Evolutionary Motion Synthesis for a Modular Robot using Genetic Algorithm

Eiichi Yoshida*† Satoshi Murata** Akiya Kamimura*
Kohji Tomita* Haruhisa Kurokawa* Shigeru Kokaji*

* Distributed System Design Research Group, Intelligent Systems Institute,
National Institute of Advanced Industrial Science and Technology (AIST)
1-2-1 Namiki, Tsukuba-shi, Ibaraki 305-8564 Japan
** Department of Computational Intelligence and Systems Science,
Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology,
4259 Nagatsuta-cho, Midori-ku, Yokohama, Kanagawa 226-8502 Japan

Abstract

An evolutionary motion synthesis method using genetic algorithm (GA) is presented for self-reconfigurable modular robot M-TRAN designed to realize various robotic motions and three-dimensional structures. The proposed method is characterized by its capacity to derive feasible solutions for complex synthesis problem of M-TRAN through natural genetic representation. For this purpose, the behavior of the robot is described using a motion sequence including both the dynamic motions and configuration changes of the robot. It is a series of segments each of which can specify simultaneous motor actuations and self-reconfiguration by connection/disconnection, starting from a given initial configuration. This simple description can be straightforwardly encoded into genetic representation to which genetic operations can be applied in a natural manner. We adopt traveling distance achieved by the evolved motion as the fitness function of GA. To verify the effectiveness of the proposed method, we have conducted simulations of evolutionary motion synthesis for certain initial configurations. Consequently, we confirm various adaptive motions are acquired according to different initial configurations and fitness functions. We also verify the physical feasibility of the evolved motions through experiments using hardware module M-TRAN II.

Key Words: Modular Robotic System, Self-reconfiguration, Evolutionary Computation, Motion Synthesis, Genetic Algorithms

*† Corresponding author (e.yoshida@aist.go.jp)

1 Introduction

Self-reconfigurable robots composed of simple robotic modules can change their shape by changing their connection and generate various motions as a combination of each module's movement. Their applications include robots that must move around in unstructured environments, such as a rescue robot that searches for survivors, a planetary exploring vehicle, or an inspection robot in hazardous environments. Modular robots can also be used as a static structure adaptive to environment. Recently, many types of three-dimensional self-reconfigurable modular robot have been proposed [1)−8)]. We have been developing a modular robot called M-TRAN (Modular TRANsformer) [9)] and its software for reconfiguration planning [12, 13)]. M-TRAN highlights its ability to realize both static structures and dynamic motions in three dimensions, thanks to its compact simplified design and dextrous connection mechanism. M-TRAN has successfully demonstrated its self-reconfigurability and dynamic full-body motion capability through experiments [9)]. For example, a cluster of M-TRAN modules can move as a crawler, then it can transform itself into a walking quadruped robot.

Due to their many degrees of freedom, motion synthesis of modular robots becomes a computationally difficult problem. Many planning methods have been proposed for three-dimensional self-reconfigurable robots [5, 6, 10, 11)] including our reconfiguration planning method [12, 13)]. Most researches focus on generating a reconfiguration sequence from one configuration to another. In addition to this "static" planning, dynamic motion synthesis must also be investigated so that the robot can work effectively in each configuration. Especially for robots with full-body dynamic motion capacity like M-TRAN, this motion synthesis is an important issue to fully exploit the high mobility of modular robot. A motion synthesis method is therefore necessary that can generate feasible three-dimensional motion with certain performance and is also widely applicable to different modular configurations. Nevertheless, high complexity of this motion synthesis problem has been a major barrier to development of such a method. It is difficult to apply frequently used schemes that learn the policy of behavior decision as an action-selection table "`if`-state then `then`-behavior" [14, 15)] because of dynamic motion and huge combinations of parameters describing states and behaviors. Path planning method often used for mobile robots [16, 17)] can difficulty in handling both the complex configuration space and the dynamic motion of modular robot either.

As to robots' motion generation itself, not limited to modular robots, a number of researches can be found. Usage of central pattern generator (CPG) based on neural oscillators is a common method for generation of biped or quadruped locomotion [18, 19)]. Kamimura also proposes a method for motion generation of M-TRAN [20)] using a CPG for fixed configuration. There have been studies on evolutionary computation of robot configuration and its motion. Sims showed virtual creatures that

evolves their morphology and control mechanism [21]. Hornby proposed evolutionary acquisition of static structure as well as moving robots based on evolutionary computation based on L-System [22]. Although these evolutionary methods are promising for modular robots, motion synthesis methods have hardly been addressed that integrates both dynamic motions and reconfiguration.

In this paper, we develop a simple and efficient motion synthesis method that can cope with its high complexity to derive feasible solutions with sufficient performance for self-reconfigurable robot M-TRAN. For this purpose, we introduce an evolutionary method using a genetic algorithm (GA). In this method, we devise a genetic representation that encodes both robots' motion and self-reconfiguration by using a genotype that describes a sequence of *segments* including both motor actuation and connection/disconnection of each module. The original point of this method lies in this integration of the motion synthesis problem with high complexity into a GA-solvable form through a natural segment-based motion description and encoding, which was difficult by action-selection table or bit-string encoding. We have applied this method to randomly generated M-TRAN configurations and conducted simulations to derive suitable motions using the moving distance as the fitness function in GA. A dynamic simulation software library is used to model the real robots' dynamics precisely. This paper concentrates on verifying the ability of synthesis of dynamic motions, while the description can potentially integrate self-reconfiguration as well. We will present various motions adaptive to fitness functions and initial configurations, which are obtained as simulation results of GA. Physical feasibility of evolved motions is experimentally confirmed using hardware prototype M-TRAN II.

2 Hardware and Motion Description

2.1 Module Structure

We briefly outline the hardware to give a basic idea of M-TRAN whose detailed hardware description is given elsewhere [9]. One module of the self-reconfigurable robot M-TRAN has two semi-cylindrical parts connected by a link as shown in Fig. 1. Each part rotates by $\pm 90^\circ$ by a servomotor embedded in the link and has three magnetic connecting faces (0 – 2). One of the two semi-cylindrical part has *active* connecting faces with movable magnets that can disconnect themselves from other connecting *passive* faces using shape memory alloy (SMA) actuator. A connecting face has electrodes for communication and power supply from external power source. Therefore, all the connected modules can communicate to each other through inter-module communication network. Each module has microprocessors for communication and driving actuators as well as batteries to allow the robot's autonomous operation.

