Evolutionary Synthesis of Dynamic Motion and Reconfiguration Process for a Modular Robot M-TRAN

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

* Intelligent Systems Institute, National Institute of Advanced Industrial Science and Technology (AIST)
Tsukuba East, 1-2-1 Namiki, Tsukuba-shi, Ibaraki 305-8564 Japan
e.yoshida@aist.go.jp http://unit.aist.go.jp/is/dsysd/
** 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

In this paper we present a couple of evolutionary motion generation methods using genetic algorithms (GA) for self-reconfigurable modular robot M-TRAN and demonstrate their effectiveness through hardware experiments. Using these methods, feasible solutions with sufficient performance can be derived for a motion generation problem with high complexity coming from huge configuration and motion possibilities of the robot. The first method called ERSS (Evolutionary Reconfiguration Sequence Synthesis) applies GA (Genetic Algorithm) to evolution of motion sequence including configuration changes though natural genetic representation. The effectiveness of the generated full-body dynamic motions are verified through hardware experiments. The second method called ALPG (Automatic Locomotion Pattern Generation) Method seeks locomotion pattern using a neural oscillator as a CPG (Central Pattern Generator) model and GA to optimize the parameters for locomotion. A number of efficient locomotion patterns has been derived, which are also experimentally verified.

Key Words: Modular Robotics, Self-Reconfiguration, Evolutionary Computation, Motion Generation, Genetic Algorithms, CPG

1 Introduction

Self-reconfigurable robots composed of simple robotic modules can reorganize their shape by changing their connection and generate various motions as a combination of each module's movement. Thanks to their flexibility, versatility and fault-tolerance, self-reconfigurable modular robots are the most useful in situations where they should move and work in hazardous, unstructured or unknown environments. This type of robots are also suitable in environments where they are required to achieve different tasks, which cannot be specified beforehand. The potential applications of self-reconfigurable modular robots range from static structure to mobile robots, especially in extreme environments inaccessible to humans. They can be applied to satellite antennas, space stations, or deep-sea structures.

Recently, many types of three-dimensional self-reconfigurable modular robot have been proposed [1]–[11]. We have been developing a modular robot called M-TRAN (Modular TRANsformer) [7]–[11] that can work as a vehicle as well as a static structure. Especially, the mobility of recently developed hardware M-TRAN II [11] is greatly improved so that it can move as a legged robot, a snake, or a crawler. To fully exploit this high mobility of modular robot, the motion generation synthesis is as important as the reconfiguration planning [10] developed so far.

However, due to many degrees of freedom of modular robot, motion synthesis of modular robots also becomes a computationally difficult problem. For these reasons, we have been developing methods for evolutionary motion pattern generation methods using genetic algorithms (GA).

In this paper we present two methods, ERSS (Evolutionary Reconfiguration Sequence Synthesis) [9] and ALPG (Automatic Locomotion Pattern Generation) Method [11], each of which have different features. In the first method ERSS, the GA is utilized to evolve the motion sequence. This method is characterized by its simplicity and ease of extension to evolutionary synthesis of plans including configuration changes. Interesting full-body dynamic motions have been obtained using this method. The second method ALPG is dedicated to evolve locomotion pattern for fixed module configuration using a neural oscillator as a CPG (Central Pattern Generator) model and GA to adjust the parameters for locomotion. A number of efficient locomotion patterns has been derived. Their performance of output motion patterns generated by those methods have been verified through hardware experiments.

2 M-TRAN II Hardware

Figure 1 shows the newest model "M-TRAN II." Each module has two semi-cylindrical parts, active and passive

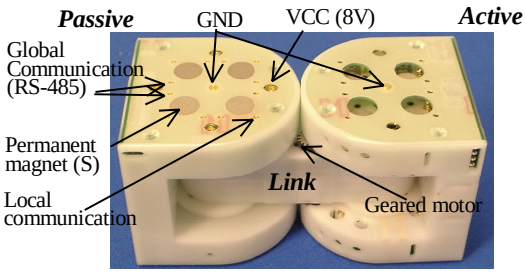


Fig. 1: A hardware module of M-TRAN II.

