

自律分散機械による 3 次元形状の自己組み立てと自己修復[†]吉 田 英 一*・村 田 智*・小 鍛 治 繁*
富 田 康 治*・黒 河 治 久*

Self-assembly and Self-repair of 3-D Structure by an Autonomous Distributed Machine

Eiichi YOSHIDA*, Satoshi MURATA*, Shigeru KOKAJI*,
Kohji TOMITA* and Haruhisa KUOKAWA*

A distributed self-assembly and self-repair method is proposed for a 3-D reconfigurable machine, composed of many identical mechanical units. The method aims to enable the machine to transform itself into desired structure from an arbitrary initial configuration. The proposed method is implemented in such a way that each unit has identical software, so that any unit can play any role in the system. The method consists of two parts, for small and large scale systems. The first part is featured by a stochastic relaxation process, which allows the system to converge to a given target structure by searching for a proper unit motion over many degrees of freedom. We have also developed another self-assembly and self-repair method dedicated to effective self-assembly and self-repair for large scale systems by using layered and recursive description of target shape. The effectiveness of the proposed method is verified through a series of computer simulations.

Key Words: Autonomous distributed system, Self-reconfigurable machine, Self-assembly and self-repair

1. はじめに

近年、機械システムを構成する要素が増加し、その構造が複雑化するのに伴い、それらの保守や維持がますます困難になってきている。一方、システムを、

- 局所的な相互作用に基づいて (分散性)
- 上位構造からの命令によらずに、主体的に (自律性)

行動を決定する多数の要素で構成し、全体として与えられた要求を満たすように構成要素とその相互作用を設計する自律分散システム^{1), 2)}の研究が進んでいる。この概念を応用することにより、目的や環境に応じてその構造を変更する柔軟性と、システムの一部が故障しても機能を回復する耐故障性を持つ機械システムの実現が期待される。本論文では、多数のユニットで自律分散型の機械システムを構成し、さまざまな立体形状を形成する自己組み立てと、一部が故障しても自律的に修復する自己修復とを、局所的な情報に基づいて分散的に実現する手法について述べる。

本論文では、均質なハードウェア・ソフトウェアを持つユニットにより機械システムを構成する。これにより、ユニットは全体のどの部分でも受け持つことができるため、故障時

の取り換えが容易になる。また、組み立て・修復に対する制約が少なく、これを効率的に行うことが可能になる。さらに、分散的な自己組み立て・自己修復の実現の観点から、ユニットには局所的な通信と能動的な再構成能力が重要である。

以上の考えに基づき、筆者らは 2 次元形状を生成する機械ユニット「フラクタム」を開発し、分散的な自己組み立て・自己修復手法を提案してきた^{3), 4)}。これらの研究により、自律分散型機械システムの有効性が示されたので、より広範な応用を目指し、3 次元形状を構成可能な機械ユニットを開発した⁵⁾。具体的には、宇宙や深海における構造物やロボットなど、極限環境で長時間稼働する機械への応用が考えられる。特に宇宙応用に関しては、折りたたんで運搬し、自己組み立てによる展開と故障時に自らの修復が可能な衛星アンテナや惑星探索ローバなどの実現が期待される。そこで、3 次元形状の分散的な自己組み立て・自己修復の手法の開発が重要な課題となる。

これまで、再構成可能な機械・ロボットシステムとして、さまざまな研究が行われている。非均質な可変構造ロボットとして、福田らの一連の研究がある⁶⁾、また、ユニット型の可変構造ロボットの動作計画を集中的な制御で行う研究も多く見られる^{7)~10)}。一方、セルオートマトン (CA) を用いて、形態形成をシミュレーションする研究^{11)~13)}も近年盛んである。筆者らの研究^{3), 4)}は、上記で説明した通り、均質なハードウェア・ソフトウェアを有するユニットを開発し、分散制

[†] 第 10/11 回分散シンポジウムで発表 (1998/1999・1)

* 機械技術研究所 茨城県つくば市並木 1-2

* Mechanical Engineering Laboratory, Tsukuba-shi, Tokyo
(Received January 29, 1999)
(Revised June 3, 1999)

御により自己組み立て・自己修復を実現した点で特徴づけられる。

しかしながら、自律分散型の機械システムによる3次元形状の自己組み立て・自己修復に関する研究はこれまでほとんど行われていない。そこで本論文では、多数の均質な機械ユニットによりこれを分散的に実現する手法を構築することを目的とする。3次元形状の自己組み立て・自己修復の手法においては、2次元の場合と比較して多くの自由度に対処する必要がある。個々のユニットは、ソフトウェアの均質性を保ちつつ、近傍との通信を用いて多くの自由度の中から行動を決定しなければならない。本論文では、まず2章で開発した3次元機械ユニットの概要について述べる。次に3章で接続状態の単純な記述を導入し、確率的な緩和手法に基づく手法を提案する。これにより、10~20ユニット程度の小規模なシステムで与えられた3次元形状の自己組み立てが分散的に行われることを計算機シミュレーションにより示す。しかし、システムが大規模になると、この手法をそのまま適用することが困難になる。そこで、4章では大規模システムを再帰的な方法で簡便に記述する方法を新たに導入し、段階的に役割を分化させて自己組み立て・自己修復を行う手法について述べる。計算機シミュレーションにより、提案した手法が50ユニット以上の大規模なシステムについて有効に動作することを確認する。

2. 3次元機械ユニットの概要

本章では、設計した3次元機械ユニット⁵⁾の動作を簡単に説明する。Fig. 1に示すように、各ユニットは直交する軸方向に6本の結合腕を持ち、それぞれ回転が可能である。腕の回転と結合は、一つのDCモータを駆動源として電磁クラッチ・ソレノイドを通じて制御する。Photo. 1は、結合腕を取り付けたベース上で動作する4台のユニットを示す。

ユニットは、2台組になって移動する動作を基本的な動作とする。Fig. 2(a)に示すように、4個のユニットからなる平面上にあるユニットX, Yを考える。ユニットYは、Zとの結合を切り離し、Xがb-b軸周りに90°回転することにより、Fig. 2(b)に示す位置に移動する。Fig. 2からわかるように、ユニットは3次元格子上の2近傍距離上の点を移動するので、格子を黒と白の「チェッカーボード」に分割する

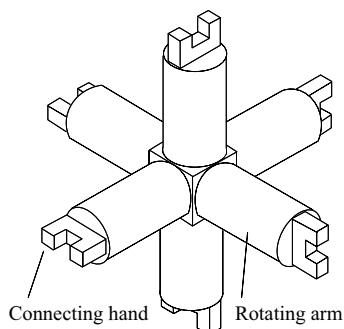


Fig. 1 A schematic view of a 3-D unit.

