An Experimental Study on a Self-repairing Modular Machine

Eiichi Yoshida [a] Satoshi Murata [a] Kohji Tomita [a]
Haruhisa Kurokawa [a] Shigeru Kokaji [a]

[a] *Mechanical Engineering Laboratory, 1-2 Namiki, Tsukuba-shi, 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 unit 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 exchange. In experiments, more than ten units successfully configured themselves and recovered from a fault. Besides the research of the 2-D model, a model of 3-D system and its self-assembly algorithm are also developed.

Key words: Distributed Autonomous System, Homogeneous Modular Machine, 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 exchange 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 [1] and Tetrobot [2] are mobile robots with variable structures. CEBOT [3] and modular robots [4] [5] are dynamically reconfigurable.

In contrast to the above studies, our approach is characterized by its strict homogeneity in both hardware and software, which makes the whole system 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. A hardware system composed of 20 mechanical units called "fracta" was used for the validation.

We are extending the concept of modular machine to 3-D space. The last part of the paper addresses its hardware realization and self-assembly software.

2 Unit Hardware

We built a system with twenty units whose structure is shown in Fig. 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 concentrically 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 connecting position 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 bilateral local communication.

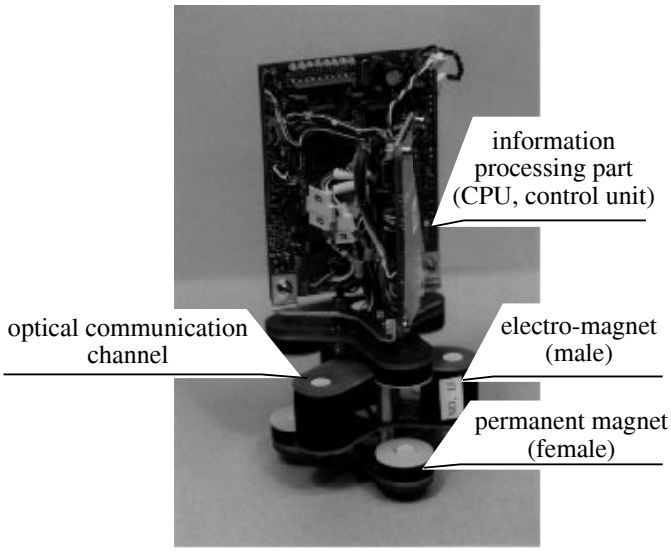


Fig. 1. Autonomous mechanical unit "fractum"

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 [6] and a nucleation algorithm [7]. 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 [8].

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

3.2.1 Target description

We describe the target formation using the connection types. Let us consider a 10-unit triangle shown in Fig. 3. Any unit with the target 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 target shape is written as a collection of the following statements:

$$\begin{aligned} &\mathrm{o}\,\{\mathrm{K},\ \mathrm{K}\},\\ &\mathrm{K}\{\mathrm{o},\ \mathrm{K},\ \mathrm{K},\ \mathrm{s}\},\\ &\mathrm{s}\,\{\mathrm{K},\ \mathrm{K},\ \mathrm{K},\ \mathrm{K},\ \mathrm{K},\ \mathrm{K}\}. \end{aligned} \tag{1}$$

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

3.2.2 Calculation of difference

In every iteration of control step, each unit decides its current connection state through local communication. Then each unit evaluates the sum of distances between the types of its current state and each of the target statements. The "difference" of unit i at t-th control step $\mathrm{diff}(i, t)$, is given as the minimum of these sums. Namely, each unit evaluates the difference from the closest target statement.

3.2.3 Diffusion process

To make the group of the units converge to the desired target configuration, it is desirable for a unit which has relatively larger difference to get a larger priority to move. We therefore introduce the following diffusion process. A

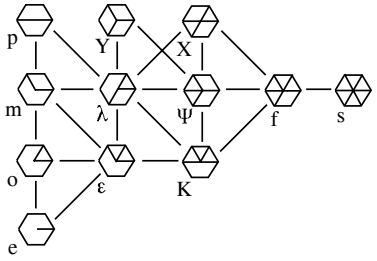


Fig. 2. Connection types and their relationship

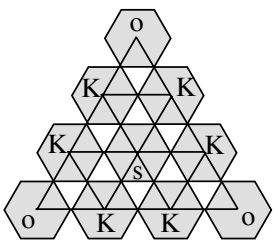


Fig. 3. Description of 10-unit triangle

diffusion variable $x(i,t)$ represents the average difference around the unit i. It is calculated as follows [1] .

