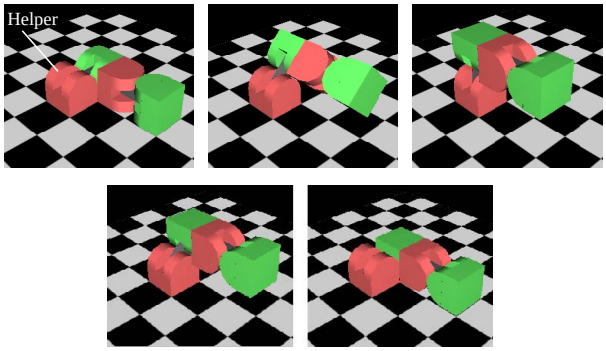



Fig. 4: Mode conversion from pivot to forward-roll.

tiled by the modules: forward-roll and pivot motions, the simplest basic motions where their orientation of rotational axes are in different directions. To generate these motions, the module should fix one of the two parts to other modules (in these cases the module-tiled floor), whereas the connection of the other part should be released. The fixed part alternates for the pivot motion as shown in Fig.3. The module is said to be in *pivot mode* or *forward-roll mode* if it can perform one of these motions. 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. 4. By combining modules in both forward-roll mode and pivot mode, a variety of three-dimensional structures are possible.

## 2.3 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 $\Sigma_m$ 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 $\Sigma_0$ 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 with respect to $\Sigma_0$,
- the orientation of basis vectors $z_m$ and $y_m$ of $\Sigma_m$ with respect to $\Sigma_0$,
- the rotation angles of each part $(\theta_1, \theta_2)$.

In the following, we assume that both parts of modules move only on orthogonal-lattice grid and that the rotation angles $(\theta_1, \theta_2)$ 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 $\Sigma_m$. The state of a

connecting face, $S(\text{face})$, 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. 5 shows the initial configuration of two modules shown in Fig. 4, 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: $[\mathtt{F}, \mathtt{F}, \mathtt{T(*)}], [\mathtt{T(2)}, \mathtt{F}, \mathtt{F}]$

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: $[\mathtt{F}, \mathtt{T(*)}, \mathtt{T(1)}], [\mathtt{F}, \mathtt{T(*)}, \mathtt{F}]$

### 2.4 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 collection of these motions. For instance, the motion sequence in Fig. 4 consists of four steps and is described as follows.

```
step 1
 ID 1 base p1 rot(90, 0)     carry 1 ID 2
 ID 2 base p1 rot(0, 0)      carry 0

step 2
 ID 1 base p1 rot(-90, 90)   carry 1 ID 2
 ID 2 base p1 rot(0, 0)      carry 0

step 3
 ID 1 base p1 rot(90, -90)   carry 1 ID 2
 ID 2 base p1 rot(0, 0)      carry 0

step 4
 ID 1 base p1 rot(-90, 0)    carry 1 ID 2
 ID 2 base p1 rot(0, 0)      carry 0
```

In step 1, there are two moving modules, and the module ID 1 rotates by ($\theta_1$ = 90°, $\theta_2$ = −0°) with base part p1 and carries another module ID 2, which does not make rotation, and so on.

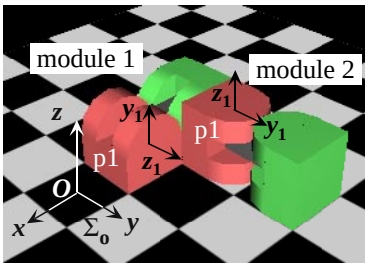



Fig. 5: Description of initial configuration in Fig.4.

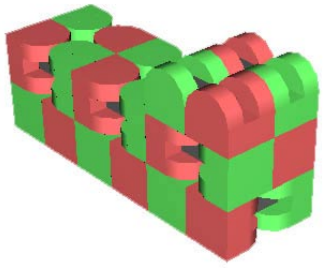


Fig. 6: A cluster composed of two layers of pivot mode modules with two converter modules.

