Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

A Hierarchical Motion Planning Strategy for a Uniform Self-Reconfigurable Modular Robotic System

Konstantine C. Prevas[1], Cem Ünsal[2], Mehmet Önder Efe[3] and Pradeep K. Khosla[4]

[1]Human Computer Interaction Institute
[2]Institute for Complex Engineered Systems
[3,4]Electrical and Computer Engineering Department
[1,2,3,4]Carnegie Mellon University, Pittsburgh, PA, 15213-3890, U.S.A.
kprevas@andrew.cmu.edu, unsal@ices.cmu.edu, efemond@andrew.cmu.edu, pkk@ece.cmu.edu

Abstract

This paper describes a multi-layered hierarchical motion planning strategy for a class of self-reconfigurable modular robotic systems, I-Cubes. The approach is based on the synthesis of motion on the basis of metacubes, which have a particular structure possessing 8 Cubes and 16 Links. The developed strategy organizes the metacube motions and the corresponding cube-level motions. At the lowest level, link motions are generated. The resulting system is demonstrated to be capable of performing a pre-specified task of moving from one position/shape to another. The paper describes the latest results of our planning strategy through some experimentally justified examples.

1. Introduction

The concept of building robots which are capable of changing their structure according to the needs of the prescribed task and the conditions of the environment has been inspired from the idea of forming topologically different objects with a single and massively interconnected system. The problems related to the design and implementation of such systems are analyzed in the field of self-reconfigurable robotics. Various applications analyzing the capabilities of self-reconfigurable robotic systems are reported in the literature. Early studies on this topic report manual configuration [1], a modularly synthesized kinematics structure called *Tetrabot* [2], and cellular mobile robots with reconfiguration capability [3]. Later studies have reported examples in 2D such as *metamorphing hexagonal modules* [4], *self-repairing machines* [5], the *Cristalline robot* [6], which moves in a horizontal plane, and *Inchworm* [7], which moves in a vertical plane. The examples operating in 3D are *Polypod/Polybot* [8], *CONRO* [9], *robotic molecule* [10], self-reconfigurable structure [11], modular robot [12], *Proteo* [13] and *I-Cubes* [14]. A common property of what is presented in [8-14] is the capability of exploiting neighboring modules to fulfill the task.

In the literature, several approaches have been proposed to discover the associations between the task described in the operational space and the corresponding actions to be carried out simultaneously in the cell level. Planning of motion has exploited the strength of simulated annealing [15], distributed approaches [13,16-17], closed-chain reconfiguration [18] and multi-layered solvers [10,12,19].

Although it is possible to treat the I-Cube entity as a closed-chain or non-homogeneous lattice-based reconfigurable robot, we limit our discussion to a uniform structure described in Section 3. This structure provides a significantly simpler planning strategy.

The following section briefly introduces the I-Cube system, the third section illustrates the group of modules composing I-Cube systems, the fourth section presents the multi-layered planner, and the concluding remarks are given in the final section.

2. I-Cube System

The I-Cube system is composed of 3-DOF Links enabling the self-reconfiguration, and Cubes determining the resolution of the object realization by constituting the elements of a lattice. A simple object with 3 Cubes and 2 Links (3C2L) is shown in Figure 1. The motors A and C provide 360º and the motor B provides 270º of rotational motion. The motors are independently controlled thereby enabling a high degree of mobility in terms of planning issues.

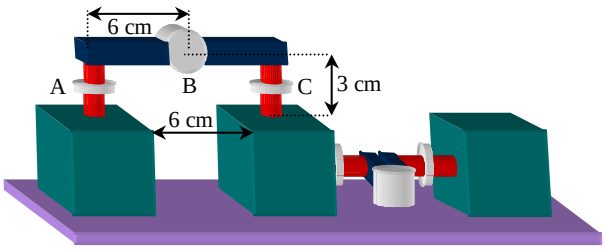

Figure 1. Physical view of a 3C2L I-Cube system

In the physical implementation stage of the I-Cube links, each link is actuated by a Cirrus CS 21-BB Hi Performance Sub-Micro BB Servo. The controller for each servo is a PID controller realized on a dedicated PIC (16C63A) controller. Feedback signals are obtained through the use of 24-hole optical encoders and the control system is operated at 160 msec of sampling rate. The powering of the system is achieved through the batteries placed inside the cubes and the presented design is proven to be able to carry its own mass, i.e. 195gr per link and 205gr per cube. The latching mechanism designed for the attachment and detachment is shown in Figure 2. When an attachment is performed, the

communication between the cube GUI and I-Cube system is maintained through a USART serial bus adapter.

