Experiment of Self-repairing Modular Machine

EIICHI YOSHIDA, SATOSHI MURATA, KOHJI TOMITA,
HARUHISA KUROKAWA and SHIGERU KOKAJI

Mechanical Engineering Laboratory, AIST, MITI, 305-8564 Japan

Abstract. A self-assembling and self-repairing mechanical system is experimentally studied to demonstrate its effectiveness. We developed a 2-D model of autonomous mechanical units capable of dynamic reconfiguration and inter-unit communication. Self-assembly and self-repair experiments have been carried out using a distributed algorithm developed under the constraints of the system's homogeneity and locality of information. In experiments, more than ten units successfully configure themselves and recovered from a fault. Besides the research of the 2-D model, a model of 3-D system is also designed.

Key Words. Distributed Autonomous Machine, Homogeneous Mechanical System, Self-assembly, Self-repair

1 Introduction

We have been developing a distributed mechanical system composed of many identical autonomous units. The key issues to build such a hyper-distributed mechanical system are homogeneity of both hardware and software, locality of information and dynamic reconfigurability. Owing to its homogeneous structure, our system can realize self-assembly and self-repair functions.

It can be used as a versatile self-maintainable machine, which can work as a mobile robot, as well as a static structure in hazardous environments.

Many studies have been made on reconfigurable mechanical systems. Polypod (Yim, 1994) and Tetrobot (Hamlin and Sanderson, 1996) are mobile robots with variable structures. CEBOT (Fukuda *et al.*, 1989) and a modular robot (Chirikjian *et al.*, 1996) are capable of reconfiguring dynamically.

In contrast to the above studies, our approach is characterized by its strict homogeneity in both hardware and software, such that the whole system is flexible and robust against faults. Another important point is the simplicity of the hardware and the usage of local communication.

This paper focuses on hardware experiments to confirm self-assembling and self-repairing capability using a hardware system composed of 20 mechanical units called "fracta." Based on a distributed algorithm, self-assembly and self-repair functions are validated by a series of experiments.

The extension of the concept of modular machine to 3-D space is under way. We will give a brief account of its current stage of development.

2 Unit Hardware

We built a system with twenty units whose structure is shown in Figure 1. The unit is mainly composed of actuation part using electro- and permanent magnets and information processing part using a microcomputer.

A unit has three-layered structure, with three pairs of permanent magnets (arranged with 120 degrees) in the top and the bottom layer and three electro-magnets (rotated by 60 degrees from outer layers) in the middle. By changing the polarity of an electro-magnet, it is attracted into, or repulsed from the gap between outer layers of another unit. Two units change their connection by an appropriate sequence of electro-magnet operations. A unit can connect with maximum six units. The unit is also equipped with serial optical channels for local bilateral communication.

3 Self-assembly and Self-repair Algorithm

In self-assembly and self-repair, each unit has the same software and decides its movement only according to local information. Under these constraints, we have been developing a difference-based algorithm (Murata *et al.*, 1994) and a nucleation algorithm (Tomita *et al.*, 1996). The latter, however, needs rather large amount of information processing, and does not suit the current stage of hardware development. Thus we extend the difference-based algorithm to enable self-repair function.

In the algorithm, all the units are assumed to be connected and synchronized in communication. The synchronization is realized by a simple distributed method (Kokaji *et al.*, 1996).