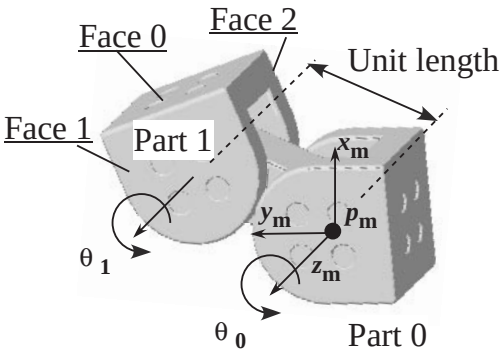


Fig. 1: A module of M-TRAN.

2.2 Description of Motion and Reconfiguration

To describe how the robot moves and reconfigures itself, we introduce a simple syntax that denotes connections and motor actuators of the modules that compose the whole robot. This syntax is designed so that the robot's motion is described in a simple manner by designating the initial state and the motion sequence.

(A) Initial State

The position and orientation of one module are uniquely determined by specifying the position and orientation of one part, together with the rotation angles of the servomotors. Therefore, the configuration of the whole robot can be given by listing them for all the modules. Figure 2 illustrates an example of initial state of a three-module configuration. We use an absolute coordinate system where one *unit length* is defined as the length between the two rotational axes of a module. Each semi-cylindrical part of module m is identified as 0 or 1. The position and orientation of module m are defined as the position p_m and the directions of the rotational axis z_m and link y_m of part 0 respectively in terms of the absolute coordinate system. The angles of servomotors are given by the absolute rotation angles of each part (θ_0, θ_1). The initial state in Fig. 2 is given as follows:

$$
\begin{array}{lll}
\text{ID 0} & p_m(0,0,0) \quad z_m(0,1,0) & y_m(-1,0,0) \\
& (\theta_0,\theta_1) = (0^\circ, 0^\circ), & \\
\text{ID 1} & p_m(-1,1,0) \quad z_m(-1,0,0) & y_m(0,1,0) \\
& (\theta_0,\theta_1) = (0^\circ, 0^\circ) & \\
\text{ID 2} & p_m(-1,-1,0) \quad z_m(-1,0,0) & y_m(0,-1,0) \\
& (\theta_0,\theta_1) = (0^\circ, 0^\circ) &
\end{array}
$$

Given those parameters, the connection and configuration of the modular robot can be uniquely determined. Hereafter, the initial rotation angles are assumed to be 0° for simplicity. We also assume

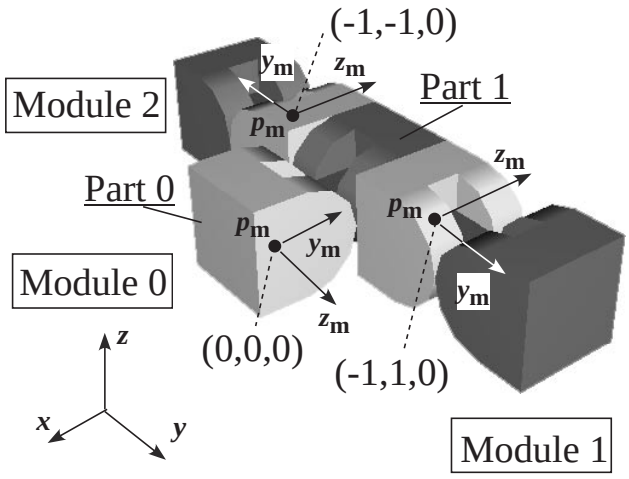


Fig. 2: Description of initial state.

that the connectivity of all the involved modules are maintained, which means that there are no isolated modules.

(B) Motion Sequence

The motion sequence is represented by a series of *segments* that describe the connective states and motor actuation of the modular robot based on our formerly developed interface software [23]. A segment has two parts, motion and connection operation.

The motion operation has the form "`m [ID basepart]` θ_0, θ_1" to specify how the servomotors of each module are actuated. Here command "`m`" denotes the operation *motion* and `[ID basepart]` are the ID of the module and its *base part* during the operation respectively. For each motor actuation, either of part 0 or 1 must be given as *base part* that is explicitly fixed to another module, while the other part is referred to as *moving part* (Fig. 3). The motor actuation is specified by "θ_0, θ_1" as absolute angles between -90° and 90°, with 30° step for simplicity. We also assume the rotation velocity is constant for any rotation angles.

The interface software keeps track of the whole configuration during the given operations and computes all the necessary connections. Therefore, there is no need for providing explicitly all the connections except for the following case. By default, the software assumes that the motion operation automatically cuts all the connections of *moving part* that changes its position. To maintain these connections, the operation *connection* must be provided explicitly using the command "`c`" as "`c [ID part] dir`." The connection operation is specified by command `c` with ID and the part that maintains its connection. The connecting face is designated by "`dir`" as either of connecting faces 0, 1, or 2 as shown in Fig. 3.

