Self-Assembly and Self-Repair Method for a Distributed Mechanical System

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

***Abstract*— We propose a self-assembly and self-repair method for a homogeneous distributed mechanical system. We focus on a category of distributed systems composed of numbers of identical units which can dynamically change connections among themselves. Each unit has an on-board microprocessor, and local communication between neighboring units is possible. In this paper, we discuss a distributed method for a group of such units to metamorphose from an arbitrary configuration into a desired configuration through cooperation by the units. This process, called self-assembly, is realized by identical software on each unit with local inter-unit communication. An extension of self-assembly, self-repair, is also examined. In this process, an occasional cut-off of an arbitrary part of the system is assumed. When some part of the system detects damage, the whole system degenerates and reconstructs itself. Computer simulations show the feasibility of self-assembly and self-repair.**

***Index Terms*— Distributed mechanical system, self-assembly, self-repair.**

I. Introduction

A NOVEL structure of a mechanical system is presented in this paper—a homogeneous structure in which all the components of the system are identical. Each component, called a *unit,* is an active mechanical element which is capable of information processing and changing local connections with its neighbors. These units cooperate autonomously to form the given shape of the whole system (this function is called *self-assembly*), and reconstruct damaged parts without outside assistance (*self-repair*), as shown in Fig. 1.

In homogeneous systems, each unit is replaceable by other units, which makes the procedure of self-assembly and self-repair very simple compared with heterogeneous systems.

Replaceability: All the units are the same, have the same functions, and can be replaced by any other units.

Moreover, homogeneous systems have several advantages:

Reduction of production/maintenance cost: The homogeneity of the components is effective for mass production, and also simplifies inspection and maintenance procedures.

Design freedom: One can freely design the system by connecting the units like LEGO blocks.

Scale extensibility: A homogeneous system is capable of scale extension or contraction; the scale of the system can be changed by adding or removing units.

Manuscript received September 22, 1998; revised May 13, 1999. This paper was recommended for publication by Associate Editor E. Pagello and Editor S. Salcudean upon evaluation of the reviewers' comments.

The authors are with the Mechanical Engineering Laboratory, AIST, MITI, Ibaraki 305-8564, Japan.

Publisher Item Identifier S 1042-296X(99)09889-4.

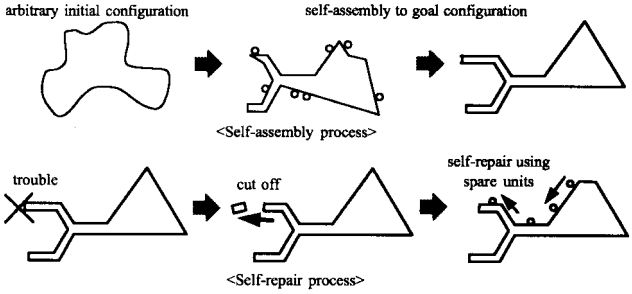


Fig. 1. Self-assembly and self-repair process of homogeneous system.

Homogeneous structures are common in nature; living things are made of cells, and the cells store identical genetic information. Many studies have been inspired by the homogeneous structures of natural things, and there is great potential in the design of homogeneous systems made of mechanical units.

In this paper, we concentrate on the software of self-assembly and self-repair, without any detailed discussion about the design of the unit hardware. The model is based on hardware we developed.

First, we propose decentralized software for self-assembly that is applicable to large systems composed of hundreds of units. It enables the units to assemble a pre-designed shape under two assumptions: system homogeneity and locality of communication. We introduce a method to locally describe and construct the complicated goal shape in a developmental fashion, which is a key technique in large-scale assembly.

We then extend the concept of self-assembly to "self-repair" by utilizing the developmental structure. In self-repair, the system can recover its original shape following arbitrary damage by backtracking the development process.

Self-assembly and self-repair are indispensable in many areas that human operators cannot access. This study will lead to a novel methodology for designing systems that operate continuously without outside help (e.g., in space, deep sea and nuclear power plants). Reconfigurability enables constructing many shapes, and thus performing many functions, using the same units. In space application, for instance, after transporting the units in a compact form, we can construct a solar panel at first, and then change it into an antenna. In the micro scale version, it will be possible to transport units in a narrow pipe and carry out some functions in another shape. Functions can be changed according to the number of units, and by supplying additional units, we can easily extend the system. Moreover, in these applications, fault tolerance is quite important because

external assistance cannot be expected. In our system, even when several units are damaged, the whole system can repair by itself.

This paper consists of eight sections. We summarize related work in the following section, describe the unit hardware briefly in Section III, and introduce an assembly method for a small system in Section IV. In Section V we show a method for developmental self-assembly. A method for self-repair, which is an extension of self-assembly, is given in Section VI. Simulation results are included in these two sections. We discuss limitations and extensions of the method in Section VII. Section VIII concludes the paper.

II. Related Work

Many studies have been inspired by the homogeneous structures of natural things. We will now summarize the research in both hardware and software.

The first and most important research in this field is von Neumann's *Theory of Self-Reproducing Automata* in 1966 [1]. In this book, he gave the detailed design of a self-reproducing pattern on a plane of cellular automata. Using the model, he opened a path to theoretical representation of self-reproduction. Unfortunately, it is not easy to apply his work to the design of a physical system since it is very abstract. Nevertheless, many followers continued the research of cellular automata, and a considerable number of studies have been conducted [2]–[9].

In contrast, few attempts have so far been made to construct "real" homogeneous artificial systems, due to many physical limitations such as geometric and dynamic constraints. Penrose made a brick model which symbolizes the metabolic process of living organisms [10]. The bricks connect to each other with simple interlocking mechanisms. Since the model does not have any information processing functions, it exhibits only very primitive behavior. The rapid progress of electronics made various microdevices available, and several artificial homogeneous systems using microchips appeared after the late 1980's. Ichikawa's "one-dimensional self-reproducing robot" [11] is an extension of Penrose's brick model. In the system, each brick has small driving wheels controlled by an on-board microprocessor and exhibits behavior similar to Penrose's model. Kokaji made a complex link mechanism called a "fractal machine" [12], which is made of identical triangular link units. Each unit is equipped with a microprocessor, and a network of microprocessors effectuates an inchworm-like movement of the structure. Homogeneity of both the physical structure and the software are realized in this system.

Several robotic systems made of "reconfigurable" mechanical units have recently been proposed. These systems can change their shape autonomously.