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

where $\bar{x}(i,t)$: "flux" of $x(i,t)$ in the unit i

$$(= \sum_{j \in \text{ neighbors of } i} \{x(j,t) - x(i,t)\}),$$

K : diffusion coefficient,

L : leak constant (effective for movable units only).

If $x(i,t) < 0$ in eq. (2), it is reset to zero. The diffusion variable K affects the velocity of the diffusion, and L helps to converge to target 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:

$$Gx(i,t) < \mathrm{diff}(i,t). \quad (3)$$

This makes units of relatively large difference move earlier. 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 target configuration is accomplished, the differences become zero in all of the units and no more motion is made.

3.2.5 Introduction of irritation

Sometimes self-assembly process is trapped in a deadlock state. The right two configurations in Fig. 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.

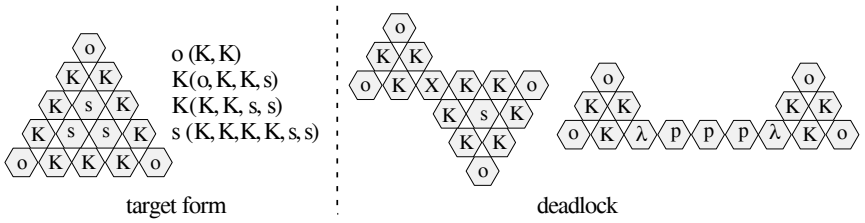


Fig. 4. Examples of deadlock

To dissolve deadlock, we add another variable called "irritation" which is augmented if the diffusion variable $x(i,t)$ becomes stable in undesired configuration [9]. When non-movable units are in deadlock, their irritation values continue to increase and are transmitted also to movable units through local communication.

By adding a new activation condition that "the irritation exceeds than certain threshold," deadlock states can be dissolved in a local way. Computer simulations revealed that the performance of self-assembly was greatly improved by the algorithm using irritation [9].

3.3 Self-repair Algorithm

The above algorithm can be used for "self-repair" operation by introducing an extended target description. This extension makes it possible to express target shapes including spare units [9].

In adding a spare unit to the target of 10-unit triangle, we introduce descriptions with priorities shown in Fig. 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.

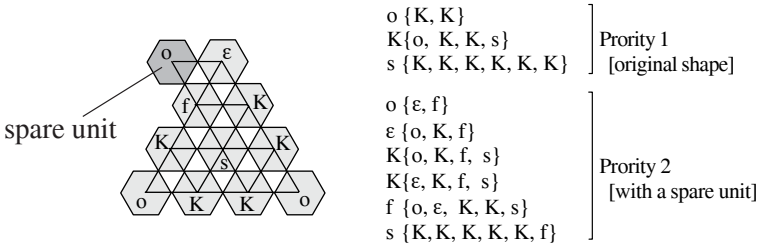


Fig. 5. Description with priorities for self-repair

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 decides 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 [7] and integrated it with the upper-level algorithm.

In the following experiments, a control step including these two control levels takes 23.25[sec]. The most time-consuming part is the control of electro-magnets which should be operated appropriately 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 step includes 18 sequential operations of electro-magnets, each of which takes 1.0[sec].

The pictures in Fig. 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 target 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 control steps. Figure 6 (left column) shows snapshots taken from the shortest case of 6 control 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 an "OK" signal to its neighbors at the beginning of each control 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 Fig. 6.

We examined the self-repair capacity by giving the simulated fault to each unit except the center "s" type. As a result, self-repair completed successfully in all of these 9 experiments and the average self-repair time was 2.2 control 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.