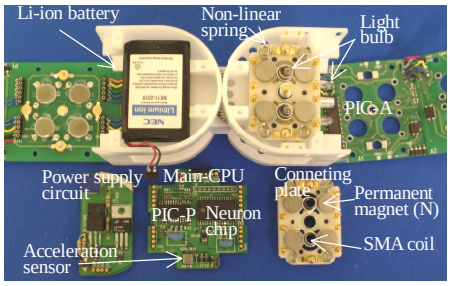


Fig. 2: Internal structure of a M-TRAN II module.

parts, and two geared motors inside the link. The active part has a movable plate with magnets that rises to the surface by the attractive force of the magnets embedded in the passive part of another module, to connect electrically and mechanically. Two surfaces can be detached automatically by heating the shape memory alloy coils by small light bulbs.

Figure 2 illustrates the internal structure of a module. Electrodes are placed symmetrically on the same surface for the power supply, global communication, and local communication. A CPU circuit board is equipped inside the passive part, including a microprocessor for global and local inter-module communication. A power supply circuit board and a battery are also embedded. Power for the module can be supplied by an internal battery or by connecting wires from outside to one of the modules.

3 ERSS – An Evolution of Motion Sequence

As mentioned earlier, development of motion synthesis method is 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 [1]. 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" [12, 13] because of dynamic motion and huge combinations of parameters describing states and behaviors. Path planning method often used for mobile robots [14, 15] has difficulty in handling both the complex configuration space and the dynamic motion of modular robot either.

For the above reasons, we adopt GA in such a way that a motion sequence can be represented in a natural way; 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 of each module. Previous research [18, 19] has hardly addressed unified approaches including both dynamic motions and reconfiguration. Genetic operations like crossover and mutation are applicable to this genotype. 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 simple and natural description and encoding. This integration of GA makes it easier to obtain feasible solutions by searching a complex problem space with certain sparseness.

4 ERSS Implementation and Experiments

4.1 Describing 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 [20]. A segment has two parts, motion and connection operation.

Figure 3 shows examples of segment where those operations are combined. As shown in the upper left of Fig. 3, the parts are distinguished as parts 0 and 1 with the corresponding rotation angles θ_0 and θ_1, and the connection faces are named $0 \sim 2$.

The motion operation is described using the command "`m`" such as "`m [ID basepart]` θ_0, θ_1" to specify the actuation of each motor. Here `[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] 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.

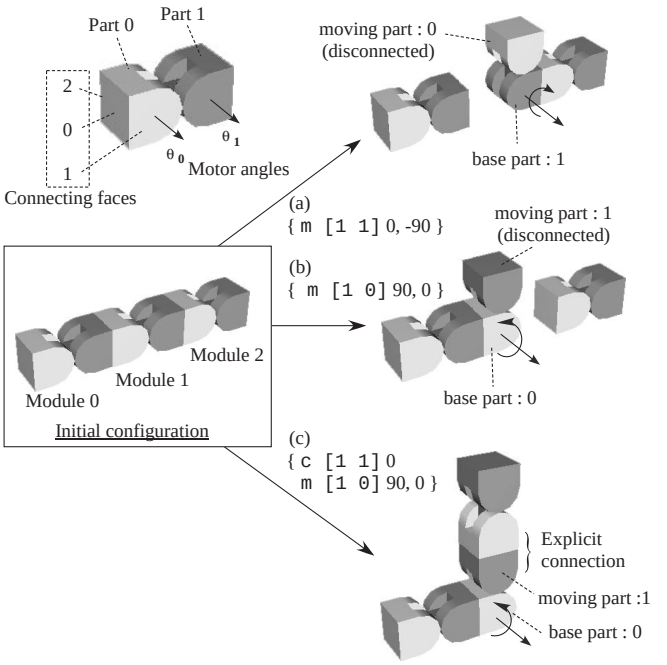


Fig. 3: Segments with motion and connection commands.