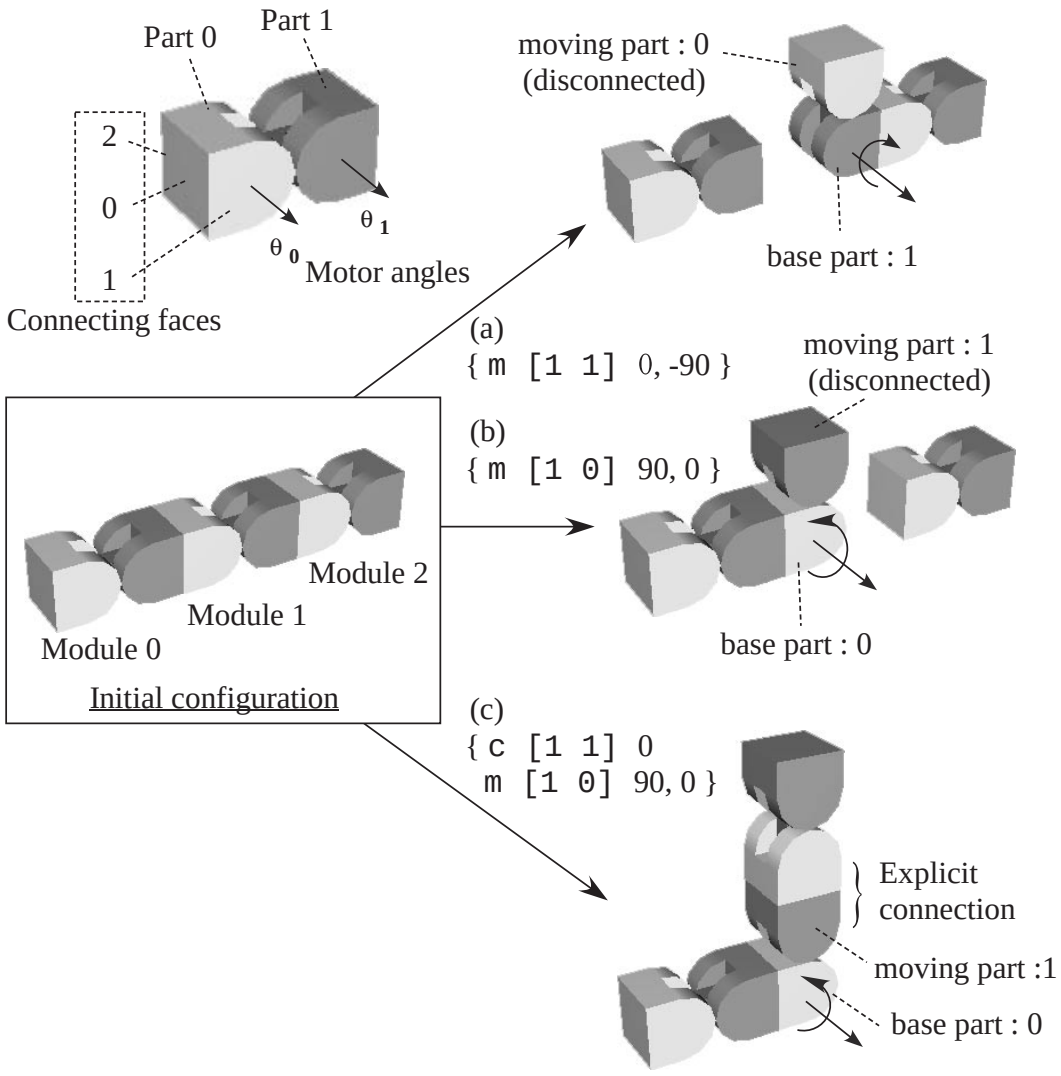


Fig. 3: Segments including simple motion and connection operations.

A *segment* can comprise simultaneous motion operations with necessary connection operation. A segment is shown as a collection of those operations quoted by "{}." There is also a command *loop* for iterative segments, as the segments quoted by "`L` N { ... }," where N is the iteration number.

Figure 3 shows examples of segment where the above operations are combined. In this figure, there are three modules 0–2 that are serially connected with all the motor angles 0°. For all three segments (a)–(c), only module 1 is actuated. The segments (a) "`m [1 1]` 0, -90" and (b) "`m [1 0]` 90, 0" are with base part 1 and 0 respectively. As can be seen, the results are different depending on the base part. Note that the connection of moving part is cut in both cases of (a) and (b). The segment (c) includes the connection operation "`c [1 1] 0`" that keeps the connection between modules 1 and 2.

Even though this simple syntax has limitations in rotation angles and velocities, it turned out that

various motions of M-TRAN can be described that include simultaneous motor actuations with connection/disconnection operations [9]. This syntax also reduces the amount of descriptions because of the above implicit connection representation and its automatic interpretation by the interface software.

3 Applying Genetic Algorithm to Motion Sequence

In this section genetic algorithm (GA) is integrated into motion synthesis of self-reconfigurable modular robot M-TRAN through segment-based encoding of motion sequence. Before explaining the implementation, the advantages of using GA, such as simplicity, ease of obtaining feasible solutions and analyzing the results, are described compared to other approaches.

The frequently used approach describes the robot's behavior as an action-selection table of "`if`-state (configuration) `then`-behavior (connection, actuation)" and applies such schemes as reinforcement learning or neural networks so that the robot can obtain the behavior policy based on this table through trial and error. This is very effective for those problems that have complexity below certain level, for example motion acquisition of simple robot or mobile robot navigation [14, 15]. However, especially for mobile robots, its motion synthesis has very high complexity [1] and also includes dynamic motions; it is not realistic to implement a system that learns such a large behavior table due to combinatorial explosion.

Another promising approach is evolutionary motion generation and there have been a number of researches. Among them is method based on CPG that can generate suitable motion patterns for fixed body mechanisms in an emergent manner [18, 19]. Other studies show another method that evolves both body topology and control system [21, 22]. However, those studies do not assume the topology changes by self-reconfiguration using connection/disconnection.

Yet another alternative is path planning method [16, 17] often used for mobile robot navigation. However, it is not appropriate for modular robot either, because the planner must deal with complex configuration space with modular robots' many degrees of freedom as well as dynamic motion.

For the above reasons, we adopt GA in such a way that a motion sequence can be represented in a natural way; segment-based genetic representation is devised that encodes a motion sequence into a genotype to which genetic operations like crossover and mutation are applicable. Besides its simplicity, this integration of GA has several advantages. First, since this method searches in a complex problem space by covering it with certain sparseness, we can expect that it derives feasible solutions

[1] One M-TRAN module has 2^6 possible connection states and 7^2 possible angular states (30° step). Since the number of action is same, the size of state-action table becomes approximately 10^7. This size grows exponentially with the number of modules.

for various initial modular configuration, but at the cost of optimality. Next, by giving an enough length to the evolving motion sequence, we can also expect that meaningful motions are derived with certain performance according to various the fitness functions. Finally, segment-based encoding enables evolutionary acquisition of the modular robot's motions as well as configuration changes, and also makes it easy to analyze the resultant motion sequence. The simple motions obtained for small modular robots can also be used as primitives that are combined into a larger system as described in Section 6.

In the following, encoding and genetic operations are detailed in 3.1 and 3.2 respectively.

3.1 Encoding

We adopt a direct representation that encodes a segment-based motion sequence into a genotype string where one gene corresponds to one segment (Fig. 4). This section explains the genetic simulator that allows the modular robot to evolve its motion using a genetic algorithm. Since the motion sequence defined in Section 2.2 is a series of segments that are the smallest elements of a genotype, the segment-based encoding can be understood in a straightforward manner. Configuration changes may occur during executing the motion sequence specified by a genotype.

In this framework, an arbitrary configuration can be chosen as an initial state for which an appropriate motion sequence will be obtained through the evolution. On running GA, population is provided as a collection of genotypes for given initial configurations.