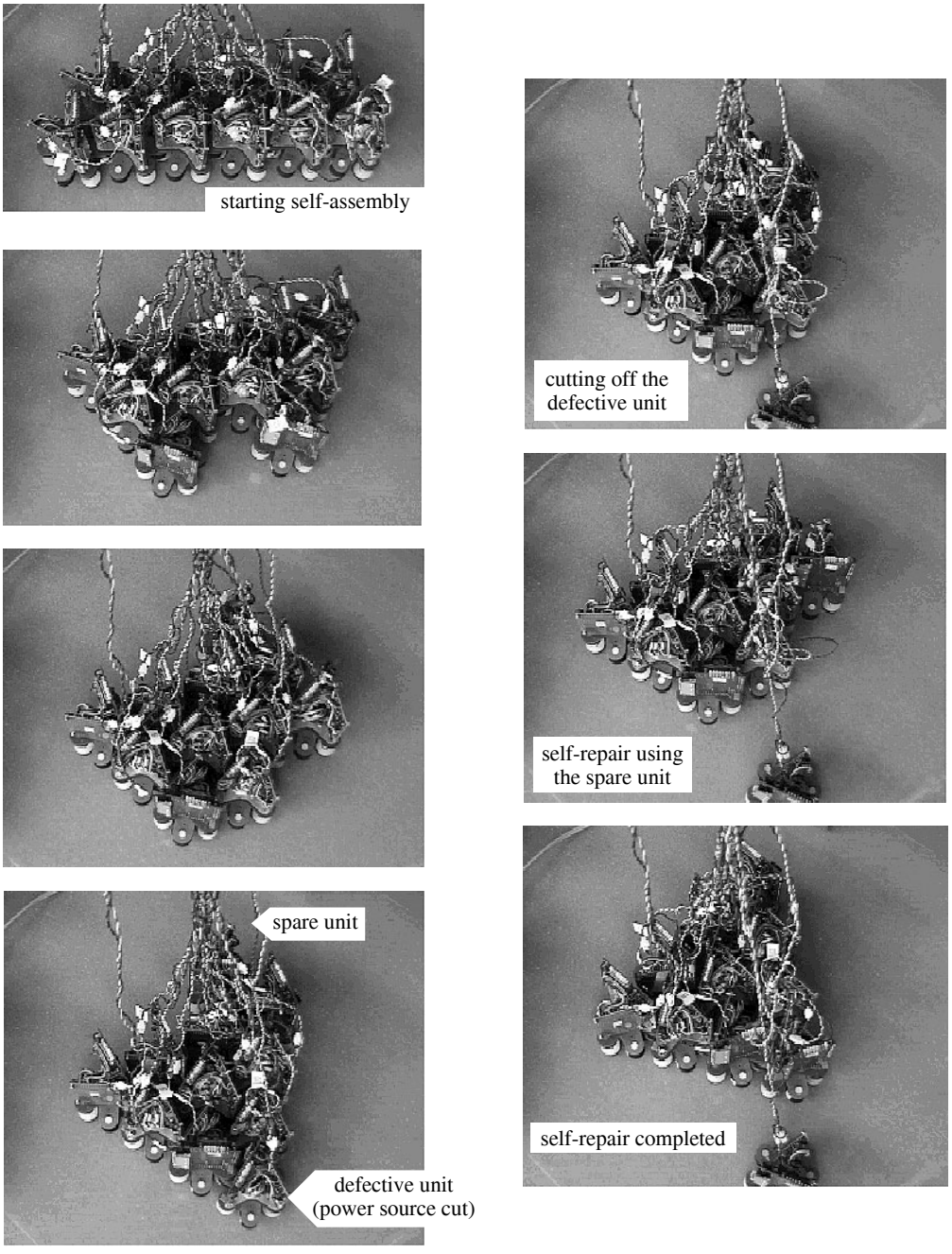


Fig. 6. Experiment of self-assembly and self-repair

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 development of hardware realization and software for distributed self-assembly.

