Proceedings of 2004 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan

Distributed Adaptive Locomotion by a Modular Robotic System, M-TRAN II

From Local Adaptation to Global Coodinated Motion Using CPG Controllers

Akiya Kamimura, Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita and Shigeru Kokaji
National Institute of Advanced Industrial Science and Technology (AIST)
Namiki 1-2-1, Tsukuba, Ibaraki, 305-8654 Japan
{kamimura.a, kurokawa-h, e.yoshida, k.tomita, s.kokaji}@aist.go.jp

Satoshi Murata
Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama, 226-8502 Japan
murata@dis.titech.ac.jp

Abstract—**A modular robot has a distributed mechanical composition which can make various configurations and also make locomotion in a wide variety of configurations. Modular robots are thought to be useful in extreme or unknown environments by adaptively changing their shape and locomotion patterns. As for locomotion, two types can be used; one is whole-body fixed-configuration locomotion and the other is locomotion by self-reconfiguration. In this paper we deal with the former type of locomotion which is realized by coordinated joint actuation. So far, proposed control methods for whole-body locomotion by modular robots have been based on predefined locomotion sequences. However, locomotion based on predefined sequences cannot adapt to changing terrain conditions such as uphill, downhill, slippery and sticky grounds. To solve such problems, we propose a distributed control mechanism using a CPG controller which enables adaptive locomotion by modular robots. Besides the real-time CPG control we introduce a decentralized control mechanism for detecting the situation that the robot is stuck and initiating transformation to another shape for recovering the situation. The results of various hardware experiments by 4-legged structure prove the feasibility of the method for adaptive locomotion and transformation by our M-TRAN II modules.**

Keywords-Self-reconfiguralble modular robotic system; locomotion; Central Pattern Generator (CPG); adaptation;

I. INTRODUCTION

In recent years, many hardware and software investigations on feasibility of self-reconfigurable robotic systems have been carried out [1–13]. Existing self-reconfigurable modular robots comprise homogeneous or heterogeneous robotic modules and can be connected in a variety of configurations according to given tasks. Most modules have an actuator for driving a joint, an automatic inter-module connection mechanism, an inter-module communication system, and a microprocessor as a controller. Modular robots, which we address in this paper, can change their configuration by releasing connections between modules and changing positions of the modules using the actuator. This capability is effective for adaptation to the external environment or self-repair through replacement of disabled parts with spare modules. The module actuator is used not only for self-reconfiguration, but also for producing a whole-body motion such as walking and crawling. Self-reconfigurable modular robots are applicable to extreme or unknown environments such as on distant planets, in deep seas, inside nuclear plants, and in disaster area for exploration or search-and-rescue operations where human access is difficult.

In software research of the self-reconfigurable modular robotic systems, 2-D and 3-D structural formation and locomotion by modules have been the main topics. Most studies have focused on a distributed algorithm or a planning method for structural formation [14–20]. As for locomotion, two types of locomotion have been considered so far. One type is realized by repeating self-reconfiguration, e.g. sending a module from tail of the module structure to head one by one [7][18–20]. The other is whole-body locomotion realized by controlling joint motors coordinately without any configuration change [21-24]. The former locomotion requires much time compared to the latter locomotion, but is useful to surmount high obstacles that are difficult by a specific module structure. The latter type is suitable for faster motion in case of a small scale of configuration such as a 4-legged robot or a snake-like robot.

For practical application of modular robots we consider a scenario in which a robot searches for injured humans in a disaster area. In this case, locomotion by self-reconfiguration might not be desirable, since speed is of vital importance and self-reconfiguration on rough terrains is difficult for current modular robots. Therefore, we investigate locomotion for several smaller, faster moving structures that can self-assemble to a larger structure or disassemble to small pieces when needed.

To realize the above scenario the following aspects should be studied. First is the design of efficient locomotion patterns for a given module configuration. Since there are many possible configurations made by modules, it is difficult to design locomotion patterns

0-7803-8463-6/04/$20.00 ©2004 IEEE

manually one by one. Second is the control method for stable locomotion by many connected modules which has many degrees of freedom. Decentralized control is desirable to reduce calculation cost on each module. Third is the adaptation mechanism such as changing locomotive motions according to the conditions of the terrains or changing to another shape when it is difficult to proceed with the original shape. For the first, in our recent work [25], we proposed an Automatic Locomotion Pattern Generation (ALPG) method for modular robots which produces an efficient locomotion pattern for any given configurations automatically. We demonstrated through hardware experiments that module structures could make locomotion successfully on the flat ground using the results made by the ALPG method, but adaptive locomotion was not realized. As a next step, in this paper we deal with the second and the third topics above and propose a distributed control method using a Central Pattern Generator (CPG) controller to realize stable whole-body locomotion and adaptation to various terrains. A decentralized control method for detecting a situation that the robot is being stuck is also proposed.

In the next section we introduce our latest M-TRAN II module hardware. A CPG model for making globally coordinated motion in a distributed manner is described in section III and the ALPG method for designing locomotion patterns is briefly introduced here. Section IV describes hardware implementation of the CPG model and the drift detection mechanism for adaptive locomotion. Results of hardware experiments are provided in section V.

