Automatic Locomotion Design and Experiments for a Modular Robotic System

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

***Abstract*—This paper presents a design method and experiments for whole-body locomotion by a modular robot. There are two types of locomotion for modular robots: a repeating self-reconfiguration and whole-body motion such as walking or crawling. For whole-body locomotion, designing a control method is more difficult than for ordinary robots because a modular robotic system can form various configurations, each of which has many degrees of freedom. This study proposes a unified framework for automatically designing an efficient locomotion controller suitable for any module configuration. The method utilizes neural oscillators (central pattern generators, CPGs), each of which works as a distributed joint controller of each module, and a genetic algorithm to optimize the CPG network. We verified the method by software simulations and hardware experiments, in which our modular robotic system, named M-TRAN II, performed stable and effective locomotion in various configurations.**

***Index Terms*—Central pattern generator (CPG), evolutionary computation, locomotion, modular robot, neural oscillator network.**

I. Introduction

In RECENT years, hardware and software experiments have addressed the feasibility of reconfigurable robotic systems [1]–[18]. Existing reconfigurable modular robots comprise homogeneous or heterogeneous robotic modules, whose configuration and locomotion patterns can change according to given tasks or the environment. Self-reconfigurable systems are able to self-adapt to the external environment by changing configurations, as well as self-repair through replacement of disabled parts with spare modules. The unique capabilities of self-reconfigurable modular robots are thought to be applicable in extreme or unknown environments, such as on distant planets, in deep seas, inside nuclear plants, and in disaster areas for exploration or search-and-rescue operations where human access is difficult. Our goal is to realize such a versatile self-reconfigurable modular robotic system by studying both module hardware designs and control algorithms for adaptive structure formation and behavior.

Manuscript received August 25, 2003; revised December 14, 2004. This study was supported by the Science and Technology Research Grant Program for Young Researchers with a Term from the Ministry of Education, Culture, Sports, Science, and Technology (MEXT) of Japan. Recommended by Technical Editor M. Meng.

A. Kamimura, H. Kurokawa, E. Yoshida, K. Tomita, and S. Kokaji are with the National Institute of Advanced Industrial Science and Technology (AIST), Ibaraki 305-8564, Japan (e-mail: kamimura.a@aist.go.jp; kurokawa-h@aist.go.jp; e.yoshida@aist.go.jp; k.tomita@aist.go.jp; s.kokaji@aist.go.jp).

S. Murata is with the Tokyo Institute of Technology, Yokohama 226-8502, Japan (e-mail: murata@dis.titech.ac.jp).

Digital Object Identifier 10.1109/TMECH.2005.848299

The main research topics on the self-reconfigurable modular robotic systems have been module design, development of module hardware, distributed algorithms for two- and three-dimensional structural formation [9], [10], [14]–[18], and locomotion by modules.

There are two types of modular robot locomotion. One type is a repeating configuration change, e.g., individually sending a module from the tail of the module structure to the head [7], [16]–[18]. The other is whole-body locomotion, such as walking and crawling, which is achieved by controlling joint motors coordinately without any configuration change [7], [11], [13]. In this paper, we deal with whole-body locomotion and develop a unified framework applicable to any given module configuration for generating a locomotion pattern and controller.

The following aspects should be considered for whole-body locomotion by modular robots.

1) Design of a locomotion controller (centralized or decentralized)
2) A method for generating locomotion structures and patterns.

An asynchronous decentralized controller is desirable to reduce calculation cost on each module.

Few studies have dealt with whole-body locomotion for modular robots. Yim demonstrated caterpillar-like locomotion and a rolling track in which each module motion is based on the specific gait control table corresponding to each module configuration [13]. Shen, Salemi, and Will proposed a distributed control method to accomplish locomotion and self-reconfiguration using a unified framework, which is based on hormonelike message propagation and a rule base for a particular global behavior [11]. We demonstrated various locomotive motions and reconfigurations using our M-TRAN I module. We used the same control method used by Yim, in which several locomotive motions and reconfigurations were achieved mechanically [7]. Although these studies provide a good means to control a multidegree of freedom system like modular robots, they do not describe how to make gait control tables or rule bases suitable for various module configurations.

Sims [19] and Lipson and Pollack [20] proposed generation methods for both robotic structures and locomotion patterns (locomotion controllers) with an evolutionary computation method. They succeeded in generating various simple robotic structures that can move in unique locomotion patterns.

In the biomimetic research field, several studies have addressed generating biped or quadruped locomotion by using neural oscillators [21]–[23], where phase differences between joint angles are determined by interactions among connected

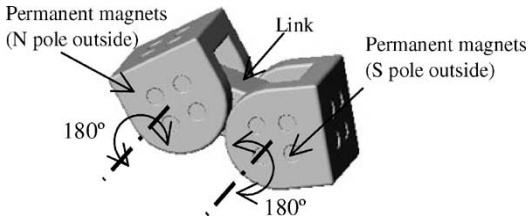

Fig. 1. Schematic view of an M-TRAN module.

nonlinear oscillators called central pattern generators (CPGs). There are several studies that automatically find a locomotion pattern by optimizing CPG parameters by using the evolutionary computation method [24], [25]. These methods are promising because they are asynchronous, decentralized, and suitable for modular robots.

The possible modular robot locomotion structures and patterns are infinite because this robot has many joints, and the scale and shape of the module structure are changeable. Designing a stable or optimal locomotion pattern only for a fixed-module configuration is difficult, in general because there are many degrees of freedom. Therefore, in this study we focus on generating whole-body locomotion by modular robots and develop a unified framework for generating practical locomotion patterns for any given module configuration. To realize this, we applied the CPG model ([21]–[23]) as a distributed locomotion controller and generalized the model to be applicable to a scalable module structure. We also developed software combining a dynamics simulator with an evolutionary computation method to evaluate module locomotion performance and to maximize efficiency of the locomotion controller.

We completed hardware experiments to demonstrate the feasibility of the proposed method. The obtained locomotion controller based on the neural oscillator model was implemented, and stable module locomotion was achieved in a distributed manner.

The contents of this paper are as follows. The Section II presents an overview of our M-TRAN module. In Section III, a distributed locomotion controller based on the CPG model and details of automatic locomotion pattern generation (ALPG) software are clarified, and simulation results for locomotion design are provided. Section IV describes the hardware design of M-TRAN II and describes hardware experiments on various locomotion patterns. The obtained results are discussed in Section V, and conclusions are provided in Section VI.

II. Overview of M-TRAN Module

We study whole-body locomotion for modular robots using our self-reconfigurable modular robot M-TRAN (Modular Transformer) (shown in Fig. 1). This module comprises three components: two semicylindrical parts and a link part. Each semicylindrical part can rotate from -90° to 90° independently using a geared motor embedded in the link. Each semicylindrical part has three connecting surfaces with permanent magnets. The modules can connect with each other by magnetic force because the polarity of the magnets between the two parts differs.

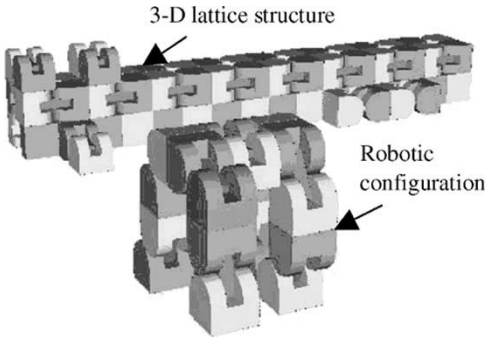

Fig. 2. Example of possible configurations, a three-dimensional lattice structure above and a robotic configuration below. Each semicylindrical part of the module is shown in a different color, and checkerboard-like structures can be seen.