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 Fig. 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 pairwise operation. Consider two connected units (shaded) on a plane **A** made by 4 units in Fig. 8 (a). Unit **Y** moves to the position in Fig. 8 (b) when unit **X** rotates about the axis **b**–**b** after releasing the connection of **Y** to **Z**. By repeating this elementary step motion, various 3-D structures can be configured without external help. We have constructed four 3-D hardware units as shown in Fig. 9 and verified their reconfiguration capacity by experiments [10].

5.2 Self-assembly of 3-D System

A distributed self-assembly algorithm for 3-D units has been developed in parallel with the hardware development.

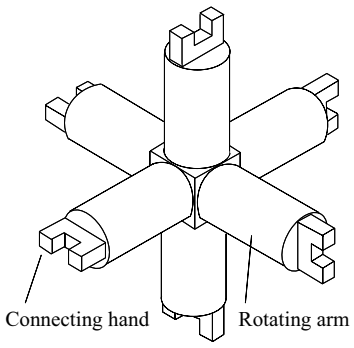


Fig. 7. Schematic view of 3-D unit

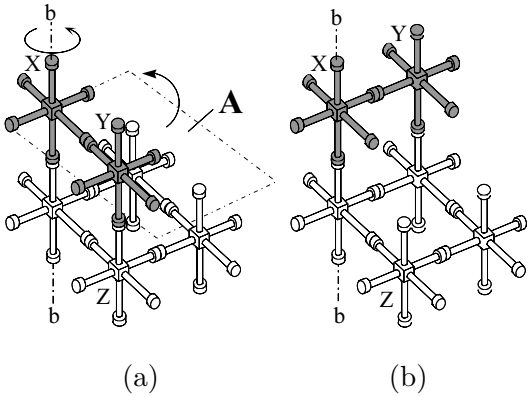


Fig. 8. Typical motion of 3-D units

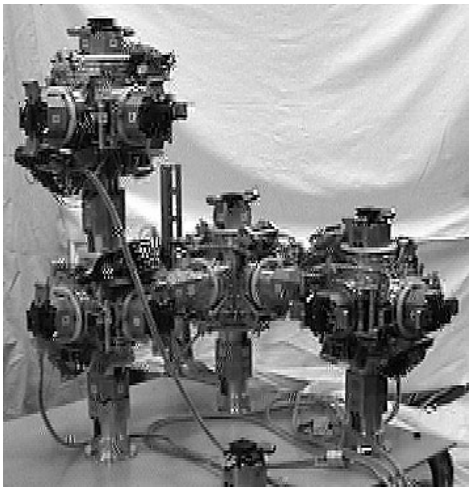


Fig. 9. Hardware setup of the four 3-D mechanical units

One of major difficulties of developing 3-D self-assembly algorithm lies in the multiplicity of degrees of freedom, whereas 2-D system has only to choose one of two directions, clockwise or counterclockwise. The algorithm should be implemented in a distributed manner in order to avoid premature convergence to undesired shape. We introduce a stochastic relaxation process based on simulated annealing to cope with these problems.

We classified the connection types of 3-D unit in 1-neighborhood system in cubic lattice as shown in Fig. 10. 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.

The target shape is described in a similar manner as the 2-D algorithm. For instance, the target shape of a box with twelve units is simply written as:

$$\begin{aligned} &\text{C31}\{\text{C31, C31, C41}\}, \\ &\text{C41}\{\text{C31, C31, C41, C41}\}. \end{aligned} \tag{4}$$

The 2-D self-assembly algorithm is applied to the 3-D system with many

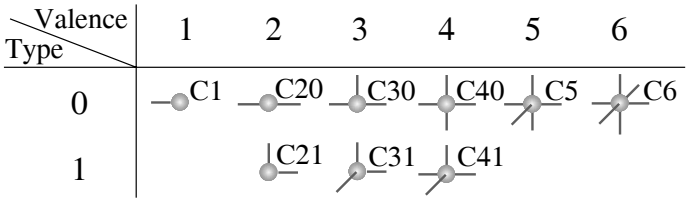


Fig. 10. Connection types for 3-D units

degrees of freedom by introducing stochastic relaxation method. When a unit has multiple reachable positions, it decides where to move according to the probability P_i of moving to the alternative i based on Markov Random Field (MRF) [11] as follows:

$$P_i = \frac{1}{Z} \exp\left(-\frac{D_i}{T}\right) \qquad Z = \sum_{i \in \text{ reachable positions}} \exp\left(-\frac{D_i}{T}\right) \quad , \tag{5}$$

where D_i is the difference between the current and the target connection state (4) and T corresponds to the "artificial temperature" in the simulated annealing method.

