2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Self Assembly of Modular Manipulators with Active and Passive Modules

Seung-kook Yun and Daniela Rus

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
yunsk@mit.edu,rus@csail.mit.edu

***Abstract*— We describe self-assembling robot arm systems composed of active modular robots and passive bars. We present a case study where the robotic module is the Shady3D robot and the passive component is a rigid bar with embedded IR LEDs. We propose algorithms that demonstrate the cooperative aggregation of modular robotic manipulators with greater capability and workspace out of these two types of elements. We present results from physical experiments in which two 3DOF Shady3D robots and one rigid bar coordinate to self-assemble into a 6DOF manipulator. We then demonstrate cooperative algorithms for forward and inverse kinematics, grasping, and mobility with this arm.**

I. INTRODUCTION

This paper explores the development of low-cost modular manipulators. Drawing from the theoretical, practical, and existing experience in manipulation and modular robotics, we propose an approach to synthesize modular manipulators that match a desired workspace by self-assembly. We envision robot systems capable of scavenging raw materials from the environment to adaptively create dynamic programmable structures that integrate robotic elements with passive components. We describe how a collection of simple robotic modules can grasp rigid bars and coordinate to self-assembled robotic manipulators with a higher number of degrees of freedom and a larger workspace than the components. The resulting robot arms are distributed mobile manipulation systems that can be controlled to accomplish the basic functionality of a robot arm: inverse kinematics, forward kinematics, grasping, and pick and place. These arms can move autonomously to different places in the workspace. The specific type of arm we study alternates robotic elements with rigid bars. The presence of the rigid bars enhances the structural rigidity of the system and also contributes to the total number of degrees of freedom of the system. The total number of elements is determined by the required workspace size. We aim to synthesize the smallest robot structure that meets the workspace requirements.

The challenge in building self-assembled modular arms ranges from issues related to designing simple and robust active modules capable of interacting with other passive and active modules, to problems of control and planning. Control is challenging because each active link is a separate robot. The many degrees of freedom of these systems have to be coordinated using distributed and efficient controllers.

We present algorithms for the self-assembly of multi-link robot arms out of 3DOF robot modules with the structure and capabilities of our robot Shady3d [1] and rigid bars with embedded LEDs for guiding grasping. We assume to know the location of the robot modules and of a cache of smart passive bars. Given a desired workspace, we determine the number of needed links. A distributed self-assembly algorithm constructs the robot arm as an alternation of robot elements and passive bars. We demonstrate this algorithm in the context of creating a 6DOF manipulator out of two Shady3D elements and one passive bar. We also present cooperative algorithms for forward and inverse kinematics, grasping, and pick and place and give data from physical experiments.

A. Related work

We build on prior work on modular manipulators [2] and self-reconfiguring robots [2]–[7]. Previous work on modular manipulators considers how to manually assemble an arm, given a collection of modules. Most previous work on self-reconfiguring robots considers a homogeneous system that has multiple copies of the same part. In our project we depart from the premise of homogeneous robots and describe heterogeneous modular robots with active and passive links.

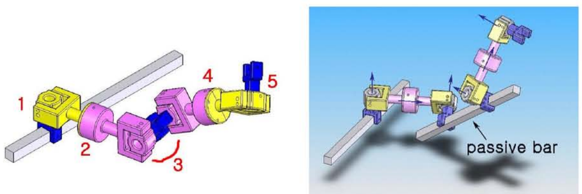
The robot arms we propose are closely related to truss climbing robots such as Staritz et al's "Skyworker" [8], Amano et al's handrail-gripping robot for firefighting [9], Ripin et al's pole climbing robot [10], Nechba, Xu, Brown et al's "mobile space manipulator SM2" [11], Kotay and Rus' "Inchworm" [12], and Almonacid et al's parallel mechanism for climbing on pipe-like structures [13] and our own Shady2D and Shady3D modules [14].

B. Outline

This paper is organized as follows. Section II reviews the capabilities of Shady3D, the robot module we use in this work. Section III presents the self-assembly algorithm and experimental results for the self-assembly to a 6DOF arm. We propose control algorithms for the inverse kinematics control of serial linkages in Section IV. Section V describes the algorithms and the results of physical experiments for kinematics, inverse kinematics, grasping, and pick and place with the 6DOF arm.