We made a two-dimensional mechanical unit called a "fractum" [13] that utilizes electromagnets for the connection, and verified by experiments its basic functions, such as the change of connections between units, transportation of units, and information exchange between units. Along with the hardware research, we developed homogeneously distributed software for self-assembly. Using this software, a system composed of ten units can start from an arbitrary configuration and metamorphose into any desired shape by cooperation through local information exchanges.

Chirikjian and Pamecha proposed a mechanical unit which has basically the same functions as ours [14]. The unit is a hexagonal link equipped with three servo motors for changing angles between adjacent links. The unit can change its local configuration by a combination of changing the shape of the links and the connecting mechanism between the links. Although the mechanical structure of this unit is complicated compared with our fractum, their unit has almost the same movement capability. Moreover, the mechanism is suitable for constructing a rigid structure and generating large torque. Another type of unit they proposed is a square-shaped unit with a sliding mechanism [15]. The unit has a sliding-connecting mechanism on each side.

Fukuda *et al.* proposed a type of hexagon-shaped mechanical unit [16]. It is one of various versions of a cellular robotic system called "CEBOT." Their unit has a function of connection and release between the units. When the units change configuration, a unit releases the connection, moves to the next position by itself, and reconnects again.

The features of our fractum compared with other units are as follows.

1) It has a simple mechanism.
2) It satisfies homogeneity of both software and hardware.

There are several ongoing projects to make 3-D reconfigurable systems [17]–[19]. Although these are still in the conceptual stage, they indicate the feasibility of a realistic 3-D reconfigurable structure.

Little research has been conducted on assembly software.

Lindenmayer proposed a mathematical model of development [20]. It is called an L-system and describes the development process based on cell division, which is difficult to realize through conventional mechanical units.

A self-assembly model of a bacteriophage was proposed by Thompson and Goel using movable finite automata [21]. A bacteriophage is constructed based on the interaction of protein subunits. In this model, the subunits are assumed to have connection points and can connect freely with other subunits according to their parameters, if they are close enough.

These assumptions (cell division and connection mechanisms) are not realistic with respect to current hardware technologies. Moreover, their purpose is the understanding of the biological development process, not engineering applications.

Chirikjian *et al.* proposed a configuration method for their units [14]. It is intended for optimality from a centralized point of view. Another method was proposed by Beni using ring structures [22].

In this paper, we assume the actual hardware units described in the following section, and describe self-assembly and self-repair.

III. Hardware Unit

We adopted a strict homogeneity of the structure, i.e., all the units in the system are of the same kind. This helps in making the system's design less complicated.

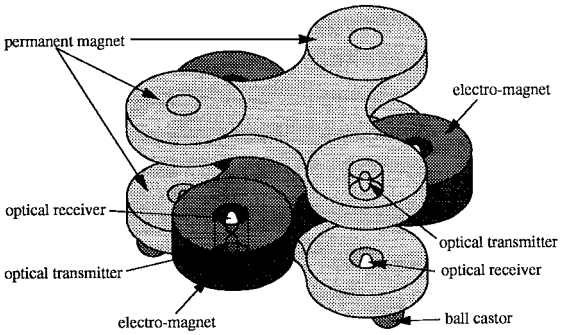


Fig. 2. Schematic of the unit.

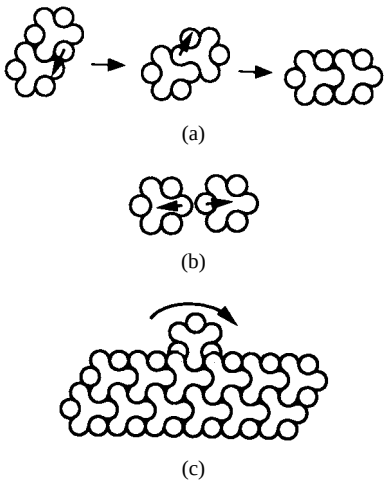


Fig. 3. Basic functions: (a) connection change, (b) connection cut, and (c) transportation.

We assume that each unit has an information processing capability and communication channels with neighboring units. A capability to change local connections among the units is also assumed. This capability is necessary to realize autonomous reconfiguration and replacement of (faulty) units.

We developed a two-dimensional (2-D) mechanical unit, called a *fractum,* which has the above capabilities. Fig. 2 shows a schematic view of the unit. Each unit has six connecting arms, three male and three double female arms. Each connecting arm is equipped with an electromagnet (male arm) or a pair of permanent magnets (female arm). All of the permanent magnets have the same polarity. If the polarity of the electromagnet of a neighboring unit coincides with the polarity of the permanent magnets, then the electromagnet is pulled into the gap between the permanent magnets. Disconnection takes place simply by reversing the polarity of the electromagnet. An on-board microprocessor controls the connection/disconnection of these arms independently. In addition, the on-board processor can communicate with neighboring processors by means of an optical transmitter and a receiver embedded in each connecting arm. Fig. 3 illustrates the basic movements of the units—Fig. 3(a) changing the connecting position, Fig. 3(b) cutting the connection between units, and Fig. 3(c) transporting a unit along the periphery of the unit group—which are possible through appropriate switching control. More complicated functions, such as forming or rearranging the shape, are possible by combining these basic movements. The simplicity of the unit structure (for instance, there are no internal moving parts) is very advantageous for micro-scale production. In the micro-scale version, electrostatic force instead of electromagnetic force might be used as the driving force of the units.

Fig. 4. Simple representation.

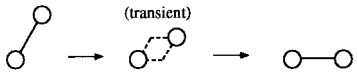


Fig. 5. Step movement.

Hereafter, to concentrate on the software side of the system, we abstract the unit hardware and introduce a simplified model and representation of the unit.

Model of the unit: The unit is represented by a circle with six lines, as shown in Fig. 4. Each line corresponds to a connection arm. The unit can rotate using the connection arm. Each unit is capable of communication with the adjacent units.

We often omit connection arms that are not connected. The connection change in Fig. 3(a) can be described as Fig. 5. This connection change, i.e., rotation by 60°, is one step movement of the unit. (The middle state in the figure is regarded as transient.)

In the following part of this paper, we will introduce a method for self-assembly and self-repair by using the above model of reconfigurable units.

IV. Simultaneous Assembly

In this section, we summarize the method of self-assembly which we had already developed [13].

This method is designed for a small system that includes about 10 units. The target of the method is self-assembly, to form a desired shape from an arbitrary initial configuration. We assume homogeneity (in particular, each unit has the same program) and locality (i.e., each unit can communicate with neighbors only).