II. M-TRAN II Module Hardware

A. Overview of the M-TRAN Module

We study whole-body locomotion of modular robots using our M-TRAN (Modular Transformer) module shown in Fig. 1. This module comprises three components: two semi-cylindrical parts and a link part. Each semi-cylindrical part can rotate from –90° to 90° independently by use of a geared motor embedded in the link part. Each semi-cylindrical 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. 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 semi-cylindrical parts, through repetition of simple procedures such as detaching the connection, rotating the semi-cylindrical part, and reconnecting.

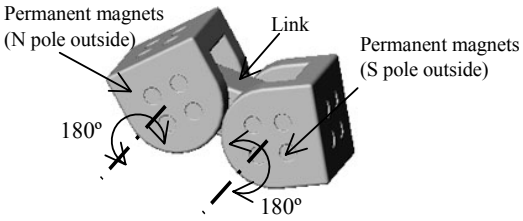

Figure 1. Schematic view of an M-TRAN module.

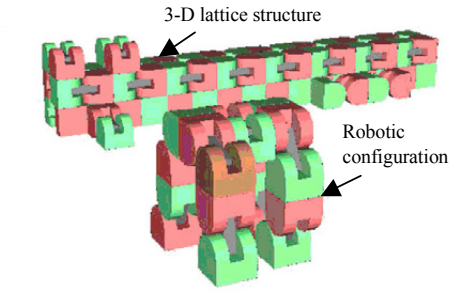

Figure 2. Example of possible configurations, a 3-D lattice structure above and a robotic configuration below. Each semi-cylindrical part of the module is shown in a different color and checkerboard-like structures can be seen.

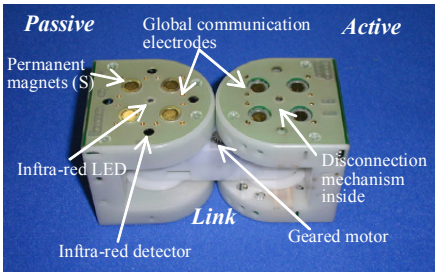

Figure 3. Appearance of the M-TRAN II module.

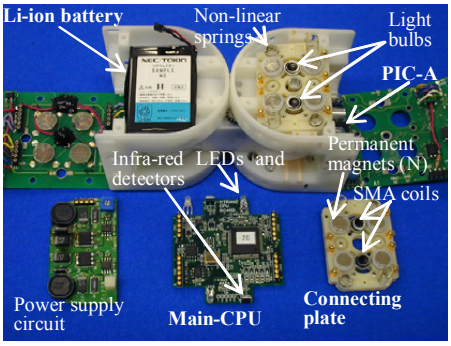

Figure 4. Inner structure of the M-TRAN II module.

In addition to self-reconfiguration, this modular robot system can generate various robotic motions, such as a crawler-like locomotion and quadrupedal walking [7], by utilizing many degrees of freedom. The following method deals with this motion.

B. M-TRAN II Module Hardware

As the detailed explanation on M-TRAN II module are described in our previous paper [25], here we present improved parts compared to the previous version.

We developed twenty M-TRAN II modules shown in Figs. 3 and 4. Figure 3 shows two semi-cylindrical parts: a passive part and an active part. As shown in Fig. 4, a CPU circuit is inside the passive part. We have developed a new CPU circuit board using SH7047 by RENESUS (Main-CPU) to improve calculation and communication capability for implementation of the CPG model. We applied CANBUS system for global inter-module communication. The communication speed is improved up to 1Mbps which is much faster than the previous version, 39kbps. By

applying the new CPU chip, real-time CPG calculation (described later) in each module can be realized. We also added infra-red LEDs and detectors on passive connection surfaces and both sides of the CPU board as a proximity sensor. They are for future application and currently not used. Table I summarizes M-TRAN II module specifications. More details regarding the mechanical and

TABLE I. SPECIFICATIONS OF THE M-TRAN II MODULE

Item	Value
Dimension	60x120x60mm
Weight	0.4kg (including battery)
CPU	SH7407 (Renesus) and two PICs
Global communication	CANBUS, 1Mbps
Power supply (battery)	DC 3.8V
Max. torque of each axis	1.9 Nm (rating)
Max. rotation speed	0.5π rad/s
Connecting force	83 N
Battery	Li-ion (3.8V, 900mAh)
Total power dissipation	0.4W (8V)
Proximity sensor	Infra-red LEDs and detectors
RF module	RF Solutions, Inc. Receiver 315MHz

electrical design of the M-TRAN II module are available in ref. [26, 27].

III. CENTRAL PATTERN GENERATOR (CPG) MODEL AND AUTOMATIC LOCOMOTION PATTERN GENERATION METHOD (ALPG)

We describe details of the CPG model first and then introduce Automatic Locomotion Pattern Generation (ALPG) Method [25] for our M-TRAN II module which is based on the CPG model.