Each connecting surface can be connected to another connecting surface in every orthogonal relation; thereby, various lattice structures are formed easily, as illustrated in Fig. 2. The lattice structure can be reconfigured by changing positions of the semicylindrical parts, through repetition of simple procedures such as detaching the connection, rotating the semicylindrical part, and reconnecting.

III. Automatic Locomotion Pattern Generation Method

This section presents the method that finds a practical locomotion pattern for a given module configuration. An efficient locomotion pattern, realized by a proper locomotion controller, is automatically generated by genetic algorithm (GA). The GA optimizes the locomotion controller by evaluating the performance of locomotion by dynamic simulations.

A. Overall Structure for Generating the Locomotion Pattern

We constructed automatic locomotion pattern generation software (ALPG hereafter), combining Vortex (CM Labs Simulations, Inc.) as a three-dimensional dynamic simulation library, a dynamic model of the M-TRAN II module, a decentralized locomotion controller based on a CPG model (Section III-B), and an optimization method for a CPG network using GA (Section III-D). The ALPG software outputs an efficient locomotion pattern for any given configuration moving in a straight line in a certain direction.

Fig. 3 presents a flow chart of the ALPG software. The module configuration and the initial posture are determined first. Here, "configuration" means a connecting relationship between modules and "posture" means a set of joint angles for a specific configuration. In the dynamics simulation, a determined module configuration and posture are placed on a flat and horizontal ground in the virtual world. Performance of the locomotion pattern is evaluated sequentially by using a fitness function, and the GA optimizes the CPG network. The obtained results can be verified by hardware (Sections IV-C and IV-D).

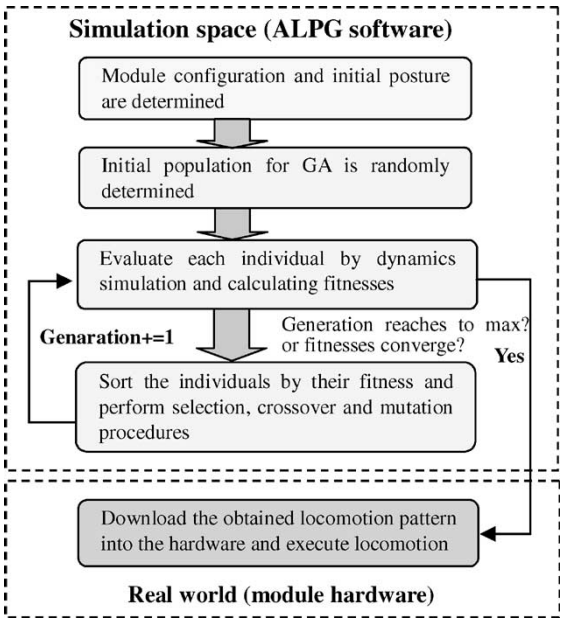

Fig. 3. Flow chart of the ALPG software.

B. Decentralized Locomotion Controller Based on the CPG Model

A locomotion controller for each module drives two joints according to the outputs of two CPGs as shown in Fig. 4. The interactions among CPGs enable a cooperative motion by modules.

We applied the CPG model used in [21]–[23] and generalized the model to be applicable to a multidegree of freedom system with an arbitrary number of modules and with any configuration. The CPG is composed of the same two inhibitively connected neurons corresponding to extensor and flexor (Fig. 4). Each CPG, described by (1) and (2), is a nonlinear oscillator having four state variables $(u_{1i}, v_{1i}, u_{2i}, v_{2i})$.

$$\begin{cases} \tau \dot{u}_{1i} = -u_{1i} - w_0 y_{2i} - \beta v_{1i} + u_{\mathrm{e}} + f_{1i} + a \cdot s_{1i} \\ \tau' \dot{v}_{1i} = -v_{1i} + y_{1i} \end{cases}$$
$$y_{1i} = \max(0, u_{1i}), \quad i = 0, \ldots, \mathrm{num} - 1 \tag{1}$$

$$\begin{cases} \tau \dot{u}_{2i} = -u_{2i} - w_0 y_{1i} - \beta v_{2i} + u_{\mathrm{e}} + f_{2i} + a \cdot s_{2i} \\ \tau' \dot{v}_{2i} = -v_{2i} + y_{2i} \end{cases}$$
$$y_{2i} = \max(0, u_{2i}), \quad i = 0, \ldots, \mathrm{num} - 1 \tag{2}$$

where the subscripts 1 and 2 represent extensor and flexor, and num represents the number of joints, i.e., CPGs (twice as many as the number of modules). In those equations, y_i is the output of each of the two neurons, u_{e} is an external input with a constant value, and w_0 is a connecting coefficient between extensor and flexor neuron. The system without the last two terms of the first equation of both (1) and (2) represents a self-excited oscillator. τ and τ' are time constants of this oscillator, and the cycle becomes longer as those variables become larger. Williamson analyzed that the natural frequency of the oscillator is proportional to $1/\tau$ if the ratio between τ and τ' is constant [26]. It was also reported that oscillation amplitude is nearly proportional to the external input u_{e} [21]. By adding the two feedback terms, the oscillation of the system is entrained and the system works cooperatively with others.

The variables s_1 and s_2 in the first equations of (1) and (2) represent the CPG interaction calculated by (3) and (4).

$$s_{1i} = 2.0 \cdot \left\{ 1 + \exp\left(\frac{-\mathrm{feed}_{1i}}{\mathrm{num}} \right) \right\}^{-1} - 1.0$$
$$\mathrm{feed}_{1i} = \sum_j \mathrm{weight}_{ij} u_{1j} \tag{3}$$

$$s_{2i} = 2.0 \cdot \left\{ 1 + \exp\left(\frac{-\mathrm{feed}_{2i}}{\mathrm{num}} \right) \right\}^{-1} - 1.0$$
$$\mathrm{feed}_{2i} = \sum_j \mathrm{weight}_{ij} u_{2j} \tag{4}$$

where weight_{ij} is a connection weight between the ith and jth CPGs. feed_{1i} and feed_{2i} are weighted sums of the state variables (u_1, u_2) of the ith CPG. The same connection weight is applied to both u_1 and u_2. The variable s is normalized from -1.0 to 1.0 by the sigmoid function after feed is divided by num, the number of joints, to be applicable to any size of system.

A locomotion controller drives two joint motors according to outputs of two CPGs. The control signals to motors are calculated by (5) and defined as a voltage input to general DC motors. In the following simulation, the dynamics model of the motor implemented in the Vortex simulator was used.

$$\mathrm{Output}_i = -m_1 y_{1i} + m_2 y_{2i}. \tag{5}$$

The motion of each joint is directly fed back to (1) and (2) as an angle deviation f between a current angle and the nominal angle calculated by (6).

$$\begin{cases} f_{1i} = k(\mathrm{angle}_i - \mathrm{nominal\_angle}_i) \\ f_{2i} = -f_{1i} \end{cases} \tag{6}$$

where k is a feedback coefficient, angle_i represents a current angle of ith joint, and the $\mathrm{nominal\_angle}_i$ is a nominal one, which is given by an initial angle of the joint. The feedback term f provides the oscillation around the nominal angle. The amplitude of oscillation and the cycle become smaller when k increases.

The CPG dynamics calculation and the dynamics simulation of the physical system by the Vortex simulator are achieved independently in the calculation step (15 ms). In each step, the CPG dynamics is calculated by using the Runge–Kutta method, and the results are used for driving the joints in the virtual world. The resultant module locomotion is evaluated by the GA process.

C. Basic CPG Behavior

Without connection, each CPG oscillates independently. When connected, all the CPGs oscillate together and converge to a specific pattern (limit cycle) determined by the network connection. Such behavior is widely known as a locking phenomenon or entrainment among connected nonlinear oscillators.