The probability P_i is uniform for each reachable position i with large T value, while P_i for the lowest potential approaches 1 if T is small. T is decreased according to the following formula:

$$T = \frac{C}{\log\left(1+t\right)} \quad , \tag{6}$$

where t and C are the number of control steps and constant parameter concerning system size, respectively.

By decreasing T gradually in this way, units search uniformly in "searching space" in earlier steps. As time elapses, the units come to make more concentrated motion to the closest position to complete target structure. The premature convergence to undesired shapes with local minima can be avoided in this way.

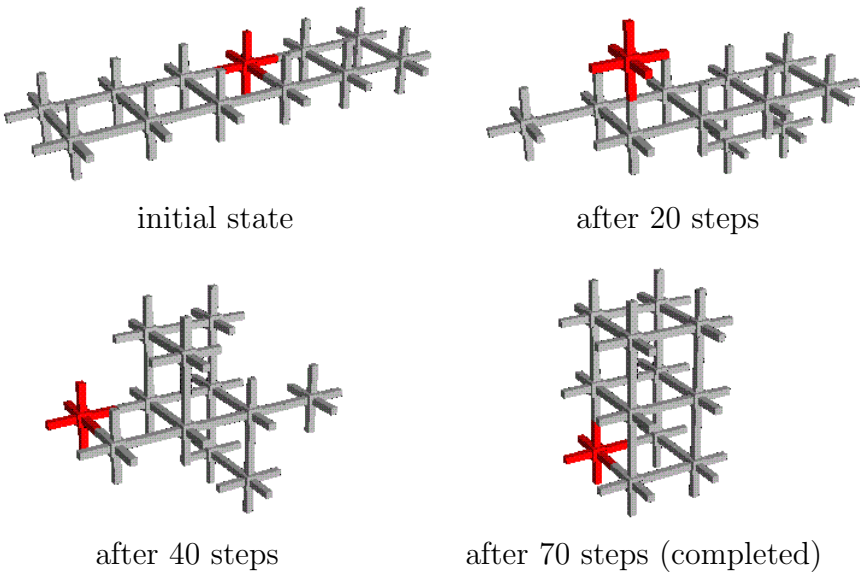


Fig. 11. Self-assembly simulation of 12 units

Figure 11 shows a graphical view of a simulation of 12-unit box described in (4) from initial ladder shape. The system succeeded in self-assembly in 70 steps. The effectiveness of the algorithm has been verified through a quantitative evaluation [12].

5.3 Towards Self-assembly of Large-Scale System

The self-assembly algorithm based on stochastic process has such advantages as flexibility and simple implementation, and works well for small number of units, say, up to twenty units. However, it becomes less effective for larger system in which ambiguity in description and importance of assembly "sequence" are significant. Since few studies has been made on distributed self-assembly of large-scale 3-D structure, we are currently working on this issue based on a developmental scheme like nucleation method in 2-D system [7].

We introduce primitive structures which determines the geometrical relationship between "nodes", which denotes a group of units here, as shown in Fig. 12. By assigning another sub-structure to each nodes of a primitive structure, various complex shapes as shown in Fig. 13 can be described. Figure 14 illustrates the recursive representation of the shape in Fig. 13(a).