Figure 2. Physical view of connection mechanism (a) on the cube faces and (b) at the link ends for cable twist prevention

The proof of concept of self-reconfiguration is shown in Figure 3, in which several intermediate steps of the tower construction task are shown.

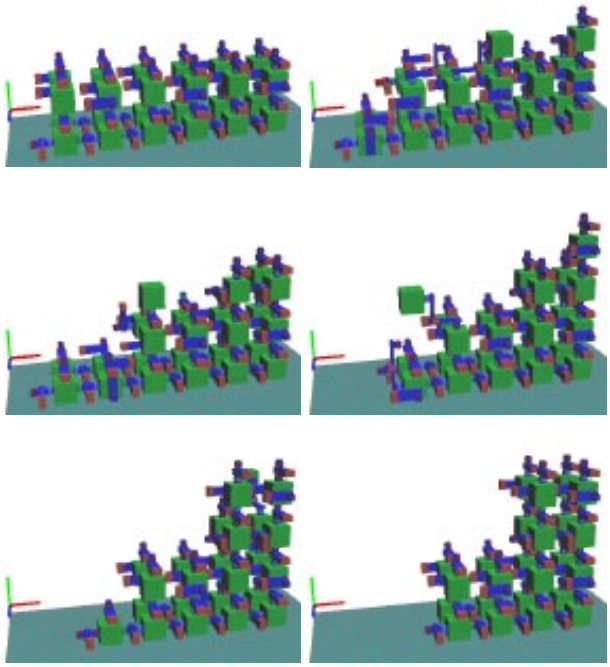

Figure 3. 24C48L group (i.e., 3 metacubes) forming a step based on metacube moves

3. Homogeneous Groups of I-Cube Modules

Our earlier attempts to find a solution for relatively smaller groups of modules also used the substrate definition, and assumed that the cubes would start and stop at the cells fitting a cubic lattice, although it is possible to treat a group of cubes and link as a closed-chain modular system. However, our earlier attempts did not pose any constraints on the structure of the system other than the one mentioned above. In [14], we defined a uniform structure of modules, which will simplify the path planning process at the higher level where the cube motions are considered, and at the lower level where sequences of link motions are defined for desired steps of cube moves.

This uniform structure enables us to replace the heuristic search methods of [15,19-20] with direct link motion evaluations for cube paths over the entity. This new approach also provides a solution for combining individual link/cube motions into simultaneous actions.

Even for homogeneous modular systems with simplified module design (i.e. *Proteo* [13], and MEL robot [21]), the motion/path planning is known to be a very complex problem due to the fact that a module forced to a specific location can block remaining modules from reaching their destinations.

The uniform structure we introduced in [14] has been changed slightly to take advantage of better link positioning with respect to cubes. The metacubes are still 8C16L groups as shown in Figures 4 and 5.

All the links in a metacube are in the completely open position. Each cube is associated with two links: one on the top (y+) and one on the side. As seen in Figure 4, a metacube has two connections on each vertical face, and four connections on the horizontal faces. These groups, when put together, result in a uniform structure with links between vertical layers of cubes (one link on top of each cube), and a link between every second pair of cubes aligned with the *x*- and *z*-axes (Figures 4, 5 and 6).

This definition of metacube provides at most four links per cube (top, bottom, and two on the side), and guarantees that any cube that needs to be moved to relocate the metacube will have at most four links, of which two can be easily transferred to another cube in the same metacube or an adjoining one. Due to the definition of link positions in the metacube, this will leave the cube with two links attached to non-opposing faces of the cube. This enables the planner to move the cube without breaking the geometric constraints of the modules.

By defining metacubes, we increase the size of the units that make up the I-Cube entity, i.e., we increase the 'group resolution.' These metamodules will be the 'tiles' of a 3D structure that will be relocated from one position to another to reconfigure the I-Cube entity. However, for tasks requiring better modular resolution, it is still possible to use other approaches given in [19,22] to find solutions for single cube displacement after moving a sufficient number of metacubes into desired positions.