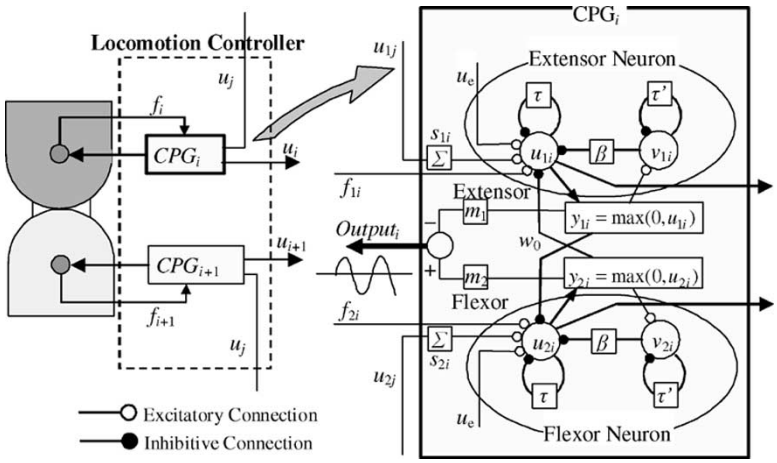

Fig. 4. Schematics illustrations of the locomotion controller (left) and details of the CPG (right). The locomotion controller controls rotation of two joints according to outputs of two CPGs.

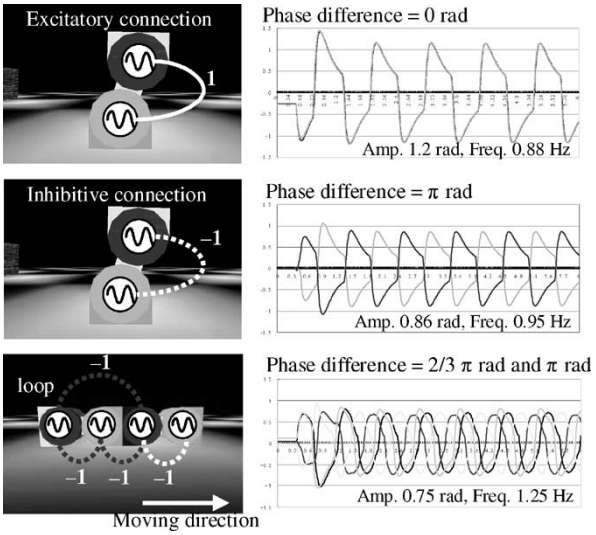

Fig. 5. Basic CPG behavior and a locomotion example by two modules. Graphs show CPG output versus time.

Moreover, each CPG physically interacts with the modules' dynamics through the motor and such an interaction affects the limit cycle. This is called global entrainment.

In our model, only three types of connection between CPGs (weight_{ij}) are used so that the GA can efficiently search a CPG network for locomotion: 1, excitatory connection; -1, inhibitive connection; or 0, no connection.

In the following, basic CPG behavior is illustrated with the simple examples shown in Fig. 5.

1) Case I: Two CPGs With Connection 1: As shown in the top graph in Fig. 5, two CPGs synchronize together (in-phase). In this case, the phase difference between CPGs always converges to 0 starting from any initial CPG state.

2) Case II: Two CPGs With Connection -1*:* In this case shown in the middle of Fig. 5, the phase difference between two CPGs always converges to π (antiphase).

TABLE I
PARAMETERS FOR CPG AND GA

Parameters for CPG	value	Parameters for GA	value
τ	**0.05**	pop_size	150
τ'	**0.6**	max_gene	150
β	1.5	s_rate	0.6
a	3.0	m_rate	0.05
m_1, m_2	0.125		
k	**8**		
w_0	2.5		
u_e	**8.5**		

3) Case III: Three CPGs in a Loop: When several CPGs are connected in a loop by connection -1, phase differences between neighboring CPGs converge to $2\pi/n$ (n, number of CPGs in the loop). In the bottom of Fig. 5, phase differences of the left three CPGs converge to $2\pi/3$ because of this loop. This CPG network makes the two-module structure move forward or back by a caterpillar-like motion according to the initial state of the CPGs. That is, two attractors exist in this case (only one is shown in the figure).

These three results also demonstrate the effect of global entrainment. While the eigen frequency of the stand-alone CPG is 0.55 Hz, the three results are 0.88, 0.95, and 1.25 Hz, for the conditions shown in Table I. The frequency and amplitude of joint motion change according to the network connection, inertia of the mechanical structure, and effects of external disturbances.

As described above, CPGs can produce various phase differences autonomously in accordance with the CPG network and physical motion of modules.

D. Evolutionary Computation

We implemented a GA in the ALPG software to find automatically the optimal CPG network for locomotion. The connection weights (weight_{ij}) and the initial values of the CPGs,

$[u_{1i}(0), v_{1i}(0), u_{2i}(0), v_{2i}(0)]$ are coevolved by using the GA. As discussed in Section III-C, the connection weights determine the phase differences among joints and the shape and stability of the limit cycle. The initial values have no relation to the locomotion pattern. However, they are important parameters for smooth convergence to a limit cycle when starting from the initial shape to the locomotive motion.

The constants used in the simulation are summarized in Table I, with important constants boldfaced. Each value is determined by trial and error, considering mechanical properties such as maximum motor torque and speed, and weight and inertia. It is possible to implement these values as variables in the GA process, but using them will expand the search space. Although it is also possible to use different values of k or m for each module, they are identical here to take into consideration module exchangeability and scalability.

By the following GA processes, locomotion patterns for moving straight with little energy consumption are obtained by optimizing both connection weights and initial values of CPGs.

1) Parameter Optimization: Connection weights among CPGs (weight_{ij}) are selected from three values, -1 (inhibitive connection), 0 (no connection), and 1 (excitatory connection). In the CPG model used in this study, the sign of the connection weight is essential to determine phase differences. Therefore, we express the connection weights in discrete values even though connection weights can be expressed in real numbers. Initial values of CPGs $(u(0), v(0))$ are real numbers from -8.0 to 8.0 and from 0.0 to 3.0. These ranges were determined empirically by repeating simulation.

2) Fitness Evaluation: In the first stage of the GA, connection weights and initial values of CPGs are randomly initialized by the population size, pop_size (see Table I). Virtual locomotion after a fixed interval (15 s) is evaluated individually by using the fitness function represented by (7).

$$\mathrm{fitness} = a \cdot \mathrm{length} - b \cdot \mathrm{width} - \frac{c \cdot \mathrm{loss}}{\mathrm{num}} \tag{7}$$

where a, b, and c are weighting coefficients and are fixed to 200, 250, and 0.47, considering the balance among the following three parameters: 1) *length* is the moving distance of the center of gravity, 2) *width* is the maximum deviation from the straight line, 3) *loss* is the energy loss, which is an accumulated value of the total consumed energy by all the motors (we used a time integration of the product of the torque and the angular velocity of each joint) during the evaluation interval. The GA searches for locomotion parameters with higher fitness values. The fitness function above leads the module structure to move faster along the straight line with less energy consumption.

3) Selection, Crossover, and Mutation Procedures: Each individual is sorted by the fitness value after the evaluation procedure for all individuals in one generation. The lower groups are deleted according to the selection rate, s_rate (see Table I).