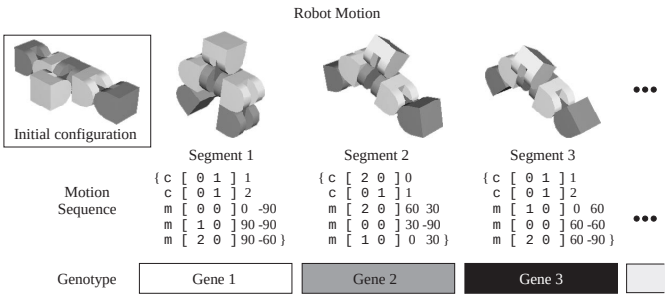


Fig. 4: Encoding a robot motion sequence into a genotype by assigning a gene to a segment.

1, or 2 as shown in Fig. 3.

A *segment* can comprise simultaneous motions 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.

This syntax can reduce the amount of descriptions because of the above implicit connection representation and its automatic interpretation by the interface software.

4.2 Applying Genetic Algorithm to Motion Sequence

This section explains the genetic simulator that allows the modular robot to evolve its motion using a genetic algorithm. 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). Since the motion sequence defined in Section 4.1 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. An appropriate motion sequence will be obtained through the evolution by repeating the following processes after reproduction; genetic operations including crossover and mutation, evaluation, and selection.

Genetic Operations

Figure 5 shows how crossover and mutation are applied to the introduced genotype. The crossover is applied to two different genotype strings that represent motion sequences using one cross-point crossover. The mutation is performed to one string from which random number of segments are chosen and are replaced by those randomly generated.

Evaluation

Evaluation is conducted in the following two phases for the newly generated genotypes.

(1) Rejecting physically infeasible genotypes due to hardware limitations like collision or loss of connectivity.
(2) Evaluating performance using fitness functions, either traveling distance, velocity, energy consumption, or their combinations, depending on the tasks.

Selection

We adopt here a hybrid selection to prevent premature convergence and maintain genetic variety by combining elite, ranking and random selections, with the ratio of 0.1, 0.3, 0.6 respectively. Elite and ranking selections keep the fittest and relatively fit genotypes. Random selection is introduced in order not to lose the variety of the population.

4.3 Experiments of Evolved Motion Sequence

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. Here we adopt the traveling distance between the robot's initial and the final positions of center of mass through the motion described by the genotypes during a

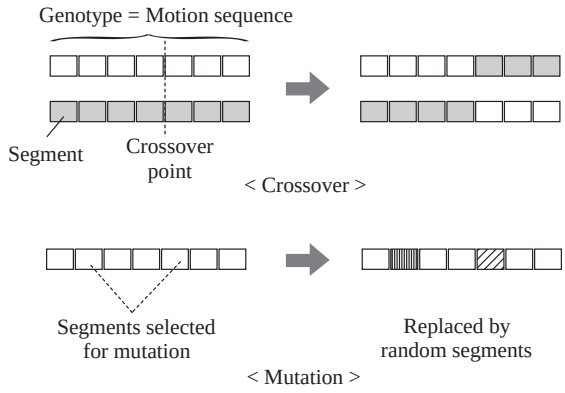


Fig. 5: Crossover and mutation operations.

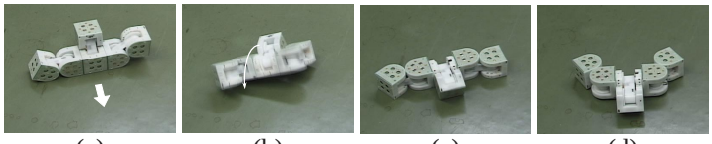
certain period of time.