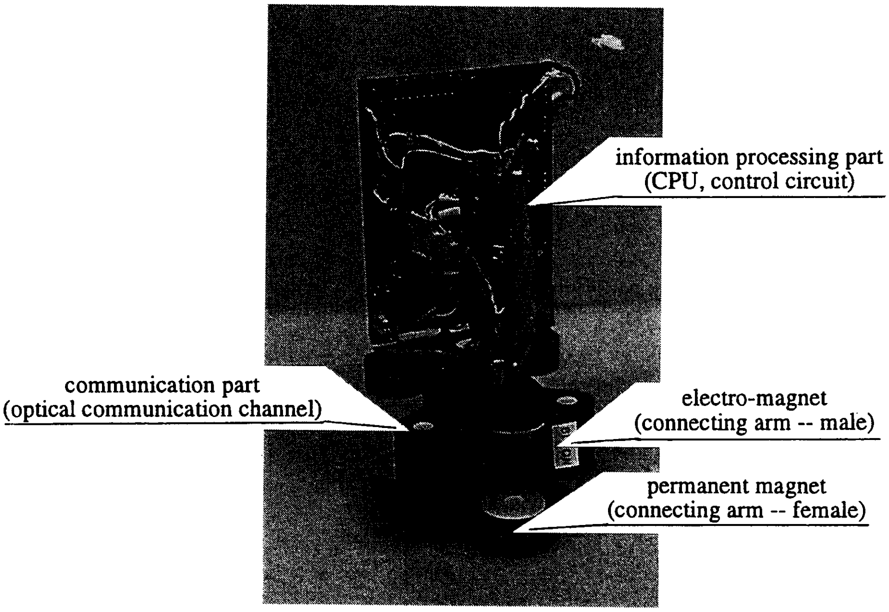


Figure 1 *Autonomous mechanical unit "fractum"*

3.1 Connection Types

The "connection type" of a unit is introduced to describe the global configuration. A unit can be formally described as a hexagon with six possible bonds. The connection of a unit is classified into 12 connection types shown in Figure 2. The link between two types denotes that they are transferable by a single step of unit motion. The notion of "distance" between two types is defined as the number of links between them.

A "movable unit" is a unit which can rotate without carrying other units and keeps the whole system connected after the movement. A unit with connection type "e", "o", or "ε" is movable.

3.2 Self-assembly Algorithm

The algorithm consists of the following three procedures. Each unit

(i) calculates such measures as
  - "difference" between the current state and the goal.
  - "irritation" which increases during deadlock state.

(ii) estimates the average difference around the unit using a diffusion process through the inter-unit communication.

(iii) if the unit has relatively large difference, it moves towards the direction to make it smaller.

A "step" of the algorithm is composed of above (i)–(iii) procedures. In the algorithm, this step is iterated by each unit in parallel.

3.2.1 Goal description We describe the goal formation using the connection types. Let us consider a 10-unit triangle shown in Figure 3. Any unit with the goal type "K" is connected to 4 units whose types are {o, K, K, s}. This situation is expressed in a "statement" K{o, K, K, s}. In this example, the goal shape is written as a collection of all such statements:

$$\begin{array}{l} o\{K, K\} \\ K\{o, K, K, s\} \\ s\{K, K, K, K, K, K\} \end{array} \qquad (1)$$

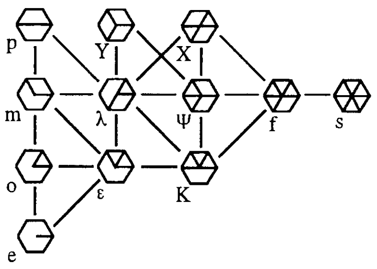


Figure 2 *Connection types and their relationship*

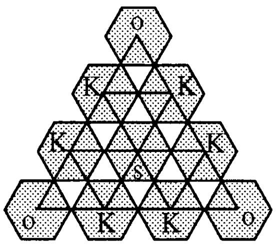


Figure 3 *Description of 10-unit triangle*

Connection types of statements in (1) are sorted in increasing order of the number of connecting units. Although there are cases where this description is not unique, it suffices as long as the goal shape is not too complicated.

3.2.2 Calculation of difference In every iteration step, each unit decides its current connection state in the form of "statement" through local communication. Then each unit evaluates the sum of distances between the types of its current state and each of the goal statements. The "difference" of unit i at t-th step diff(i, t), is given as the minimum of these sums. Namely, each unit evaluates the difference from the closest goal statement.