In the method, a global target configuration is embedded in every unit and is described by a collection of local connection specifications. *Connection types* are introduced to represent the local connection.

The connecting state of each unit is represented by the arrangement of the connecting arms. The arrangements are classified into twelve types, which we call *connection types*, as shown in Fig. 6.

Using the connection type, we can describe the global configuration. For example, consider a triangle composed of ten units (Fig. 7). In the figure, the symbols indicate the connection types of the units. This configuration is described by three statements

$$\mathrm{o}\{\mathrm{K},\mathrm{K}\}$$
$$\mathrm{K}\{\mathrm{o},\mathrm{K},\mathrm{K},\mathrm{s}\}$$
$$\mathrm{s}\{\mathrm{K},\mathrm{K},\mathrm{K},\mathrm{K},\mathrm{K}\}.$$

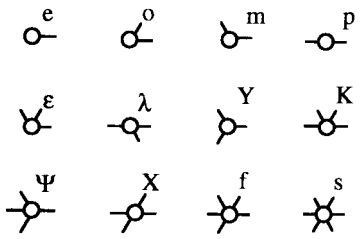


Fig. 6. Connection types.

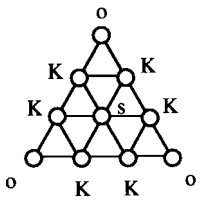


Fig. 7. Triangular configuration.

These statements describe three different local connective conditions in the configuration (i.e., corner, edge, and center). In each statement, the first letter indicates the type of the unit itself, and the letters in braces indicate the types of its neighbors. (These types are sorted in a fixed manner.) For instance, the second statement asserts that "there is a type K unit which connects with one type o unit, two type K units, and one type s unit in the final configuration."

Self-assembly proceeds as follows. Each unit evaluates the difference between the current connecting situation (connection types of itself and neighbors) and the goal statements. It then moves randomly if the difference is not 0. The frequency of the motion is decided according to the magnitude (degree) of the difference. By repeating this process, the overall system converges to the target configuration. In a simulation study, the success rate was 97% in assembling a triangle made of ten units. (Non-success includes two cases: the terminated assembly of a different shape and a nonterminating assembly activity in a fixed time.)

This method is simple and very effective when the system is small and the target configuration is symmetric. However, the efficiency declines for large scale construction. For example, the success rate in the case of a triangle made of 15 units (described by four statements) dropped to 73%. As the target configuration is described by local connections only, it is difficult to generate enough driving force to the global configuration. Also, the method is not suitable for constructing shapes with less symmetry, because goal statements becomes long and complicated, which make the assembly difficult.

V. Developmental Assembly

In this section, we propose a new method of self-assembly for large-scale construction, which is analogous to the development process of living things. A target configuration is embedded in every unit, which is similar to the genetic information in the DNA of living things. Mechanical units cannot divide, which is the critical difference between mechanical units and living cells. We will therefore provide a sufficient number of units from the beginning, and design a configuration method in which the units can be moved to the correct places.

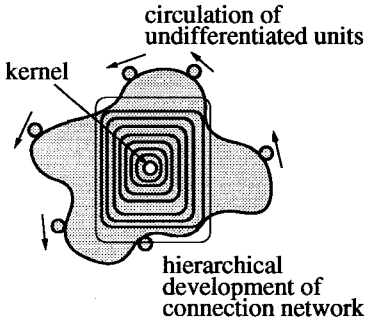


Fig. 8. Nucleation method.

The conceptual outline of the method is as follows. We assume that we have a set of connected units with an arbitrary shape. First, it chooses a unit as the origin of the construction. This unit is called the *kernel*. Next, the kernel gathers adjacent units to compose a *logical connection network* according to the embedded plan. This network is the first stage. The units involved in the first stage network then gather some surrounding units and form the second stage network. Repeating this process increases the stages, and the network grows stage by stage, approaching the target configuration. In this method, a unit joining the network mimics the biological differentiation process. The units in the network are called *differentiated*, and others are called *undifferentiated.*

Fig. 8 illustrates the three main points of the method—the kernel, the hierarchical development of the connection network, and the circulation of undifferentiated units. We call this a *nucleationmethod* because the network grows layer by layer from a unit, like the development of snowflakes. Compared with the method in the previous section, the difficulty in construction is drastically reduced by introducing the layer, because it acts as a kind of coordinate system and reduces the volume of search spaces. In the following sections, we will describe the method, first as an outline and then in detail, along with some simulation results.

A. Outline of the Method

1) Logical Connection Types: We introduced the connection types in Section III. Now we will define another kind of connection type, called a *logical connection type*. Hereafter, the former connection type will be called *physical connection type*.

If two units have a physical connection, and they estimate the connection to be a part of the target configuration, it is called a logical connection. In addition to the twelve physical connection types in Fig. 6, logical connections have another type, "n," which indicates that a unit has no logical connection.

2) Description Matrix: In the nucleation method, all units have an identical blueprint of the construction. It is called a *descriptionmatrix*, and includes all the necessary information for the self-assembly process. It is represented in the form of the lower triangular matrix shown in Fig. 9. The symbol "–" indicates that the units keep the same logical type of the previous line.

stage (s) \ location index (l)	0	1	2	3	4	5	6	7	8
0	n								
1	Y	e							
2	s	o	o						
3	–	K	K	o					
4	–	–	f	ε	o				
5	–	–	–	K	ε	o			
6	–	–	–	–	K	ε	o		
7	–	–	–	–	–	K	ε	o	
8	–	–	–	–	–	–	K	ε	o

Fig. 9. Description matrix.

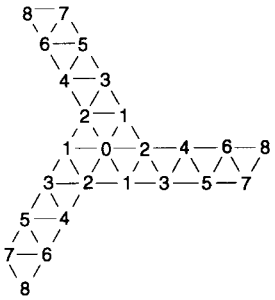


Fig. 10. Location indices.

Each line of the matrix describes a temporary goal (sub-goal) of the logical connection network. When this sub-goal is globally reached, the sub-goal is changed according to the next description, as the stage number s is incremented. Then, each unit in the network tries to find additional units to satisfy this new sub-goal. Thus, the logical connection network grows layer by layer.

Besides the stage number s, each unit possesses a number called a location index denoted by l. This number is initially -1 except that the kernel unit has the value $l = 0$. When the unit with $l = -1$ joins the network, l is set to some number. Once it is set, it does not change. Each unit accesses the description matrix with two variables, s and l, and determines which logical connection type it should take in the sub-goal network.