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. We use the crossover and mutation ratios are 0.5 and 0.05. The simulation repeats the genetically evolving motion sequence during 100 seconds. We assume that the motors have constant angular velocity $v = \pi/6$ and enough torque to lift another module against gravity. The dynamics of robot motion are simulated using a dynamics simulator library Vortex developed by Critical Mass Labs, and the simulation results are experimentally validated using the hardware modules.

Figure 6 shows the two fittest motions at different generations 27 and 40, using the absolute moving distance in the direction indicated by the arrow. At the earlier generation of 27, the GA outputs a crawling motion using friction as shown in Fig. 6(i). Then more efficient motion is discovered where a central module lifts the other two and swings them forward to gain the distance at generation 40, as in Fig. 6(ii). Figure 7 is the development of average and maximum value of fitness functions in the population, the traveling distance 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 gradually improved through generations.

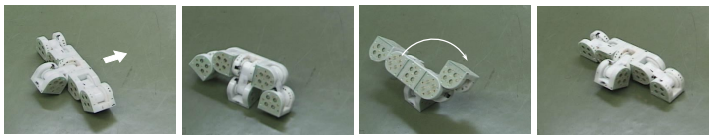
Figure 8 shows the motion evolved by applying the different fitness function to the same initial configuration; using the traveling distance in the different direction. An inchworm locomotion using two modules at both ends is obtained so that robot can move effectively in the direction specified by the arrow.

Figure 9 shows another random configuration to which total traveling distance in an arbitrary direction is applied 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 sequence composed of just 10 segments, it can keep twisting skillfully and produce a steady advance.



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

(i) Crawling motion at generation 27.



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

(ii) Lift-and-swing motion at generation 40.

Fig. 6: The evolved motions using the moving distance along the arrows as fitness function. (i) After rolling itself (b), the robot crawls using friction (c,d). (ii) The central module lifts the other two (b) and swings them (c) to gain the distance (d).

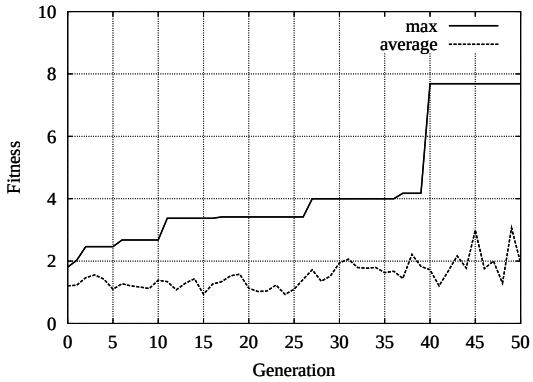
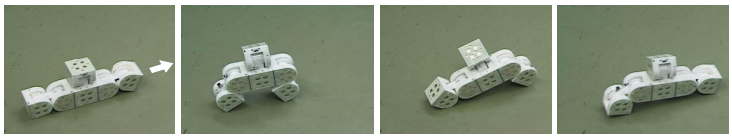
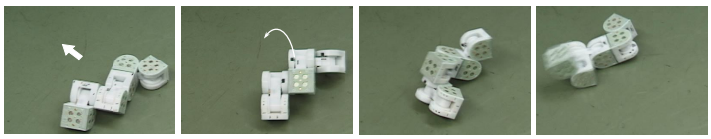


Fig. 7: Development of fitness function.



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

Fig. 8: The evolved motion using different fitness function. The robot realizes an inchworm locomotion.



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

Fig. 9: A whole-body motion evolved using the total moving distance as fitness function. (a) initial condition. The robot is rolling itself (b, c) and then total body is flipped over (d).

To summarize, we have shown that the proposed simple method can generate suitable motion sequences according to the given initial configurations and different fitness functions, in an adaptive and effective fashion. We have experimentally demonstrated that the evolved motion are physically feasible for the modular robot hardware M-TRAN.