3.2.3 Diffusion process To make the group of the units converge to the desired goal configuration, it is desirable for a unit which has relatively larger difference to get a larger priority to move than neighbors. We therefore introduce the following diffusion process. A diffusion variable $x(i, t)$ represents the average difference around the unit i. It is calculated as follows[1].

$$x(i, t+1) = x(i, t) + \underbrace{K\bar{x}(i, t) - L}_{\Delta x(i,t)}, \quad x(i, 0) = \text{diff}(i, 0) \quad \text{[initial state]} \tag{2}$$

where $\bar{x}(i, t)$: "flux" of $x(i, t)$ in the unit i $\left(= \sum_{j \in \text{neighbors of } i} \{x(j, t) - x(i, t)\}\right)$

K : diffusion coefficient

L : leak constant (effective for movable units only)

If $x(i, t) < 0$ in (2), it is reset to zero. The diffusion variable K affects the velocity of the diffusion, and L helps to converge to goal by decreasing whole diffusion variables in the system.

3.2.4 Moving strategy A unit is activated to move clockwise or counterclockwise by the following condition that the ratio of difference to diffusion variable (estimated average) is larger than an activation threshold G:

$$Gx(i, t) < \text{diff}(i, t) \tag{3}$$

This makes units having relatively large difference move. The motion is stochastic in such a way that the direction which gives smaller difference after the movement is chosen with larger probability.

If the goal configuration is accomplished, the differences become zero in all of the units and no more motion will be made.

3.2.5 Introduction of irritation Sometimes the self-assembly process is trapped in a deadlock state. The right two configurations in Figure 4 are in deadlock because all the movable type "o" units have the difference 0 with proper neighbors {K, K}.

[1] Suppose that each unit has a water reservoir connected to neighbors with pipes. Then, the water levels will converge to an average level as water flux passes through the pipes. This water level analogically corresponds to the local average difference.

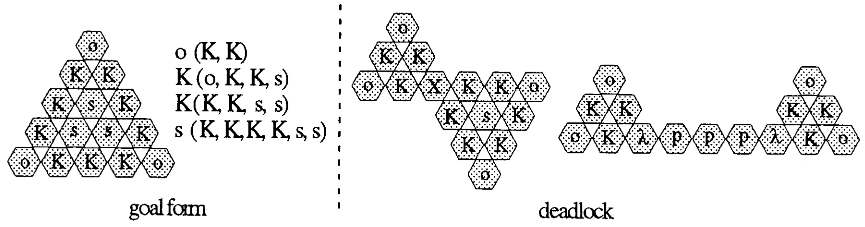


Figure 4 *Examples of deadlock*

To dissolve deadlock, we add another variable called "irritation" irr(i, t). If $\Delta x(i, t)$ in (2) remains smaller than a certain value, namely $x(i, t)$ becomes stable, irr(i, t) is augmented according to:

$$\mathrm{irr}(i, t+1) = \mathrm{irr}(i, t) + \max(|\bar{x}(i, t)|, |\mathrm{diff}(i, t) - x(i, t)|) \tag{4}$$

In particular, when non-movable units are in deadlock, $x(i, t)$ approaches zero with non-zero diff(i, t) or $\bar{x}(i, t)$. Consequently, irr$(i, t + 1)$ increases continuously (Yoshida *et al.*, 1997).

If the inequality in (5) is satisfied, then $x(i, t)$ is increased largely in proportional to $|\mathrm{diff}(i, t) - x(i, t)|$ and irr(i, t) is reset to zero in those units.

$$\mathrm{irr}(i, t) > I_{th} \tag{5}$$

Since non-movable units in deadlock supply a flux continuously, $\bar{x}(i, t)$ increases in also movable units. Then the irritation irr(i, t) in such movable units increases by evaluating (4), while $x(i, t)$ remains stable, namely, $\Delta x \simeq 0$ in (2) due to the leak constant L.