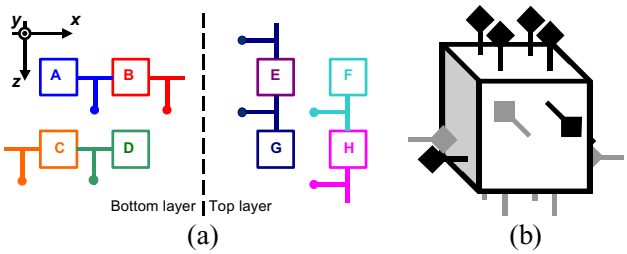

Figure 4. Illustration of (a) cube and link positions (top links not shown) in a metacube, and (b) connections to neighboring metamodules (gray connections belong to neighboring modules).

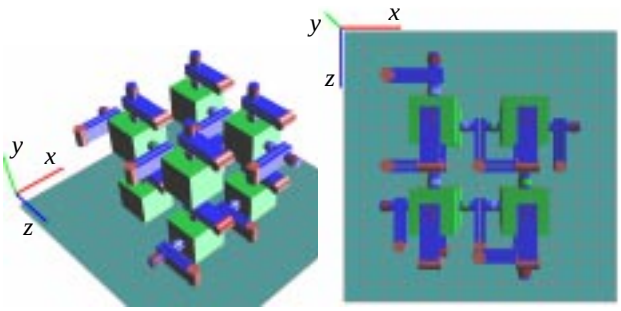

Figure 5. Different views of a metacube (8C16L) where the links at each layer are aligned along the same axis.

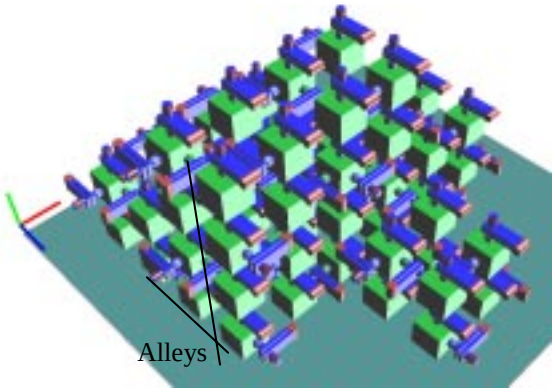

Figure 6. A network of I-Cubes with overlapping metacubes

Note that the configuration given here for the metamodules is one of many possible structures. We have decided to keep all the links attached to the top faces of the cubes as more link placements on the horizontal layer provides a more robust structure; however, for space applications, this constraint can be relaxed.

4. Multi-Layered Planner

The reconfiguration task is analyzed in a layered fashion as shown below to come up with a hierarchically organized solution. The following subsections discuss these levels in detail.

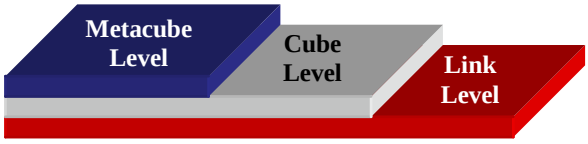

Figure 7. Hierarchical levels of motion planning

4.1. Metacube Level: High-Level Solvers

When a group of I-Cube modules, each of which has the uniform structure shown in Figures 4 and 5 having *n* cubes and *2n* links, is to reconfigure itself to fulfill a prescribed task, the highest level of planning synthesizes the displacements for each metacube. First results on this issue have been presented in [14], which describe two solvers named SolverIC and SolverICF, which are explicitly defined in [22]. The underlying idea is to perform a mapping between metamodules of the initial configuration and those of the final one. In realizing the mapping, the cost of each operation is defined to be the Manhattan distance between initial and final positions. The prime objective is to keep the cost as low as possible while handling the application-specific difficulties such as metamodule motion precedence, overlappings, and modules acting as obstacles for the others.

The primary difference between Solver IC and Solver ICF is that the latter gives priority to the metamodules that are closer to their destination locations. Furthermore, the decision flow accounts for the consecutive moves and can perform alterations if the action to be performed requires the removal of the first.