A. CPG Model

We applied the following CPG model as a CPG controller for each joint represented by the 4th order non-linear differential equation. The CPG model is based on the well-known model by Matsuoka [28], which is widely used for making stable locomotion [29-31]. We extended the model to be applicable to a multi-degree of freedom system of any configurations with an arbitrary number of modules:

$$\begin{cases} \tau \dot{u}_{pi} = -u_{pi} - w_0 y_{\bar{p}i} - \beta v_{pi} + u_e + f_{pi} + a \cdot s_{pi} \ , \\ \tau' \dot{v}_{pi} = -v_{pi} + y_{pi} \ , \end{cases} \tag{1}$$

$$i = 0, \cdots, num - 1, \quad (p, \bar{p}) = (1,2), (2,1),$$

$$Output_i = -m_1 y_{1i} + m_2 y_{2i} \ , \tag{2}$$

$$y_{pi} = \max(0, u_{pi}) \ , \tag{3}$$

$$\begin{cases} f_{1i} = k(angle(t)_i - initial\_angle_i) \ , \\ f_{2i} = -f_{1i} \ , \end{cases} \tag{4}$$

$$s_{pi} = 2.0 \cdot \left\{1 + \exp(-feed_{pi} / num)\right\}^{-1} - 1.0 \ , \tag{5}$$

$$feed_{pi} = \Sigma_j \, weight_{ij} u_{pj} \ ,$$

where u_{pi} and v_{pi} are the inner state of the ith neuron. The variable y_{pi} is the output of the ith neuron, u_e is an external

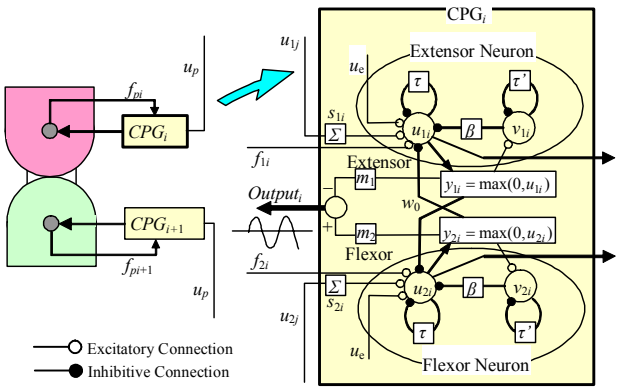

Figure 5. Schematics of the neural oscillator (CPG). Each module has two CPGs; each CPG controls rotation of a joint.

input with a constant value, and τ and τ' are time constants of u_{pi} and v_{pi}, which determine the frequency of the oscillation. In addition, f_{pi} and s_{pi} represent feedback signals from each joint and other neurons respectively. The variable f_{pi} works as a spring tension around the *initial_angle* for making a rhythm. The variable s_{pi} is a normalized value from –1.0 to 1.0 calculated by sigmoid function in which $feed_{pi}$ is divided by the number of modules, *num*, to maintain the balance of the amplitude between feedback signals. The variable $feed_{pi}$ represents an accumulated value of feedback signals from connected neurons, and $weight_{ij}$ is a connecting weight between the ith and jth neurons (each neuron in a CPG is connected to the same type of neuron in other CPGs with the same weight). The implemented GA (described later) in the ALPG software optimizes the initial value of the state variable, (u_{pi}, v_{pi}) and the connection matrix, $weight_{ij}$ for efficient locomotion.

The parameters τ, τ', β, a, m_1, m_2, k, w_0, and u_e were determined by trial and error considering mechanical properties such as maximum motor torque, maximum motor speed and weight of the real hardware. They are summarized in Table II. Each CPG outputs a periodical pattern by itself. It oscillates coordinately with other CPGs when connected. Such behavior is widely known as a locking phenomenon or entrainment among connected non-linear oscillators. We applied the model to realize coordinated motion with several modules. As shown in Fig. 5, each CPG is placed at a joint; then, joint actuation is controlled directly by the CPG output, y, which is a voltage value for a joint motor. In each module, dynamics of two

TABLE II. FIXED PARAMETERS FOR CPG

Parameters	Value
τ	0.05
τ'	0.6
β	1.5
a	4.0
m_1, m_2	0.125
k	8
w_0	2.5
u_e	8.0

CPGs are calculated by eq. (1).

Oscillations of connected CPGs are mutually entrained corresponding to feedback signals f_{pi} and s_{pi}, which are expressed by eq. (4) and eq. (5). The s_{pi} is determined by the CPG network. For stable locomotive motions, not only entrainment between CPGs but also entrainment between dynamics of the whole body and CPGs is important: they are called global entrainment. In other words, when the swing of the mechanical structure as a pendulum does not match the rhythm made by CPGs, locomotive motion becomes neither periodic nor stable. In this model, the rhythm made by the mechanical structure is fed back to each CPG by eq. (4).

B. *CPG Behaviors in Various Connection*

We show basic behaviors of CPGs in Fig. 6 when they are connected by 1: excitatory connection, –1: inhibitive connection and connected in a loop by –1. The fixed parameters of the CPG shown in Table II were used in the simulation.