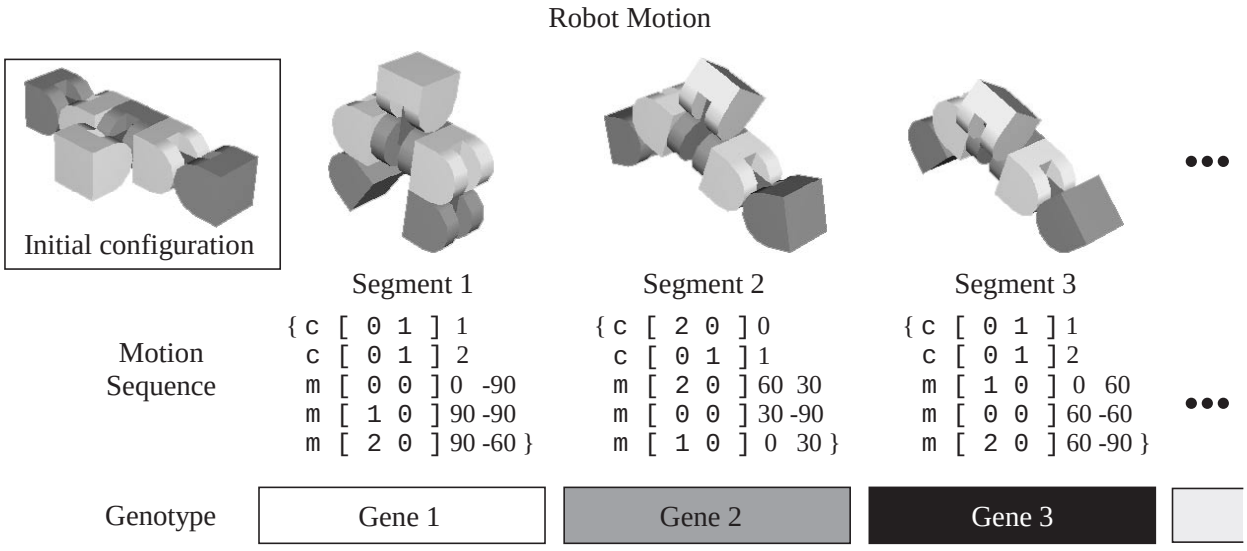


Fig. 4: Encoding a robot motion sequence into a genotype by assigning a gene to a segment.

3.2 Genetic Operations

Genetic algorithms mainly consist of the following processes in each generation; reproduction, genetic operations including crossover and mutation, evaluation, and selection. We assume that initial configurations are given in the first place. In the simulations in the next section, we will choose examples from among randomly generated initial states.

(A) Crossover and Mutation

Figure 5 shows how crossover and mutation are applied to the introduced genotype. The crossover operation is applied to two different strings that represent motion sequences. Two of new strings are generated from those strings as illustrated in Fig. 5. This operation can be done in a similar way to the crossover for bit strings. Here we adopt one cross-point crossover in this paper.

The mutation operation is applied to one string from which random number of segments are chosen to be mutated. The selected segments are replaced by those randomly generated (Fig. 5). These segments include connection "`c`" and motion "`m`" operations in which their ID, part, rotation angles (by 30° step), and connecting faces are specified randomly.

The crossover and mutation operations are applied to the ratio r_c and r_m of the total population respectively.

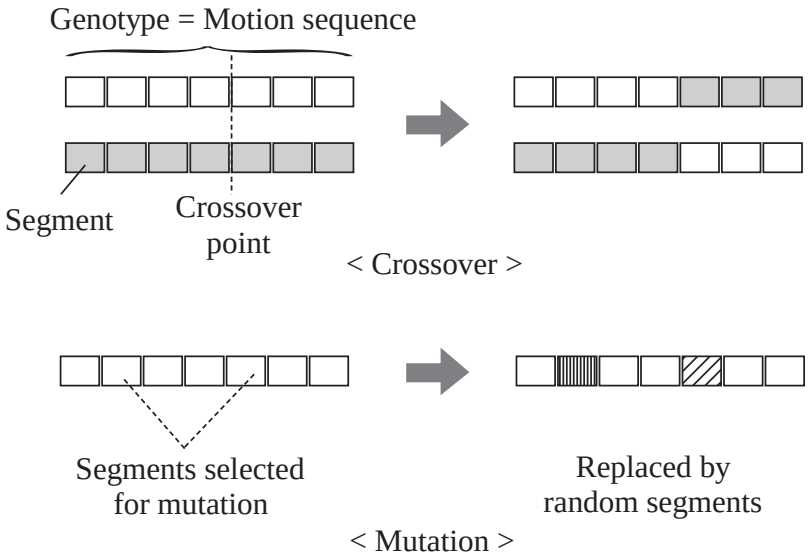


Fig. 5: Crossover and mutation operations.

(B) Evaluation and Selection

After these operations, evaluation is conducted by two phases for the newly generated genotypes; first by its physical feasibility and next by performance evaluation using fitness functions.

The first evaluation is necessary because some motions represented by the resulting genotypes after the genetic operations may not be physically feasible due to the following hardware limitations of the robot. The desired motion cannot be completed when:

(1) connectivity is not maintained by inappropriate disconnections,

(2) collision between the modules occurs, or

(3) excessive force or torque are applied to the connection faces or servomotors.

The above motion feasibility is verified by the interface software introduced in 2.2(B). Therefore, if a resulting genotype turns out to be infeasible, it is rejected and other possibilities are explored by genetic operations. This is an important step to maintain the population of genotypes representing feasible motion sequences.

Next, the physically feasible genotypes undergo the second evaluation for selection. This evaluation is performed based on the fitness function, which must be chosen depending on the task and the constraints such as hardware limit of environments. The fitness function can be traveling distance, energy consumption, reconfiguration steps, or their combinations.

In this paper as the fitness function, the robot's traveling distance between the initial and the final positions is computed through the motion described by the genotypes during a certain period of time. Here the position of the robot is represented as its center of mass. All the genotypes in the population are evaluated using this fitness function. Later in the simulation section, the reconfiguration steps are fixed as all the length of genotypes representing motion sequences is the same. However, this can also be used as a part of the fitness function in other applications.

Those genotypes that correspond to suitable motions are preserved in the next generation according to the calculated fitness function. We adopt here a hybrid selection to prevent premature convergence and maintain genetic variety. Three selection schemes are combined in this hybrid selection, elite, ranking and random selections, which are applied to the ratio s_{elite}, s_{rank} and s_{rand} of the population respectively. Elite and ranking selections are to keep the fittest and relatively fit genotypes. Random selection is introduced in order not to lose the variety of the population.

In this way, a new population is succeeded to the next generation and the same genetic process is repeated.