The latest version of the developed solver is called SolverM, whose initial version is already presented in [14]. The current version is based on the instant updates of the entity's perimeter, which has experimentally been shown to be successful in reducing the time required to find the solution and the number of link motions required for fulfilling the task.

At this level, the planner also generates a data structure representing the entity's current perimeter. The structure used is a hash table, which stores the position of each cube as well as the position of each cell directly bordering a cube. Generation of this structure involves storing the initial position of each cube, and adding all bordering positions as border cells. Since each cube borders at most six cells, the computational complexity of generating the perimeter is linear in the number of cubes.

4.2. Cube Level: Generating Paths for a Single Cube

The path planning problem at the cube level involves the development of strategies for resolving and reconstructing metacubes at the level of individual cubes. For this purpose, we considered the entire set of possible cases and their attachment and detachment strategies. There are 17 such cases based on the positions of a metacube that is capable of moving to another location (i.e., is on the 'edge' of the group).

In Figure 8(a), an exemplar case is illustrated. The metacube of interest is the one drawn as a wire frame, and the solid ones are its neighbors. The numeric notation for this case is depicted in Figure 8(b). It should be clear that if x+=x-=1, the metacube cannot move, or equivalently, a metacube cannot be constructed in such an empty location. The same constraint is also valid for y and z directions.

Furthermore, there is a second set of cases in which y+=1. These are excluded, but for possible space applications, a corresponding attachment and detachment strategy needs to be developed. Having the structure illustrated in Figure 4 in mind, the detachment and attachment graphs for case 38 (100110) are depicted in Figure 8(c) and (d). When the graphs characterizing the attachment-detachment sequences for all cases that are likely to occur are formed, a static codebook describing the best sequences is prepared, and in the middle level of planning, the cube motions are synthesized according to what is prescribed by the codebook. Given the current position and destination for a moving cube,

the optimal path for that cube is found using a heuristic search algorithm similar to A*, where the heuristic used is the Manhattan distance to the cube's destination.

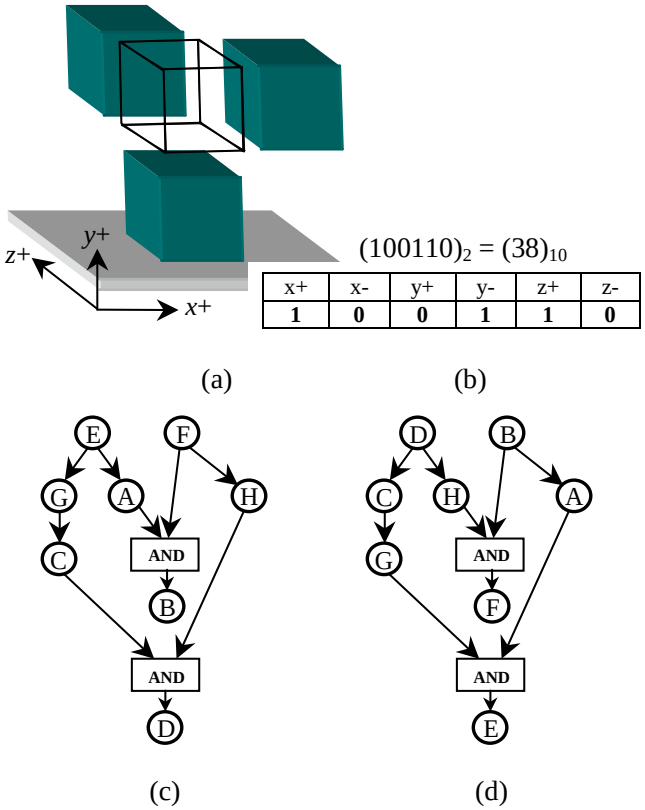

Figure 8. (a) Case 38 (100110) and (b) its representation. The detachment (c) and the attachment (d) strategies for case 38. AND operator determines the attachment/detachment condition depending upon the movement of multiple cubes.

This algorithm uses the perimeter structure generated by the metacube level planner. The moving cube is removed from its original position and added at its destination position. Removing a cube from this representation involves examining each of the six bordering cells. Any cells which no longer border a cube are removed, and the position of the cube itself is marked as a border cell. Similarly, adding a cube involves examining all neighboring cells and changing those which did not previously border a cube to border cells.