5 ALPG – Locomotion Pattern Evolution using CPG

We are also developing another evolutionary method ALPG dedicated to locomotion generation for a fixed configuration, while ERSS aims evolution of motion sequence including configuration changes [11]. By focusing on the acquisition of oscillatory pattern using CPG, the method can deal with the evolution of locomotion for many-module structures more easily than ERSS method.

In biomimetic research field there have been a number of research on generation of biped or quadruped locomotion using neural oscillators [16, 17]. It is considered possible to apply the same principle to the modular robots to generate locomotive motions.

In this method, for a fixed configuration, each module's motor is controlled by an assigned oscillator, which is connected to other modules' oscillators. Using GA, the locomotion pattern is evaluated based on fitness function to optimize the such parameters as connection weights between oscillators. The generated locomotion pattern is finally transferred in the hardware to verify its effectiveness.

6 ALPG Implementation and Experiments

6.1 Neural Oscillator Model

We applied a neural oscillator as a model of the Central Pattern Generator (CPG) as shown in Fig. 10 [16, 17], where each neuron is represented by the non-linear differential equations (1) for total N modules. Each module's motor has its own CPG and is controlled directly by the CPG output expressed in (2).

$$\begin{aligned}\tau\dot{u}_{\{1,2\}i} &= -u_{\{1,2\}i} + w_{12}y_{\{2,1\}i} - \beta v_{\{1,2\}i} + u_e \\ &\quad + f_{\{1,2\}i} + s_{\{1,2\}i} \\ \tau'\dot{v}_{\{1,2\}i} &= -u_{\{1,2\}i} + y_{\{1,2\}i}\end{aligned} \tag{1}$$

$$\begin{aligned}y_{\{1,2\}i} &= \max(0, u_{\{1,2\}i}) \\ f_{1i} &= k(\mathit{angle}(t)_i - \mathit{initial\_angle}_i) \\ f_{2i} &= -f_{1i} \\ s_{\{1,2\}i} &= \sum_{j=1}^{n} \mathit{weight}_{ij} y_{\{1,2\}j} \\ \mathit{Output}_i &= p_1 y_{1i} + p_2 y_{2i}\end{aligned} \tag{2}$$

where subscripts 1, 2 corresponds to extensor and flexor neurons respectively (Fig. 10). For CPG_i of total $2N$, u_i and v_i are the internal states, y_i is the output of the neuron, f_i and s_i are the feedback signals from each motors and other neurons, and weight_{ij} denotes the connecting weight between CPG_i and CPG_j. Other parameters are constant values; time constants $\{\tau, \tau'\}$ of u_i, v_i are $\{0.05, 0.6\}$, fatigue coefficient of neuron β=1.5, connection weight between neurons w_{12}=2.5, output weight of neurons p_1, p_2=1.25, the feedback coefficient of motor angles k=8, and is an external input u_e=8.5.

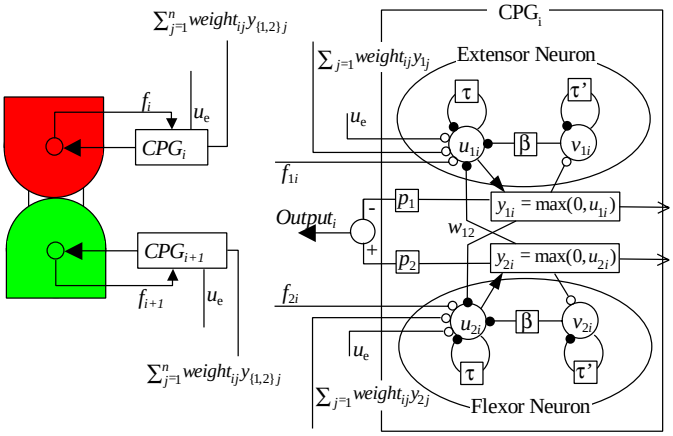


Fig. 10: Schematic view of neural oscillator.