(a) (b)

Fig. 1. (a) Shady3D and its structure: 3-joints and 2 grippers (b) Shady3Ds are moving on the trusses



(a) (b)

Fig. 2. (a) A 5DOF manipulator with directly connected two Shady3Ds (b) A 6DOF one with inserting a passive bar between two robots [16]

II. Experimental infrastructure: Shady3D and rigid bars with LEDs

In this work, we use Shady3D [1] as an active module and a bar with embedded IR LEDs as a passive one. The resulting arm can be anchored anywhere on a truss (See Figure 8). The algorithms presented here depend on the abstract capabilities of Shady3D and can be instantiated on any other robot module with similar capabilities. We introduce the hardware and how they build a self-assembled tower.

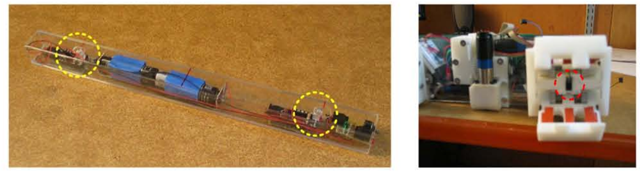
A. Shady3D

Shady3D was originally designed with the goal of climbing 3-dimensional trusses as a first step toward tree-climbing robots. It has three joints for 3-D motion and two grippers on each side as shown in Figure 1. The number of joints is chosen to be minimal for moving on the 3-D trusses. Unlike Shady [15] which was designed to climb planar trusses, the middle joint enables Shady3D to switch from one plane to another as Figure 1(b).

The three joints of Shady3D enable a robot to traverse 3D trusses. By connecting two Shady3Ds (See Figure 2(a)) directly we can generate a 5DOF linkage. The DOF is not six due to the fact that the axes of two gripper joints lie on the same line. A 6DOF linkage is obtained by using a truss element as a medium of connection as in Figure 2(b).

B. Passive bar

The self-assembly operation requires many grasping steps that need to be robust. We choose an approach that embeds beacons in the passive object. Solutions that rely on other sensors such as vision are possible but require more computation. A passive bar emits IR signals via the IR LEDs embedded in the bar is shown in Figure 3(a). Two LEDs are located at each side of the bar (indicated by yellow dotted circles) and inform a robot about existence of the bar.



(a) (b)

Fig. 3. (a) A passive bar with embedded IR LEDs (b) an IR sensor attached beneath the gripper

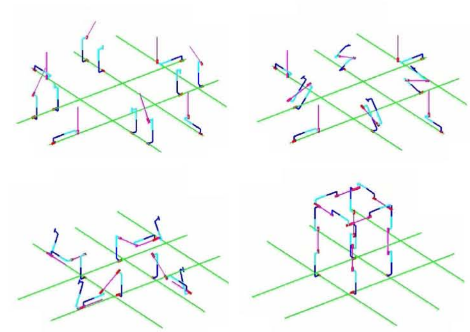


Fig. 4. Four snapshots of the tower building simulation. The Shady3D robot modules are drawn as an elongated U-shapes with light and dark halves; the free bars and the grid are drawn as straight segments [16].

C. Self-assembled linkage: walking tower

Figure 4 shows snapshots of the self-assembly of a truss tower. Twelve active modules and eight passive bars are employed to build a three-dimensional tower. Figure 5 shows the resulting structure moving.

III. Algorithm for Cooperative Self-assembly of Modular Arm

We describe how to build a serially linked manipulator - one leg of the tower - from arbitrary numbers of Shady3Ds. This algorithm is used to build a 6DOF linkage from two 3DOF Shady3D modules.

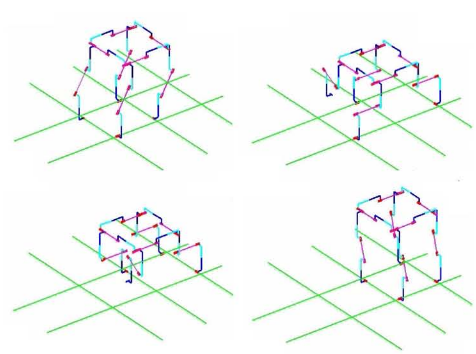


Fig. 5. Four snapshots of the tower moving simulation [16].