4 Simulation Results

Simulations have been conducted to verify that the proposed method can generate proper motions according to its configuration. In this paper, we focus on motion synthesis for fixed configurations as a first stage of development, although the proposed evolutionary method can designed to deal with configuration changes.

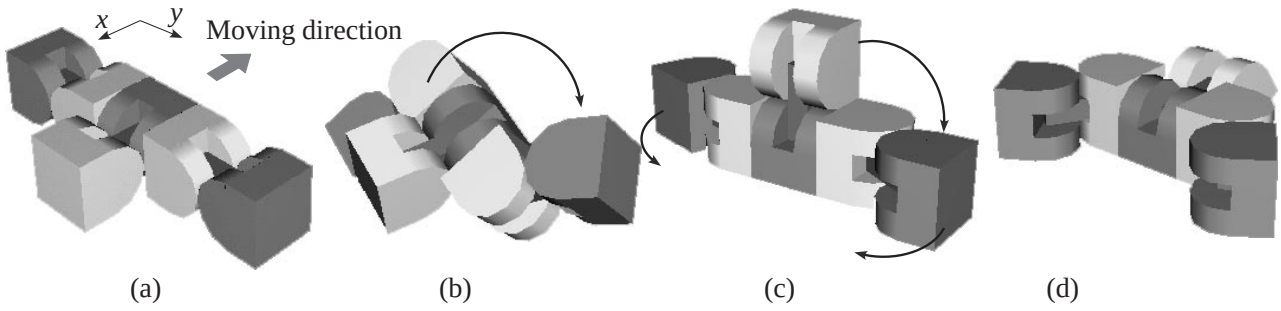
In the following simulations, evolutionary motion synthesis is simulated for randomly generated initial configurations composed of three modules on a flat plane with friction. Given an initial configuration, three fitness functions, the absolute traveling distance in $\pm x$ and $\pm y$ directions and the total traveling distance are used to observe how the motions evolve according to different fitnesses. For the simulations, we use the crossover and mutation ratio $r_c = 0.5$ and $r_m = 0.05$. The ratios of hybrid selection are $s_{elite} = 0.1$, $s_{rank} = 0.3$, and $s_{rand} = 0.6$ respectively. The GA is conducted with population size 40 and total generations 50 starting from a randomly generated initial population. Here the length L of genotype string is constant as $L = 10$ for simplicity. The simulation repeats the interpreted motion sequence until a period of $T = 100$ seconds elapses to evaluate the traveling distance.

The dynamics of robot motion are simulated using a dynamics simulator library Vortex developed by Critical Mass Labs. In the simulation, we assume that a velocity control for servomotors with constant angular velocity $v = \pi/6$ and maximum torque to lift another module against gravity. The connection between modules is considered to be sufficiently rigid. Such parameters as size and mass of the robot, gravity, and friction are also appropriately modeled.

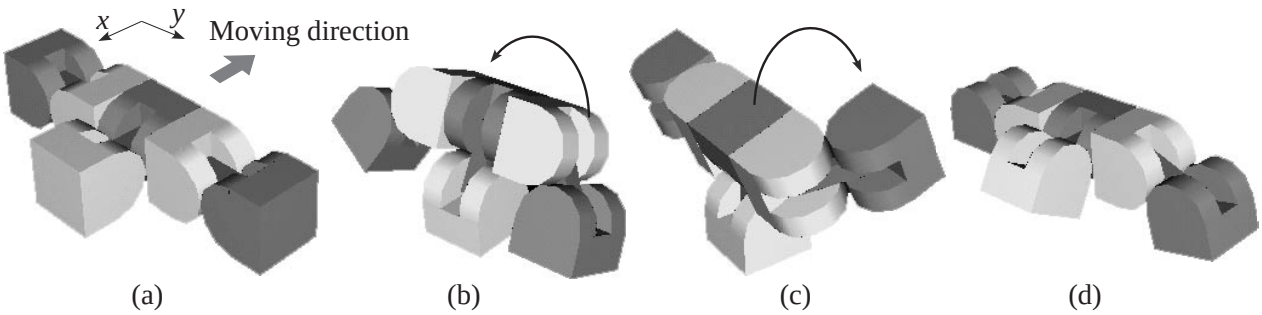
In this section we report some simulation results of evolved motion. The first result shows how different motions develop for different fitnesses, and the second one is an example of a dynamic whole-body motion.

4.1 Evolution of Motions for Different Fitnesses

The first simulation result demonstrates that different motions have been obtained for the same initial configuration depending on fitness functions. The genetic algorithm was applied to an initial configuration in Fig. 2 placed on a flat plane using two fitness functions; the total moving distances in $\pm x$ and $\pm y$ directions. Figure 6 shows two fittest motions at different generations 27 and 40, using the absolute moving distance in $\pm x$ direction as the fitness function. At the earlier generation of 27, the GA outputs a crawling motion using friction as shown in Fig. 6(1). Then more efficient motion is discovered where a central module lifts two other modules and swings them forward to gain the



(1)



(2)

Fig. 6: Two different motions evolved using the absolute moving distance in $\pm x$ direction as the fitness function. (1) Crawling motion using friction at generation 27. (a) initial condition. The robot rolls itself (b), and make crawling motion to move in $-x$ direction (c, d). (2) "Lift and swing" motion acquired at generation 40. (a) initial condition. The central unit lifts the other two (b), swings them toward $-x$ direction (c) and gain distance (d).

distance at generation 40, as in Fig. 6(2). Figure 7 is the development of average and maximum value of fitness functions in the population, the traveling distance in $\pm x$ direction after 100 seconds. We can observe that the maximum fitness function drastically rises from 4.2 to 7.7 at generation 40 when the lifting motion was acquired. The average fitness is also gradually improved through generations.

Figure 8 shows the evolved motion after total 50 generation using the moving distance in $\pm y$ direction as the fitness. This is an inchworm locomotion using two modules at both ends and the robot moves by the distance of 18.1.