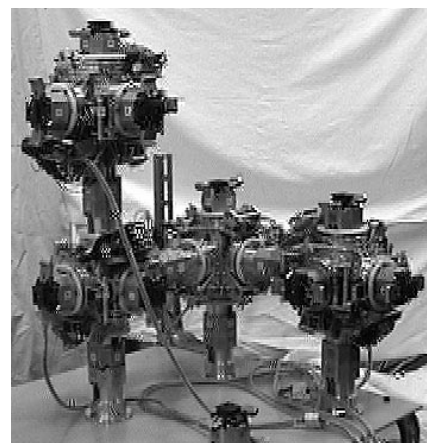


Photo. 1 Hardware setup of four 3-D mechanical units.

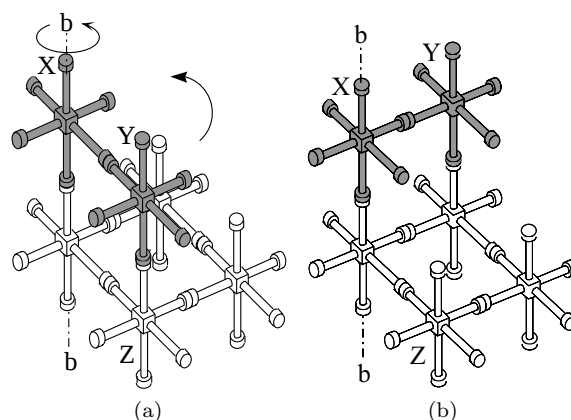


Fig. 2 Typical pairwise motion of 3-D unit (a) initial configuration. (b) final configuration after pairwise motion of X and Y (X rotated about axis b-b).

と、そのどちらかの色の位置のみ移動できるという性質を持つ。このような動作を繰り返して、ユニットは3次元格子点上を移動し、さまざまな構造物を構成できる。ユニットにマイクロコンピュータを搭載し、結合ハンドに光通信デバイスを装着することにより、自律的な情報処理と結合している隣接ユニットとの相互の情報の伝達が可能となる。

3. 小規模システムの自己組み立て

本章では、与えられた目標形状の自己組み立てを行うための分散的な手法について述べる。1章でも述べた通り、個々のユニットが全体のどの部分でも受け持つことができるように、ハードウェアと同様にソフトウェア面についても均質性が満たされる必要がある。そこで、最初に基本事項であるユニットの結合型の分類と到達可能点について述べた後、分散・均質システムに適した目標形状の記述法を導入する。

3次元ユニットシステムでは、移動が2方向だけの2次元のフラクタムに比較して、到達可能点の数が接続の状態によって変化し、5個以上の可能性もある。このように自由度が大きいシステムにおいて、局所情報に基づく分散的な手法で自己組み立てを実現するため、確率的緩和法を導入することとした。

ここで, if[connection state]-then[motion] 式のルールによるヒューリスティックを用いることも可能である. しかし, すでに述べたように接続状態により変化する到達可能点をすべて列挙することは困難であり, また異なる目標形状ごとに別のヒューリスティックを用いなければならない. 本論文では, 単純で一般に適用できる手法を構築するため, 上記の手法を用いることにする.

3.1 結合型と到達可能点

ここでは, 本章で用いる結合型の記法と, ユニットの Fig. 2 の基本動作で移動できる位置である到達可能点について述べる.

3.1.1 結合型の分類

ユニットの中心点からなる3次元の格子点上の距離1の近傍でユニットの接続形態を考え, Fig. 3 のように分類した. それぞれの結合型は, 接続数 (Valence) と型 (Type) により表現する. 回転変換により重なる結合形態は同じ結合型とみなすことにより, 各接続数に対する型は2種類以下となり, 記法が単純になる. 型が2種類存在する接続数では, 便宜的に型を0と1に分類する. 例えば, “C30” は, 接続数3の0型を表し, 型が1個しかない場合には単に “C1” などとする. この結合型は, 自己組み立てで目標とする「目標形状」の記述に用いられる.

3.1.2 到達可能点

最も単純な C1 型の到達可能点を考える. 回転角を 90° とすると, 移動する C1 型ユニット (passenger) の到達可能点は, Fig. 4 に示すように最大で2点 (×印) がある. ユニットの隣接する運搬ユニット (carrier) の回転で移動するので, 格子点の1近傍でなく2近傍に移動する.

ただし, 到達可能点を数えあげるときには, 移動の軸となるユニット (support) が必要であることを考慮しなければならない. 別のユニットがすでに存在する点も到達可能点から除かれる. また, 移動時に他のユニットと衝突しない, ユニットの構成する構造体の接続を維持する, という制約も満

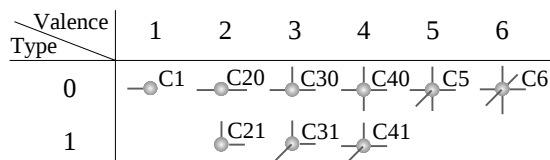


Fig. 3 Connection types for 3-D units.

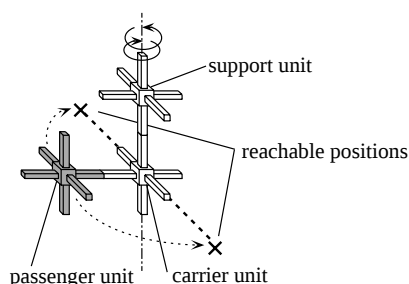


Fig. 4 Reachable positions of C1 type unit.

たさなければならない¹⁴⁾.

そこで, 移動することが許される結合型である「可動型」を定義する. ここでは, 上記の制約を考慮し, 簡単のため可動な結合型は C1, C21, C31 型とする. C21, C31 型ユニットの到達可能点は C1 型の到達可能点を重ね合わせ, 上記の衝突回避や接続性維持などの制約を考慮して求めることができる.

3.2 分散的自己組み立て手法

ここでは, 3.1 節で定義した記法を用いて目標形状を記述し, 局所情報を用いた確率的緩和法により自己組み立てを分散的に行う方法を述べる. 各ユニットには目標形状が与えられているとし, 以下の処理を繰り返す.

- (1) 自分が可動型かを調べ, 到達可能点を列挙する
- (2) 自分が到達可能点に移動したとした場合の結合型を計算して目標と比べ, 差 D を計算する.
- (3) D から計算したポテンシャルがより大きく減少する方向に, より大きな確率で移動する

これを, 自己組み立ての1制御ステップとし, 各ユニットは同期してこれらの処理を行う. この同期も分散的な手法によって実現することができる¹⁵⁾.

3.2.1 目標形状の記述

まず, 自分の型と接続する他のユニットの型を列挙する以下の「型リスト」を定義する.

$$\text{type} [\text{type}_1, \dots, \text{type}_n] \quad (\text{接続数の降順}) \quad (1)$$

ただし n : 接続数

type: ユニットの結合型

type_j: j 番目の隣接ユニットの結合型

目標とする形状は, この型リストの集合として表す. 例えば Fig. 5 (a) の8ユニットからなる立方体の記述は, 型リストを一つだけ含む以下の形となる.

$$\text{C31} [\text{C31} \quad \text{C31} \quad \text{C31}] \quad (2)$$

また, Fig. 5 (b) の目標形状は次のように記述される.

$$\begin{aligned} &\text{C31} [\text{C41}, \text{C31}, \text{C31}] \\ &\text{C41} [\text{C41}, \text{C41}, \text{C31}, \text{C31}] \end{aligned} \quad (3)$$

ただし, この記述方法の適用は, 記述に対して形状が一意に対応する場合に制限される. ユニットの多数のときには一つの記述に複数の形状が対応することが多くなるため, 4章の方法を用いることが適当である.