1) *Case I: Two CPGs are connected by 1*

As shown in the top graph in Fig. 6, two CPGs synchronize together. In this case, the phase difference between CPGs always converges to 0 in any initial states of CPGs.

2) *Case II: Two CPGs are conneted by –1*

In this case shown in the middle graph in Fig. 6, the phase difference between two CPGs always converges to π.

3) *Case III: Three CPGs are connected in a loop by –1 and one of them is connected to other CPG by –1*

In the case that several CPGs are connected in a loop by –1, phase differences between CPGs converge to 2π divided by the number of CPGs in a loop. In the bottom of Fig. 6, phase differences converge to 2π/3 and π respectively and this makes the two module structure move forward or back by a caterpillar-like motion according to the initial states of the CPGs. That is, two attractors exist in this case (shown only one of them). When there are many CPGs and they are connected in various ways, a wide variety of phase differences can be expressed. The GA implemented in ALPG software seeks one of the efficient locomotive motions moving straight by optimizing both initial states of CPGs and connection matrix represented by –1, 1, or 0.

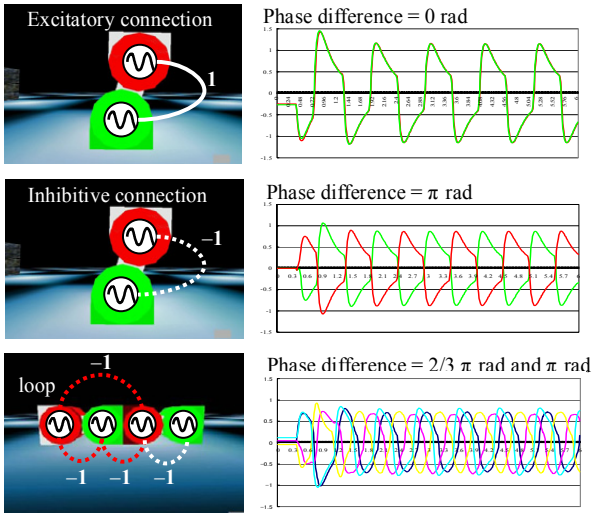

Figure 6. CPG behaviors in various connections. Graphs show CPG output vs. time.

As described above, CPGs can produce various phase differences autonomously in accordance with connection matrix without any explicit synchronization signals. They also can keep phase differences against disturbances to some extent. It makes locomotion by modules robust. It is very suited to implement the model as a joint controller into the module hardware because CPGs are equal with each other and they work in a distributed manner. Even if the configuration has changed, changing locomotion patterns can be easily performed only by replacing the connection matrix.

C. *An Outline of ALPG Software*

We developed ALPG software to seek efficient locomotion patterns automatically for a given module configuration [25]. The ALPG software is realized by combining Vortex (CM Labs Simulations, Inc.) as a three-dimensional dynamic simulation library, a dynamics model of the M-TRAN II module, the CPG model described before, and an optimization method for a CPG network using a genetic algorithm (GA). The ALPG software outputs an efficient locomotion pattern to move for any given robotic configuration.

Figure 7 shows 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 robot with certain configuration and posture is placed on a flat ground shown in Fig. 8. Then each module's joints begin to oscillate periodically with its phase, frequency, and amplitude determined by the CPG network. Performance of the locomotion is evaluated using a fitness function expressed by eq. (6) and the GA

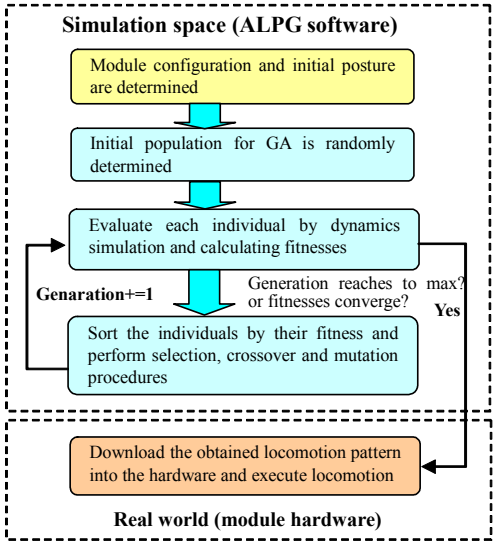

Figure 7. A flow chart for making a locomotion pattern in ALPG software.

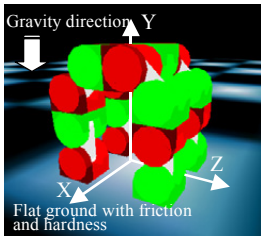

Figure 8. Simulation space on ALPG software. The software implements a dynamic model of the M-TRAN II module with environmental features such as gravity acceleration, friction and hardness of the ground.