6.2 Evolution of CPG Network using GA

Genetic Representation

For the CPG network to output adaptive locomotion pattern, the initial values $u_{0\{1,2\}i}$ and $v_{0\{1,2\}i}$ of each CPG and the connection weights weight_{ij} are evolved together by using GA. The initial values are important for converging the oscillation of the CPG to a limit cycle attractor smoothly. The connection weights determine the phase differences between the oscillations of CPGs and make the limit cycle robust against external disturbances.

The values of $u_{0\{1,2\}i}$ and $v_{0\{1,2\}i}$ are real numbers from -8.0 to 8.0 and from 0.0 to 3.0. The connection weights between CPGs, weight_{ij}, take discrete values, -1 (inhibitory connection), 0 (no connection) or 1 (excitatory connection) to narrow the search space [17]. Individual genotypes represented as a string of those values and crossover is applied separately to initial values and conneciton weights.

Evaluation and Selection using Fitness Function

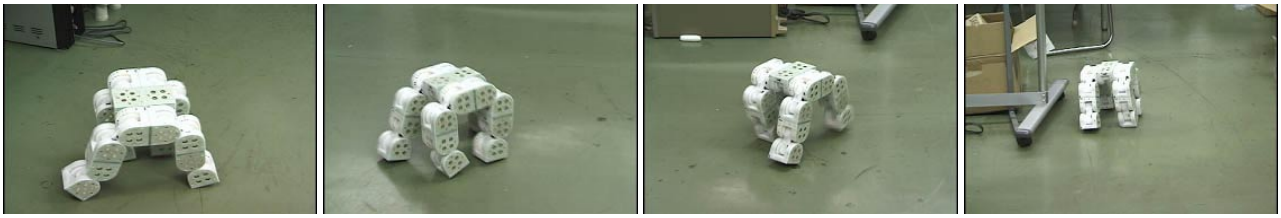
Locomotion by each individual is evaluated by the fitness function represented by (3) in 15 seconds in simulation space, where a, b, and c are weighting coefficients and are fixed to 200, 250, and 0.47. Here, *length* is the moving distance of the center of gravity in a fixed direction, *width* width is the deviation from the direction, *energy* is the energy consumption during the motion, and N is the total number of modules. The fitness function is designed for the straight locomotion in the specified direction.

$$\mathit{fitness} = a \cdot \mathit{length} - b \cdot \mathit{width} - c \cdot \mathit{energy}/N \tag{3}$$

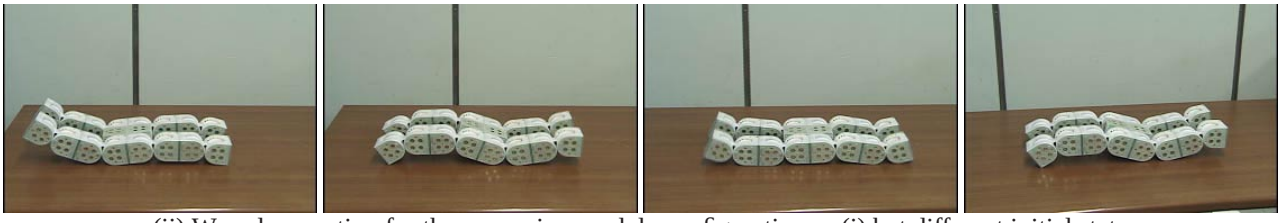
In the selection process, first 40% of fittest individuals are preserved in the next generation as elites. Then the rest 60% of individuals are generated through crossover operation of two individuals chosen arbitrarily from those elites. Mutation is finally applied to the newly generated individuals with the rate of 0.05. GA simulation is applied to the population of 150 individuals. The simulation terminates when the average fitness converges within certain range, or the generation exceeds the maximum number 150.