4.2 Example of Whole-Body Motion

Figure 9 shows another random configuration for which total traveling distance in an arbitrary direction is used as the fitness function. As a result, a whole-body twisting motion was finally acquired after simulation of 50 generations. During this motion, the robot continuously twists its body and repeats flipping itself to move in a certain direction. Although it is a simple motion that iterates a

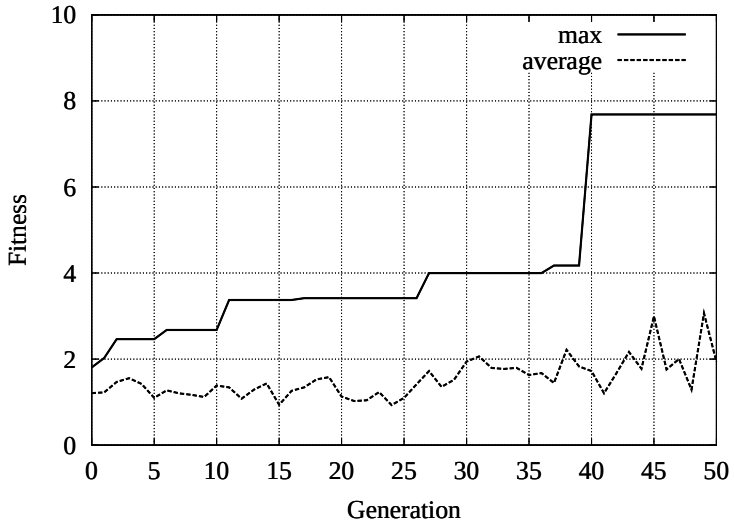


Fig. 7: Development of fitness function of distance in $\pm x$ direction.

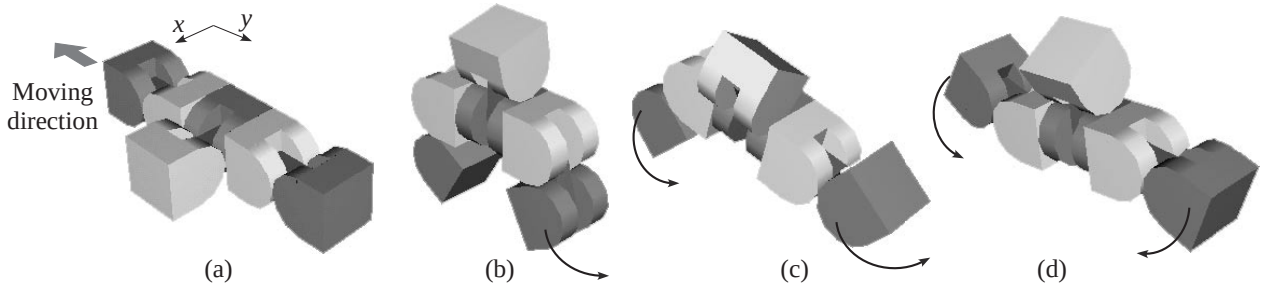


Fig. 8: Motion evolved using the absolute moving distance in $\pm y$ direction as the fitness function. (a) initial condition. The robot moves by an inchworm locomotion by the modules at both ends.

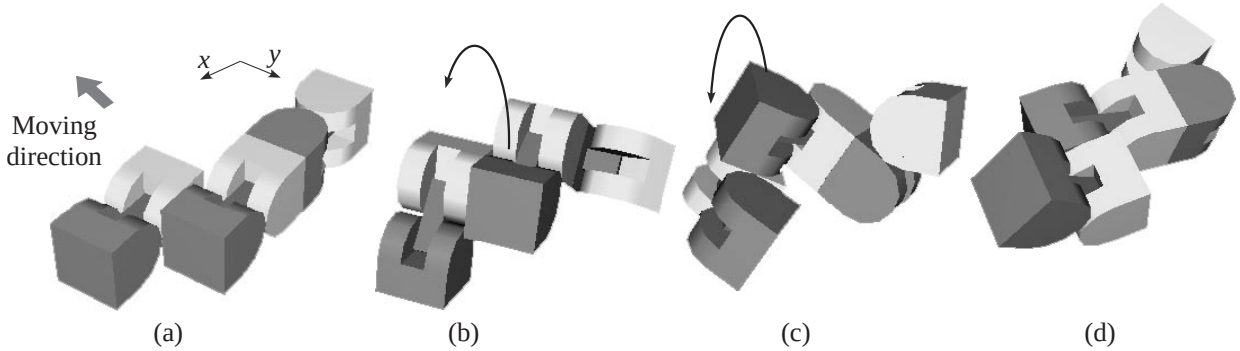


Fig. 9: A whole-body motion evolved using the total moving distance as the fitness function. (a) initial condition where the arrow shows the moving direction. Using almost all the degrees of freedom, the robot is rolling itself (b, c) and then total body is flipped over (d) to move in the desired direction.

sequence composed of just 10 segments, it can keep twisting skillfully and produce a steady advance. The development of fitness function is shown in Fig. 10. The simulation output a motion that achieves the maximum distance of 25.0 in 100 seconds after 50 generations. We can also observe the gradual

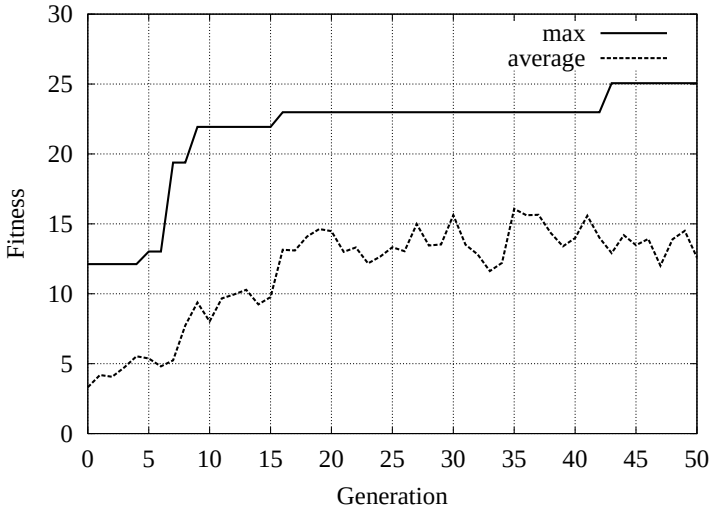


Fig. 10: Development of fitness function of total moving distance.

improvement of the average fitness.

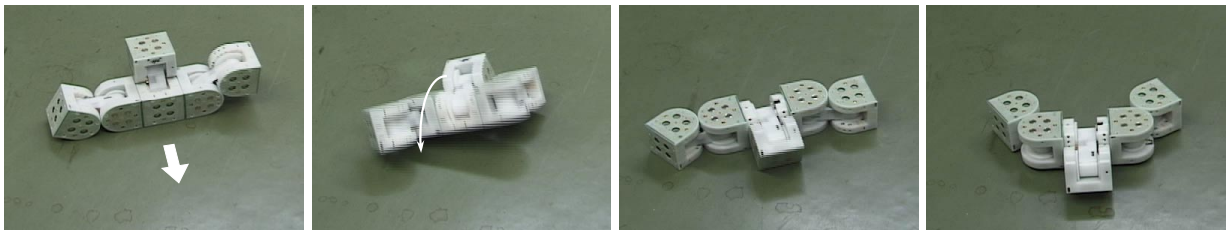
To summarize, we have shown that suitable motion sequence has been obtained according to the given initial configurations and different fitness functions. From the above simulation results, we demonstrated that the proposed evolutionary method can generate motions of modular robot in an adaptive and effective fashion.