The description matrix of Fig. 9 corresponds to a pinwheel-like shape of 25 units (Fig. 10). Each number in Fig. 10 indicates the location index, and the lines indicate logical connections.

By using the description matrix, the units expands the logical network until the target configuration is constructed. We illustrate these steps using the same example. The logical network in each stage is shown in Fig. 11.[1] Only the kernel exists in the logical network in the initial state. The second row "(Y e)" of the description matrix indicates that the kernel $(l = 0)$ should take logical type "Y," and should try to establish logical connections with three[2] of the surrounding

[1]The logical network is usually surrounded by undifferentiated units, which are omitted in the figure.

[2]In this description method, the number of units to be added to the logical network in each stage is not explicitly given in the matrix. Instead, it is determined implicitly by the consistency of the network. Also, in this example, there are two possible locations for these three units. It is not explicitly determined.

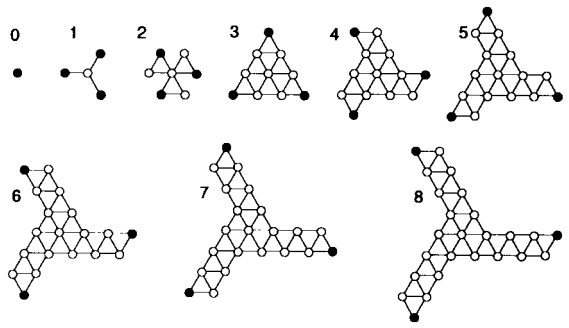


Fig. 11. Logical network in each stage.

units. The three units added in the network should take type "e" and their location indices are set 1. Thus, the first stage network is formed. Then the stage number is incremented and the sub-goal is changed as the third description "(s o o)."

Each unit in the connection network changes its logical type according to the plan, and at the final configuration the physical types coincide with the logical types[3]. Apparently, many different routes are possible to achieve the same target configuration, thus many description matrices exist for the same goal configuration.

B. Procedures of Nucleation Method

1) Structure of One Step Execution: We assume that all the units share a synchronized clock. This is accomplished by the method introduced in [23] under the restrictions of homogeneity and locality.

The structure of the self-assembling procedure is simple. Each unit repeats the following procedures synchronously, after executing the "kernel selection" procedure once at the beginning.

1) Joining logical network (Undifferentiated units get location indices if possible).
2) Constructing logical connections (Units search for the direction of the logical connection).
3) Stage update (Units evaluate the completion of the current stage network, and increment their stages if completed).
4) Unit circulation (Units on the periphery circulate counterclockwise).

One execution of procedures 1) through 4_ is called a *step* of the nucleation method.

It is a distributed parallel process and also homogeneous. At the initial state, each unit has the same program and data (values of variables). Communications between adjacent units are performed in each procedure. All information transferred between units is an integer value, such as the random number for kernel selection, physical type, stage, and logical connection request. Each procedure is described in the following sections.

2) Kernel Selection: In the first stage of the nucleation method, it is necessary to choose a particular unit from which

[3]If the system has spare units, physical types and logical types are not necessarily the same around the spare units when the target configuration has been constructed.

self-assembly starts among the units. Note again that all the units are driven by identical programs. In the Appendix A, we will show one possible method, which is based on random number generation.

The kernel sets $l = 0$ and $s = 1$ when it is chosen. (Other units have $l = -1$ and $s = 0$.)

3) Joining Logical Network: At the beginning of each repetition of the step, the undifferentiated units try to join the logical connection network.

Joining Logical Network: An undifferentiated unit (unit A) is called a {\it candidate unit} if it detects that its adjacent unit (unit B) is already involved in the network, and that B has a logical connection arm (request) to A. When A becomes a candidate, it tentatively sets its location index and stage. The values are the same as the stage of unit B. (If the stage of unit B is s_B, unit A sets $s = s_B$ and $l = s_B$.)

The candidate units extend logical connection arms according to the connection types decided by its stage and location index.

4) Constructing Logical Connections: Because there is no information about the directions of logical connection arms in the description matrix, each unit must find appropriate ones by itself. For instance, if a unit is assumed to take logical type "o," the unit needs to ascertain which two arms should be connected among its six arms. (This is performed by both differentiated units and candidate units.) We will omit the detail, but each unit tries one possible direction in each step, and finds appropriate directions of logical connections after some steps. Thus, the logical connection network fits the new units into itself in the correct direction. In the process, some candidate unit may becomes noncandidate again.

5) Stage Update: We will describe here how the units locally confirm the global completion of the logical network at each stage and proceed to the next stage. The completion is defined as follows. A unit is called *satisfied* if and only if all the logical connection arms of the unit are connected with other units' logical connection arms. The *completion* of a connection network at any stage indicates that all the units in the connection network are satisfied. (To be precise, connection network described here is composed of both differentiated units and candidate units.)

The decision of completion of a stage must be performed locally in a distributed manner. Each unit evaluates local completion of the connection network of each stage by exchanging values called *completeness*. If the value exceeds the threshold TH, then the unit deems that the stage has been completed locally for that unit.

Completion of the network at each stage can be decided by the following method [12].

Least completeness propagation: Each unit decides the completeness of the current stage as follows: If the unit is satisfied, then it detects the completeness in the same stage of its logically adjoining units, and lets the minimum value be c. The new completeness of the unit is then $c + 1$. If the unit is not satisfied, the completeness is 0.

Using this method, if the logical network of a stage is completed, each unit increments the completeness of the stage, and finally the value exceeds the threshold. Note that all the units do not necessarily proceed to the next stage simultaneously. More precisely, if every unit in a circle of radius[4] TH is satisfied in the current stage, then the center unit proceeds to the next stage.

When the units update their stages, candidate units become differentiated, and the location indices are fixed to the tentative value. The units in the completed network change their logical types according to the description matrix. Thus, in the next step, the units can try to extend the logical connection network by adding logical connection arms.

6) Unit Circulation: We devised a method of supplying units to necessary places. In the nucleation method, as the stage proceeds, the connection network expands beyond the (physical) edge of the units. If it is possible to supply undifferentiated units to those places, the logical network can grow, as long as there are some undifferentiated units, and it will form the target configuration.

Because the periphery of the units is a closed curve, units can be simply supplied by the following method:

Unit supply by circulation: We divided physical types into movable and unmovable. If a unit is of a movable physical type and is undifferentiated, then it moves one step counterclockwise at the probability of P_{move}.

By this method, undifferentiated units circulate on the periphery counterclockwise. This is simple but needs some modification as shown in Appendix B.