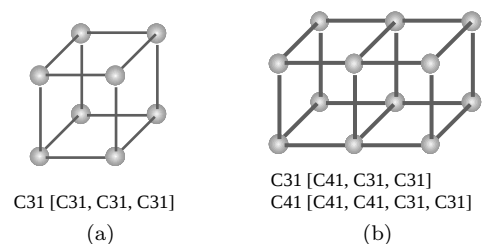


Fig. 5 Descriptions of target structure.

3.2.2 緩和的計算手法による目標形状の分散的生成

ユニットは、複数の到達可能点から、目標形状に近づく方向を近傍の情報をもとに選択する必要がある。

目標との差 D の計算

まず、各候補点 i (「移動しない」という候補を $i=0$ とし、て含む) に対して、目標と現状との差 D_i を求める。各ユニットの D_i は、式 (4) を用いて求めることができる。

$$D_i = \min_{k=1}^M [W \text{diff}(\text{type}^g(k), \text{type}^i) + \sum_{n \in \text{neighbors at point } i} \text{diff}(\text{type}_n^g(k), \text{type}_n^i)] \quad (4)$$

ただし

M : 目標形状中の型リストの数

type^i : ユニットが現在位置から到達可能点 i に移動した場合の結合型

type_n^i : ユニットが現在位置から到達可能点 i に移動した場合の n 番目の隣接ユニットの結合型

$\text{type}^g(k)$: 目標形状中の k 番目の型リストの結合型

$\text{type}_n^g(k)$: 目標形状中の k 番目の型リストの n 番目の隣接ユニットの結合型

また、 $\text{diff}(A, B)$ は型 A と型 B の差を表し

$$\text{diff}(A, B) = \begin{cases} 1 - \delta(A, B) & \text{for } n(A) = n(B) \\ a \times |n(A) - n(B)| & \text{for } n(A) \neq n(B) \end{cases} \quad (5)$$

とする。 $\delta(\text{type}_1, \text{type}_2)$ は Kronecker のデルタ関数、 $n(\text{type})$ は型 type の接続数を表す。式 (4) の $\min[\]$ 中の第 1 項はあるユニットの結合状態の現状と目標との差を表す。また第 2 項は、局所的通信から得られた情報を用いてその差を隣接ユニットについて同様に計算し、和をとったものである。 W は第 2 項に対する第 1 項の重みである。目標と現状の型リストの長さが異なる場合には、短い方に結合ゼロの “C0” を挿入して同じ長さとなるようにし、 $n(\text{C0}) = 0$ とし式 (5) を計算する。目標形状は通常複数の目標型を含むので、式 (4) では、目標型ごとに差を計算し、そのうちの最小の値を D_i としている。 $D_i=0$ となれば、候補点 i で目標形状に含まれる目標型の記述の一つが達成されたことを意味する。

確率的緩和法の導入

次に、候補点 i に移動する確率 P_i を、画像復元に用いられるランダム確率場 (MRF)¹⁶⁾ に基づき、 D_i をポテンシャルに用いて計算する。

$$P_i = \frac{1}{Z} \exp\left(-\frac{D_i}{T}\right) \quad (6)$$

$$\text{ただし } Z = \sum_{i \in \text{移動可能点}} \exp\left(-\frac{D_i}{T}\right)$$

T はアニーリング法における人工的な温度である。 T が大きいとき確率 P_i は一様であるが、 T が小さくなると最低のポテンシャルを持つ状態がほぼ確率 1 で実現される。 T は式 (7) にしたがって減少させる。

$$T = \frac{C}{\log(1+t)} \quad (7)$$

t は組み立て開始からのステップ数、 C は定数である。これにより、最初はユニットは乱雑に動いて探索空間を一様に探索し、時間が経過すると目標の形状により近い点に移動しやすくなる。以上の緩和的計算により、移動の自由度が大きくても目標形態に近づくように全体の構造が変化すると考えられる。

以上の計算は、局所情報を用いて行うことができる。具体的には (1) 到達可能点の列挙に 2 近傍 (2) 到達可能点での結合型の導出にこれに 1 を加えた 3 近傍の接続情報が必要である。これは、各ユニットがまず 1 近傍の接続情報を獲得し、次にこの情報を隣接ユニットと互いに交換することを 2 回繰り返すことにより得られ、計 3 回の通信で可能となる。

3.3 シミュレーションによる有効性の検証

以上で述べた手法を用い、与えられた初期状態から Fig. 5 (a), (b) に示す目標形状を構成するシミュレーションを行った。ここでは、 $W=3.0$, $a=2.0$, $C=10.0$ とした。簡単のため、1 ステップで移動するユニットは 1 個とする^(注 1)。

8 ユニットはジグザグ形から、12 ユニットは梯子形の初期状態から自己組み立てを開始する。これらは、自己組み立て順序が自明でない初期状態である。成功した自己組み立ての過程の例をそれぞれ Figs. 6, 7 に示す。これらの結果は、提案した手法により目標形状を分散的に探索して組み立てることが可能であることを示している。

自己組み立て手法の有効性を検証するため、500 回シミュレーションを行って、ユニット当たりの平均の目標との差 D を調べた。シミュレーションは同じ初期状態から開始されるが、乱数発生種を変えることにより、異なる確率的緩和過程が生起するようにする。これらの結果を、式 (6), (7) の MRF によらずにランダムに到達可能点を選択した場合と、つねに差 D の最急降下方向に移動する場合から得られた結果と比較する。

ステップ数に対する平均の D の値の変化を Figs. 8, 9 に示す。グラフには、提案した手法による結果 (proposed) を実線で、ランダム (random) と最急降下 (steepest) の場合を細かい点線と粗い点線の場合をそれぞれプロットした。初期値が二つのグラフで異なるのは、トポロジが異なる初期状態から開始しているからである。

これらのグラフから、目標との差の平均 D は、時間が経過するにしたがってほぼ 0 に近付いていることがわかる。これは、提案した手法により、ほとんどの場合目標形状に収束していることを示す。これに対して、ランダムな移動では D は大きな値 (> 10) をとり続け、与えられた目標がほとんど達成されていない。ランダム移動のとき、Fig. 9 では初期状態より D が増加する現象がみられる。これは、初期状態の梯子

(注 1) 単一ステップに複数のユニットを移動させる場合には、例えば一定範囲にリレー式で移動の意図を伝え、相互選択によりその範囲内で移動するユニットを 1 台選ぶことで可能となる¹⁷⁾。