## 3 Motion Planning Architecture

We deal with the motion planning for a particular class of module clusters (Fig. 6). The cluster is a chain of rectangular prisms composed of two layer of pivot mode modules where the lengths of the prisms are variable. A couple of forward-roll mode modules called converters are attached to a side of the cluster. The converter modules are used to change the rotation axes of the pivot modules. The connectivity condition of the whole cluster is satisfied by placing the modules of each layer so that the directions of $y_m$ axis are orthogonal.

The goal of planning is to let the cluster trace a certain given three-dimensional trajectory in the lattice grid correcponding to the center of mass of the four-module block (Fig. 7). For instance, the paths are given as a route that a plant inspection robot or a planetary explorer should trace. This allows the module cluster to move into narrow space or to climb over the obstacle. The planner should generate appropriate motion sequence that realizes the cluster motion guided along the desired trajectory.

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. If this motion sequence is not carefully planned, the resultant sequence may not be possible for many reasons such as inappropriate orientation of rotation axes, collision between modules, or loss of connectivity during the motion. This kind of coordinated motion sequence must be carefully chosen in each case of particular local configuration. For example in Fig. 8, the motion of module #1 to the target

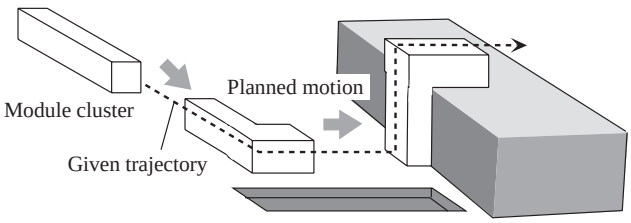



Fig. 7: Planning of cluster motion.

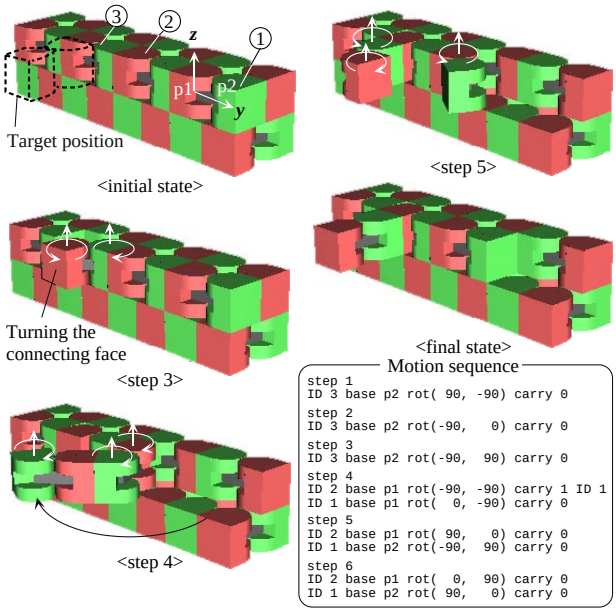



Fig. 8: A coordinated motion sequence by 3 modules.

position requires coordinated motions such as module #3's turning a connection face, or module #2's carrying module #1. We have not found any generally applicable law for planning these motion sequences, so some database of rules to look up is necessary.

In this paper, we take a two-layered approach to cope with the complexity of the planning problem. The upper layer decomposes the planning problem into subproblems solvable by the lower layer. The lower layer is designed to solve simple planning problems based on a database of rules for each local configuration.

More precisely, the upper and lower layer are called the *global flow planner* and the *local motion scheme selector* respectively. As shown in Fig. 9, the global flow planner searches possible module paths and motion orders 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 reconfiguration motion sequences called *motion schemes*. Otherwise it tries another possible module path generated by the global planner. The selector copes with the non-isotropic property of module movability by associating the coordinated motion with the corresponding local configuration. Note that this is a centralized planning method assuming that all the information of modules in the cluster is available.