By adding, therefore, a new activation condition (5) for movable units, deadlock states can be dissolved in a local way.

3.3 Self-repair Algorithm

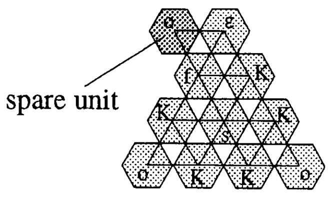
The above algorithm can be accommodated to "self-repair" function by introducing an extended goal description. This extension makes it possible to express goal shapes including spare units (Yoshida *et al.*, 1997).

In adding a spare unit to the goal of 10-unit triangle, we introduce descriptions with priorities shown in Figure 5. We give the first priority to the original triangle shape and the second to the shape with a spare unit. In this example, up to three spares can be included by slightly modifying the description.

This simple method allows units first to build the structure with spare units, and to repair themselves if some of units are removed. Computer simulations showed the effectiveness of the introduced algorithm.

4 Self-assembly and Self-repair Experiments

Using the developed hardware setup, self-assembly and self-repair experiments have been conducted. The algorithm presented so far is a high-level control which de-



o {K, K}
K{o, K, K, s}
s {K, K, K, K, K, K}
	Prority 1
	[original shape]

o {ε, f}
ε {o, K, f}
K{o, K, f, s}
K{ε, K, f, s}
f {o, ε, K, K, s}
s {K, K, K, K, K, f}
	Prority 2
	[with a spare unit]

Figure 5 *Description with priorities for self-repair*

cides whether a unit moves or not and the direction of the motion. In the real mechanism, a unit's motion requires other units' local cooperation. We have developed a low-level control which consists of map generation, collision avoidance and coil drive (Tomita *et al.*, 1996) and integrated it with the upper-level algorithm.

In the following experiments, a step time including these two control levels is 23.25[sec]. The most time-consuming part in the step time is the control of electro-magnets. They should be controlled by an appropriate sequence with sufficient interval, so that undesired motion may not occur. It is also necessary that resultant connection type after the motion should not be unstable "e" type. The control sequence consists of 18 steps, each of which takes 1.0[sec].

The pictures in Figure 6 are taken from one of the experiments. Starting from a two-line configuration, 11 units form a 10-unit triangle with one spare unit based on the proposed self-assembly algorithm. This goal shape remains stable as long as all the units are working without faults.