A crossover procedure is achieved to fill the deleted parts by selecting parents from the remaining individuals by using a roulette selection method. The crossover procedures for connection weights and initial values are achieved separately because the former are discrete whereas the latter are continuous. The

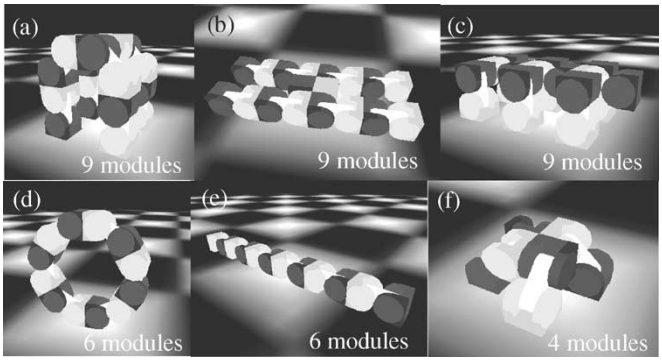

Fig. 6. Examples of tested configurations. (a) Four-legged configuration. (b) H-shaped structure whose configuration is the same as (a). (c) Six-legged configuration. (d) Wheel configuration. (e) Thread configuration. (f) Another type of four-legged configuration.

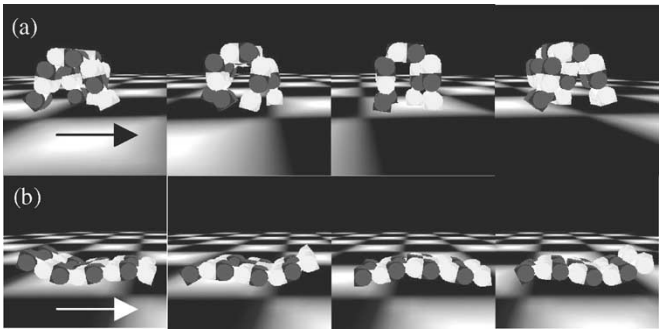

Fig. 7. Obtained locomotion patterns. (a) Walking pattern. (b) Wave-like pattern. For configurations in Fig. 6(a) and (b). Arrows in the figure show the moving direction.

N-point crossover method is used for the connection weights (weight_{ij}), while the unimodal normal distribution crossover (UNDX) method [27] is used for the initial values. The latter method is known to be superior for optimizing multimodal functions, i.e., functions with many local minima.

After the crossover procedures, several individuals are selected randomly according to the mutation rate, m_rate (see Table I). In the mutation procedure, the initial values $[u(0), v(0)]$ of the selected individual are varied in a narrow range and a part of the connection weight matrix is initialized randomly to -1, 0, or 1.

The evaluation procedure then restarts with the new generation. The GA process stops when the number of generations exceeds a maximum number of generations, max_gene (see Table I), or the fitness average becomes constant. A quasioptimized locomotion pattern will emerge after the GA processes above are repeated.

4) Application to Various Module Configurations: A locomotion controller suitable for locomotion in any module configurations can be produced by the ALPG. To show the feasibility of this method, we applied it to various module configurations (Fig. 6). Stable locomotion was realized for all configurations. The obtained locomotion patterns by two structures in Fig. 6(a) and (b) are shown in Fig. 7. These two patterns differ completely despite their identical configurations (connecting relationship

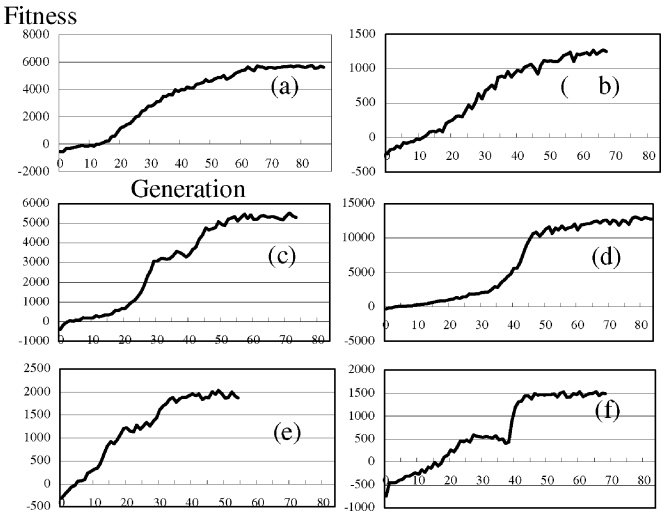

Fig. 8. Fitness versus generation curves for configurations shown in Fig. 6.

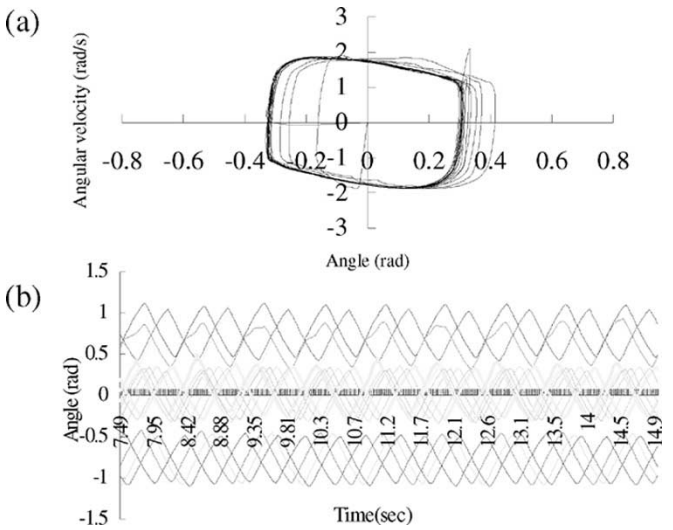

Fig. 9. (a) Phase space trajectory (angle versus angular velocity) of the base joint of the front leg in Fig. 6(a). (b) All joint angle trajectories for the motion in Fig. 7(a). The oscillation cycle is 1.15 Hz.

between modules). The former is a walking pattern and the latter is a wavelike motion. This result shows that the initial posture affects the locomotion pattern: walking is efficient for legged configurations and a wavelike motion is better for a flattened configuration. This comes from feedback terms, $f_{1,2}$, in (1) and (2).

Fig. 8 plots the fitness versus generation curve of each configuration from Fig. 6(a)–(f). Each fitness value is calculated in the same metrics, and it is confirmed that the fitness value of the wheel shape [Fig. 6(d)] is the best among the six module structures, i.e., most effective for moving rapidly with lower energy on flat ground. Almost the same result (fitness value) was obtained for the four-legged configuration after several trials.

Fig. 9(a) illustrates motion in the phase space (the angle and angular velocity) of the front leg base joint in Fig. 6(a). Fig. 9(b) shows the cyclic change of every motor angle in Fig. 6(a). Every motor angle oscillates with a constant frequency (1.15 Hz) and a fixed-phase difference by entrainment among CPGs and the mechanical structure. For stable locomotive motions, both entrainment between CPGs and entrainment between mechanical structures and CPGs (global entrainment) are important. In other words, when the pendulum swing of the mechanical structure is not matched to the rhythm made by CPGs, locomotive motion becomes neither periodic nor stable. In our model, the rhythm made by the mechanical structure is applied to each CPG model as a change of the rotation angle of each joint represented by (6).

All oscillation cycles of the obtained locomotion patterns are listed in Table II. The oscillation of the six-legged configuration, Fig. 6(c), is the slowest among six samples. This is due to the weight of the whole body being supported on only several of the six short legs; the cycle of the legs becomes long. As a result, all the oscillations are entrained and the cycle is long.

The software simulation above confirmed that the ALPG method can generate various locomotive motions according to the given module configurations. The required time for the ALPG to create a locomotion pattern depends on the number of modules and the number of contacts in the virtual world between modules and the ground at each step. It took about 6 h to obtain the stable walking pattern for the nine-module configuration in Fig. 6(a) with a 2.53 GHz Pentium 4 processor PC.

TABLE II
OSCILLATION CYCLES OF THE OBTAINED LOCOMOTION PATTERNS

Configuration	Cycle Hz
(a)	1.15
(b)	1.11
(c)	1.06
(d)	1.27
(e)	1.23
(f)	1.21

5) Analysis of the CPG Network and Stability: We analyzed the acquired CPG network for the four-legged configuration [Fig. 6(a)] and locomotion stability by the CPG network.