A. Self-assembly algorithms

The output of the workspace computation algorithm is a linkage manipulator which consists of serially connected n robots and $(n-1)$ passive links and meets the task requirement.

We want n robots to come together on the support structure necessary to anchor the arm and them to build the linkage together. The cooperation among the robots relies only on local communication with a limited range.

More specifically, we assume:

- The support environment and location of the robots is given in the form of a graph with nodes and edges as described in [17].
- The target location is reachable from every robot's starting point.
- A robot has a unique ID.
- Robots can communicate only when they are nearby: located on adjacent nodes.
- using communication, a robot can read the state of the other robot and send a control command.
- Locations of the passive bars are given.

The coordination of the robots is done using a state variable. robots use the state to indicate to other robots what they are doing. We use the following states:

- *Idle*: waiting for a command
- *Moving*: motors are working
- *Assembling*: doing self-assembly now
- *Assembled*: assembled with other robots

Algorithm 1 Building serially linked robots

```
Locomote to the anchor point (Algorithm 2)
if anchor is empty then
    Be the root of the linkages
else
    while The root robot's state≠Assembled do
        Delay
    end while
    Add myself to the linkages (Algorithm 3)
end if
```

Algorithm 1 gives the high-level planning algorithm. Once robots are initialized, they independently travel to the location of the arm with Algorithm 2. Firstly, each robot finds the best route to the anchor point by Dijkstra's algorithm. Every time they advance to the next node, they check whether that node is already occupied. If not, they move. Otherwise, they wait until the node is empty. Since the anchor is the same for all the robots, there is no deadlock. Details of distributed deployment for the general case was extensively reviewed in [17].

The first robot to arrive at the anchor location becomes the root. The root robot communicates to the robots approaching the anchor. The robots are assembled into the linkages in order of their arrival at the anchor point using Algorithm 3. Note that Algorithm 3 can be used for the root robot of the arm.

Algorithm 2 Locomotion to the anchor point

```
Find the shortest path to the anchor
while not reached the target or not met the linkages do
    Communicate with adjacent robots
    if next node is empty then
        Move to the next node
    else
        Wait until the next node is empty
    end if
end while
```

The addition of new modules starts from moving the robot to the nearest approachable node. The robot then communicates with the root robot of the arm, in order to let it grasp the other side of the bar and assemble it. The robot uses information about the location of the passive bar and grasps it, guided by the emitter on the bar's grasping point. To reach the grasping point, the robot and the arm uses the inverse kinematics algorithm described in Section IV.

Algorithm 3 Addition of a robot to serial linkages by Self-assembly

```
State = Moving
Move to the nearer approachable node
State = Idle
while the other approachable node is empty do
    Communicate with the adjacent robot
end while
Let the root robot assemble me
State = Assembling
Move to check the passive bar
if sense a bar then
    Grasp the bar
    State = Idle
    while The root robot's State≠Assembled do
        Delay
    end while
    State = Assembled
else
    Return to the original position
    State = Idle
end if
```

After self-assembly to new serial linkages, we assign two roles to the first and the last robot: a root and a leaf, where the former has a fixed gripper at the truss while the latter is the end point of the linkages. The fixed gripper of a root is called an anchor and the free gripper of a leaf is an end-effector.

B. Experiments with self-assembly of two Shady3Ds

The proposed algorithms are implemented in experiments with two Shady3D robots and one bar. Figure 6 shows snapshots from the experiment. Firstly, given a specified position for the passive bar within the Shady3D experimental environment, each Shady3D module optimally positions itself so as to be able to reach the bar. Details of the optimal deploying algorithm are addressed in [17]. In the first step of the algorithm each Shady3D module moves independently and in parallel to reach and grasp the bar. The bar is detected using the LED sensors within the Shady3D grippers. Upon grasping the bar, the Shady3D modules signal to each other using Bluetooth to coordinate the completion of the grasping step and the self-assembly of a 6DOF manipulator. We tested the self-assembly, and a sequence of 10 executions resulted in no error. Each self-assembly experiment took 1 minute (See Table I).