6.3 Experimental Results of Evolved Pattern

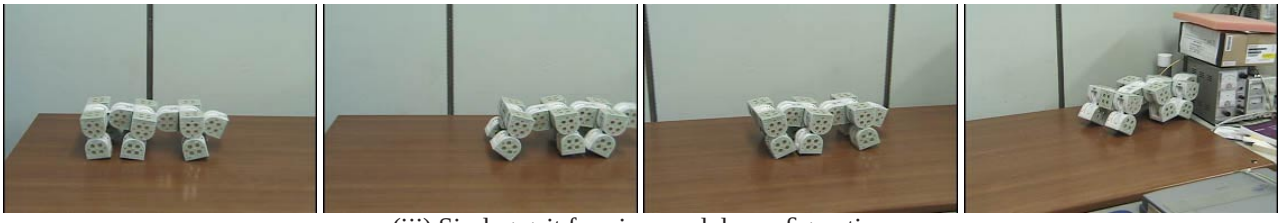
We applied the ALPG method to the various module configurations to the hardware modules as shown in Figure 11. The initial states of Fig. 11(i) and (ii) have the same connective configuration but the different initial motor angles.



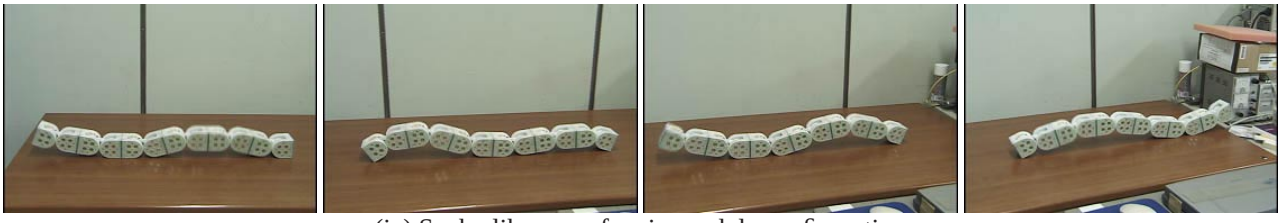
(i) Four-leg gait for nine-module configuration



(ii) Wave locomotion for the same nine-module configuration as (i) but different initial state



(iii) Six-leg gait for nine-module configuration



(iv) Snake-like wave for six-module configuration

Fig. 11: Hardware experiments of various evolved locomotion patterns.

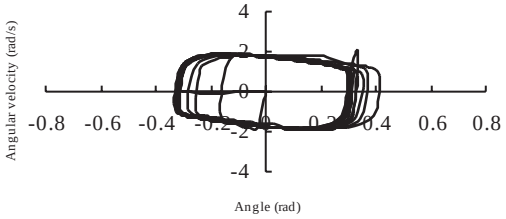
This difference leads to different locomotion pattern, one is a gait pattern and the other is a wave-like motion. For other configurations, stable locomotion patterns have been obtained successfully for all the configurations.

Figure 12(a) illustrates the simulation results of motion in the phase space (the angle and angular velocity) for one of the motors depicted in Fig.11(i) and Fig. 12(b) shows the cyclic change of all the motors angle. The motor angle oscillates with a constant frequency (about 1.2Hz), amplitude and phase difference, making the locomotion pattern stable.

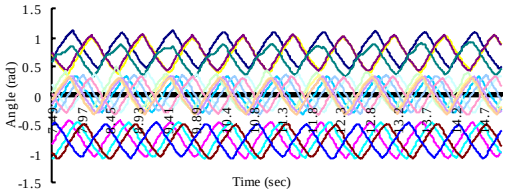
The above experiments verified that the proposed evolutionary method can effectively and robustly generate appropriate locomotion pattern to various configuration.