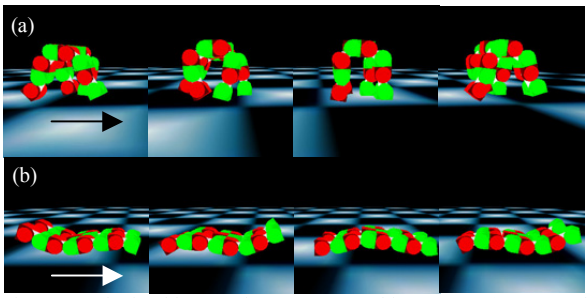

Figure 9. Obtained locomotion patterns, waking pattern (upper) and wave-like pattern (below), for configurations, (a) 4-legged and (b) H-shaped. Arrows in the figure show the moving direction.

optimizes the CPG network.

$$fitness = a \cdot length - b \cdot width - c \cdot loss / num \ , \qquad (6)$$

where *length* is the moving distance, *width* is a drift from the objective path, and *loss* is the energy loss. Figure 9 shows two example results. The ALPG software can automatically produce an efficient locomotion pattern depending on the initial configuration and posture.

Two types of results can be used after simulation. One is time series data for every joint angle and the other is a connection matrix representing the CPG network for an efficient locomotion. In our previous work, as the calculation power of the Main-CPU was limited, we used the former, time series data, as a sequence and confirmed that locomotion is effective on a flat ground [25]. However such a method was not applicable to various terrain conditions because the sequence was optimized for the flat ground with specific friction. To solve the above problem, in the following, we utilize the CPG model also on the real hardware and evaluate performance through hardware experiments.

IV. Hardware Implementation of the CPG Model and the Drift Detection Mechanism

In this section we describe the CPG model implementation into hardware and the drift detection mechanism for adaptive locomotion. The drift detection mechanism is added to the CPG controller shown in Fig. 10 to compensate a drift error of joint rotation and also to detect the situation that the joint is stuck when a heavy load is put on the joint successively.

A. CPG Model Implementation

We implemented the CPG model (III-A) into the microprocessor in the Main-CPU of each module. We call

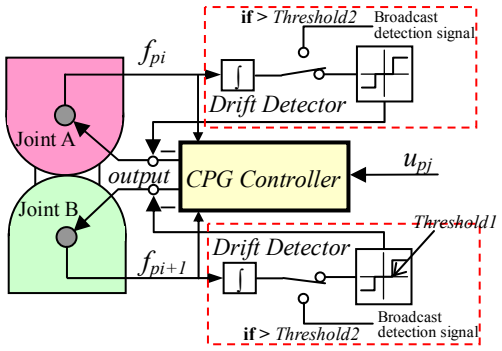

Figure 10. Schematics of the CPG controller and the drift detection mechanism (Drift Detector). The drift detector is surrounded by the dotted line. The CPG controller implemented in each module calculates dyamics of two CPGs in a cycle.

it as a CPG controller (Fig. 10). Since each module has two joints, each CPG controller has only to calculate dynamics of own two CPGs locally.

A calculation cycle for CPG dynamics is 15 msec, which is the same in the simulation. First, a CPG controller takes in feedback signals represented by u_{pj} from connected CPG controllers by using inter-module communication bus and then calculates dynamics of two CPGs by eq. (1) using Euler method. Then, CPG controller outputs the calculation results to the joint controller for driving two joints and gets feedback signals, f_{pi} and f_{pi+1}, as shown in Fig. 10. By repeating these processes locally in each module, phase differences among joints determined by connection matrix are autonomously made and globally coordinated locomotion is realized. There are no needs for synchronization between modules and a master as in conventional systems. The CPG method is considered suitable for such a modular robotic system, a multi-CPU system.

B. Drift Detection Mechanism for Adaptive Locomotion

As already described, to compensate a drift error and detect being stuck of a joint, we added the drift detection mechanism named drift detector to the CPG controller (Fig. 10). In normal situation, each joint oscillates periodically around its initial angle (reference point) shown by the dotted line in Fig. 12 by the spring effect expressed by eq. (4). When a heavy load is put on the joint successively as in Fig. 11, center point of oscillation will drift from the reference point (shown by the continuous line in Fig. 12). The drift leads to unstable locomotion in several cases. In real situation such as going up a slope by 4-legged robot, the drift occurs at hip joints where mostly a heavy load is put and the robot can not go up. To avoid such situation,

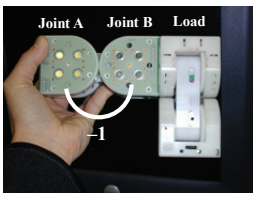

Figure 11. Experimental setup that two modules are connected; the left module is controlled by the CPG and the right is just a load. In the experiment here, two CPGs are connected by –1.

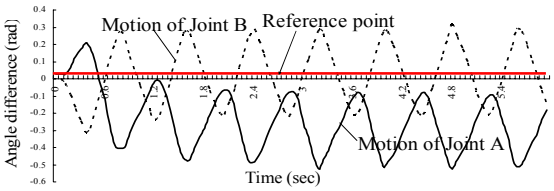

Figure 12. Angle changes of joint A (continuous line) and joint B (dotted line) in the experiment without drift detector.