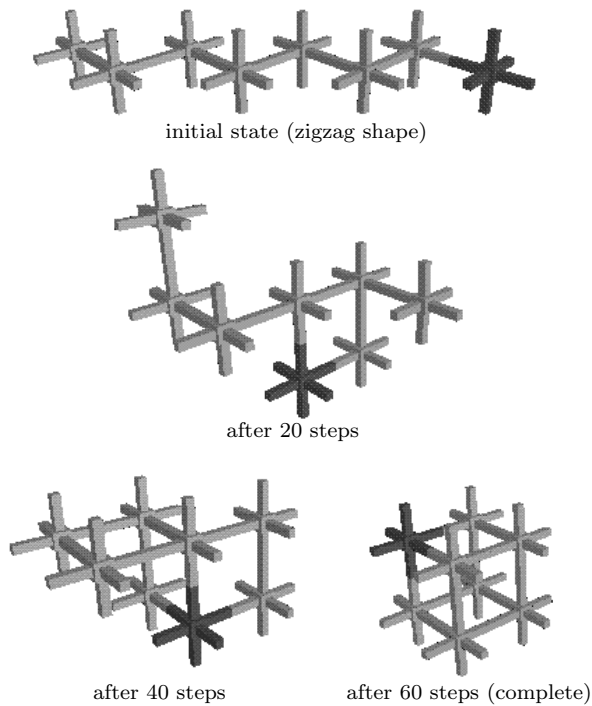


Fig. 6 Self-assembly simulation (8 units).

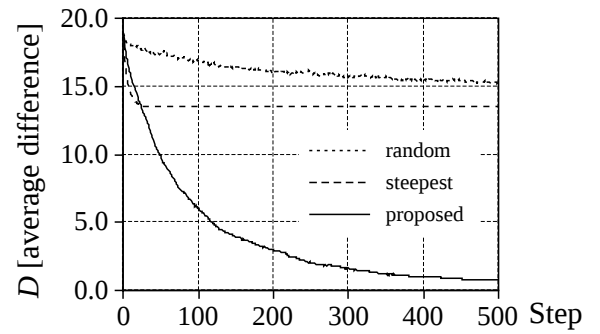


Fig. 8 Difference per unit (8 units).

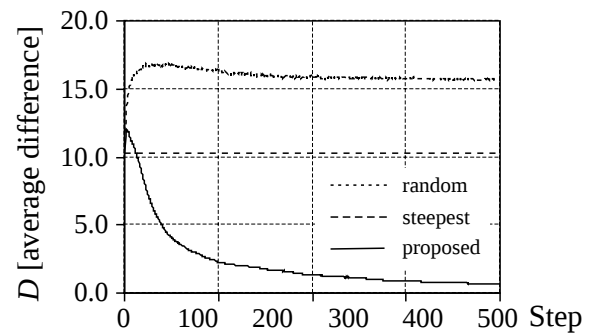


Fig. 9 Difference per unit (12 units).

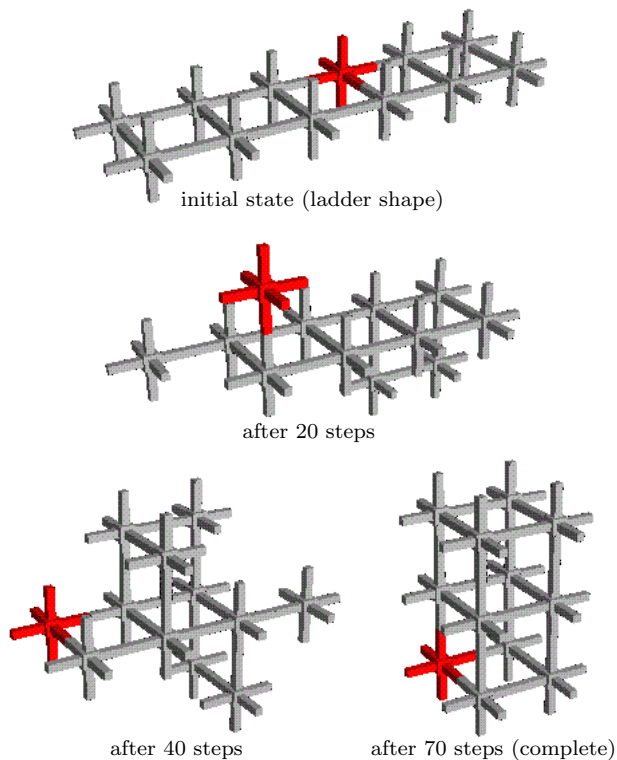


Fig. 7 Self-assembly simulation (12 units).

形がランダム移動により変化した構造より目標に近いからであると考えられる。最急降下方向の場合には、Fig. 8 では初期段階で D の値が高速に減少するが、すぐに大きな値に収束

してしまうことがわかる。また、Fig. 9 では初期状態に収束してしまっている。

これらの結果は、自由度が大きい3次元ユニットシステムにおいては、ランダムな移動や最急降下法では目標形状が達成されないことを意味する。また、提案した方法は、問題に依存してルールを変更しなければならないヒューリスティックとは異なり、目標の記述の変更によりさまざまな形状の自己組み立てが可能であるより一般的な手法であることも付け加えておく。以上により、提案した分散的自己組み立て手法が3次元形状の自己組み立てに有効であることが明らかにされた。

また、予備のユニットを加えた目標形状を与えることで、一部のユニットが故障したり消失したりした場合に自己修復を行うことが可能である。さらに、環境に応じて目標形状を変更することにより、動的に形状を適應させるように本手法を拡張することができる¹⁸⁾。

4. 大規模システムの自己組み立てと自己修復

本章では、大規模なシステムの分散的自己組み立てと自己修復を行うための手法について述べる。3章で示した方法は、単純という利点があり小規模なシステムでは効率よく機能する。しかし、システムを構成するユニットが20個を超えるような大規模なシステムでは、式(1)の目標形状を記述が大きくなる一方で、それらに対応する目標形状中のどの部分も対等に組み立てようとするため、自己組み立ての完了が困

難になる．

このような大規模な系に対しては，生物系の発生にも見られるように「段階的な組み立て」の導入が必要であると考えられる⁴⁾．そこで，簡単な基本構造を再帰的に繰り返して適用することによって対称性を持つ大規模な構造を単純に表現し，それを用いた分散的自己組み立て手法を提案する．

4.1 提案する手法の概要

本章で提案する手法は，階層的な形状の表現（4.2 節）と，それに基づく組み立て・修復手法（4.3, 4.4 節）からなる．Fig. 10 に，本章で述べる形状の記述法と自己組み立て手法の概念を示す．

4.2 節で説明するように，大規模な形状の表現は，階層的なグラフのノードに単純な基本構造を再帰的に当てはめて行くことにより行う．これは，3次元形状の幾何的な関係をグラフにより抽象的に記述するもので，4.2.2 項に示す対応づけにより，実際のユニットの位置が決定される．