Fig. 10 illustrates the obtained connection network on CPG numbers 6 and 8, which are placed on the base joints of the front legs of the four-legged configuration. In that figure, nine modules and 18 CPGs are shown. The solid lines show an excitatory connection and the dotted lines show an inhibitive connection. A nearly symmetrical connection network is obtained by the evolutional computation. Fig. 11 shows the feedback signals, S_{16} and S_{18}, input to CPG numbers 6 and 8. The antiphase feedback signals are generated by a CPG interaction determined by the connection network, enabling the four-legged configuration to move straight with a trotting gait.

To confirm the stability of the locomotion made by the CPG network, we carried out a simulation in which an external force is added twice on the front right leg of the four-legged configuration while walking. Fig. 12 shows that the disturbed rhythm was soon recovered. This demonstrates that locomotion by CPGs is robust to external disturbances.

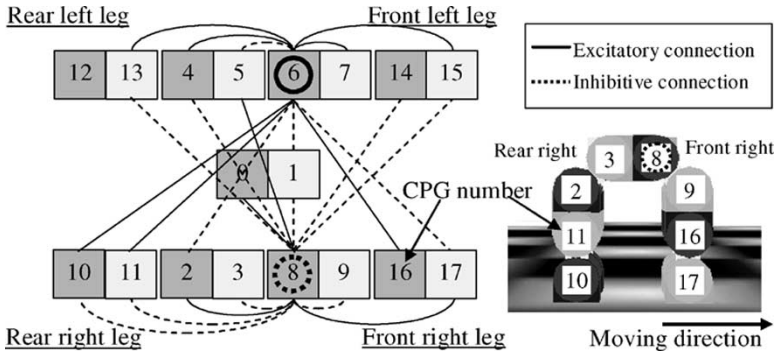

Fig. 10. CPG network connections of two CPGs (numbers 6 and 8) of the four-legged configuration in Fig. 6(a).

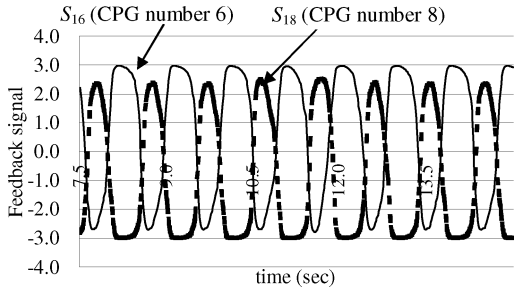

Fig. 11. Feedback signals S_{16} and S_{18} for the motion in Fig. 7(a).

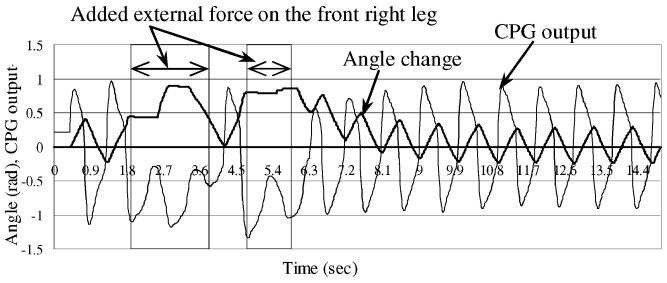

Fig. 12. Trajectories of the joint angle (in thick line) and the CPG output when external force is added on the front leg. The rhythm of the joint rotation is soon recovered after the external force is removed.

IV. Hardware Experiments

This section describes hardware experiments of modular robot locomotion. The developed module hardware named M-TRAN II and its control system are explained first. Two types of locomotion experiments are then presented.

The first locomotion experiments were carried out by downloading the obtained locomotion sequences (time series data) for all joints to the modules and playing back locomotion according to the sequence. The objective of the experiments is to confirm the validity of dynamic simulation by ALPG software.

The second locomotion experiments were carried out to achieve adaptive locomotion on varying terrain conditions by applying real-time CPG control. We compared results by playback control.

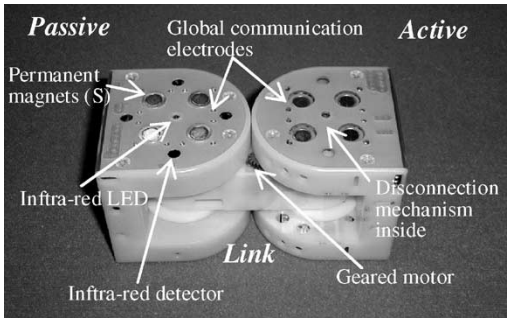

Fig. 13. M-TRAN II module.

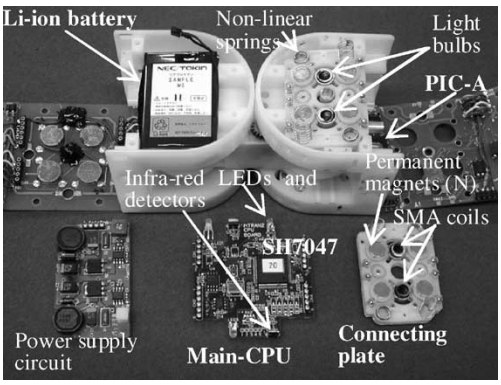

Fig. 14. Inner structure of the M-TRAN II module.

A. Development of M-TRAN II Module Hardware

We showed robotic motion and transformation performance using our modular robotic system, M-TRAN I [7]. The newly developed system (M-TRAN II), shown in Figs. 13 and 14, is much improved compared with M-TRAN I, especially in CPU processing speed, the communication system, the motor controller, maximum torque and speed, the power supply using a battery, power consumption, and size and weight. These improvements enabled real-time CPG control and stand-alone operations.

Fig. 13 shows two semicylindrical parts, called a passive part and an active part. The passive part has four permanent magnets on each of three surfaces (S pole outside). Global communication electrodes are placed symmetrically on the same surface. As shown in Fig. 14, the passive part contains a CPU circuit, power

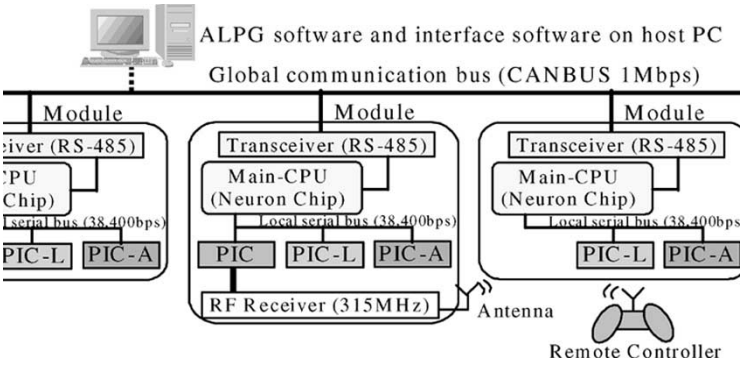

Fig. 15. Control and communication system used in M-TRAN II.

TABLE III
SPECIFICATIONS OF THE M-TRAN II MODULE

Item	Value
Dimension	60×120×60mm
Weight	0.4 kg (including battery)
CPU	SH7047 (Renesus) and two PICs
Global communication	CANBUS, 1 Mbps
Power supply (battery)	DC 3.8 V
Max. torque of each axis	1.9 N·m (rating)
Max. rotation speed	0.5π rad/s
Connecting force	83 N
Battery	Li-ion (3.8 V, 900 mAh)
Total power dissipation	0.4 W (8 V)
Proximity sensor	Infra-red LEDs and detectors
RF module	RF Solutions, Inc. Receiver 315 MHz

supply, and a battery. The CPU circuit includes a microprocessor (SH7047, Renesas Technology Corp.) called a Main-CPU. We applied a CANBUS (1 Mbps) system for intermodule communication. An infrared LED and a sensor are installed on each CPU circuit board and each passive connection surface as a proximity sensor. Each module can operate independently using its CPU and the battery. Currently, battery life is about 30 min.