The procedure of adding or removing a cube is computationally inexpensive, as it requires at most 36 hash table look-ups, regardless of the total number of cubes present, and therefore will run in $O(1)$ time.

4.3. Link Level: Generation of link motions

Once the optimal path for the cube has been found, it is passed to an algorithm which constructs a path along the faces of the cubes. For each step taken by the cube, this algorithm examines the surrounding cubes and determines the optimal path for moving the cube one step along their faces. For example, if the cube is on the top face of another cube, and must move along the x-axis in a positive direction, the planner will check for a cube immediately below the destination cell. If a cube exists there, it will add that cube's top face to the path. Otherwise, it will check for cubes bordering the cell along the z-axis, and so forth. Once again, since only the immediate area is considered, constant computational time is assured.

The face path is then passed down to the lowest level of the planner, which determines and executes the sequence of link motions that will move the cube in question along the path. This level is divided into three distinct phases: (1) detachment, (2) movement, and (3) attachment. The movement phase may include one or more transitions between the orthogonal planes.

During the detachment phase, the cube is moved from its starting position to the first face on the path belonging to a metacube other than the one to which the moving cube belongs. If this face is on the side of a metacube, the moving cube's two associated links are moved as well; if the face is on the horizontal plane, a sequence of link motions is recorded which will move each associated link, one at a time, to the position of the link under the destination of the moving cube, but this sequence is not executed until later.

The actual link motions to be executed during the detachment phase are hard-coded into the planner, based on the detachment plan being followed, the initial position of the cube to be moved, and the position to which the cube is being detached. For each combination of these variables, there is a predefined sequence of link motions which, due to the regularity of the metacube structure, is guaranteed to move the cube and its associated links into their desired positions.

For each of the 17 detachment plans, there are 8 possible initial cube positions, and at most 12 possible destination faces, so the number of hard-coded cases is less than 1632. Since most detachment cases have less than 12 possible destination faces, the number is in reality significantly smaller. In addition, one would expect about half of these cases to be mirror images of the other half. Currently, 10 of these cases have been implemented in our simulation.

The movement phase determines the link motions necessary to move the cube from its position at the end of the detachment phase to the last face on the face path belonging to a metacube other than the one to which the moving cube is moving. As with the detachment, a predefined sequence of link motions is looked up for each step in the face path based on the cube's current and desired positions relative to the metacube to which it is moving.

The link motions executed during this phase are designed in such a way that, after each step on the path, the links associated with the metacube to which the moving cube is currently attached are in a predefined configuration based on the position of the moving cube. This is possible due to the uniform structure of the metacubes. In addition, unless the moving cube is traversing the horizontal face of a metacube, the links for all other metacubes are in their resting (idle/default) positions.

For movement along the side of a metacube, the moving cube's associated links are moved along the "alleys", which do not contain any links. For movement along the horizontal

face of a metacube, the cube is moved with the metacube's links on the horizontal plane (i.e. "forest"), and the associated links are left behind. When a cube moves from the horizontal plane to a vertical plane, the last two links used to move the cube become the cube's associated links. The rest of the links on the horizontal plane follow the cube's path two steps along the top, and the cube's original associated links are moved into the two open positions, using the sequence recorded during detachment. When a cube moves from a vertical plane to the horizontal plane, the associated links are left behind and a sequence of link motions is recorded which will move them to the horizontal plane. This procedure is similar in its operation to a buffer structure; since exactly one link is moving for each movement of the cube, the procedure takes constant amortized time.

The attachment phase is similar to the detachment phase, in that a predefined sequence of motions is executed based on the attachment plan, the destination of the moving cube, and the moving cube's current position. The number of hard-coded cases is the same as for detachment; at this time, 16 have been implemented.

There are three primary advantages to this approach to generating link motions. First, the procedure is almost entirely modular. Since the link motions guarantee that the links are in a predefined position based on the relative position of the moving cube, the position of all links can be determined given only the cube's position, and in constant time. As a result, the generation of link motions for each step is almost completely independent of all other steps. This simplifies the procedure by eliminating the need for search algorithms and reducing the problem to a simple look-up based on several variables.

Second, the procedure can easily be parallelized. Since the link motions guarantee that, for movement along vertical planes, the links on all other metacubes will be in their rest positions, the motion of multiple cubes can be parallelized by ensuring that only one cube at a time is moving along any given metacube, allowing cubes to "lock" the metacube on which they are moving. Cubes moving along the horizontal plane would lock multiple metacubes.