IV. Controlling the modular arm by distributed inverse kinematics

Reaching an arbitrary point in space by serial linkages requires robot coordination. Unfortunately, the structure of Shady3D does not allow a closed-form inverse kinematics solution even for the simplest 6DOF linkages from two robots. Instead of using an explicit solution, we use an approximation algorithm based on the manipulator jacobian. We select a Damped Least Square (DLS) method because it has good robustness and performance [18].

Only the root robot calculates the joint angles to grasp a bar when needed. The solution propagates along the linkages by successive communication to the leaf robot. When sending the angles, each robot cut off the first three angles from the solution and send the remaining ones. Then it rotates by those three angles. After finishing moving, the robot await that the leaf-side robot stop rotating, by checking its status. Therefore, the status of the robots change in one by one from the leaf to the root.

V. Manipulation Tasks with Modular 6DOF Arm

We have developed algorithms for four kinds of tasks with the manipulator. The algorithms were implemented on our physical prototype 6DOF modular manipulator. In each case, task information is given to the robots in the form of a command stack. The robots decide which role to play based on the task specification and its location.

A. Distributed control algorithm for task execution

Each task is a stack of command sets for the two robots, and how a robot execute the task is shown in Algorithm 4. Parameters of the command set are:

- *RootNode (#)*: the root location to anchor the arm
- *Displacement (x,y,z,roll,pitch,yaw / $\theta_1 \cdots \theta_6$)*: 6 joint movements and end-effector displacement
- *Grasp (G/R)*: grasp/release of the end-effector

Each robot starts by finding out if it is a root. The root robot calculates the joint displacement of two robots directly or indirectly by inverse kinematics. The leaf robot waits for a command. The root sends the corresponding joint displacements to the leaf robot. Then they both execute their next command in parallel. The root checks the command completion, and then pops the next command set until the stack is empty.

Algorithm 4 Task execution

```
while Task Stack not empty do
  Pop the next queue
  if Anchor = Root then
    Get the commands from the queue
    Send the command for the leaf
    State = Moving
    Execute my command
    while The leaf's State = Moving do
      Delay
    end while
    State = Assembled
  else
    Wait for the command from the root
    State = Moving
    Execute my command
    State = Assembled
  end if
end while
```

B. XYZ-directional movement

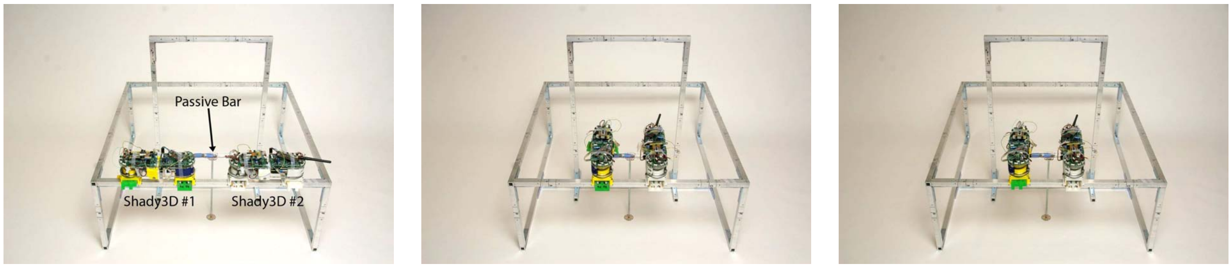
In this task, the distributed inverse kinematics protocol is used to implement the positioning of the arm's end effector at a desired location (x,y,z). The arm's initial configuration is shown in Figure 7(a). The left gripper of the arm is the anchor and the right gripper is the end-effector. We have tested different (x,y,z) locations for the 6DOF manipulator built in Section V as shown in Figure 7(b-c). Each experiment was done 10 times without error and it took 20 seconds(See Table I.) In this case, the task stack has only one command set with a single end-effector displacement.