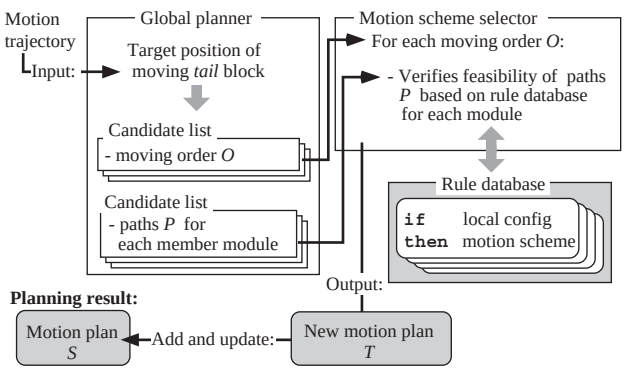



Fig. 9: Planner architecture.

In the following planning method, we give the following assumptions:

(1) One module can lift only one other module.
(2) Only one motion scheme is allowed at a time.
(3) At least two converter modules are assumed in the whole cluster.
(4) The flow direction should go straight at least by two unit lengths.

The first assumption comes from the limited torque capacity of the hardware. The remainders are introduced to simplify the planning problem.

## 4 Cluster Flow and Global Planner

The input to the global planner is the desired trajectory of 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*, as shown in Fig. 10. By one block motion, the head of the cluster moves by two unit length on the lattice grid. While there are several way of generating this kind of a block motion, we adopt simple

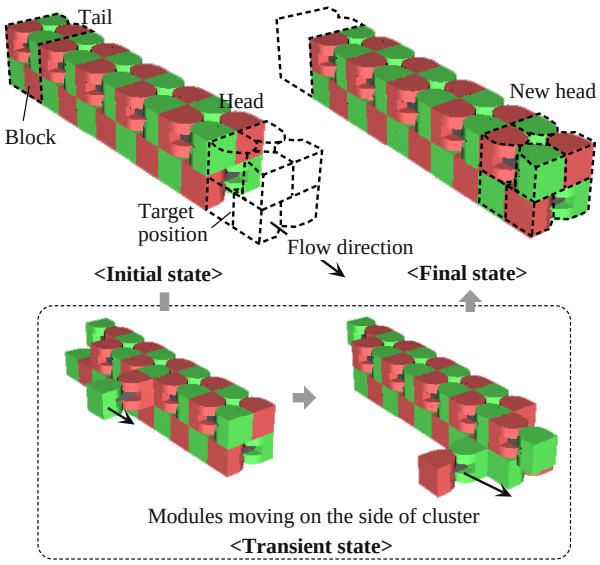



Fig. 10: Example of block motion.

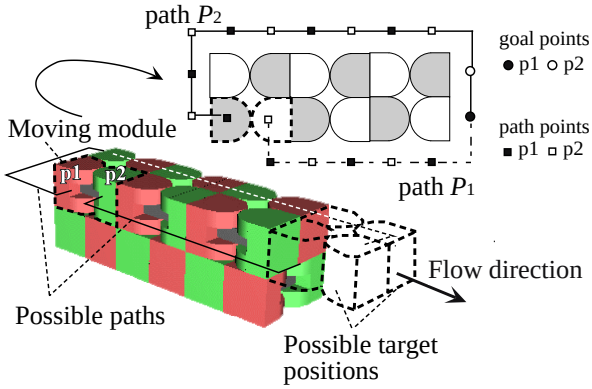



Fig. 11: Path of a module for block motion.

motion schemes sending modules one by one towards the head. The modules move using forward-roll and some coordinated support motions on the side of the cluster (Figs. 8 and 10).

The output of the global planner are the possible paths $\mathcal{P}_{mi}$ $(i = 0, 1, \ldots, N_P)$ for each member module $m$ in the tail block and its motion orders $\mathcal{O}_i$ $(i = 0, 1, \ldots, N_o)$, where $N_P$ and $N_o$ are the numbers of candidate paths and orders respectively.