5 Experiments

We have conducted experiments to verify the evolved motion can be realized using hardware module of M-TRAN II model developed recently [20]. Compared to the previous model M-TRAN, this hardware is downsized one box part from 6.6cm cube to 6 cm, and the performance is improved in motor control, communication capacity, and power consumption. These improvements allows the modules to operate autonomously using battery.

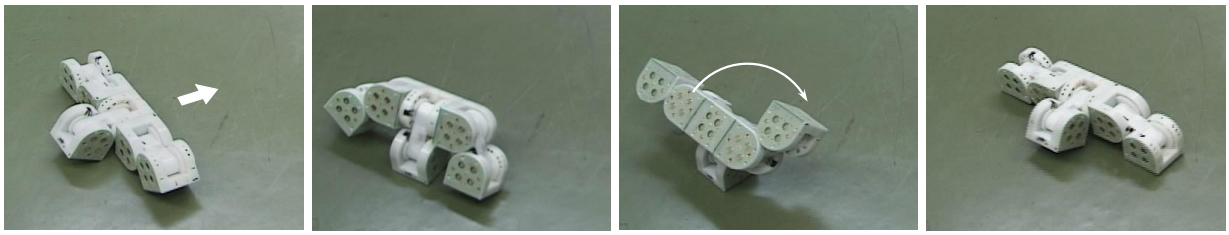
The evolved motion sequence is downloaded to each module to move the modules in the same way as the simulation. During the experiments, the modular robot is driven by the batteries embedded in each module. Figures 11 and 12 show the experiment using the results in Figs. 6 and 8 in Section 4.1. Figure 13 is the experimental result of the whole-body twisting motion obtained in Section 4.2. In general, the evolved motions are executed properly by the hardware and the robot moves in the expected direction.

The above experiments verified the motion derived by the proposed evolutionary motion synthesis is physically feasible. However, we also observed that some motions are not executed as expected. To improve the accuracy of simulation, modeling of friction and motor property must be refined.



(a) (b) (c) (d)

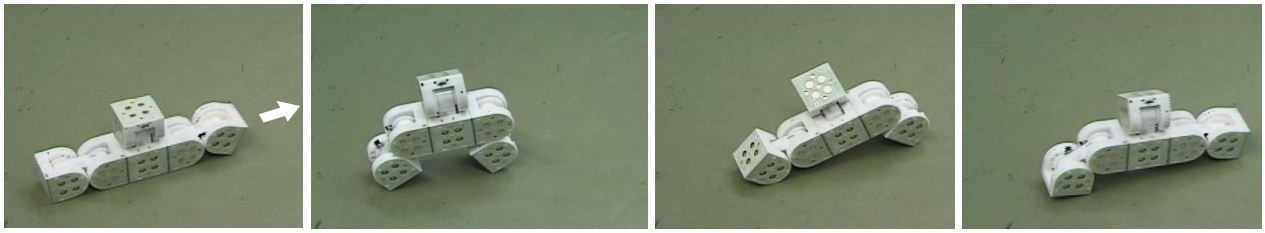
(1) Crawling motion.



(a) (b) (c) (d)

(2) Lift-and-swing motion.

Fig. 11: The evolved motions in Fig. 6 using the absolute moving distance in $\pm x$ direction as the fitness function. (1) After rolling itself (b), the robot crawls in the desired direction using friction. (2) To achieve the evolved "lift and swing" motion, the central module successfully lifts and swings the other two modules.



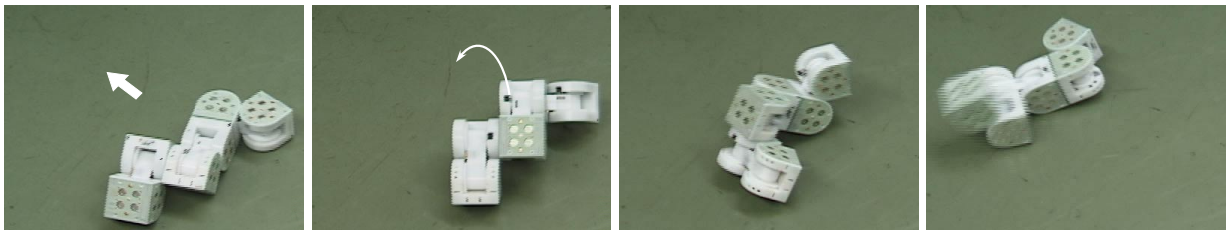
(a) (b) (c) (d)

Fig. 12: The evolved motion in Fig. 8 using fitness function of moving distance in $\pm y$ direction. The robot realizes an inchworm locomotion.

6 Conclusions and Future Work

6.1 Conclusions

This paper presented an evolutionary motion synthesis for a modular robot using the genetic algorithm (GA). The method is featured by the ability to obtain feasible motions for the complex sysnthesis problem and to describe the evolution of both the motor actuation and configuration changes of the robot. We first introduced a simple description of motion and reconfiguration of modular robot M-TRAN, in the form of motion sequence composed of segments. Next, a natural way of encoding those



(a) (b) (c) (d)

Fig. 13: The whole-body motion evolved in Fig. 9 using the total moving distance as the fitness function. The twisting and flipping motion is properly executed.

motion sequences into genotypes by regarding a segment as a gene, and such genetic operations as crossover and mutation are defined accordingly. We have conducted simulations using the absolute traveling distance as the fitness functions of GA to verify that suitable motions can evolve for given initial configurations. The simulations are conducted so as not to include configuration changes as we focused on the evolutionary acquisition of robot motion. Using the proposed framework, various motions were evolutionarily obtained in accordance with different fitness functions. It is noteworthy that completely different motions evolved depending on different fitness functions that are moving distances in different directions. Finally, the feasibility of evolved motions are verified using hardware modules. We confirmed that the proposed method can output physically feasible motions by the hardware.

6.2 Future Work

As noted above, this paper concentrates on the evolution of the robot motions and although configuration changes can be integrated in the genetic representation, they are not included in the simulation. The next step of this work is to investigate how the configuration changes must be included to evolve the behavior of the robot effectively.