C. Simulation

We demonstrated the feasibility of the method using simulations. The main parameters of the method were P_{move} and TH. These values must be chosen carefully since these parameters affect each other. For example, if we require TH to be large, which means that a stage of a unit is not updated until all the units in large radius TH are satisfied, then we need to take small P_{move} to suppress the supply of undifferentiated units. Otherwise, the network cannot complete the stage because even units in the correct position move too often. However, if this is performed to excess, completion becomes very slow due to stagnation of the unit supply. We use the following values in simulation studies throughout this paper: $P_{move} = 0.8$ and $TH = 5$. (Several simulation with different values showed that the method is not so sensitive to the values.)

We conducted several simulation studies. The first example uses the description matrix in Fig. 9. Fig. 12 shows a simulated configuration sequence. In this figure, the numbers indicate the stages of the units. All the units are connected physically; the lines indicate logical connections. All self-assembly simulations succeeded in 1000 trials. The total simulation steps ranged from 134 to 2323 with an average of 590.

Another example is a larger configuration composed of 75 units, shown in Fig. 13. The numbers in the figure indicate the location indices. Self-assembly succeeded in 979 cases out of 1000 in this configuration. The required steps ranged from

[4] In this paper we will assume that the distance is measured based on the minimum number of existing (or connecting) units between two units. The distance of the adjoining units is 1.

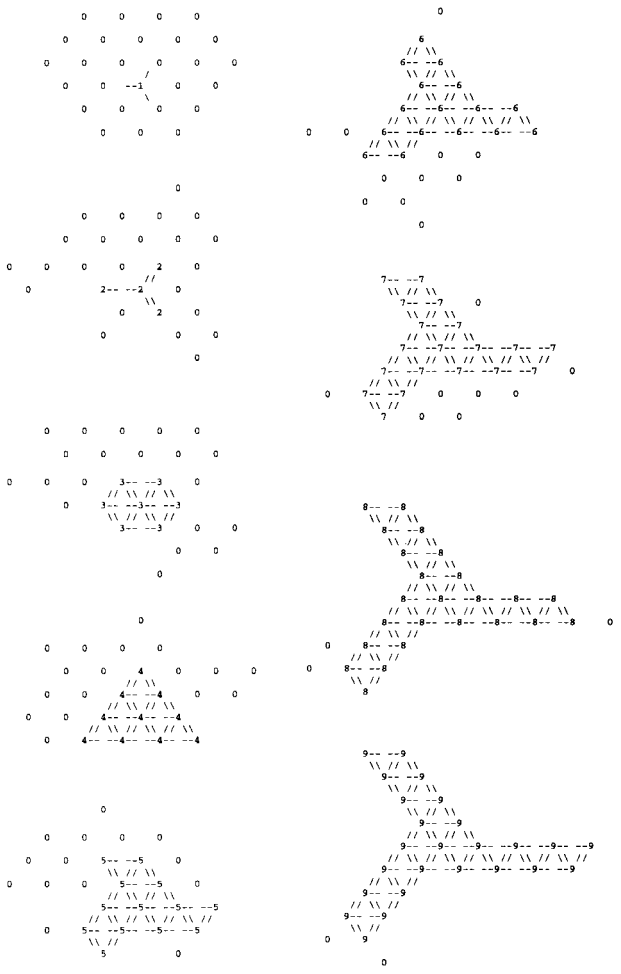


Fig. 12. Simulated sequence of self-assembly.

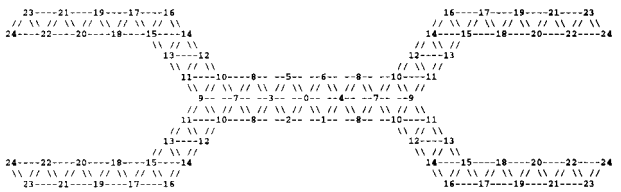


Fig. 13. Configuration of wrench-like shape.

1702 to 4775 and the average was 2600. A simulated sequence is shown in Fig. 14. The reason for these failures is that the units made a hole inside of their configuration, and it is difficult to fill such a hole when it is generated in the early stage of the development.

VI. Self-Repair

In this section, we describe self-repair of the system following a breakdown. Self repair can be performed with a simple procedure due to the layered structure of the system.

There are many types of failures to be considered. In this paper, we deal with one category of failure wherein some units are removed from the system. We assume that the remaining units work correctly. The removal may include several units and may occur at any place in the system at any time. Thus, not only the units on the outer surface but also the ones inside of the logical network may be removed, if the remaining units are connected.