The paths are derived by tracing lattice positions as $\mathcal{P}_{mi} = (p_{mi_0}, p_{mi_1}, \ldots, p_{mi_n})$ 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 (Fig. 11). A module may have multiple target positions and paths, and their number varies depending on the cluster configuration. After the tail block motion is completed, it becomes a new head block. Then the next tail will be sent to the head, and so forth.

The order of applying motions is defined as $\mathcal{O}_i$ $(o_{i1}, o_{i2}, o_{i3}, o_{i4})$ where $o_{ij}$ is ID number of member module. It should be decided in such a way that the connectivity of whole cluster is maintained. For instance, consecutive motions of the two modules in the upper layer of the tail block in Fig. 11 are not allowed because when the two upper modules are moved the connectivity condition of two lower modules is violated.

## 5 Motion Scheme Selector

Based on the output of the global planner, appropriate *motion schemes* should be selected to achieve the planned block motion, considering connectivity and collision avoidance. The *motion scheme selector* does this job using a database of rules. In the following, after outlining the selection procedure, we will detail rule description, matching, and validity check of module paths.

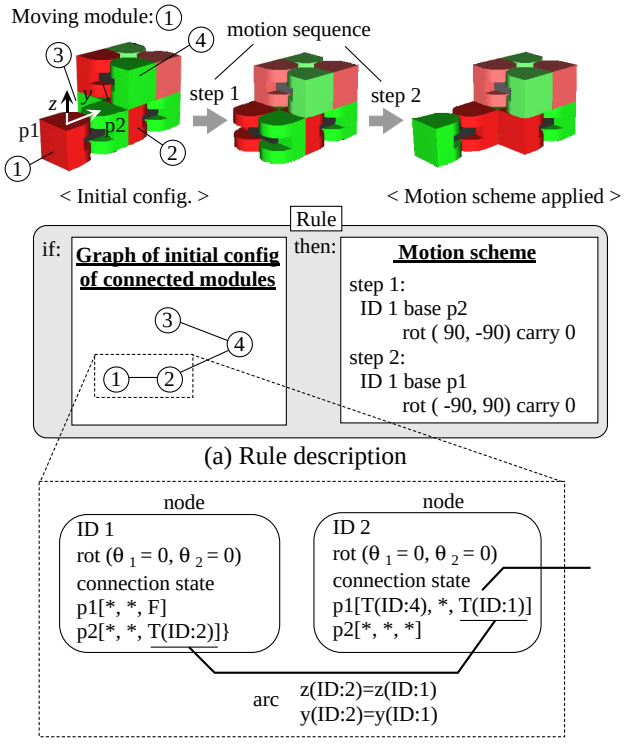
### 5.1 Selection procedure

According to the motion order $\mathcal{O}_i$ $(i = 0, 1, \ldots, N_o)$ given from the global planner, the selector verifies the validity of possible paths $\mathcal{P}_{mj}$ $(j = 0, 1, \ldots, N_P)$ of each member module $m$ in the block, 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. 12a). Among the rules that matches the current local configuration, a motion scheme that gives the largest forward movement is selected. The motion scheme of the selected rule is stored in the temporary motion sequence $\mathcal{T}$. If all the motions of the member modules are correctly determined, the planner updates the motion plan $\mathcal{S}$ by appending the output sequence $\mathcal{T}$ to it. Otherwise, the selector tries next possibilities of $\mathcal{P}$ or $\mathcal{O}$.

### 5.2 Rule description

A rule $R_k$ $(k = 1, 2, \ldots, N_R)$ in the database is composed of a `if`-condition part and a `then`-action part, where $N_R$ is the total number of rules. The former is a connectivity graph $G_k$ that describes a local connection state to be matched to the current local configuration of the moving module. The latter corresponds to a motion scheme $M_k$ written in the form of motion sequence.