One challenge is coping with the position error along the vertical axis - in this case, Z-directional - because of tilting of the arm due to gravity. About 20mm error was measured regardless of the Z-directional displacement. The error mainly comes from mechanical weakness of a robot (e.g. backlash, tolerances, and plastic material).

C. Reaching nodes unreachable by one robot

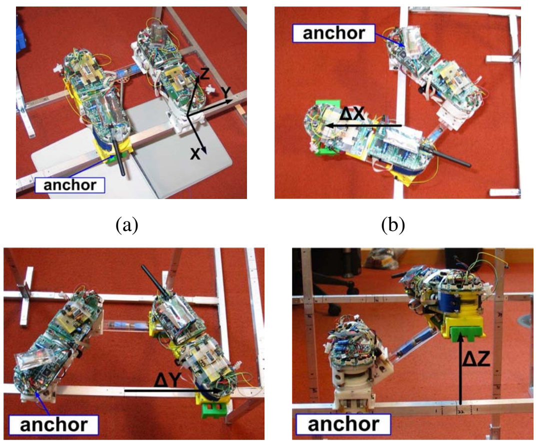
Consider an inspection task which requires reaching every point on the truss. As pointed out in [16], some points on the truss are unreachable by one robot due to its fixed length and 3DOF. When we model the truss environment as a graph where nodes are points of interest and edges correspond to reachability among the nodes, such unreachable points are nodes without an edge. Upon self-assembly, many unreachable points become reachable by the 6DOF linkage because of enhanced workspace and additional DOFs.

Figure 8 shows the self-assembled robot built in Section V. reaching the unreachable nodes(denoted by the arrows). The task stack has one command set with a single end effector displacement according to 3-D locations of the nodes. Three unreachable nodes were tested ten times each without error. Each task took 40 seconds(See Table I.) The position error along the vertical axis due to the mechanical weakness of the arm persists for the task as well with an observed maximum



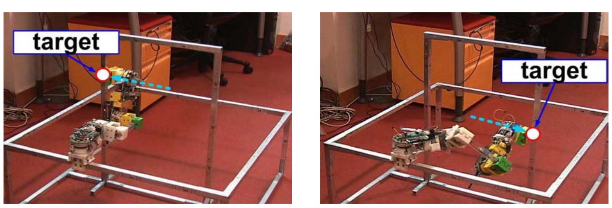
(a) (b) (c)

Fig. 6. Implementation of self-assembly of 6DOF modular arm and an example of moving a bar. (a) Two robots have moved to the approachable nodes. (b) They are swinging their body to find the bar. (c) They have grasped each side of the bar.



(a) (b) (c) (d)

Fig. 7. Uni-directional movement of a 6DOF manipulator composed of self-assembled Shady3Ds. (a) A self-assembled manipulator with two Shady3Ds. The left gripper is anchored at the truss and the right one is free to move. (b) X-directional movement with 150mm displacement (c) Y-directional movement with 150mm displacement (d) Z-directional movement with 150mm displacement



(a) (b)

Fig. 8. A 6DOF manipulator with two Shady3Ds reaches some nodes which are unreachable by one robot. The robot can be anchored anywhere in the environment.

60mm tilting. In our environment, the self assembled 6DOF can reach all the nodes.

D. Pick and drop by forward kinematic control

In this task, the arm collects an object(a bar), moves to a different location where it drops the object. This task requires a 6DOF manipulator. The locations of pick and drop are given by joint angles. The robot moves by distributed forward kinematic control.

The task stack is composed of 7 command sets each of which has one joint displacement or grasping/release. As the task starts, one of the modules releases its grasp of the environment. Figures 9(a, b, c) shows two modules controlled independently and in parallel to demonstrate the movement of the arm. An additional bar is manually presented to the free gripper of the 6DOF manipulator. The bar is grasped, transported, and dropped at a specified location (see Figure 9 (d, e, f).) We have performed this experiment 10 times in a row during the course of one hour. Each experiment consisted of 9 joint movements and 5 grasping/release operations, and it took about 140 seconds. All the control steps succeeded for all the experiments. However, due to a hardware failure at the end of the 7th experiment one of the gripper motors had to be replaced(See Table I.)