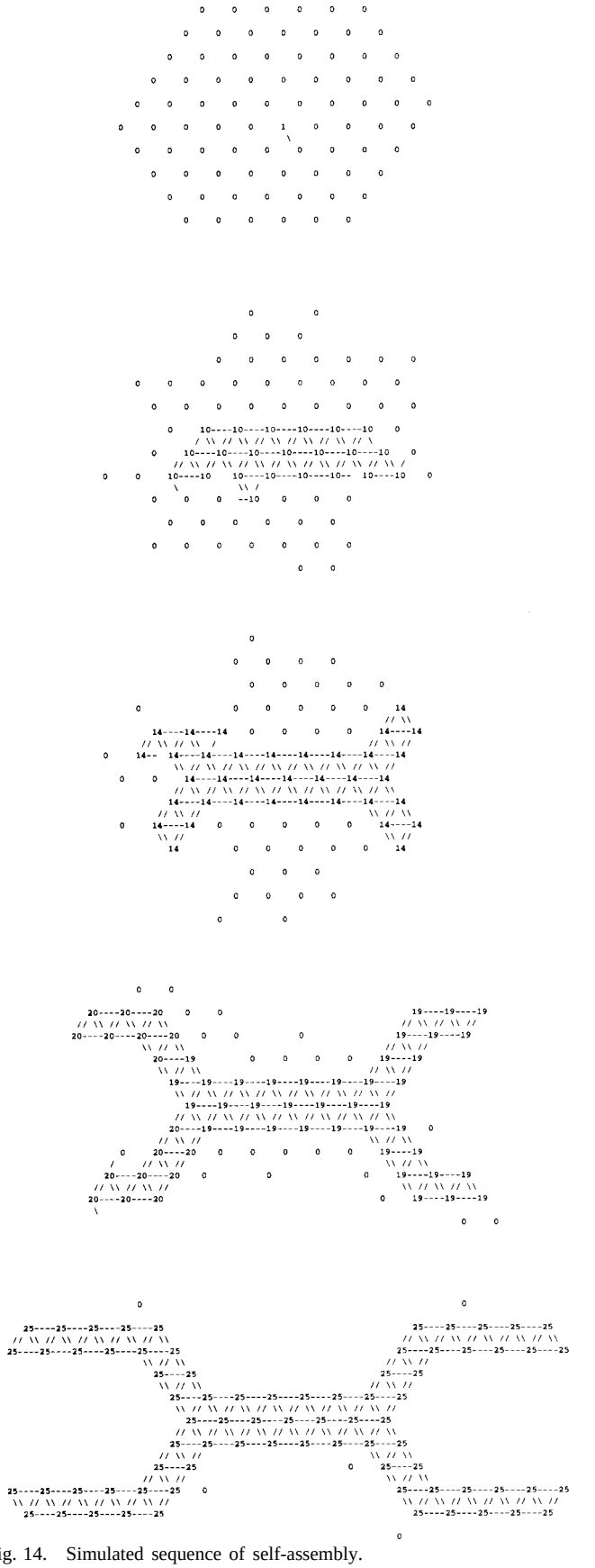


Fig. 14. Simulated sequence of self-assembly.

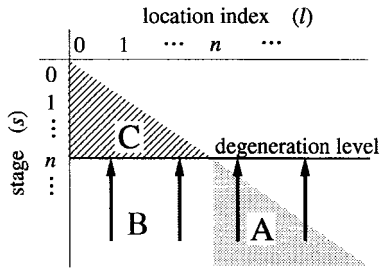


Fig. 15. Actions of degeneration.

The strategy is to transport spare units to the area of the damage and refill it. If necessary, some units become undifferentiated again during this process. This self-repair can be performed by degeneration of the system to the previous stage, which is the same location index as the removed units. The system stops the ongoing development, returns to an earlier stage where negative effects of the removal do not yet exist, and resumes construction. The procedure consists of failure detection, transmission of a signal, and degeneration.

A. Self-Repair Method

1) Failure Detection: Failures can be detected by the units involved in the connection network as follows: if unit A has a logical connection with unit B, and unit B does not respond to a signal from A (e.g., querying its logical connection, etc.), then unit A deems that unit B has been removed.

2) Degeneration Signal: Degeneration of the stage is performed according to a signal called a degeneration signal. It is transmitted by the units that have detected failure, and the signal is propagated to the overall system. The signal includes the level of the failure, which determines the stage to which every unit should return. The level is the location index of the removed units. To make provisions against failure, each unit must store the location indices of the adjoining units.

3) Degeneration: If some units are removed, and their minimum location index is n, then the system must return to stage n. To accomplish this, every unit with a stage greater than n (areas A and B of the description matrix in Fig. 15) must return to stage n. Units in area A, which are involved in the network after stage n, become undifferentiated again. Each unit in area B must set the previous logical type and direction at stage n again. For that purpose, each unit also needs to store the history of the directions of the logical type at each stage. Units in area C do not do anything. They are designed to stop the signal and to ignore unnecessary degeneration signals. The precise procedure is formulated as follows.

Let l be the location index of a unit, s the stage of the unit, and n the smallest level of degeneration signal received from the neighbors. If $n < s$, then the unit transmits a degeneration signal of level n to all neighboring units. Then, according to the level of n, it performs the following.

1) If $n < l$ (area A), the unit is initialized and it returns to the undifferentiated state.
2) If $l \leq n < s$ (area B), the stage of the unit is set to n. The units adjust their variables (logical type, etc.) according to the stage.
3) If $s \leq n$ (area C), the unit does not take any action, i.e., it ignores the signal.

Through this procedure, the signal is spread to the overall system and the system degenerates to the appropriate stage. If several units are removed, the minimum location index is consequently effective. Also, the signal does not remain after the completion of degeneration.

The construction and degeneration may be performed concurrently; i.e., after the transmission of a degeneration signal, the units resume construction immediately. If several failures occur in the system, signals of several levels are spread, and the system goes back to the lowest level. It must be noted that if the kernel is removed, the units must begin again with the kernel selection process.

B. Simulation

We conducted simulation studies for self-repair. The procedure of self-repair is embedded between procedures 3) and 4) in Section IV-B1.

A simulated sequence of self-repair for the configuration of Fig. 13 is shown in Fig. 16. The numbers indicate the stages of the units. There are ten spare units in the initial configuration. Three units are removed from the upper right position (b) after the target configuration is finished. Six units are removed from the upper left position while the degeneration signal is propagated (d). The overall system degenerates to stage 19 (g), resumes construction, and successfully forms the target configuration again (h).

VII. Discussion

In this section we discuss some limitations and extensions of the method.

The nucleation method constructs several shapes according to the description matrix, and any connected shape has corresponding description matrices. Therefore, in principle, we can construct any connected shape using the nucleation method, if we have sufficient number of units. The easiness of construction differs widely depending on the target shapes, but it is difficult to describe it quantitatively. Construction is easy if the shape is symmetric, has few protruding parts, has no hole inside, and not be made of many units.

It would be useful to introduce several extensions to the description matrix, like programming languages. One extension is to use several description matrices. The overall system is described by a collection of several components. This extension is possible by introducing a new entry which indicates a link to another description matrix. It is effective in concise description, especially for iterative structures. Fig. 17(a) is an example of a configuration with a iterative structure, and Fig. 17(b) shows its two description matrices.

As a failure of the system, we considered only the simplest case that several units are removed from the system and remained units work correctly. Many other cases of failures can be considered. Another simple failure is halting failure, in which failed units do nothing. Our method can be applied to this case after cutting off the failure units from the system.