4.3 節では，ユニットがメッセージ通信を行いながら，上記の表現手法により記述された目標形状の自己組み立てを階層グラフの上位から下位へ行う手法を述べる．具体的には，

（1）初期段階では，目標形状を記述する階層グラフの上位ノードに対応する部分を組み立て，大まかな形をつくる．ここで，ノードの組み立ては以下に行われる．

- ローカルに各ノードの組み立てをチェックするユニット（Local Coordinator Unit, LCU）を相互に選び，ノードの階層レベルや相対位置を含むメッセージを発生する．
- 各ユニットは，受け取ったメッセージをもとに，上位レベルのノードが優先して組み立てられるように移動する方向を決定する．
- LCU は，対応するユニットが集合すればノードの組み立ては完了したと見なす．

（2）ノードの組み立てが完了したら，LCU はその下位を展開して，メッセージを発生して近傍に伝え（1）と同様に組み立てを進める．

これを繰り返すことにより，下層の自己組み立てが局所的・並列分散的に行われることになる．

4.4 節では，この組み立て手法を拡張した自己修復手法に

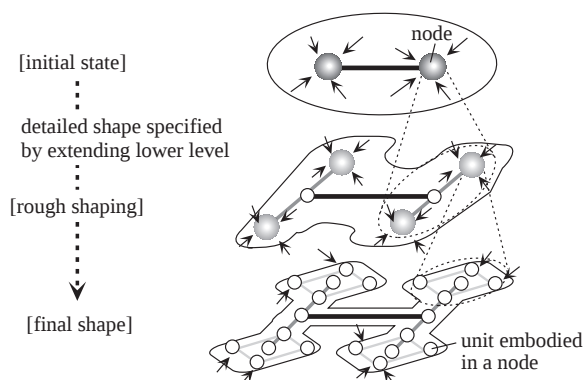


Fig. 10 Hierarchical graph structure for self-assembly.

ついて述べる．

4.2 対称性を利用した構造の表現

ここでは，まず階層的なグラフ表現を用いて，3次元形状の幾何的な構造を記述する手法を述べる．次に，このグラフ表現に対して，実際のユニット配置を対応づける方法について説明する．

Fig. 11 (a) と (b) は，本章で導入する記述法により表現された多数ユニットによる3次元形状の例で，それぞれ48個，56個のユニットからなる．

4.2.1 階層的グラフ表現による3次元形状の記述

3次元形状の幾何的な構造を表現するために，階層的なグラフ表現を導入する．グラフは，以下を用いて表現する．

- レベル L : 幾何的な構造における階層を表す．
- ノード N : 3次元格子点座標上の位置を表す．位置 Pos と特徴ベクトル $FVect$ により表現される．
- 基本構造 S : ノード間の幾何的な位置関係を示すプリミティブ．基本構造は，Fig. 12 に示す4種類とする．

ある階層レベルの基本構造とそれを構成するノードをそれぞれ S_L, N_L と表す．

3次元形状は Fig. 13 の形で記述する．グラフ表現は，この記述を最上位から順にレベル 1, 2, ... とした階層関係を持つ基本構造を考え，これを再帰的に展開することで得られる．すなわち，上位レベルの基本構造に含まれるノードに下位の

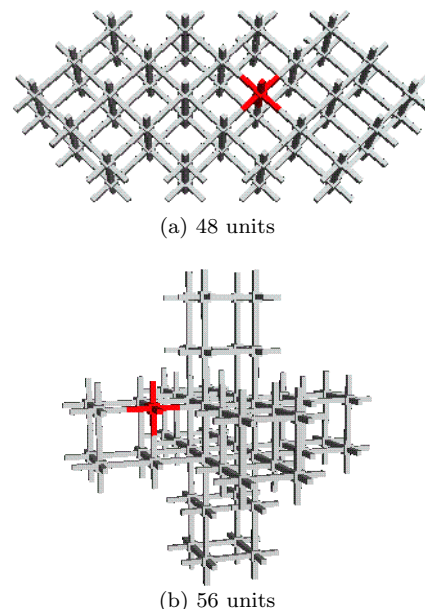


Fig. 11 Example of large structure.

Type	line	square	octahedron	cube
FVect of composing nodes:	direction vector	normal vector	vector from center	same as FVect of upper node

Fig. 12 Primitive types of description.

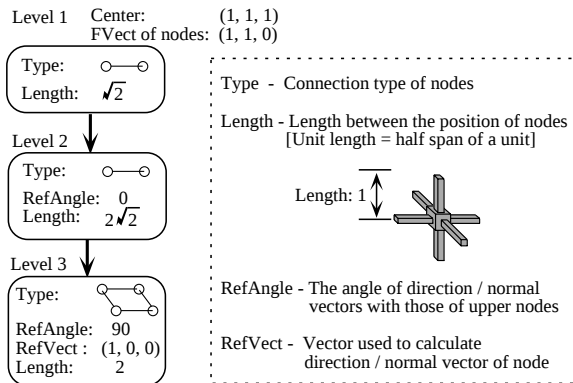


Fig. 13 Description of structure in Fig. 11(a).

基本構造を対応させることにより、最終的に全てのノードを決定することができる。このグラフ表現に対して、後述する実際のユニットの配置の対応づけを行うことによりユニットによる 3 次元形状が決められる。

具体的に階層を展開してグラフ表現を生成する方法を次に説明する。まず、レベル L の基本構造 S_L は上位レベルに対して相対的な位置関係を決定するため、次の属性を持つ。

Type_L: 型 (line, square, octahedron, cube)

Length_L: S_L に属するノード間の距離

RefAngle_L (0° または 90°) , **RefVect_L**:

上位レベルのノード N_{L-1} の特徴ベクトル FVect_{L-1} から S_L を構成するノードの FVect_L を決定するために用いる相対角度と参照ベクトル

これを用い、階層レベル L の基本構造 S_L に属する各ノード N_L から、次のレベル $L+1$ の S_{L+1} , N_{L+1} への展開は次のように行う。

- (1) N_L の位置 Pos_L に S_{L+1} の対称中心を対応させ、 S_{L+1} の Type_{L+1} と Length_{L+1} から S_{L+1} に属するそれぞれのノード N_{L+1} の Pos_{L+1} を決定する。
- (2) N_L の特徴ベクトル FVect_L から、 S_{L+1} の Type_{L+1} に応じて、 S_{L+1} に属するそれぞれのノード N_{L+1} の FVect_{L+1} を求める。

line, square のとき

N_{L+1} の FVect_{L+1} は方向ベクトル d (line の場合) または法線ベクトル n (square の場合) とする。

- $\text{RefAngle}_{L+1} = 0^\circ$ のとき:

$$d \text{ (or } n) = \text{FVect}_L.$$

- $\text{RefAngle}_{L+1} = 90^\circ$ のとき:

$$d \text{ (or } n) = \text{FVect}_L \text{ と } \text{RefVect}_{L+1} \text{ の外積}$$

octahedron のとき

$$\text{FVect}_{L+1} = S_{L+1} \text{ の対称中心からのベクトル}$$

cube のとき

$$\text{FVect}_{L+1} = \text{FVect}_L$$

以上で述べたグラフ表現により、最上位レベルの基本構造 S_1 の対称中心位置とノードの特徴ベクトル (Fig. 13 の Center と FVect of Nodes) を与えれば、順次それらを展開してい