A summary of our experiments is shown in Table I.

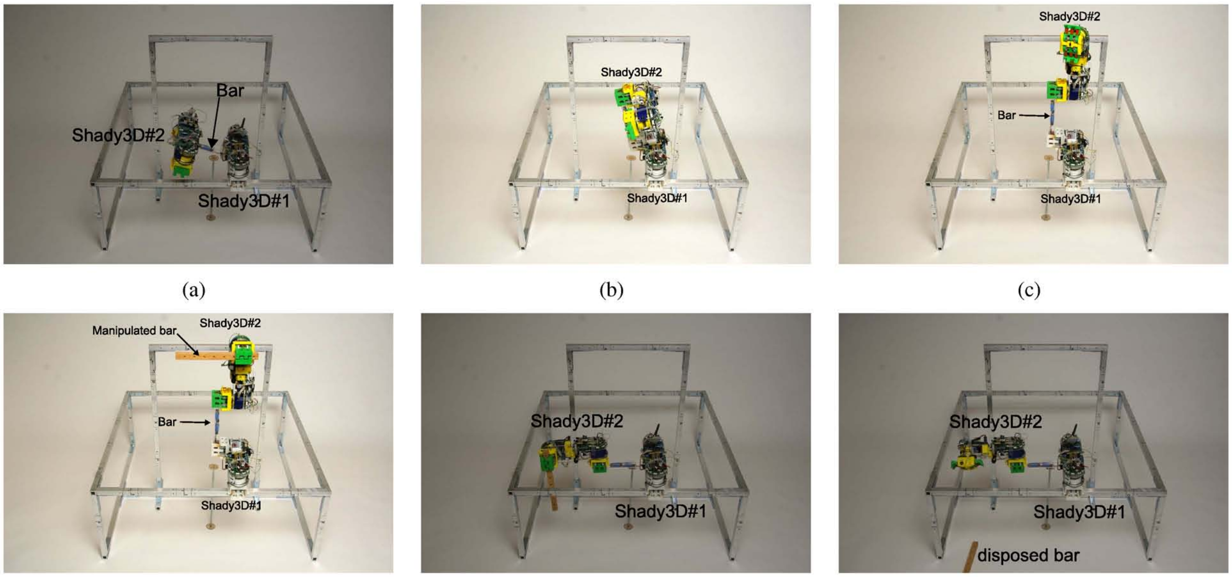
VI. CONCLUSIONS AND FUTURE WORKS

In this paper we discussed our first step toward building a self-assembled robot composed of passive components and modular manipulators. As a first stage, we designed a module with the minimal number of joints for 3D movement and building a 6DOF manipulator. By combining two modules and one passive bar, we can generate a more capable robot. We described a suite of algorithms and experiments for building a serial linkage. We proposed the inverse kinematics to control multi-robot in 3D space without serous position error and long convergence time. Hardware implementation of building a 6DOF manipulator and several tasks show how the proposed self-assembly works in the real world. The coordinated manipulation algorithms perform well. They are generally robust and the response time is adequate for the tasks we considered. However, the materials used in the prototype cause a structured tilting error which has to be eliminated in future versions.

Much work remains to be done in the direction of building modular robots with active and passive components. There is a great need for a better hardware design and high-level distributed planning algorithms.

VII. ACKNOWLEDGMENTS

This work was supported in part by NSF IIS-0426838, NSF EFRI-0735953 and NSF CNS-0707601. Seung-kook



(a) (b) (c)

(d) (e) (f)

Fig. 9. Implementation of self-assembly of 6DOF modular arm and an example of moving a bar. (a) A 6DOF manipulator combined by two modules and the base module pulls the other upward. (b) The base module has fully moved the other module up. (c) The manipulator is stretched to the maximum height. (d) The end effector is given a bar to be moved. (e) The bar is moved to the dropping position. (f) The manipulator has dropped the bar.