Figure 12b illustrates the graph description of local configuration. In the connectivity graph $G_k$, 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




(b) Detailed graph description

Fig. 12: Example of a rule for a rolling motion scheme

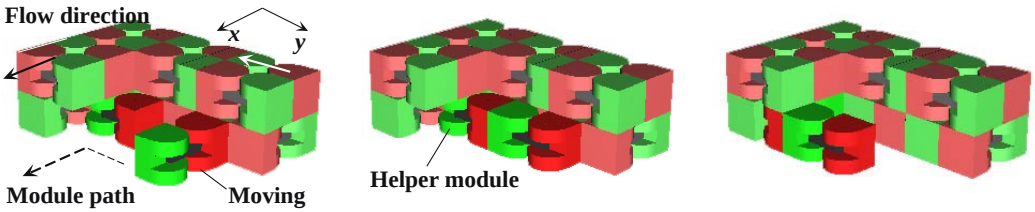



Fig. 13: Direction change of cluster flow on a plane.

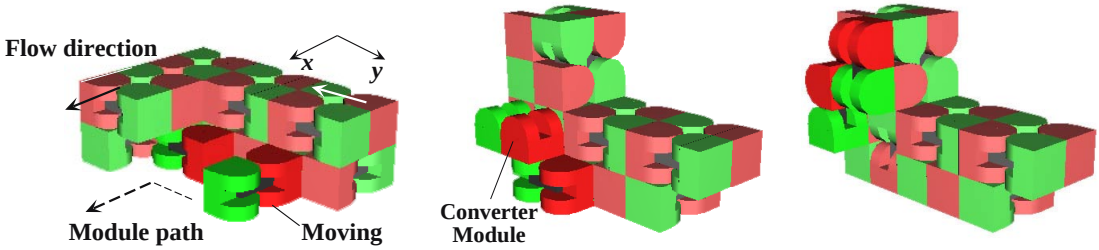



Fig. 14: Direction change of cluster flow to orthogonal direction to a plane.

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 $[(z(m), y(m)]$. The top node of a connectivity graph corresponds to the module to be matched to the current local configuration. In the rule database, sufficient number of rules are required to realize various module motions. These rules are currently hand-coded.

### 5.3 Rule matching

To find a motion scheme of a module $m$ for the given path $\mathcal{P}_{mj}$, the selector searches rules that matches the local configuration of $m$.

Let $G_m$ be the current connectivity graph of module $m$. Matching between $G_m$ and rule templates $G_k$ ($k = 1, 2, \ldots, N_R$) proceeds from the top node down to the connecting nodes. During the matching process, the connection states and rotation angles are compared for corresponding nodes, as well as the connecting directions in each arc. The graph matching succeeds if all the nodes and arcs turned out to be compatible. All the matching possibilities are tested for each rule, such as mirrored configurations and configurations where the parts p1 and p2 are swapped. The selector makes a list of all the matched rule $R_k$ to check the validity as described next.

### 5.4 Validity check of a module path

Suppose that $N_f$ rules were found that match to the current configuration $G_m$. For each rule $R_k$ ($k = 1, 2, \ldots, N_f$), the validity of associated motion scheme $M_k$ is checked. If there exist valid motion schemes, then the selector chooses the one that gives the largest forward movement for the path $\mathcal{P}_{mj}$.

The validity check is performed from two aspects, collision avoidance and connectivity of total cluster. By applying the motion scheme $M_k$ to the module $m$, collision can be detected by calculating the sweeping area of its motions. Similarly, the connectivity is examined during the motion by tracing the connected modules from module $m$ in the cluster.

When more than one rules are found valid, one of the motion schemes is selected based on some additional criteria, such as the maximum traveling distance along the path.

### 5.5 Typical motion schemes

In order to implement the motion scheme selector, we extracted several fundamental motion schemes as follows.