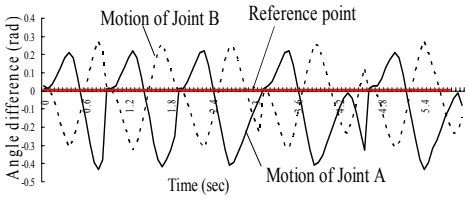

Figure 13. Angle changes of joint A (continuous line) and joint B (dotted line) in the experiment with drift detector

we introduced two thresholds, threshold1 and threshold2, in the mechanism (Fig. 10). One is to suppress the drift and the other is to detect the situation beyond recovery. The latter threshold (threshold2) is larger than the former (threshold1).

The drift detector shown in Fig. 10 is described by the following C-like code and it is carried out in the dynamics calculation loop in each module locally.

$$f = cur\_angle - init\_angle;$$

$$if\ (old\_f \cdot f > 0)\ accum\_val += f;\ else\ \ accum\_val = 0;$$

$$old\_f = f;$$

$$if\ (|accum\_val| > threshold1)\ output -= normalize(accum\_val);$$

$$if\ (|accum\_val| > threshold2)\ broadcast\_message;$$

where an angle difference (f) of a joint calculated by subtracting initial angle (*init_angle*) from current angle (*cur_angle*) is checked in every cycle of the CPG calculation. The value is accumulated while the angle difference is the same sign. The accumulated value (*accum_val*) will be cleared by zero when it crosses the reference point. The absolute value of the *accum_val* is compared with threshold1 and threshold2. If it is over threshold1, output value of the CPG controller expressed by eq. (2) is subtracted by normalized *accum_val* (–1.0 or 1.0) to suppress the drift.

Figure 13 shows the experimental results in the case in Fig. 11 with the additional control. It is confirmed by comparing the graphs in Fig. 12 that center point of oscillation is forced back and the drift is removed. Phase difference between two joints is also kept by CPG interaction. In this case the drift error was compensated successfully but in the case that the situation is not recovered and the *accum_val* is over threshold2, the module detects the situation as being stuck and broadcasts a detection signal to all the connected modules for initiating transformation to another shape. Then all the modules start transformation to a predefined target shape currently.

The actual values of the two thresholds must be determined according to its shape. In the following experiments using a 4-legged robotic configuration, the values are determined empirically and implemented.

V. HARDWARE EXPERIMENTS

We carried out several hardware experiments to show the feasibility of the real-time CPG control and the drift detection mechanism. In the following experiments, we selected the 4-legged configuration as an example to test the method. This is because locomotion by 4-legged configuration is more critical to external disturbance than any other configurations which we have tested in ref. [25].

A. Comparison with Previous Method and Real-Time CPG Control

We examined a locomotion pattern when a heavy load, two modules here, is put only on one side of the 4-legged configuration. Since the 4-legged configuration is composed of nine modules, two ninth of weight of the whole body is put on one side in the experiment. In such case, it is difficult to walk with the same walking pattern in the previous method using a predefined locomotion sequence. Figure 14 shows comparison with our previous method and the real-time CPG control. The former could not walk straight and finally was stuck because phase differences among joints were disturbed by the heavy load and not recovered at all. On the other hand, the latter could walk straight by regulating walking steps cooperatively with each other as shown in the figure. The results prove robustness and adaptability of the CPG control method.

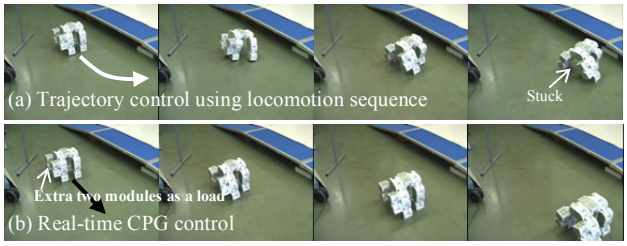

Figure 14. Comparison on locomotion between trajectory control and the real-time CPG control when a heavy load is put on one side of the whole body.

B. Adaptation to Various Ground Conditions

We evaluated locomotion by the 4-legged configuration on normal, sticky and slippery ground. Although the locomotion was optimized for the normal condition of the ground by ALPG software, the 4-legged robot could adaptively walk on them as shown in Fig. 15. Figure 16 shows angle changes of hip joints of the 4-legged robot. It is confirmed that walking steps are automatically regulated according to conditions of the ground and phase difference is always kept by CPGs.

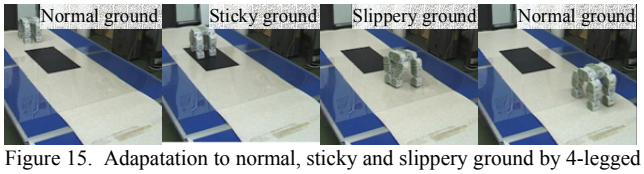

Figure 15. Adapatation to normal, sticky and slippery ground by 4-legged robot.