From the evolutionalism point of view, the results obtained in this paper corresponds to individual evolution, whereas the evolution of configuration seeks to evolve the species. The latter becomes significant where the robot encounters a drastic change of the external environment that cannot be handled solely by the limited motion variety within a fixed configuration. One of the ways to represent those topology changes is a network between different configurations. Figure 14 illustrates an image of adaptive system including evolution of both individual and species. In this system, repertoire of evolved primitive motions is stored in each individual configuration. If the environmental change is considered to be significant, appropriate reconfiguration process must be invoked based on the configuration network; in Fig. 14, this corresponds to reconfiguration between quadruped and crawler.

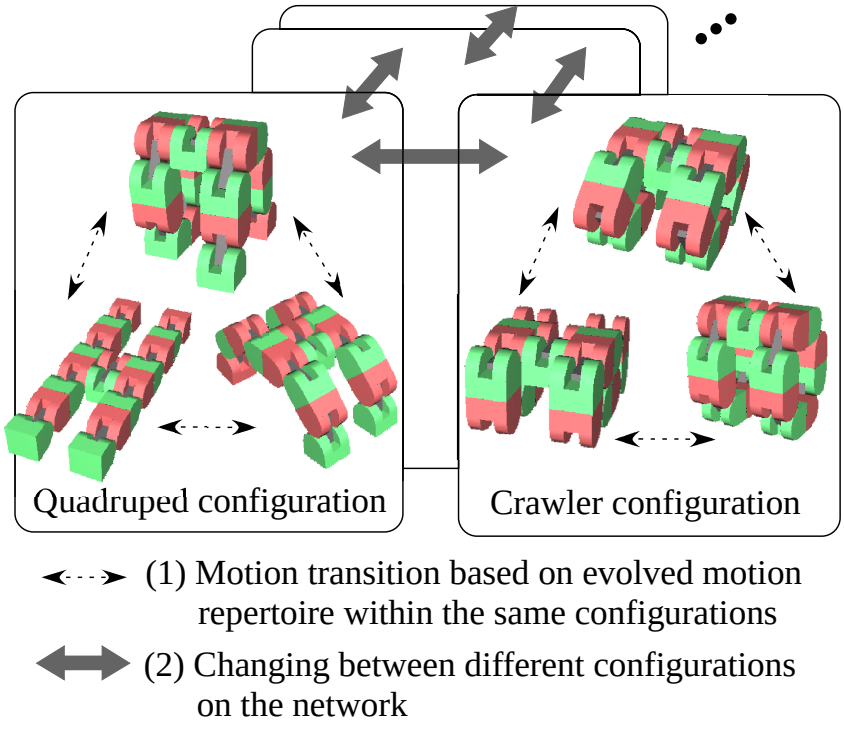


Fig. 14: Adaptive system including evolution of motion and configuration.

To realize such a system, integration of sensors becomes an important research topic. We will continue investigating the design of an adaptive system for modular robots. Improvement of precision of modeling in simulations is also an issue to be tackled in future work, as well as selection of appropriate parameters of GA.

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) A. Castano, R. Chokkalingam, and P. Will: "Autonomous and Self-Sufficient CONRO Modules for Reconfigurable Robots," *Distributed Autonomous Robotics 4*, Springer, 155–164, 2000.

4) 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.

5) C. Ünsal, H. Kılıççöte, and P. Khosla: "A modular self-reconfigurable bipartite robotic system: Implementation and Motion Planning," *Autonomous Robots*, **10**-1, 23–40, 2001.

6) M. Yim, Y. Zhang, J. Lamping, and E. Mao: “Distributed Control for 3D Metamorphosis,” Autonomous Robots **10**-1, 41–56, 2001.

7) D. Rus and M. Vona. “Crystalline Robots: Self-reconfiguration with Compressible Unit Modules,” *Autonomous Robots*, **10**-1, 107–124, 2001.

8) E. Yoshida, et al: “Micro Self-Reconfigurable Modular Robot Using Shape Memory Alloy,” *Journal of Robotics and Mechatronics*, **13**-2, 212–219, 2001.

9) A. Kamimura, et al. : “Self-Reconfigurable Modular Robot – Experiments on Reconfiguration and Locomotion –,” *Proc. 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 606–612, 2001.

10) 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.

11) E. Yoshida, et al. : “A distributed method for reconfiguration of 3-D homogeneous structure,” *Advanced Robotics*, **13**-4, 363–380, 1999.

12) E. Yoshida, et al: “A Motion Generation Method for a Modular Robot,” *Journal of Robotics and Mechatronics*, **14**-2, 177–185, 2002.

13) E. Yoshida, et al: “A Self-Reconfigurable Modular Robot: Reconfiguration Planning and Experiments,” *Internal Journal of Robotics Research*, to appear.

14) M. Asada, et al: “Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning,” *Machine Learning*, **23**-2/3, 279–303, 1996.

15) H. Kimura and S. Kobayashi: “Reinforcement Learning for Locomotion of a Two-linked Robot Arm,” *Proc. of the 6th Europian Workshop on Learning Robots (EWLR-6 1997)*, pp.144–153, 1997.

16) J.-C. Latombe: Robot Motion Planning, *Kluwer Academic Press*, 1991.

17) S. M. LaValle and J. J. Kuffner: “Rapidly-Exploring Random Trees: Progress and Prospects,” *Int. Journal of Robotics Research*, **20**-5, 378–400, 2001.

18) G. Taga: “A model of the neuro-musculo-skeletal system for human locomotion II – real-time adaptability under various constraints,” *Biolog. Cybern.*, **73**, 113–121, 1995.

19) H. Kimura, et al.: “Realization of dynamic walking and running of the quadruped using neural oscillator,” Autonomous Robots, **7**-3, 247–258, 1999.

20) A. Kamimura, et al. : “Automatic Locomotion Pattern Generation for Modular Robots,” *Proc.*

2003 IEEE Int. Conf. on Robotics and Automation, submitted.

21) Karl Sims: “Evolving Virtual Creatures,” *Computer Graphics, Annual Conference Series (SIGGRAPH ’94 Proceedings)*, 15–22, 1994.

22) S. Hornby, et al. : “Body-Brain Coevolution Using L-systems as a Generative Encoding,” *Proc. Genetic and Evolutionary Computation Conference (GECCO)* , 868–875, 2001.

23) H. Kurokawa, et al. : “Motion Simulation of a Modular Robotic System,” *Proc. 2000 IEEE Int. Conf. on Industrial Electronics, Control and Instrumentation*, 2473–2478, 2000.