The active part holds three disconnection mechanisms using permanent magnets, nonlinear springs, and shape memory alloy (SMA) coils based on an internally balanced magnetic unit (IBMU) [28]. A microprocessor (PIC16F873) called PIC-A is also installed for controlling connection/disconnection and checking connection status. The detailed explanations for the mechanism are provided in [29].

Two geared motors and their control circuit board are installed inside the link. The control circuit board includes a microprocessor (PIC16F877) called PIC-L, which either realizes a PID position control or directly drives the motor.

Table III summarizes M-TRAN II module specifications. More details regarding the mechanical and electrical design of the M-TRAN II module are available in [30].

B. Control and Communication System in M-TRAN II

Fig. 15 shows the communication system used in the M-TRAN II system. Each module is connected to a global communication bus by way of a Main-CPU with CANBUS (1 Mbps). The modules can utilize a virtual shared memory realized by this intermodule communication. The data written in the memory will be implicitly transferred to all the connected modules and each module can refer to the status of other modules only by reading its own memory.

Inside the module, two PICs (PIC-A and PIC-L) are connected to the Main-CPU by a local bus, each of which is controlled by the Main-CPU. The communication speed of the local bus is 38 400 bps.

Two of the 20 developed modules include an RF receiver (RF Solutions, Inc.) inside, and it is possible to send commands by remote control to all the connected modules. The received commands, such as switching directions, changing module configurations and locomotion modes are executed by all modules at the same time.

C. Hardware Implementation of the Controllers

We performed two types of locomotion experiments: one using a playback controller and the other using a real-time CPG controller.

In the playback control mode, each module selects a sequence by each location in an assembled module structure and plays back the sequence by positioning motors every 15 ms according to the sequence. All the modules start the playback simultaneously by the command from the remote controller. As a result, whole-body module locomotion can be achieved. There is no communication between modules in the playback control mode, i.e., each module is just repeating the playback of its own sequence.

In real-time CPG control mode, locomotive motions by modules are achieved by calculating CPG dynamics, expressed by (1)–(6) and driving the joints every 15 ms, similar to the ALPG software. In the real hardware, dynamics for every CPG dynamics are calculated in a distributed manner. Each module calculates its own two CPG dynamics corresponding to its two joints, and CPG interactions are realized by intermodule communication. The advantage of the proposed system is that calculation cost for locomotion does not change even if the number of modules increases. Communication cost will increase when the number of modules increases, but it is not a critical problem compared with calculation cost, because the current communication speed is fast enough.

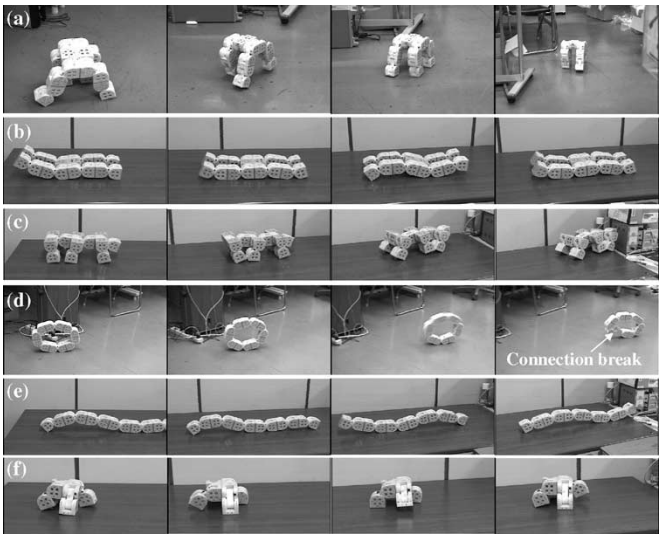

Fig. 16. Hardware experiments on various locomotion patterns (a) to (f) in the playback control mode. A part of the connection in the wheel configuration (d) was broken after a few rolls. In the experiments above, all modules operate on an internal battery and no tethers are attached.

TABLE IV
COMPARISON OF MOVING SPEEDS IN HARDWARE EXPERIMENTS AND IN SIMULATION

	Hardware experiments cm/s	Simulation cm/s
(a)	20	23
(b)	4.5	6.8
(c)	11	19
(d)	N/A	49.3
(e)	6.0	7.9
(f)	6.0	6.7

D. Locomotion Experiments in Playback Control Mode

Locomotion experiments were carried out by playing back the sequences with all the configurations (Fig. 6). Photos of the experiments are shown in Fig. 16. Movies of the hardware experiments and simulations, along with other experiments, are available from our web site [31]. All locomotive motions were achieved by real hardware on flat ground using the same conditions as in the simulation.

Table IV lists moving speeds of module structures in the hardware experiments and the simulations. These results show the validity of the dynamic simulation and the implemented model.

Locomotive motion by real hardware was not achieved for the wheel configuration [Fig. 6(d)]. One part of the connection was broken after several rotations. One reason for this failure is that the rhythmic joint motion diverged from the rolling motion (the entire rhythm) because there was no feedback from the physical system in the playback control mode. Excessive force thus occurred at one connection and exceeded the connection force.

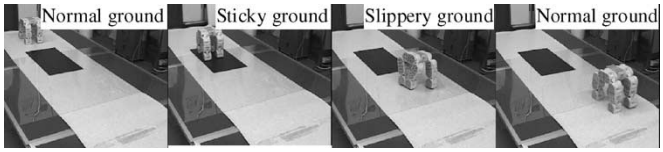

Fig. 17. Adaptation to normal, sticky, and slippery ground by four-legged robot with real-time CPG control.

E. Locomotion Experiments by Real-Time CPG Controller

We completed locomotion experiments on the four-legged configuration [Fig. 6(a)], the wheel configuration [Fig. 6(d)], and the thread configuration [Fig. 6(e)] by using the real-time CPG controller.

Fig. 17 shows locomotion of the four-legged configuration on normal, sticky, and slippery ground. Fig. 18 shows angle changes of hip joints while walking. In the playback control mode, hardware locomotion often failed, e.g., moving direction was changed drastically or the robot did not move forward, when experimental conditions such as friction and evenness of the ground differed from those of the simulations.

In the real-time CPG control mode, as shown in Fig. 17, the four-legged robot could adaptively walk on sticky and slippery ground. This is because walking steps were automatically regulated according to the conditions of the ground, and phase differences were always maintained by CPGs (Fig. 18).

Fig. 19 shows the rolling motion of the wheel configuration by using the real-time CPG controller. As shown in the figure, the wheel configuration could roll successfully on flat ground without breaking the connection. However, on a downslope or upslope, locomotion became unstable and eventually was out of control. Results are briefly discussed in Section VD.

Fig. 20 shows caterpillar-like locomotion along uneven terrain using the thread configuration. In the playback control mode, as the robot plays back the same locomotion pattern on the flat ground as in the simulation, the connection breaks or there is excessive motor load on uneven ground (Fig. 20). Using the real-time CPG controller, smooth locomotion was spontaneously realized because the locomotion pattern is autonomously regulated by both real-time CPG interaction and global entrainment.

V. DISCUSSION

This section discusses issues of the CPG connection used in ALPG software, a comparison between two locomotion controllers, and the necessity of external sensor information and an upper controller.