Self-assembly proceeds by constructing first top-level structure, then down to sub-structure, and so forth. Using this algorithm, rough shaping is performed in the early stages of self-assembly, and then detailed shape is assembled in a

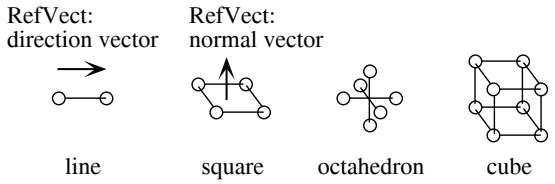
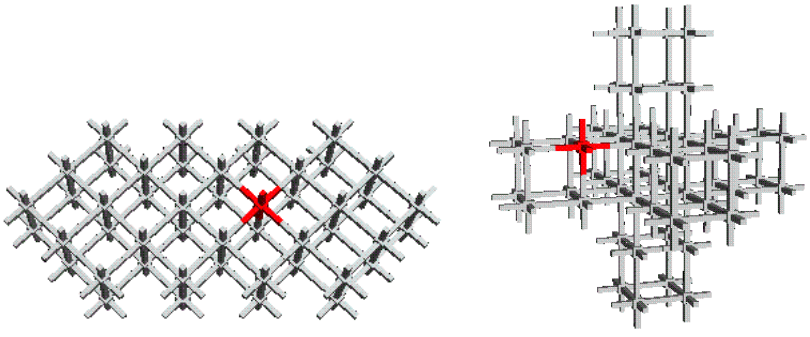


Fig. 12. Primitive types of description



(a) 48 units (b) 56 units

Fig. 13. Examples of large-scale structure

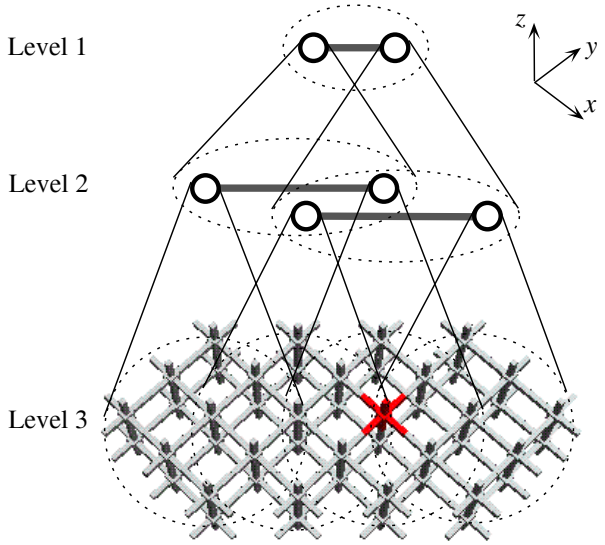


Fig. 14. Recursive description of structure in Fig. 13(a)

concurrent and distributed manner later as shown in Fig. 15.

Figure 16 shows a self-assembly process of 56-unit structure in Fig. 13(b), where dark colored units have reached a position in the target shape. As can be seen, a large structure composed of more than 50 units is successfully constructed in a distributed way.

Possible applications of such a system include machines used in environments inaccessible to humans, for instance, planetary exploring vehicles or satellite antennae. Transported in a compact folded form, they can expand to the original structure when working, and repair themselves if some part becomes damaged.