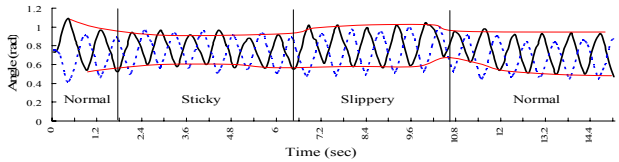

Figure 16. Angle changes of hip joints. The range between red curves shows double of amplitude. It is confirmed that the amplitude while walking on the sticky ground is smaller than others. This means waking steps are automatically changed according to conditions of the ground. It also can be seen that phase difference is always kept.

C. *Comparison between CPG control and CPG control with Drift Detector*

We tested walking along the 10 degrees uphill slope. As shown in Fig. 17 (b), the 4-legged robot using the CPG with the drift detector could go over the slope successfully, but the other was stuck on the way and could not go up as in (a). In case of the 4-legged robot, it can go over the slope up to 10 degrees currently.

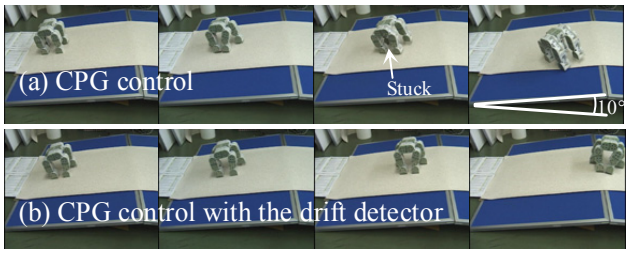

Figure 17. Comparison between (a) CPG control and (b) CPG control with the drift detector.

D. *Adaptation to Uphill and Configuration Change*

We carried out an experiment that the 4-legged robot walked on the 10 degrees uphill, flat ground and the 15 degrees uphill to confirm the feasibility the CPG control and the drift detector. As shown in Fig. 18, the 4-legged robot could go over the first hill and passed the flat ground. Then one of the modules detected the situation of being stuck of the joint as shown in the graph in Fig. 19 at the start point of the 15 degrees uphill. The module broadcasted a detection signal to all the modules and the 4-legged robot transformed its shape into the H-shaped configuration. The H-shaped robot could go up the 15 degrees uphill successfully. The H-shaped robot was also controlled by CPGs. A series of motions were automatically carried out without any human intervention. It was confirmed that the detection mechanism works in such real case.

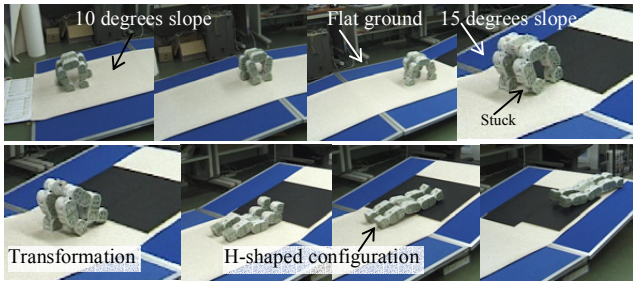

Figure 18. Experiment on adaptation to uphill including transformation when the robot is stuck. In the experiment, 10 degrees slope, flat ground and 15 degrees slope were provided.

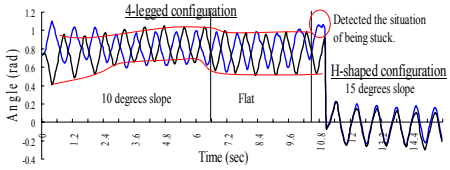

Figure 19. Angle changes of hip joints. It is confirmed that steps of the 4-legged robot becomes small to adapt the slope. At the start point of the 15 degrees slope the situation of being stuck is detected by the drift detector successfully. Angle changes while transformation are not shown in the graph.

VI. Conclusions and Future Works

We introduced a distributed control method for locomotion by modular robots using a CPG controller and a drift detection mechanism for adaptive locomotion. By implementing the method into the module hardware, robustness for locomotion and adaptability to various ground conditions were shown through hardware experiments using 4-legged configuration. We also carried out several experiments on other configurations, e.g. thread type, 6-legged and other 4-legged, by using the same method. It was confirmed that the method was also applicable to others. They are not included in the paper but will soon be uploaded to our M-TRAN II web site [32]. The merits of the proposed CPG control for modular robots are summarized as follows.

1. Very suited for a distributed system. CPGs work in a distributed manner.
2. No explicit synchronization procedure is needed.
3. Phase differences among joints are autonomously created and kept by real-time CPG interaction. It is not needed for every joint's time series angle data for locomotion.
4. Adaptive locomotion on various terrains can be realized.
5. Applicable to any configurations only by changing connecting weights among CPGs (connection matrix).
6. As CPG interaction is represented by three discrete values, –1, 1, or 0, it is easy to implement on any optimization methods such as GA to search for efficient locomotion patterns.

In this paper we demonstrated a simple adaptation to the terrains and transformation when one of the modules was stuck only by using internal information of each module. Through hardware experiments we noticed the followings. Changing configuration on the steep slope or rough terrains is difficult in several cases. Detecting the environment and making a decision in a distributed manner of what configuration is desirable to the situation is still a challenging issue. To solve the problems, any other detection mechanisms and decision making algorithms using external sensors must be necessary. In parallel with the work in this paper, a software research on planning for making a moving path by a whole body using information of distributed external sensors is now in progress.