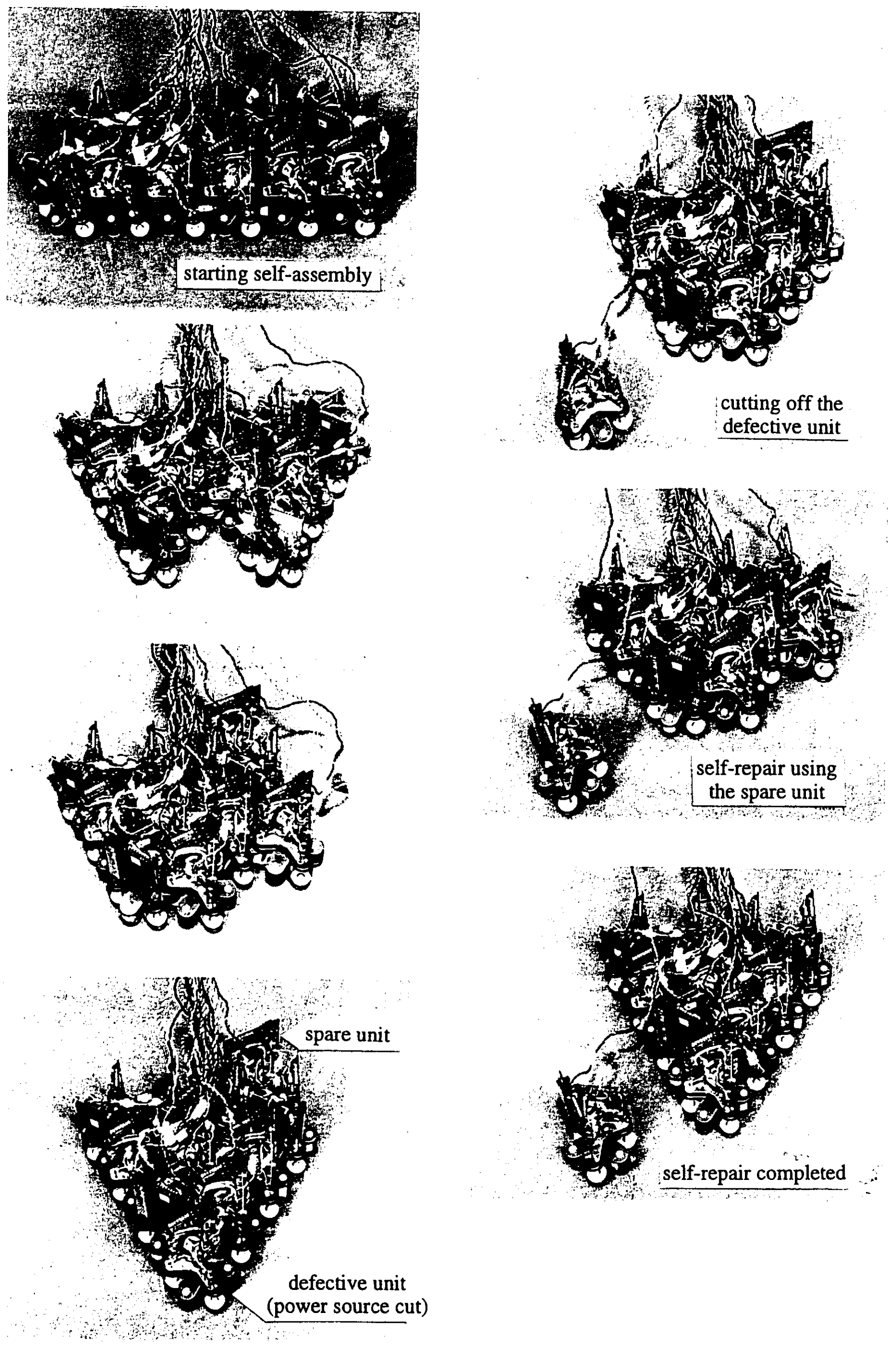
The 1000 simulations of this self-assembly resulted in the success ratio 98.3% with the average convergence time 57.7 steps. Figure 6 (left column) shows snapshots taken from the shortest case of 6 steps.

To verify the self-repair capacity, a simulated fault is given to one of the 11 units by cutting its electric power source (except for "s" type unit). Each unit sends out a "OK" signal to its neighbors at the beginning of each step while working properly. The fault can be detected if a unit does not receive the expected "OK" signal. In this case, the unit reverses the polarity of the corresponding electro-magnet to cut off the faulty unit.

Then self-repair process proceeds automatically. Even though total units in the system has changed, the global self-repair process is launched thanks to its distributed characteristics of the algorithm. We can see the original 10-unit triangle is constructed in the right column of Figure 6.

We examined the self-repair capacity by giving the simulated fault to each of 10 units except the center "s" type. As a result, self-repair completed successfully in all of these 10 experiments and the average self-repair time was 2.2 steps.

The experiments demonstrate that a mechanical system consisting of many identical hardware and software can maintain themselves using self-assembly and self-repair functions in a distributed manner.



Self-assembly experiment Self-repair experiment

Figure 6 *Experiment of self-assembly and self-repair*

5 Development of 3-D System

We are currently attempting to extend the concept of our self-assembling modular machine to 3-D space. This section outlines its present developments stage.