TABLE I
RESULT OF THE EXPERIMENTS

experiment	number of execution	number of joint displacement	number of grasping/release	success ratio	operation time(sec)	remark (error)
Self assembly	10	6	4	10/10	60	
Pick and drop	10	5	3	9/10	140	motor failure
XYZ move	30	6	0	30/30	20	tilting error
Reaching	30	6	0	30/30	40	tilting error

Yun is supported in part by Samsung Scholarship. We are grateful for this support. We thank Marsette Vona and Carrick Detweiler for their help and insightful discussions and Luliu Vasilescu for photography.

REFERENCES

[1] Y. Yun and D. Rus, "Shady3d:a robot that climbs 3d trusses," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4071–4076.

[2] Cem Unsal, Han Kiliccote, and Pradeep Khosla, "A modular self-reconfigurable bipartite robotic system: Implementation and motion planning," *Autonomous Robots*, vol. 10, no. 1, pp. 23–40, Jan. 2001.

[3] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirickjian, "Modular self- reconfigurable robot systems," *EEE Robotics and Automation Magazine*, vol. 14, pp. 43–52, 2007.

[4] Satoshi Murata, Haruhisa Kurokawa, Eiichi Yoshida, Kohji Tomita, and Shigeru Kokaji, "A 3-d self-reconfigurable structure," in *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, Leeuven, Belgium, May 1998, pp. 432–439.

[5] Keith Kotay, "Self-reconfiguring robots: Designs, algorithms, and applications," Ph.D. dissertation, Dartmouth College, Dec. 2003.

[6] David G. Duff, Mark Yim, and Kimon Roufas, "Evolution of polybot: A modular reconfigurable robot," in *Proceedings of the Harmonic Drive International Symposium*, Nagano, Japan, Nov. 2001.

[7] Kasper Støy, "The ATRON self-reconfigurable robot: challenges and future directions," Presentation at the Workshop on Self-reconfigurable Robotics at the Robotics Science and Systems Conference, July 2005.

[8] Peter J. Staritz, Sarjoun Skaff, Chris Urmson, and William Whittaker, "Skyworker: A robot for assembly, inspection and maintenance of large scale orbital facilities," in *IEEE ICRA*, Seoul, Korea, 2001, pp. 4180–4185.

[9] Hisanori Amano, Koichi Osuka, and Tzyh-Jong Tarn, "Development of vertically moving robot with gripping handrails for fire fighting," in *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, HI, 2001, pp. 661–667.

[10] Zaidi Mohd Ripin, Tan Beng Soon, A.B. Abdullah, and Zahurin Samad, "Development of a low-cost modular pole climbing robot," in *TENCON*, vol. I, Kula Lumpur, Malaysia, 2000, pp. 196–200.

[11] Michael Nechyba and Yangsheng Xu, "Human-robot cooperation in space: SM2 for new space station structure," *IEEE Robotics and Automation Magazine*, vol. 2, no. 4, pp. 4–11, Dec. 1995.

[12] Keith D. Kotay and Daniela L. Rus, "Navigating 3d steel web structures with an inchworm robot," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 1996.

[13] M. Almonacid, R. J. Saltarén, R. Aracil, and O. Reinoso, "Motion planning of a climbing parallel robot," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 485–489, 2003.

[14] Carrick Detweiler, Marsette Vona, Yeoreum Yoon, Seung-kook Yun, and Daniela Rus, "Self-assembling mobile linkages with passive and active modules," *IEEE Robotics and Automation Magazine*, 2007.

[15] C. Detweiler, M. Vona, K. Kotay, and D. Rus, "Hierarchical control for self-assembling mobile trusses with passive and active links," in *IEEE Intl. Conf. on Robotics and Automation*, 2006, pp. 1483–1490.

[16] Y. Yoon, "Modular robots for making and climbing 3-d trusses," Master's thesis, Massachusetts Institute of Technology, Cambridge, 2006.

[17] S. kook Yun and D. Rus, "Optimal distributed planning of multi-robot placement on a 3d truss," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2007.

[18] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose pseudoinverse and damped least squares methods," Apr. 2004. [Online]. Available: http://www.math.ucsd.edu/ sbuss/ResearchWeb/ikmethods/iksurvey.pdf