(1) rolling on a side of a cluster (Fig. 12)
(2) carrying a module by right-angle on a plane (Fig. 13)
(3) converting the rotational axis of a module (Fig. 14)

Figure 12 shows a rule corresponding to a simple motion scheme of the rolling on a side of the cluster. Figures 13 and 14 illustrate how a module configuration changes in the latter two motion schemes. Suppose that the initial cluster is put on $x$-$y$ plane, the direction change between $x$ and $y$ axes is done by alternating the layers (Fig. 13). The converter modules are used when the desired flow requires a change of the rotational axis of the module (Fig. 14).

Although the correctness of the planner has not been proven yet, we observed most of the cases were solved in simulations where the planner were applied to clusters composed of different number of robots. We believe that the proof can be done by checking necessary and sufficient rules are provided for the cluster flow. After the correct-

ness of the planner is shown, the planning of cluster flow can be reduced to a simple planning problem of module blocks which can be regarded as a “meta-module.”

### 5.6 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 15 shows 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.

## 6 Hardware Experiments

This section presents an example of cluster motion of block structure to show the planned motion can be achieved by hardware modules.

Figure 16 shows the experimental setup. Please refer to the related paper [14] for the details of hardware implementation. The motion is planned in the host PC and then converted into low-level control commands of servomotors and SMA actuators by the simulator software. These control commands are distributed to the microprocessor of appropriate modules through a serial bus line by way of electrodes on the connecting faces. The power is also supplied all the modules throughout inter-module connection from one module connected to the power source.

In this experiment, eight-module cluster flow motion is executed. The generated plan was modified so that some motions are made in parallel to reduce the execution time, whereas the original motion plan transfers the modules one by one on the side of the cluster. The total motion steps

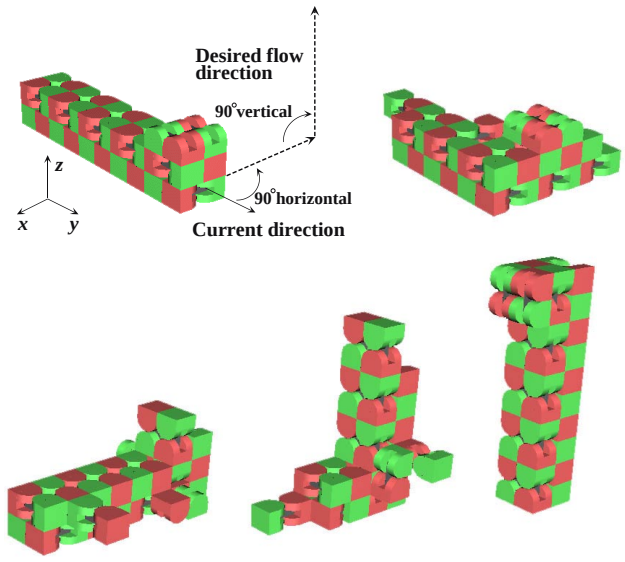



Fig. 15: Simulated plan of motions in different flow directions from initial configuration on a plane.

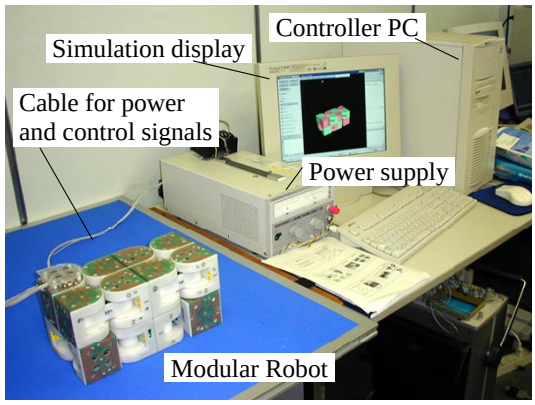



Fig. 16: Experimental setup.

were 23 steps. As shown in Fig. 17, the cluster motion has been achieved to demonstrate the validity of the planned motion.

## 7 Discussions

This section discusses the future direction of the planning method by addressing such issues as parallel module motions and generalization of applicable classes.