5.1 Design of 3-D Unit

The difficulty of design lies in how both geometric complementarity and system homogeneity are satisfied in 3-D space. A solution we reached is illustrated in Figure 7. The unit has six connecting arms in three orthogonal axes which can rotate independently. Only one DC motor of 7[W] is used as power source in a unit to realize compact implementation. Its torque is delivered by controlling solenoids and electro-magnetic clutches. We confirmed that a unit generates enough power to lift another unit.

The motion of units is basically made by a pair of them. Consider two connected units (shaded) on a plane **A** made by 4 units in Figure 8 (a). Unit **Y** moves to the position in Figure 8 (b) when unit **X** rotates about axis **b–b** after releasing the connection to **Z**. By repeating this elementary step motion, various 3-D structures can be configured without external help.

5.2 Self-assembly of 3-D System

A self-assembly algorithm for 3-D units is being developed in parallel with the hardware development. We present here the current status of development.

We classified the connection types of 3-D unit in 1-neighborhood system in cubic lattice as shown in Figure 9. We neglect rotational transformations in this classification.

Assuming that a unit rotates by 90 [deg] at one step, a unit with type C1 has maximum two reachable positions. Other than C1, types C21 and C31 are also regarded as movable here.

Here, a 3-D self-assembly algorithm is used which is derived from the difference-based 2-D algorithm.

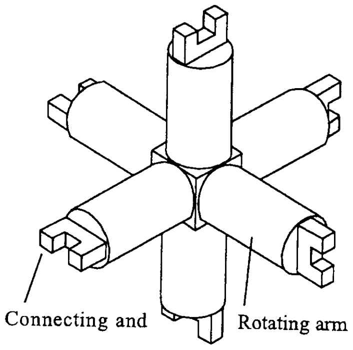


Figure 7 *Schematic view of 3-D unit*

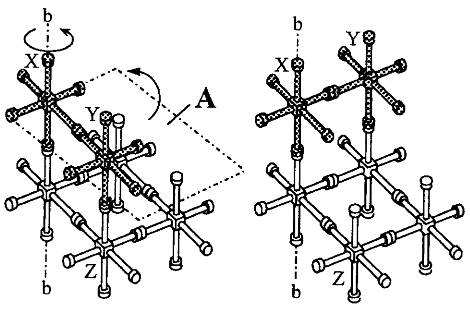


Figure 8 *Typical motion of 3-D units*

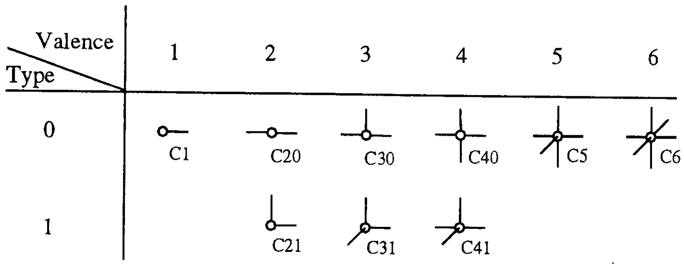


Figure 9 *Connection types for 3-D units*

We simulate a simple self-assembly to construct a box with twelve units from the initial ladder configuration. Similarly as the 2-D algorithm, the goal description is simply written as:

$$\begin{array}{l} C31\{C31, C31, C41\} \\ C41\{C31, C31, C41, C41\} \end{array} \qquad (6)$$

Units calculate the probability of moving to each of reachable positions based on Markov Random Field (MRF) according to the calculated difference (Geman and Geman, 1984). Using a kind of simulated annealing method, units move uniformly to every reachable positions in earlier steps, and less frequently as time elapses.

Figure 10 shows a graphical view of a simulation sequence. In this example, the system succeeded in self-assembly in 70 steps. Construction of more complex structure will be tackled in our future work.