7 Conclusions

In this paper we presented two methods of evolutionary motion synthesis for a modular robot, ERSS (Evolutionary Reconfiguration Sequence Synthesis) and ALPG (Automatic Locomotion Pattern Generation), and verified their effectiveness experimentally using recently developed robot M-TRAN. In ERSS, the motion sequence is described as a series of segments specifying both dynamic motion and configuration changes. This sequence description can easily be translated into genetic representation to which GA (Genetic Algorithms) is applied. Using the traveling distance as the fitness function, interesting full-body dynamic motions were evolved through dynamic simulation. Their feasibility and validity was verified through experiments using M-TRAN II hardware. The other method ALPG concentrates on the evolution of locomotion pattern using CPG (Central Pattern Generator) for fixed configurations. We adopt a CPG model using mutually connected oscillators whose output controls each module's motions. The network between CPG evolves through GA also based on traveling distance. For various different configurations,



(a) Relationship between angle and angular velocity.



(b) Transition of all the joint angles.

Fig. 12: Movement of one joint in the pattern in Fig. 11(i).

efficient periodical locomotion patterns were evolved in simulation world. Their effectiveness were also experimentally demonstrated.

One of the important issues of future work is integration of sensors into self-reconfigurable modular robots. Software aspect is equally significant for efficient reconfiguration planning and motion generation. We will also address the method of generic behavior decision for the modular robots based on distributed or hierarchical approach, by appropriately combining the developed evolutionary computations in pursuit of adaptive and robust operation of modular robots.

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, et al.: "Autonomous and Self-Sufficient CONRO Modules for Reconfigurable Robots," *Distributed Autonomous Robotic Systems 4*, Springer, 155–164, 2000.

[4] C. Ünsal, et al.: "A modular self-reconfigurable bipartite robotic system: Implementation and Motion Planning," *Autonomous Robots*, **10**-1, 23–40, 2001.

[5] M. Yim, et al.: "Distributed Control for 3D Metamorphosis," Autonomous Robots **10**-1, 41–56, 2001.

[6] D. Rus and M. Vona. "Crystalline Robots: Self-reconfiguration with Compressible Unit Modules," *Autonomous Robots*, **10**-1, 107–124, 2001.

[7] S. Murata, et al. : "M-TRAN: Self-reconfigurable Modular Robotic System," *IEEE/ASME Transactions on Mechatronics*, **7**-4, 431–441, 2002.

[8] E. Yoshida, et al: "A Motion Generation Method for a Modular Robot," *Journal of Robotics and Mechatronics*, **14**-2, 177–185, 2002.

[9] E. Yoshida, et al.: "Evolutionary Motion Synthesis for a Modular Robot using Genetic Algorithm," *J. Robotics and Mechatronics*, to appear.

[10] E. Yoshida, et al: "A Self-Reconfigurable Modular Robot: Reconfiguration Planning and Experiments," *Internal Journal of Robotics Research*, **21**-10, 903–916, 2002.

[11] A. Kamimura, et al. : "Automatic Locomotion Pattern Generation for Modular Robots," *Proc. 2003 IEEE Int. Conf. on Robotics and Automation*, to appear.

[12] M. Asada, et al: "Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning," *Machine Learning*, **23**-2/3, 279–303, 1996.

[13] H. Kimura and S. Kobayashi: "Reinforcement Learning for Locomotion of a Two-linked Robot Arm," *Proc. of the 6th European Workshop on Learning Robots (EWLR-6 1997)*, pp.144–153, 1997.

[14] J.-C. Latombe: Robot Motion Planning, *Kluwer Academic Press*, 1991.

[15] S. M. LaValle and J. J. Kuffner: "Rapidly-Exploring Random Trees: Progress and Prospects," *Int. Journal of Robotics Research*, **20**-5, 378–400, 2001.

[16] 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.

[17] H. Kimura, et al.: "Realization of dynamic walking and running of the quadruped using neural oscillator," Autonomous Robots, **7**-3, 247–258, 1999.

[18] Karl Sims: "Evolving Virtual Creatures," *Computer Graphics, Annual Conference Series (SIGGRAPH '94 Proceedings)*, 15–22, 1994.

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

[20] 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.