て階層的なグラフ表現を得ることができる。

これを図解したのが Fig. 14 であり、Fig. 11 (a) 構造を示す Fig. 13 を再帰的に展開したものである。

4.2.2 グラフ表現と実際のユニット配置の対応づけ

前項で述べた階層型のグラフは、3 次元形状の幾何的な空間構造を抽象的に表現したものである。これをもとに、実際のユニットを対応づけて配置する規則についてここで述べる。

ここでは、グラフ表現の記述能力を向上させるため、Fig. 13 に示すように、3 次元格子点座標の単位長さをユニットの半スパンの長さとし、実際のユニットの座標は偶数値をとることとする。

グラフ表現を実際のユニットの位置に対応させるとき、あるノード N の位置 Pos の座標成分が偶数ならばその位置にユニットを 1 個配置する。奇数の場合には、最近接の 2 つの偶数座標にユニットを配置することとする。したがって、ノード N の座標 (x, y, z) のうち、奇数成分が

- 0 個の場合: ノード位置 Pos にそのままユニットを配置
- 1 個の場合: ユニットの線分の両端に 2 個配置
- 2 個の場合: ユニットの正方形の頂点上に 4 個配置
- 3 個の場合: ユニットの立方体の頂点上に 8 個配置

することになる。ただしここでは、ユニットは複数の階層のノードに属することが可能であるとする。Fig. 13 の記述を展開したグラフ表現に対し、上記の配置規則によりユニットを対応させたのが Fig. 14 の最下部である。対称性を利用しているためデータ構造自体は単純であるが、階層構造による繰り返しを用いて、多数ユニットからなる平板や箱型構造はもちろん、Fig. 11 に示すような対称性を持つある程度複雑な構造にも対応できる。

4.3 自己組み立ての手法

ここでは、前節の記述を利用した自己組み立てについて述

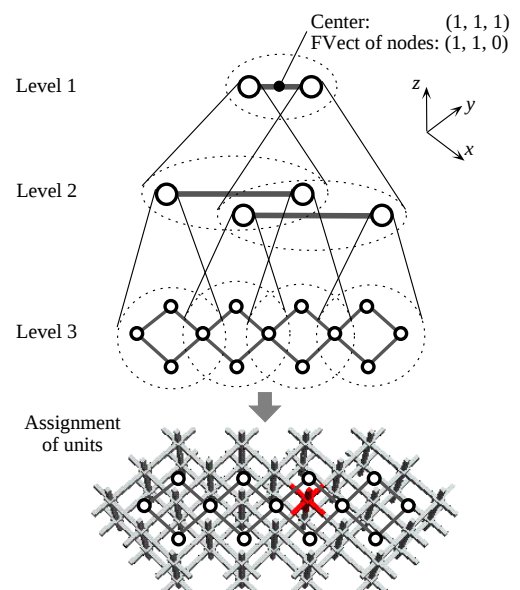


Fig. 14 Recursive description of structure in Fig. 11(a).

べる．大規模な構造の自己組み立ては，隣接ユニットの情報だけでは困難であるが，逆に多数のユニットが大域的に通信すると通信負荷が増大する．そこで，各ユニットに目標形状の記述を与えておき，4.1 節で示したように，階層グラフの上位から下位へ形状を組み立てていく手順をとる．組み立ての開始時には，最上位ノードの対称中心位置の座標を決定する必要がある．これは，例えば分散的に実現できる相互選択手法¹⁹⁾を用いてユニットの集合体から1台を選ぶことによって可能である．この自己組み立てを，1章でも述べた通り，ソフトウェアの均質性が保たれるように実現した．

ユニット間で通信されるメッセージ単位はノードに関して以下の内容を持つ．

- V ノードの位置 Pos までの相対座標
 - $level$ ノードの階層レベル L
 - $nflag$ ノードに対するユニット配置完了 (T)/未完了 (F)
 - LCU_ID ノードの局所組み立て担当ユニット (LCU) の ID
未定のときは NO とする．
- また，各ユニットが保持する情報は
- ID 一意な識別番号
 - $uflag$ ノードに組み込まれたか (T) / 否か (F)
ノードに配置されるユニットに組み込まれれば，移動しない．
 - m_{list} 配置未完了ノードに関するメッセージのリスト
 - c_{list} 配置完了ノードに関するメッセージのリスト
- とする．ユニット U が行う情報処理の1ステップの詳細は以下の通りである．

(1) メッセージの受信と内容の更新： 受け取ったメッセージ m について V を更新し，

- m が c_{list} になければ
 - m の $nflag = F$ で m_{list} になければ m を追加，
 - m の $nflag = T$ で m_{list} にあれば削除して m を c_{list} に追加
- m が c_{list} にあり， m の $nflag = F$ ならば $nflag = T$ に更新する

上記のいずれも次のステップで隣接ユニットに m を伝達

(2) 移動方向の決定： m_{list} 中の m について

- U の $uflag = F$ のとき
 - 評価値 $E = p \times level + q \times V$ のマンハッタン距離を計算する．
 - E の小さい R 個からランダムに一つを選び， V に近づく方向に移動

(3) ノードへの組み込みの判断： 各ユニットが移動した後， m_{list} 中の m について

- U が m の相対ベクトル V が示すノードに対応する位置に到達：
 - U の $uflag = F$ ならば
→ ノードの配置に組み込まれたので， $uflag = T$ ．
- m の $LCU_ID = NO$ ならば，

○ ノードを構成した最初のユニットが LCU となり， $LCU_ID = ID$ とする．

LCU の可能性が複数ある場合には，乱数などを用いた相互選択を行って一つに決める．

○ 次のステップで m を隣接ユニットに伝達

(4) LCU によるユニット配置完了の判断：

- U の $uflag = T$ のとき
 - U が m_{list} の m の LCU の場合：
 - ★ 局所通信により， m のノードの配置完了を確認
→ 完了していれば，
 - ★ m の $nflag = T$ とする
 - ★ m の下位の基本構造を展開し，それぞれのノードに対して $nflag = F$ ， $LCU_ID = NO$ のメッセージ m' を生成

次のステップで m ， m' を隣接ユニットに伝達 ■

前章と同様，各ユニットの移動と情報処理は同期して行われるとする．またステップ (2) で，リスト中の評価値の小さい上位 R 個からランダムに一つのメッセージ m を選択するのは，ユニットの動きにある程度のランダム性を持たせ，デッドロックを防ぐためである．