A. Structural Symmetry Consideration in GA

Introducing the same mechanical symmetry into the network structure may reduce the searching space for GA or lead to a better solution. Consequently, we carried out ALPG simulation taking into consideration the structural symmetry of the four-legged configuration shown in Fig. 10. In the simulation, CPG

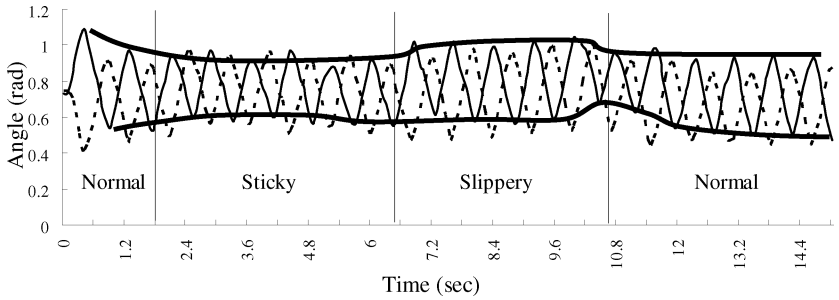

Fig. 18. Angle changes of hip joints. The range between the thick lines shows double the amplitude. The amplitude while walking on sticky ground was less than the amplitudes in other conditions. This means walking steps are automatically changed according to ground conditions. Phase difference is always maintained by the CPG interaction.

Fig. 19. Snapshots of rolling motion by the crawler configuration [Fig. 6(d)]. Rolling motion was successfully realized by the real-time CPG control.

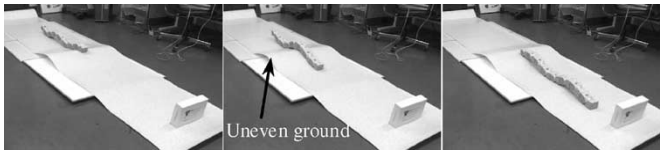

Fig. 20. Snapshots of caterpillar-like motion by the thread configuration (Fig. 6(e)) in the real-time CPG control mode. It successfully followed along the uneven part of the ground.

network connections from half of the CPGs (numbers 0, 10, 11, 2, 3, 8, 9, 16, 17) were optimized by GA. Connection weights of the facing CPG were symmetrically applied for the CPG network connections of the other half of the CPGs. Locomotion by the symmetrical CPG network was then evaluated in the GA process.

The resultant fitness value for the obtained locomotion after several trials was far less than the original results. A possible reason for this is that structure symmetry cannot be applied directly to the CPG network. However, a true reason has not been established, and further investigation is needed.

B. Locally Connected CPG Network

The current CPG controller is not applicable to modular robot systems such as CONRO [11], which have no global communication line. The CPG connection should be limited to neighboring modules for such a system. We tried ALPG simulation with this restriction, but could not obtain better locomotion in any configuration. The main reason for this is that the current CPG model cannot generate various phase differences without a loop. In future work, it will be important to propose a CPG model that can describe phase differences with neighboring CPGs directly.

C. Comparison Between Two Locomotion Controllers

The playback and real-time CPG controllers are compared as follows. Each controller's advantages correspond to the other controller's disadvantages.

The merits of the playback controller are summarized as follows.

1) Easily applied to almost all modular robotic systems because the controller does not care about a specific type of motor.
2) Position control or simple tracking control by a motor is enough.
3) Intermodule communication is unnecessary as long as the clocks are synchronized.

The merits of the real-time CPG controller are as follows.

1) Very suited to a distributed system because CPG controllers work in a distributed manner.
2) No explicit synchronization procedure is needed.
3) Possible to cope with rough terrain or unexpected conditions because phase differences among joints are autonomously created and maintained by real-time CPG interaction.
4) There is no need for storing locomotion sequence data for all joints in a large volume.
5) Applicable to any configurations simply by changing connection weights among CPGs (connection network).

D. Necessity of External Sensor Information and Upper Controller

Locomotion by the current CPG model is considered reflexive motion in creatures. External disturbances or changes in circumstances can be overcome to some extent by using a rhythm and phase difference regulation mechanism realized by local feedback (angle change, CPG interaction) to a CPG.

However, the mechanism does not work properly when rolling on the downslope by using the wheel configuration (Section IV-E). In the experiment, the wheel configuration rolled down with a fixed shape, i.e., angle changes of each joint were very

small and the rhythm regulation mechanism did not work. This shows that the stability range of the controller for this configuration is very narrow. To solve the problem, external sensor information must be introduced to each CPG to reflect the global body rhythm into the local rhythm generated by each CPG.

Furthermore, an upper level controller is necessary to realize high-level action, such as detecting obstacles and stepping over them.

VI. Conclusions and Future Work

We applied a neural oscillator model (CPG) and its network to generate stable locomotive motions for modular robotic systems. We also developed a unified framework by using the CPG model and the genetic algorithm for creating locomotion suitable for any given module structures.

Software simulation by ALPG confirmed that locomotive motions in various configurations can be generated automatically. We performed two types of locomotion experiments using our M-TRAN II modules: locomotion by the playback controller and by the real-time CPG controller. Using the playback controller confirmed that locomotion patterns made by ALPG can be utilized by real hardware under the same conditions as in the simulation. Furthermore, we achieved adaptive locomotion on various terrains using a real-time CPG controller where phase differences among joints were autonomously regulated by real-time CPG interactions.

Future work should focus on the following.

1) Construction of a new CPG model integrating external sensor information.
2) Addition of an upper level controller that enables optional high-level actions such as detecting an obstacle and stepping over it.
3) Development of algorithms for autonomously changing the whole shape as needed.

Acknowledgment

The authors thank Dr. S. Ok at the Communications Research Laboratory, Information and Network Systems Division, Keihanna Human Info-Communications Research Center, Image Group, Japan, for advice regarding the neural oscillator (CPG) and its implementation. We thank N. Fujii at the Department of Knowledge-based Information Engineering in Toyohashi University of Technology for aiding analysis of the neural oscillator network.

References

[1] T. Fukuda and S. Nakagawa, "Dynamically reconfigurable robotic system," in *Proc. IEEE Int. Conf. Robotics Automation (ICRA1998)*, May 1998, pp. 1581–1586.

[2] G. S. Chirikjian, A. Pamecha, and I. Ebert-Uphoff, "Evaluating efficiency of self-reconfiguration in a class of modular robots," *J. Robot. Syst.*, vol. 12–5, pp. 317–338, 1995.

[3] S. Murata, E. Yoshida, H. Kurokawa, K. Tomita, and S. Kokaji, "Self-repairing mechanical systems," *Auton. Robots*, vol. 10, pp. 7–21, 2001.

[4] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, and S. Kokaji, "A 3-D self-reconfigurable structure," in *Proc. IEEE Int. Conf. Robotics Automation (ICRA1998)*, May 1998, pp. 432–439.

[5] E. Yoshida, S. Murata, S. Kokaji, K. Tomita, and H. Kurokawa, "Micro self-reconfigurable robotic system using shape memory alloy," *Distrib. Auton. Robot. Syst.*, vol. 4, pp. 145–154, 2000.

[6] K. Hosokawa, T. Tsujimori, T. Fujii, H. Kaetsu, H. Asama, Y. Koruda, and I. Endo, "Self-organizing collective robots with morphogenesis in a vertical place," in *Proc. IEEE Int. Conf. Robotics Automation (ICRA1998)*, May 1998, pp. 2858–2863.

[7] A. Kamimura, E. Yoshida, S. Murata, H. Kurokawa, K. Tomita, and S. Kokaji, "A self-reconfigurable modular robot (MTRAN): Hardware and motion planning software," *Distrib. Auton. Robot. Syst.*, vol. 5, pp. 17–26, 2002.

[8] D. Rus and M. Vona, "A basis for self-reconfigurable robots using crystal modules," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS2000)*, Oct. 31–Nov. 5 2000, pp. 2194–2202.

[9] K. Kotay and D. Rus, "Motion synthesis for the self-reconfigurable molecule," in *Proc. 1998 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS1998)*, Oct. 1998, pp. 843–851.