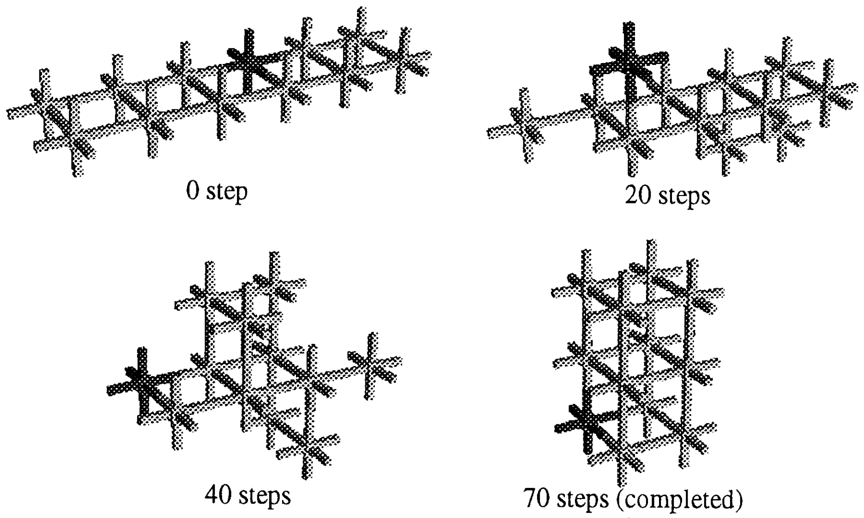


Figure 10 *Self-assembly simulation of 12 units*

6 Conclusions

We presented experimental research on self-assembly and self-repair of a distributed mechanical system. A 2-D model of identical units called "fracta" is used as hardware which realizes dynamic reconfiguration and inter-unit communication. Based on a distributed algorithm that allows a group of units to transform themselves into a desired shape, we demonstrated the self-assembly and self-repair functioned successfully. This experimental results confirmed the hardware feasibility of the self-repairing distributed mechanical system and opened its way to applications such as long-running explorer or surveillant in hazardous environments.

As a further challenge, we are now on the way to 3-D system. We designed a prototype unit and confirmed its basic function of reconfiguration capacity. A self-assembly algorithm is being implemented as well, whose potential effectiveness was shown by computer simulations. The self-assembly algorithm is being refined for robustness and adaptability.

7 REFERENCES

Chirikjian, G., A. Pamecha and I. Ebert-Uphoff (1996). Evaluating efficiency of self-reconfiguration in a class of modular robots. *J. Robotic Systems* **12**(5), 317–338.

Fukuda, T., S. Nakagawa, Y. Kawauchi and M. Buss (1989). Structure decision method for self organizing robots based on cell structure – CEBOT. In: *Proc. IEEE Int. Conf. Robotics and Automation.* pp. 695–700.

Geman, S. and D. Geman (1984). Stochastic relaxation, gibbs distributions, and the baysian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-6**(6), 721–741.

Hamlin, G. J. and A. C. Sanderson (1996). Tetrobot modular robotics: Prototype and experiments. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS'96).* pp. 390–395.

Kokaji, S., S. Murata, H. Kurokawa and K. Tomita (1996). Clock synchronization algorithm for a distributed autonomous system. *J. Robotics and Mechatronics* **8**(5), 317–338.

Murata, S., H. Kurokawa and S. Kokaji (1994). Self-assembling machine. In: *Proc. IEEE Int. Conf. Robotics and Automation.* pp. 441–448.

Tomita, K., S. Murata, E. Yoshida, H. Kurokawa and S. Kokaji (1996). Reconfiguration method for a distributed mechanical system. In: *Distributed Autonomous Robotic System 2* (H .Asama *et.al*, Ed.). Springer.

Yim, M. (1994). New locomotion gates. In: *Proc. IEEE Int. Conf. Robotics and Automation.* pp. 2508–1524.

Yoshida, E., S. Murata, K. Tomita, H. Kurokawa and S. Kokaji (1997). Distributed formation control for a modular mechanical system. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robot and Systems (IROS'97).*