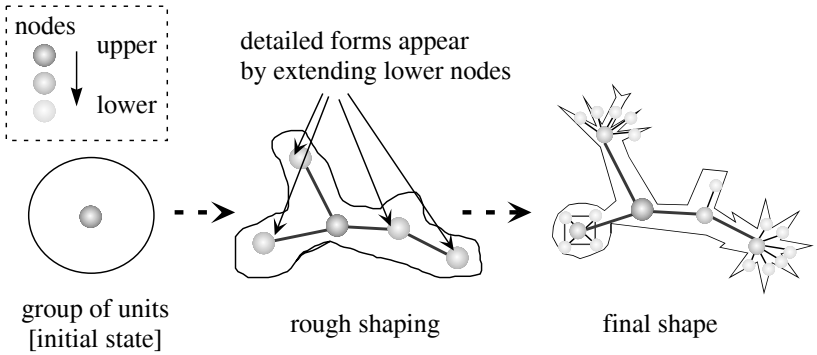


Fig. 15. Large-scale self-assembly process

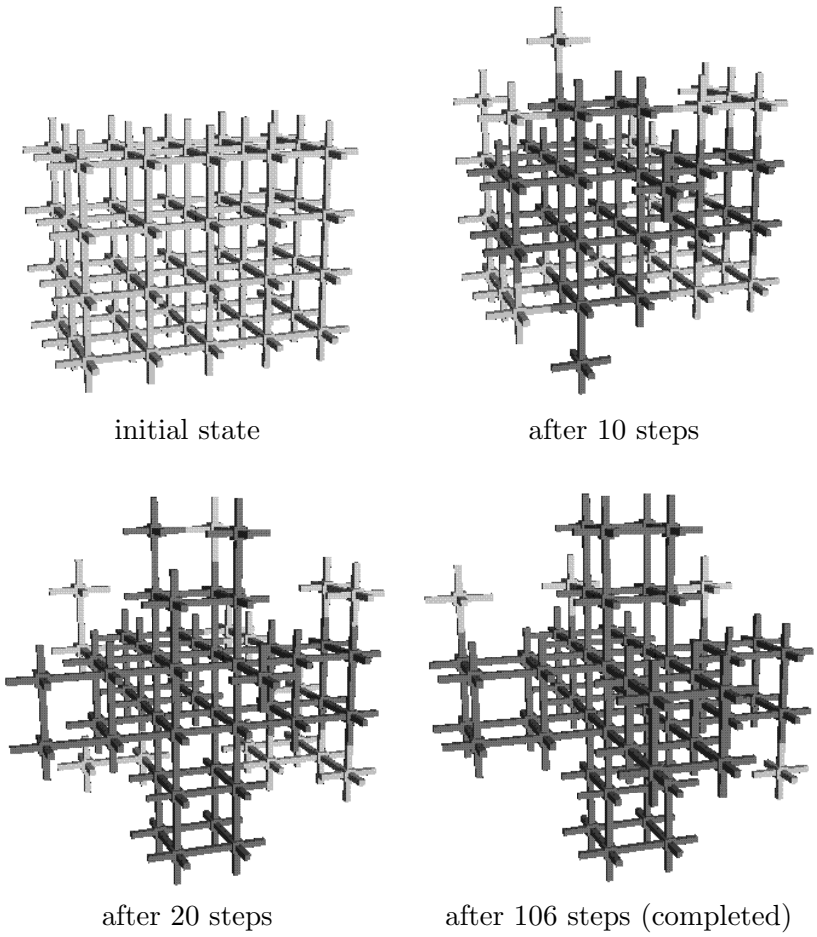


Fig. 16. Self-assembly simulation of many-unit structure

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 have demonstrated the self-assembly and self-repair functions successfully. This experimental results confirmed the hardware feasibility of the self-repairing mechanical system and opened its way to applications such as long-running explorer or surveillant in hazardous environments.

As a further challenge, we are currently on the way to 3-D system. We designed a prototype unit and confirmed its basic function of reconfiguration capacity. Self-assembly algorithms for small and large systems have been implemented as well, whose effectiveness was shown by computer simulations.

References

[1] M. Yim. New locomotion gates. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2508–1524, 1994.

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

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

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

[5] K. Kotay, D. Rus, M. Vona, and C. McGray. The self-reconfiguring robotic molecule. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 424–431, 1998.

[6] S. Murata, H. Kurokawa, and S. Kokaji. Self-assembling machine. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 441–448, 1994.

[7] K. Tomita, S. Murata, E. Yoshida, H. Kurokawa, and S. Kokaji. Reconfiguration method for a distributed mechanical system. In H .Asama *et.al*, editor, *Distributed Autonomous Robotic System 2*, pages 17–25. Springer, 1996.

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

[9] E. Yoshida, S. Murata, K. Tomita, H. Kurokawa, and S. Kokaji. Distributed formation control for a modular mechanical system. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robot and Systems (IROS'97)*, pages 1090–1097, 1997.

[10] H. Kurokawa, S. Murata, E. Yoshida, K. Tomita, and S. Kokaji. A 3-d self-reconfigurable structure and experiments. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS'98)*, pages 860–865, 1998.

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

[12] E. Yoshida, S. Murata, H. Kurokawa, K. Tomita, and S. Kokaji. A distributed reconfiguration method for 3-d homogeneous structure. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS'98)*, pages 852–860, 1998.