[10] C. Ünsal, H. Kiliççöte, and P. K. Khosla, "A modular self-reconfigurable bipartite robotic system: Implementation and motion planning," *Auton. Robots*, vol. 10–1, pp. 23–40, 2001.

[11] W. M. Shen, B. Salemi, and P. Will, "Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots," *IEEE Trans. Robot. Autom.*, vol. 18, pp. 700–712, Oct. 2002.

[12] A. Casal and M. Yim, "Self-reconfigurable planning for a class of modular robot," *Proc. SPIE*, vol. 3839, Sep. 1999, pp. 246–257.

[13] M. Yim, "New locomotion gaits," in *Proc. IEEE Int. Conf. Robotics Autom. (ICRA1994)*, May 1994, pp. 2508–2514.

[14] E. Yoshida, S. Murata, H. Kurokawa, K. Tomita, and S. Kokaji, "A distributed method for reconfiguration of 3-D homogeneous structure," *Adv. Robot.*, vol. 13, no. 4, pp. 363–379, 1999.

[15] K. Tomita, S. Murata, H. Kurokawa, E. Yoshida, and S. Kokaji, "Self-assembly and self-repair method for distributed mechanical system," *IEEE Trans. Robot. Autom.*, vol. 15, no. 6, pp. 1035–1045, Dec. 1999.

[16] Z. Butler, K. Kotay, D. Rus, and K. Tomita, "Generic decentralized control for a class of self-reconfigurable robots," in *Proc. IEEE Int. Conf. Robotics Autom. (ICRA2002)*, May 2002, pp. 809–816.

[17] K. C. Prevas, C. Ünsal, M. Ö. Efe, and P. K. Khosla, "A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system," in *Proc. IEEE Int. Conf. Robotics Autom. (ICRA2002)*, May 2002, pp. 787–792.

[18] H. H. Lund, R. L. Larsen, and E. H. Østergaard, "Distributed control in self-reconfigurable robots," in *Proc. 5th Int. Conf. Evolvable Syst.: From Biology to Hardware*, Mar. 2003.

[19] K. Sims, "Evolving virtual creatures," in *Computer Graphics (Siggraph'94 Proc.)*, July 1994, pp. 15–22.

[20] H. Lipson and J. B. Pollack, "Towards continuously reconfigurable self-designing robotics," in *Proc. IEEE Int. Conf. Robotics Autom. (ICRA2000)*, Apr. 2000, pp. 1761–1766.

[21] K. Matsuoka, "Mechanisms of frequency and pattern control in the neural rhythm generators," *Biol. Cybern.*, vol. 56, pp. 345–353, 1987.

[22] G. Taga, "A model of the neuro-musculo-skeletal system for human locomotion II—Real-time adaptability under various constraints," *Biol. Cybern.*, vol. 73, pp. 113–121, 1995.

[23] H. Kimura, S. Akiyama, and K. Sakurama, "Realization of dynamic walking and running of the quadruped using neural oscillator," *Auton. Robots*, vol. 7–3, pp. 247–258, 1999.

[24] A. J. Ijspeert, "Evolution of neural controllers for salamander-like locomotion," *Proc. SPIE*, vol. 3839, Sep. 1999, pp. 168–179.

[25] R. D. Beer and J. C. Gallagher, "Evolving dynamical neural networks for adaptive behavior," *Adap. Beh.*, vol. 1, no. 1, pp. 91–122, 1992.

[26] M. Williamson, "Neural control of rhythmic arm movements," *Neural Netw.*, vol. 11, pp. 1379–1394, 1998.

[27] I. Ono and S. Kobayashi, "A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover," in *Proc. 7th ICGA*, Jul. 1997, pp. 246–253.

[28] S. Hirose, M. Imazato, Y. Kudo, and Y. Umetani, "Internally-balanced magnet unit," *Adv. Robot.*, vol. 3, no. 1, pp. 225–242, 1986.

[29] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-TRAN: Self-reconfigurable modular robotic system," *IEEE/ASME Trans. Mechatronics*, vol. 7, pp. 431–441, 2002.

[30] H. Kurokawa, A. Kamimura, E. Yoshida, K. Tomita, S. Murata, and S. Kokaji, "Self-reconfigurable modular robot (M-TRAN) and its motion design," in *Proc. ICARCV'02*, Dec. 2002, pp. 51–56.

[31] M-TRAN II Web Site in AIST [Online]. Available: Available: http://unit.aist.go.jp/is/dsysd/mtran/English/index.html.

Akiya Kamimura received the M.E. and Dr. Eng. degrees from the Graduate School of Engineering, University of Tokyo, Tokyo, Japan, in 1997 and 2000, respectively.

He joined the Mechanical Engineering Laboratory, AIST, Massachusetts Institute of Technology, in 2000, and has been conducting research at the National Institute of Advanced Industrial Science and Technology (AIST), Ibasaki, Japan, since 2001. His research interests include modular robotics and rapid prototyping systems.

Dr. Kamimura received the Best Paper Award at the 2002 International Symposium on Distributed Autonomous Robotic Systems (DARS'02).

Haruhisa Kurokawa received the B.E. and M.E. degrees in Precision Machinery Engineering and the Dr. Eng. degree in Aero- and Astronautical Engineering from the University of Tokyo, Tokyo, Japan, in 1978, 1981, and 1997, respectively.

He is currently the Head of the Distributed System Design Research Group, Intelligent Systems Institute, National Institute of Advanced Industrial Science and Technology (AIST), Ibasaki, Japan. His main research subjects are kinematics of mechanisms, distributed autonomous systems, and nonlinear control.

Eiichi Yoshida (S'94–M'96) received the M.E. and Dr. Eng. degrees from the Graduate School of Engineering, University of Tokyo, Tokyo, Japan, in 1993 and 1996, respectively.

From 1990 to 1991, he worked at the Department of Microtechnique at Swiss Federal Institute of Technology, Lausanne (EPFL). He joined the Mechanical Engineering Laboratory, AIST, MITI in 1996, and since 2001, he has been conducting research at the National Institute of Advanced Industrial Science and Technology (AIST), Ibasaki, Japan. His research interests include decentralized autonomous systems and modular robotics.

Dr. Yoshida received the Best Paper Award at the 1998 International Symposium on Distributed Autonomous Robotic Systems (DARS'98).

Satoshi Murata (M'01) received the B.E., M.E., and Dr. Eng. degrees in aeronautical engineering from Nagoya University, Nagoya, Japan, in 1984, 1986, and 1997, respectively.

In 1986, he joined the Mechanical Engineering Laboratory, AIST, MITI. Since 2001, he has been an associate professor in Tokyo Institute of Technology. His current interests include distributed mechanical systems, modular robotics, and molecular computing.

Dr. Murata received the IEEE-IE Outstanding Paper Award and SICE Outstanding Paper Award in 1991 and 1996, respectively. He is a member of SICE, RSJ, and JSME.

Kohji Tomita (M'99–A'01) received the B.E., M.E., and PhD degrees from the University of Tsukuba, Tsukuba, Japan, in 1988, 1990, and 1997, respectively.

He joined the Mechanical Engineering Laboratory, AIST, MITI, in 1990, and has been conducting research at the National Institute of Advanced Industrial Science and Technology (AIST), Ibasaki, Japan, as a Senior Research Scientist since 2001. He was a visiting researcher at Dartmouth College, Hanover, NH, from 2000 to 2001. His research interests include modular robots, distributed software systems, and graph automata.

Shigeru Kokaji (M'00) received the B.E, M.E., and Dr. Eng. degrees in precision machinery engineering from the University of Tokyo, Tokyo, Japan, in 1970, 1972, and 1986, respectively. He is currently the Deputy Director of Intelligent Systems Institute, National Institute of Advanced Industrial Science and Technology (AIST), Ibasaki, Japan. His research interests include distributed control of mechanical and robotic systems.