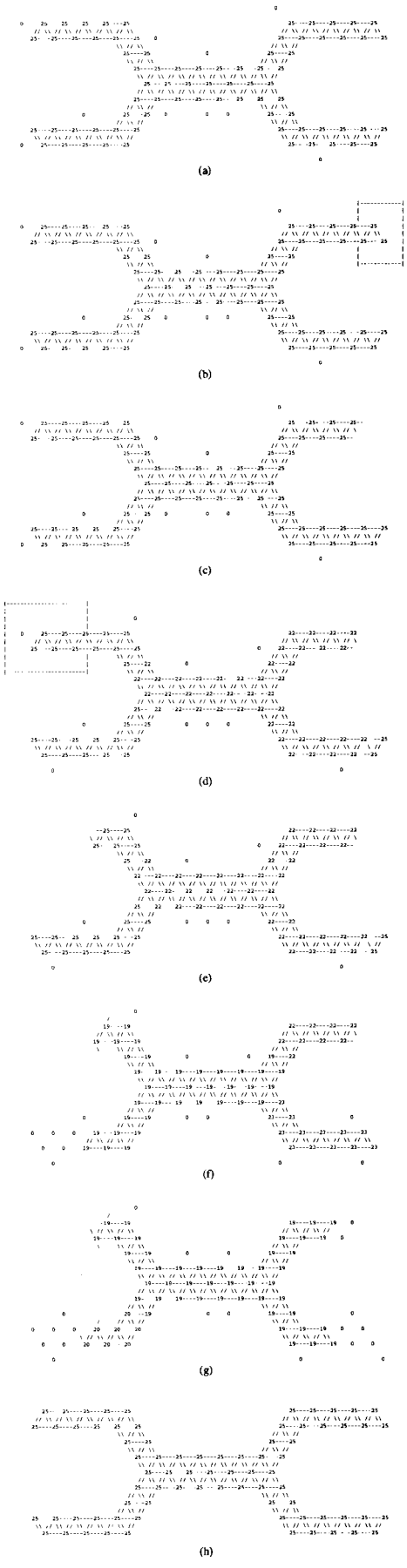


Fig. 16. Simulated sequence of self-repair.

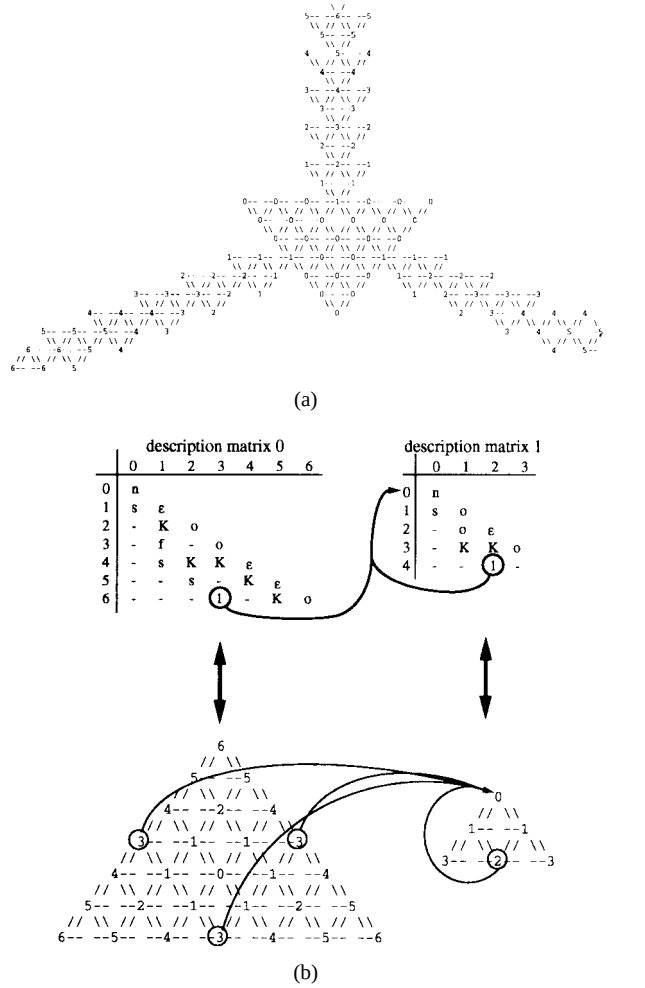


Fig. 17. Construction with two description matrices. An example of (a) iterative structure and (b) description matrices.

Recover from other failures (e.g., Byzantine failure) will be much more complicated.

In order to show the feasibility of the homogeneous distributed mechanical systems in the physical world, we have a lot of future work, mentioned a few in the following.

1) A method of efficient power supply to the units will be necessary.
2) Miniaturization for micro-scale assembly should be considered.
3) For a practical application, 3-D systems will be necessary.

As an example, we have proposed a 3-D model in [19]. By solving these issues, the feasibility of the systems will be enhanced.

VIII. Conclusion

We have proposed a design of a system composed of identical units, and a decentralized method of self-assembly and self-repair. The method, called the nucleation method, is analogous to the development process of living things. It is composed of the kernel selection and developmental growth of a logical network based on a description matrix. Due to the nature of mechanical units (i.e., they cannot self-reproduce), a

circulation method for supplying units was introduced. The system is organized into a layered structure, which makes the self-repair process simple, using a degeneration signal. Simulation studies demonstrated the feasibility of the self-assembly and self-repair methods.

Self-repair is indispensable in many areas that human operators cannot access, e.g., space and deep sea. This study might lead to a novel methodology for designing systems that operate continuously without outside help.

Appendix A
Kernel Selection

One possible kernel selection procedure is as follows:

Kernel selection: Assume that every unit is in the same state (values of the variables) at the beginning.[5] Each unit generates a random number and assigns it to its two variables m and r. It then repeats the following N times: it compares the value of m with that of the adjacent units; if another unit has a larger value, then it assigns the value to the variable m. After the repetition, if $m = r$ holds, then there is no unit with a larger value in the vicinity within the distance N. Thus, by taking N to be large enough to cover the whole system, one unit with the largest r value can be chosen as the kernel.[6]

Appendix B
Unit Ccirculation

For efficient assembly, the unit circulation procedure described in Section V-B6 needs some modifications as follows.

1) This method often makes projections (or protruding parts) with undifferentiated units; the supply of units then becomes ineffective. Some additional rules are necessary to keep the periphery smooth. For example, if the move of an undifferentiated unit makes another unit unmovable, then the move is suppressed.
2) First we chose types ''e,'' ''o,'' and ''ϵ'' as movable types, but then no unit could move in some configurations that were not the target. The easiest way to avoid such a deadlock is to define most of the types as movable. But this is not acceptable because of physical requirements (the whole system must be connected) and ineffective convergence. We therefore defined types ''m,'' ''λ,'' and ''K'' as also movable, but the probabilities of their moving are quite low compared to P_{move} for "e," "o," and "ϵ."