Let us begin with considering the assumptions given in Section 3. The assumption (2) of allowing only one module at a time is introduced for simplicity of the planning. However, a cluster of modules can perform parallel motions preserving connectivity and collision avoidance. We are now on the way to increase concurrency of motion by merging several concurrent motion sequences. To relax the latter two assumptions, more than two converter modules must be generated from block member modules. This may require more general global planning that is also related to the next issue.

Generalization of applicable classes of clusters is another important issue. Although the currently developed method applies to only a particular class, we believe that the basic framework of the two-layered approach is effective for other classes. For the global planner, we need to devise a method to narrow the search space according to the problem. We also intend to develop a global planner that can deal with wider classes of structure, using some powerful searching method such as genetic algorithm. 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. Practically, it is very difficult to construct such a large database by coding all the rules manually. Automatic rule acquisition will be required to archive this goal. We are thinking of extending the database based on some evolutionary methods, and also generating more complex rules including rule hierarchy.

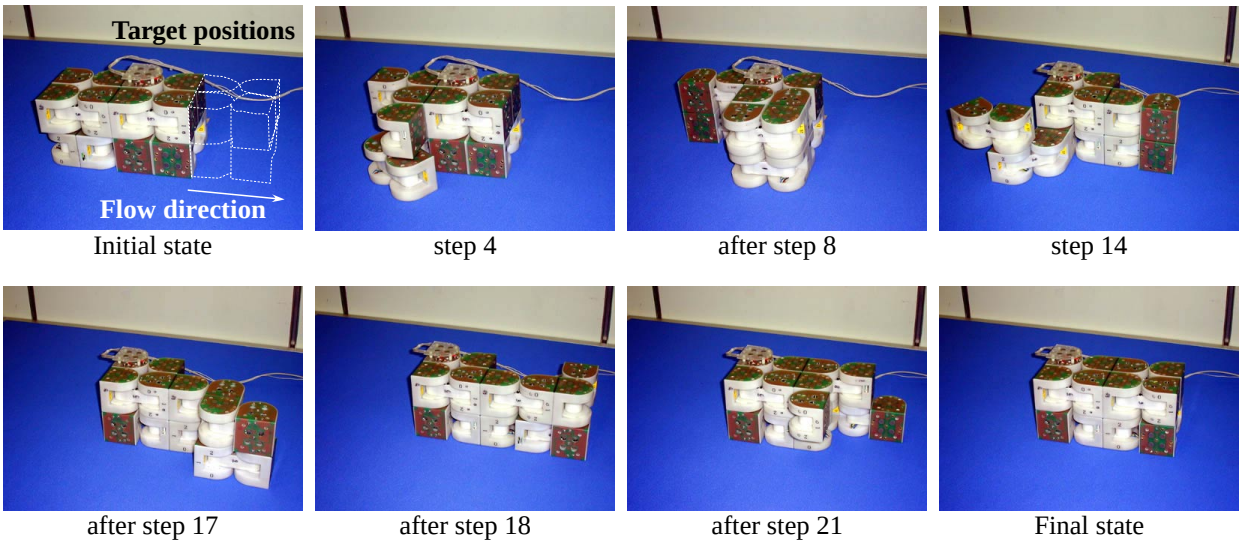



Fig. 17: Experiment of cluster motion of block structure using 8 modules.

## 8 Conclusions

This paper discussed motion planning of a self-reconfigurable modular robot designed to generate both static structure and dynamic robotic motions. We proposed a two-layered motion planning, global flow planner and local motion scheme selector. The former part provides the possible paths and motion orders to realize the flow of the cluster. The latter combines a series of motion schemes based on a rule database to make the flow.

In spite of the limited class of applicable structures, our approach will be effective for other classes by refining the global planner and extending the rule database. 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.

[14] A. Kamimura, et al. : "Self-Reconfigurable Modular Robot – Experiments on Reconfiguration and Locomotion –," *Proc. 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, in press.