ACKNOWLEDGMENT

This study was supported by the Science and Technology Research Grant Program for Young Researchers with a Term from Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

REFERENCES

[1] T. Fukuda and S. Nakagawa, "Dynamically Reconfigurable Robotic System," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 1581–1586, 1998.

[2] G. S. Chirikjian, A. Pamecha, and I. Ebert-Uphoff, "Evaluating Efficiency of Self-Reconfiguration in a Class of Modular Robots," *J. Robotic Systems*, 12-5, pp. 317–338, 1995.

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

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

[5] E. Yoshida, S. Murata, S. Kokaji, K. Tomita, and H. Kurokawa, "Micro self-reconfigurable robotic system using shape memory alloy," *Distributed Autonomous Robotic Systems 4*, pp. 145–154, 2000.

[6] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, and S. Kokaji, "Hardware Design of Modular Robotic System," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (IROS2000), pp. 2210–2217, 2000.

[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 –," *Distributed Autonomous Robotic Systems 5*, pp. 17–26, 2002.

[8] 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," *Proc. IEEE Int. Conf. on Robotics and Automation* (ICRA1998), pp. 2858–2863, 1998.

[9] D. Rus and M. Vona, "A basis for self-reconfigurable robots using crystal modules," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (IROS2000), pp. 2194–2202, 2000.

[10] K. Kotay, D. Rus, M. Vona, and C. McGray, "The self-reconfigurable robotic molecule," *Proc. IEEE Int. Conf. on Robotics and Automation* (ICRA1998), pp. 424–431, 1998.

[11] C. Ünsal, H. Kiliççöte, and P. K. Khosla, I(CES)-cubes; a modular self-reconfigurable bipartite robotic system, *Proc. SPIE*, vol. 3839, pp. 258–269, 1999.

[12] A. Castano and P. Will, "Mechanical design of a module for reconfigurable robots," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (IROS2000), pp. 2203–2209, 2000.

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

[14] E. Yoshida, S. Murata, H. Kurokawa, K. Tomita, and S. Kokaji, "A distributed method for reconfiguration of 3-D homogeneous structure," *Advanced Robotics*, 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. on Robotics and Automation*, 15-6, pp. 1035–1045, 1999.

[16] K. Kotay and D. Rus, "Motion synthesis for the self-reconfigurable molecule," *Proc.1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 843–851, 1998.

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

[18] Z. Butler, K. Kotay, D. Rus, and K. Tomita, "Generic Decentralized Control for a Class of Self-Reconfigurable Robots," *Proc. IEEE Int. Conf. on Robotics and Automation* (ICRA2002), pp. 809–816, 2002.

[19] 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," *Proc. IEEE Int. Conf. on Robotics and Automation* (ICRA2002), pp. 787–792, 2002.

[20] H. H. Lund, R. L. Larsen, and E. H. Østergaard, "Distributed Control in Self-Reconfigurable Robots," *Proc. of the 5th International Conference on Evolvable Systems: From Biology to Hardware* (ICES2003), pp. 296-307, 2003.

[21] M. Yim, "New Locomotion Gaits," *Proc. IEEE Int. Conf. on Robotics and Automation* (ICRA1994), pp. 2508–2514, 1994.

[22] M. Yim, Y. Zhang, and D. Duff, "Modular Robots," *Cover Story on February 2002 issue of IEEE Spectrum Magazine*, pp. 30–34, 2002.

[23] M. Yim, D. G. Duff, and K. D. Roufas, "Polybot: A modular reconfigurable robot," *Proc. IEEE Int. Conf. on Robotics and Automation* (ICRA2000), pp. 514–520, 2000.

[24] W. M. Shen, B. Salemi, and P. Will, "Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots," *IEEE Transactions on Robotics and Automation*, vol.18, issue 5, pp. 700–712, October, 2002.

[25] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita and S. Kokaji, "Automatic Locomotion Pattern Generation for Modular Robots," *Proc. IEEE Int. Conf. on Robotics and Automation* (ICRA2003), pp. 714–720, 2003.

[26] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-TRAN: Self-Reconfigurable Modular Robotic System," *IEEE/ASME Transactions on Mechatronics*, Vol.7, No. 4, pp. 431-441, 2002.

[27] H. Kurokawa, A. Kamimura, E. Yoshida, K. Tomita, S. Murata, and S. Kokaji, "M-TRAN II: Metamorphosis from a Four-Legged Walker to a Caterpillar," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (IROS2003), pp.2454-2459, 2003.

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

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

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

[31] K. Hase, et al., "Development of three-dimensional whole-body musculoskeletal model for various motion analyses," *JSME Int. Journal* C 40, pp. 25-32, 1997.

[32] M-TRAN II web site in AIST, http://unit.aist.go.jp/is/dsysd/mtran/English/index.html.