References

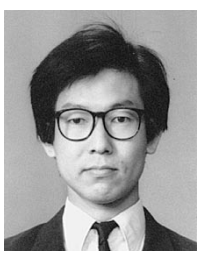
[1] J. von Neumann, *Theory of Self-Reproducing Automata*. Urbana, IL: Univ. Illinois Press, 1966.
[2] T. Toffoli and N. Margolus, *Cellular Automata Machines*. Cambridge, MA: MIT Press, 1987.
[3] E. F. Codd, *Cellular Automata*. New York: Academic, 1968.
[4] A. W. Burks (ed.), *Essays on Cellular Automata*, Univ. Illinois Press, 1970.
[5] J. Myhill, "The Abstract Theory of Self-Reproduction" in *Essays on Cellular Automata*, A. W. Burks, Ed. Urbana, IL: Univ. of Illinois, 1970.
[6] S. Wolfram, *Cellular Automata and Complexity: Collected Papers*. Reading, MA: Addison-Wesley, 1994.
[7] C. G. Langton, Ed., *ALIFE I* Reading, MA: Addison-Wesley, 1989.
[8] C. G. Langton, C. Taylor, J. D. Farmer and S. Rasmussen, Eds., *ALIFE II*. Reading, MA: Addison-Wesley, 1991.
[9] C. G. Langton, Ed., *ALIFE III*. Reading, MA: Addison-Wesley, 1994.
[10] L. S. Penrose, "Self-reproducing machines," *Sci. Amer.*, vol. 200, no. 6, pp. 105–114, 1959.
[11] Y. Ichikawa and F. Ohkido, "One-dimensional self-reproducing robot," in *Proc. 1991 Int. Conf. Ind. Electron., Contr. Instrum.*, 1991, pp. 963–966.
[12] S. Kokaji, ''A fractal mechanism and a decentralized control method,'' in *Proc. USA-Jpn. Symp. Flexible Automat.*, 1988, pp. 1129–1134.
[13] S. Murata, H. Kurokawa and S. Kokaji, ''Self-assembling machine,'' in *Proc. 1994 IEEE Int. Conf. Robot. Automat.*, 1994, pp. 441–448.
[14] G. Chirikjian, A. Pamecha and I. Ebert-Uphoff, "Evaluating efficiency of self-reconfiguration in a class of modular robots," *J. Robot. Syst.*, vol. 13, pp. 317–338, Oct. 1996.
[15] A. Pamecha, C.-J. Chiang, D. Stein and G. Chirikjian, "Design and implementation of metamorphic robots," in *Proc. 1996 ASME Design Eng. Tech. Conf. Comp. Eng. Conf.*, 1996.
[16] T. Ueyama, T. Fukuda, Y. Itou and H. Asama, "A study on dynamically reconfigurable robotic system (21st report, experimental results on automatic approach, connection and separation by series III," in *Proc. JSME Annu. Conf. Robot. Mechatron.*, 1990, pp. 345–348, (in Japanese).
[17] M. Yim, "Locomotion with a unit-modular reconfigurable robot," Ph.D. dissertation, Dept. Mech. Eng., Stanford Univ., Stanford, CA, 1994.
[18] K. Kotay, D. Rus, M. Vana and C. Mcgray, "The self-reconfiguring robotic molecule," in *Proc. 1998 Int. Conf. Robot. Automat.*, 1998, pp. 424–431.
[19] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita and S. Kokaji, "A 3-D self-reconfigurable structure," in *Proc. 1998 Int. Conf. Robot. Automat.*, 1998, pp. 432–439.
[20] A. Lindenmayer and G. Rozenburg, Eds., *Automata, Language, Development*. Amsterdam, The Netherlands: North-Holland 1976.
[21] N. S. Goel and R. L. Thompson, "Movable finite automata (MFA): A new tool for computer modeling of living systems," in *ALIFE I*, C. G. Langton, Ed. Reading, MA: Addison-Wesley, 1989.
[22] G. Beni, "Research perspectives in swarm intelligence the reconfiguration problem," in *Proc. Int. Symp. Syst. Life*, 1997, pp. 51–59.
[23] S. Kokaji, S. Murata, H. Kurokawa and K. Tomita, "Clock synchronization mechanisms for a distributed autonomous system," *J. Robot. Mechatron.*, vol. 8, no. 5, pp. 427–434, 1996.

[5] More precisely, we assume that the states differ only in the seeds for random number generation. In the physical model, this deviation can be easily achieved.

[6] When the resolution of random numbers is insufficient, two or more units may take the largest value. In this case, one unit can be chosen by repeating the procedure among these units.

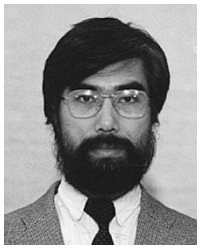
Kohji Tomita (M'99) received the B.E., M.E., and Ph.D. degrees in engineering from the University of Tsukuba, Tsukuba, Japan, in 1988, 1990 and 1997, respectively.

He joined the Mechanical Engineering Laboratory, AIST, MITI, Ibaraki, Japan, in 1990. He is now a Senior Researcher with the Machine Intelligence Division, MITI. His research interests include distributed autonomous systems and machine intelligence.



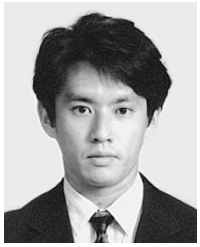
Satoshi Murata received the B.E., M.E., and Dr. Eng. degrees in aeronautical engineering from Nagoya University, Nagoya, Japan, in 1984, 1986, and 1997, respectively.

In 1986, he joined the Mechanical Engineering Laboratory, AIST, MITI, Ibaraki, Japan, and is now a Senior Researcher with the Systems Science Division. His interest includes distributed mechanical system especially self-assembling and self-repairable systems.



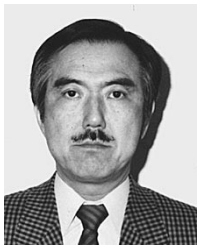
Haruhisa Kurokawa received the B.E and M.E degrees in precision machinery engineering, and the Dr.Eng. degree in aero- and astronautical engineering, from the University of Tokyo, Tokyo, Japan, in 1978, 1981, and 1997 respectively.

He is the head of the Sound and Vibration Division, Mechanical Engineering Laboratory, AIST, MITI, Ibaraki, 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 was with the Department of Microtechnique, Swiss Federal Institute of Technology at Lausanne (EPFL), Lausanne, Switzerland. He is conducting research in the Mechanical Engineering Laboratory, AIST, MITI, Ibaraki, Japan. His research interests includes decentralized autonomous systems.



Shigeru Kokaji 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 Director of the Department of Advanced Machinery, Mechanical Engineering Laboratory, AIST, MITI, Ibaraki, Japan. His research interests include distributed control of mechanical/robotic systems.