提案した手法の有効性を確認するため，60 個のユニットが直方体状の初期状態から，Fig. 11 (b) の 56 ユニットからなる形状の自己組み立てのシミュレーションを行った．ここでは Fig. 3 の結合型では C6 型以外のユニットは可動とし，ステップ (2) の $p = 2$ ， $q = 1$ ， $R = 5$ とした．Fig. 15 は，組み立てが成功したシミュレーションの一例で，濃い色は組み立てが完了した部分を示す．同時に複数のユニットの移動を許すことも可能であるが，単純のためシミュレーションの1ステップではユニットは1個だけ移動するとしてステップ数を示した．階層表現を利用した自己組み立て手法により，目標形状が完成している．目標形状に含まれないユニットはスベアである．

ここでは，組み立ての開始点となる最上位ノードの対称中心として，異なる位置を Fig. 15 の初期形状の内部で与えて計 100 回のシミュレーションを行い，自己組み立ての成功率を求めた．その結果，1000 ステップまでの成功率は 98% であり，組み立てを開始する位置にかかわらず，提案手法が有効であることが示された．組み立てが成功しなかったのは，多くのユニットからなるループ状構造を生成した場合で，均質なユニットではこれを解消できないためデッドロックとなる．これに対しては，ループとして残りやすい結合型を局所的に避けるなどの対策が必要と考えられる．

このように，この手法は大規模システムに関して有効であるが，多くの計算とメッセージ交換のための通信を必要とする．小規模システムに対しては計算・通信のコストの面から，単純な 3 章の方法が有利であることを付け加えておく．

4.4 自己修復の手法

前節で示した分散的な自己組み立て手法の利点は，システムの一部が故障したときに機能を回復する「自己修復」を実

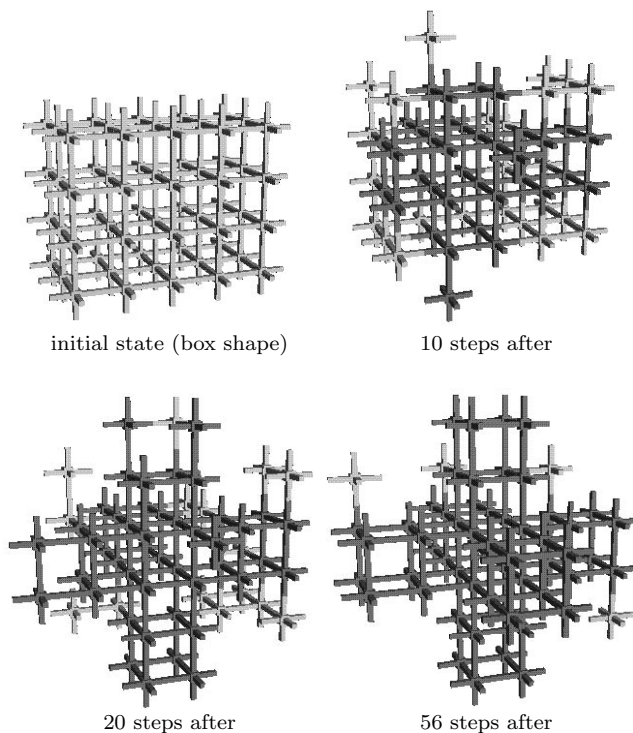


Fig. 15 Self-assembly of many-unit structure.

現することによってさらに有効性を発揮する。本章では、3次元ユニットによる大規模システムの自己修復手法について述べる。

前節で述べた手法では、並列分散的に自己組み立てが進むので、自己修復への拡張は比較的容易である。ノードを構成するユニットどうしは、局所的な信号の伝達を通じて、正常に動作しているかを一定時間ごとに確認することが可能である。ここでは、信号を送っても返答が無い場合には、そのユニットが故障あるいは欠落したとみなす。故障が発生した場合には、前章の手法に以下の処理を付加すれば自己修復を行うことが可能となる。

メッセージには、以下の内容を加える。

- $rtimes$ ノードの修復回数 [初期値:0]

ユニット U について、ノードを構成するユニットの故障を発見したとき:

(1) 故障時の情報生成

- U の $uflag = T$ のとき,
 - c_{list} からノード情報 m を削除し、内容を変更:
 - $rtimes$ インクリメント
 - $nflag = F$ とする
 - $LCU_ID = U$ の ID [U が新たな LCU のとき]

U の $uflag = F$ とし、変更した m を m_{list} に加え、隣接ユニットに伝達

(2) メッセージの処理: 伝達されたメッセージ m について

- U の $m' \in (m_{list}, c_{list})$ と m の $level$, V が同じとき:
 - m の $rtimes > m'$ の $rtimes$ のとき:

★ m' を c_{list} , m_{list} から削除, m で置換

★ 変更した m を隣接ユニットに伝達

- 他の処理は前章と同じ

以上を加えることにより、システムを構成するユニットの一部が故障しても、スペアのユニットを用いて目標の構造を回復することができる。

Fig. 16 は、Fig. 15 で完成した構造の一部 (4 ユニット) を削除したときのシミュレーションである。全体のユニット数、目標の構造に必要なユニット数はそれぞれ 60, 56 であるから、自己修復にはスペアである 4 ユニットの全てを用いなければならない。シミュレーションの結果を見ると、スペアユニットを用いて構造の回復が行われていることがわかり、本章で述べた自己修復機能の有効性が確認された。ただし、2章で述べたように、ユニットは3次元「チェッカーボード」上の2つの系列のうちのどちらかしか移動できないので、同じ系列のユニットが不足した場合には完全には修復できない。6本の腕全てが接続されたユニットが故障した場合の修復は、それを排除する操作が必要となり、現在に対応できない。今後、手法を拡張してこの場合の修復も行えるようにすることが必要である。

5. おわりに

本論文では、均質な3次元の機械ユニットを用いて分散的に自己組み立て・自己修復をする方法について述べた。3次元構造を形成可能な均質なユニットの設計について述べた後、まず小規模なシステムの自己組み立て手法を構築した。この手法は、組み立ての目標とする形状の簡便な記述法と局所的な情報を利用した確率的緩和法により構成された。計算機シ

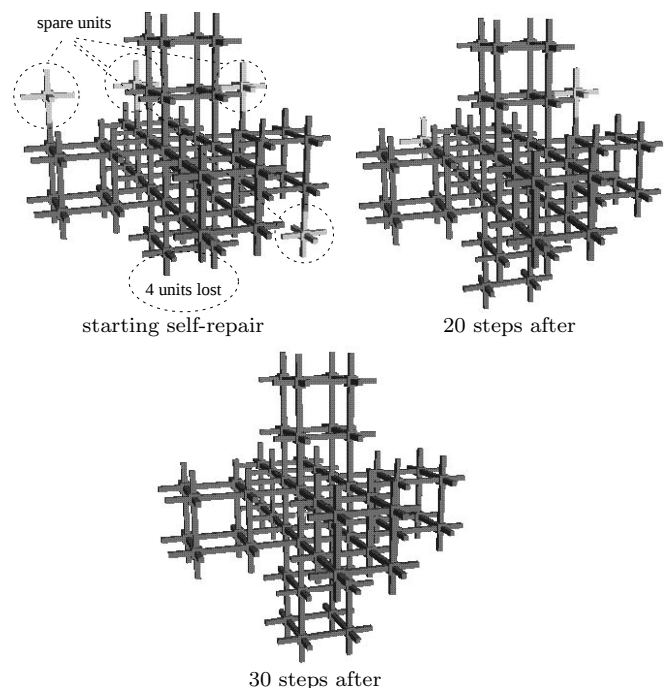


Fig. 16 Self-repair of many-unit structure.