Third, the procedure is computationally inexpensive. For each step in the cube's path, the low-level planner executes in $O(1)$ time, since it is in effect a look-up table based on a small, constant number of variables. The entire procedure is therefore linear in the number of steps in the cube path. The largest possible number of cells in the entity's perimeter is $10n+8$, where n is the number of cubes in the entity, and since cube paths will not enter a perimeter cell more than once, this bound also applies to the length of the cube path. Thus, the planner runs in $O(n)$ time for each cube.

The data given in Table I and its visualization in Figure 9 reflect the performance of the planner. The planner was run on a Pentium-III 600 MHz processor.

The rows show four different problems involving metacubes that are aligned in one axis where the metacube at one end moves to the other. These cases represent the largest possible perimeter for the given number of metacubes, i.e., the expected average distance traveled by the cubes has the maximum possible value.

TABLE I. A Complexity Assessment

Problem	# of link motions	Solution time
3 metacubes	1187	1:14
4 metacubes	1427	1:56
5 metacubes	1667	2:41
6 metacubes	1907	3:37

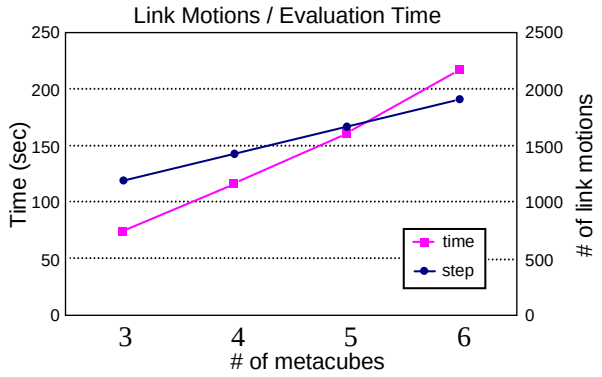

Figure 9. Visualization of the data presented in Table I.

As expected, the number of link motions is linear in the number of cubes in the entity, given the final position assignment for the moving metacube. These link motions are performed sequentially in the current simulation, but would allow for a moderate degree of parallelization. The time taken by the planner is also nearly linear, with a small amount of additional complexity being introduced by the display algorithms.

5. Discussion and Conclusions

A hierarchical motion planning strategy for a distributed bipartite robotic system, I-Cubes, is presented. The approach has been realized in three levels starting with the definition of metamodules at the highest level. The solution generated at the middle level focuses on the cube motions and the lowest level synthesizes the corresponding link actions for a given task by using the predefined link motion sequences for specific cube positions along with the calculated path.

It has experimentally been shown that the proposed technique has some prominent features in organizing itself yielding an approximately linear dependence between time expenditure and the number of cubes contained in the I-Cubes entity, as well as allowing for safe and easy parallelization of motions.

The tests performed on the physical system are still in progress towards proof of concept in the physical domain.

Acknowledgments

The work described in this manuscript was supported in part by Defense Advanced Research Projects Agency under contract DABT-63-97-1-0003, and by the Pennsylvania Infrastructure Technology Alliance, a partnership of Carnegie Mellon, Lehigh University, and the Commonwealth

of Pennsylvania's Department of Economic and Community Development.

The authors would like to thank Dominic Muren, Will Hein, Kevin Peterson, Lemi Daghan Acay and Mike Vande Weghe for their efforts in realizing the physical system.

References

[1] C. J.-J. Paredis and P. K. Khosla, "Kinematic Design of Serial Link Manipulators from Task Specifications," *Int. J. of Robotics Research*, v.12, no.3, pp.274-287, 1993.

[2] B. Neville and A. Sanderson, "Tetrabot Family Tree: Modular Synthesis of Kinematic Structures for Parallel Robotics," Proc. IEEE/RSJ Int. Symp. of Robotics Research, pp.382-390, 1996.

[3] T. Fukuda and Y. Kawaguchi, "Cellular Robotic System as One of the Realization of Self-Organizing Intelligent Universal Manipulator," Proc. of the IEEE Int. Conf. on Robotics and Automation, pp.662-667, 1990.