ミュレーションにより、各種の3次元形状の自己組み立てが可能であることを示し、本手法の有効性を明らかにした。

次に、大規模なシステムの分散的な自己組み立て・自己修復の方法を述べた。単純な基本構造を再帰的に適用した大規模構造の表現を用い、段階的に組み立てが進めることで、大規模な3次元形状の自己組み立て・自己修復が可能であることを示した。

以上の手法ではハードウェアと同様ソフトウェアの均質性も満たされている。また、さまざまな目標形状に対応できるので、自律分散型機械システムによる3次元構造の自己組み立て・自己修復手法の構築に関する一般的な指針を示すことができた。今後、これらの手法を実現するためのハードウェアの制御手法も開発していく予定である。

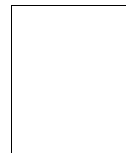
参 考 文 献

- 1) 計測自動制御学会, 特集: 自律分散システム, 計測と制御, **29**-10 (1990).
- 2) 計測自動制御学会, 特集: 自律分散システムの新たな展開, 計測と制御, **32**-10 (1993).
- 3) 村田, 黒河, 小鍛冶: 自己修復する機械 - 分散機械ユニットによる自己組立 -, 計測自動制御学会論文集, **31**-2, 254/262 (1995).
- 4) 村田, 富田, 黒河, 小鍛冶: 機械の自己組立てと自己修復, 計測自動制御学会論文集, **33**-5, 424/432 (1997).
- 5) 村田, 黒河, 吉田, 富田, 小鍛冶, 分散機械システムの設計 (第6報) - 3次元自在結合システム -, 第9回自律分散システム・シンポジウム資料, 21/24 (1997).
- 6) 例えば, 福田, 中川: 動的再構成可能ロボットシステムに関する研究 (第1報, セル間の自動接近・結合・分離制御), 日本機械学会論文集 C 編, **55**-509, 114/118 (1989).
- 7) G. Chirikjian, A. Pamecha, and I. Ebert-Uphoff, Evaluating Efficiency of Self-Reconfiguration in a Class of Modular Robots, J. of Robotic Systems, **12**-5, 317/338 (1996).
- 8) M. Yim, New Locomotion Gaits, Proc. IEEE Int. Conf. on Robotics and Automation, 2508/2514 (1994).
- 9) G. Hamlin and A. Sanderson, Tetrobot: A Modular Approach to Reconfigurable Parallel Robotics, Kluwer Academic Publishers (1998).
- 10) K. Kotay and D. Rus, Motion Synthesis for the Self-reconfiguring Molecule, Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS'98), 843/851 (1998).
- 11) 伊能, 上杉, 工学構造物を自己組織化するセル・オートマトン (さまざまな位相構造の生成とその形態比較), 日本機械学会論文集 C 編, **61**-585, 241/246 (1995).
- 12) G. Theraulaz and E. Bonabeau, Coordination in Distributed Building, Science, **269**, 686/688 (1995).
- 13) 畠山, 白, 可動有限オートマトンモデルによる T4 バクテリオファージの尾部の自己組織化, 計測自動制御学会論文集, **34**-6, 614/620 (1998).
- 14) H. Kurokawa, S. Mutata, E. Yoshida, K. Tomita and S. Kokaji, A 3-D Self-reconfigurable Structure and Experiments, Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS'98), 860/865 (1998).
- 15) S. Kokaji, S. Mutata and H. Kurokawa: Clock Synchronization algorithm for a Distributed Autonomous System, J. of Robotics and Mechatronics, **8**-5, 427/434 (1996).
- 16) 川入, 脳の計算理論, 322/341, 産業図書 (1996).
- 17) K. Tomita, S. Murata, E. Yoshida, H. Kurokawa and S. Kokaji, Reconfiguration Method for a Distributed Mechanical System, Distributed Autonomous Robotic System 2, H. Asama, et al., eds., 17/25, Springer (1996).

- 18) E. Yoshida, S. Mutata, H. Kurokawa, K. Tomita and S. Kokaji, A Distributed Reconfiguration Method for 3-D Homogeneous Structure, Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS'98), 852/859 (1998).
- 19) N. Lynch, Distributed Algorithms, 51/57, Morgan Kaufmann Publishers, (1996).

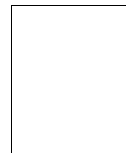
[著 者 紹 介]

吉 田 英 一 (正会員)



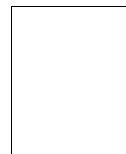
1996年東京大学大学院工学系研究科博士課程修了, 同年通産省工業技術院機械技術研究所入所。博士(工学)。1990~91年スイスローザンヌ工科大学留学。現在, 自律分散型の機械システムの設計や制御の研究に従事。日本機械学会, 日本ロボット学会, IEEEなどの会員。

村 田 智 (正会員)



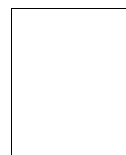
1987年名古屋大学大学院工学系研究科修士課程修了, 同年通産省工業技術院機械技術研究所入所。博士(工学)。1998~99年米国 Johns Hopkins 大学客員研究員。分散システムの研究に従事。92年 IEEE-IE 論文賞, 96年本学会論文賞受賞。日本機械学会, 日本ロボット学会, 電気学会などの会員。

小鍛冶 繁 (正会員)



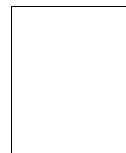
1972年東京大学大学院工学系研究科修士課程修了, 同年通産省工業技術院機械技術研究所入所。機械システムの分散制御を研究(工学博士)。精密工学会, 情報処理学会, 日本ロボット学会などの会員。

富 田 康 治 (正会員)



1990年筑波大学大学院理工学研究科修士課程修了, 同年通産省工業技術院機械技術研究所入所し現在に至る。博士(工学)。物理情報部知識工学研究室においてマシンインテリジェンスの研究に従事。ACM, IEEE-CS, 情報処理学会などの会員。

黒 河 治 久 (正会員)



1981年東京大学大学院工学系研究科修士課程修了, 同年通産省工業技術院機械技術研究所入所。宇宙における機構の制御および自律分散システムの研究に従事(工学博士)。精密工学会, 日本機械学会, 日本航空宇宙学会, AIAAの会員。