[4] A. Pamecha, C.-J. Chiang, D. Stein and G. S. Chirikjian, "Design and Implementation of Metamorphic Robots," Proc. ASME Design Eng. Tech. Conf. and Comp. in Engineering Conf., Irvine CA, 1996.

[5] E. Yoshida, S. Murata, K. Tomita, H. Kurokawa and S. Kokaji, "Experiments of Self-Repairing Modular Machine," *Distributed Autonomous Robotic Systems* 3, H. Lueth, R. Dillmann, P. Dario, H. Wörn, (eds.), Springer-Verlag, pp.119-128, 1998.

[6] M. Vona and D. L. Rus, "A Physical Implementation of the Self-reconfiguring Cristalline Robot," Proc. of the IEEE Int. Conf. on Robotics and Automation, pp.1726-1733, 2000.

[7] K. Kotay and D. L. Rus, "The Inchworm Robot: A Multi-Functional System," *Autonomous Robots*, v.8, no.1, pp.53-69, 2000.

[8] M. Yim, D. Duff and K. D. Roufas, "PolyBot: A Modular Reconfirurable Robot," Prof. IEEE Int. Conf. on Robotics and Automation, pp.514-520, 2000.

[9] A. Castano, W.-M. Shen and P. Will, "CONRO: Towards Deployable Robots with Inter-Robots Metamorphic Capabilities," *Autonomous Robots*, v.8, no.3, pp.309-324, 2000.

[10] K. Kotay and D.L. Rus, "Motion Synthesis for the Self-Reconfiguring Robotic Molecule," Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, v.2, pp.843-851, 1998.

[11] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita and S. Kokaji, "A 3-D Self-Reconfigurable Structure," Proc. IEEE Int. Conf. on Robotics and Automation, pp.432-439, 1998.

[12] E. Yoshida, S. Murata, A. Kaminura, K. Tomita, H. Korokawa and S. Kokaji, "Motion Planning of Self-reconfigurable Modular Robot," Proc. of the 7th Int. Symp. on Experimental Robotics, pp.375-384, 2000.

[13] H. Bojinov, A. Casal and T. Hogg, "Emergent Structures in Modular Self-reconfigurable Robots," Prof. IEEE Int. Conf. on Robotics and Automation, pp.1734-1742, 2000.

[14] C. Ünsal and P. K. Khosla, "A Multi-Layered Planner for Self-Reconfiguration of a Uniform Group of I-Cube Modules," to appear in Proc. 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2001.

[15] A. Pamecha, I. Ebert-Uphoff and G. S. Chirikjian, "Useful Metrics for Modular Robot Motion Planning," *IEEE Trans. on Robotics and Automation*, v.13, no.4, pp.531-545, 1997.

[16] J. E. Walter, J. L. Welch, N. M. Amato, "Distributed Reconfiguration of Hexagonal Metamorphic Robots in Two Dimensions," Proc. SPIE, Sensor Fusion and Decentralized Control in Robotic Systems III, v.4196, pp.441-453, 2000.

[17] E. Yoshida, S. Murata, H. Kurokawa, K. Tomita and S. Kokaji, "A Distributed Reconfiguration Method for 3-D Homogeneous Structure," Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp.852-859, 1998.

[18] M. Yim, D. Goldberg and A. Casal, "Connectivity Planning for Closed-Chain Reconfiguration," Proc. SPIE, Sensor Fusion and Decentralized Control in Robotic Systems III, v.4196, pp.402-412, 2000.

[19] C. Ünsal, H. Kiliççöte, M. Patton and P. K. Khosla, "Motion Planning for a Modular Self-reconfiguring Robotic System," Distributed Autonomous Robotic Systems 4, Springer-Verlag, pp.165-175, 2000.

[20] C. Ünsal, H. Kiliççöte, and P. K. Khosla, "A Modular Self-Reconfigurable Bipartite Robotic System: Implementation and Motion Planning," *Autonomous Robots*, v.10, no.1, pp. 23-40, 2001.

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

[22] C. Ünsal and P. K. Khosla, "Solutions for 3-D Self-reconfiguration in a Modular Robotic System: Implementation and Motion Planning," Proceedings of SPIE, Sensor Fusion and Decentralized Control in Robotic Systems III, November 2000.