A Distributed Blackboard-based Control System for Modular Self-Reconfigurable Robots

T. L. Lau H. Y. K. Lau Albert Ko

The University of Hong Kong
Department of Industrial and Manufacturing Systems Engineering
Pokfulam Road, Hong Kong SAR, China
e-mails: aux1496@hkusua.hku.hk

Abstract

Modular Self-Reconfigurable (MSR) robots are made up of identical mechatronic modules that can be assembled into a variety of structures to facilitate different kind of works. The controlling of MSR robots is similar to the control of a distributed cooperating system, in which all modules must be coordinated correctly to complete a common task.

In this paper we discuss the implementation of the distributed blackboard architecture for locomotion control of a MSR robot. We have used computer to construct a model of the MSR robot and used it to demonstrate the capability of a simple rule-based blackboard control system in coordinating a group of modules to perform a common task using purely local control rules and through communication between neighboring modules.

Keywords

Self-Reconfigurable Robots, Modular Robots, Blackboard Systems, Distributed Autonomous Systems, Locomotion Control.

1. Introduction

Modular Self-Reconfigurable (MSR) robots [4][16][18][14][12][6][9] are robots made up of many identical but independent mechatronic modules that can be disconnected and reconnected autonomously and to rearrange into different structures that can facilitate the robot to complete its tasks more effectively. Each individual module is a self-contained unit equipped with its own processor to control the module's movement and to facilitate communication with neighboring modules.

MSR robots' ability in self-reconfiguration makes them particularly useful for applications in unstructured, remote and hazardous environment such as deep sea exploration, space exploration, urban rescue, mining, intelligent material handling and military intelligence. Since all modules are identical, if any module in a system is damaged; the robot can simply discard the damaged module and quickly replace it with another one located nearby. This functionality gives MSR robots a distinctive advantage over conventional robots in repairing itself while far from home on a mission.

In spite of all the advantages MSR robots has to offer, there are many challenges to overcome before these robots can have practical applications outside of research. One of the biggest challenges is to develop decentralized control system that does not require a designated leader for coordination. The rationale for not having a centralized controller (a leader) is to avoid a MSR robot (i.e. the group of connected mechatronic modules) from having a concentrated (specific weak point, i.e. the leader,) weak point of failure and to achieve total homogeneity in module design.

Because of the unique advantages distributed control systems has to offer for self-reconfigurable robots, researchers around the world have examined many different approaches to the mechatronic and control algorithm design of different kinds of MSR robots. Butler et al. at Dartmouth College [5] proposed a distributed goal recognition technique called "Trace" to generate global shape using only local knowledge and local communication. Unsal et al. [15] at Carnegie Mellon University have presented a multi-layered planner for the motion of modules with a combination of distributed approaches at the high-level with low-level pre-defined rules for trajectory motions. Bojinov et al. at PARC [3] applied multi-agent control to randomly generated stable structures based on local rules.

In this paper we will present the application of a distributed blackboard-based system for controlling the locomotion of a 5-module MSR robot based on limited local information. Blackboard-based systems are systems built on a model in which a number of experts cooperate in solving a particular problem [13]. Blackboard systems are inherently suitable for MSR robots due to the following key attributes [4][13]:

Modularity – the arrangement of blackboard system components (knowledge source, blackboards, independent processors, etc.) is highly flexible, and can be arranged in self-contained groups of modules.

Merging of Knowledge – decisions are reached by consulting with multiple experts. Each module of the robot can be treated as an expert in the problem solving process.

Continuous Problem Solving – solutions are built incrementally and continuously over time, hence, there is always a current best solution. This is particularly useful for the robot to handle unexpected events.

Parallel Processing – the modularity of many knowledge sources in the system generates a coarse-grained parallelism at the module level.

Totally Distributed Design – the system has no leader, no teacher, nor any centralized elements, therefore no concentrated (specific) weak point of failure.

Through the incorporation of the abovementioned characteristics, the proposed 5-module MSR robot has been built and tested in a computer simulated environment using MATLAB. Details of system design, experiment setup and test results are presented in the following order. Section 2 provides an overview of the blackboard analogy, different blackboard architectures, and advantages of employing blackboard systems in MSR robots. Section 3 presents the simulation setup, model design, locomotion control and communication scheme. Section 4 proposes the future plan of our research and a conclusion is given in Section 5.

2. Blackboard Systems

2.1 An Overview

To visualize the idea of a Blackboard model, we can visualize three experts sitting in a meeting room equipped with a blackboard (Figure 1). The blackboard is viewable to all experts at all time but only one piece of chalk is available for writing. Without any particular order, expert Beta goes up to the blackboard, picks up the only piece of chalk and writes down her problem. Expert Alpha sees the problem and writes on the blackboard what he thinks is useful in solving the problem. Expert Delta analyzes the information posted by Beta and Alpha and writes down his opinion. This scenario is repeated continuously. Each expert adds their knowledge to the blackboard, and reevaluates their own opinion with respect to the new information. Eventually the problem can be solved when enough information is gathered. This model of a basic blackboard system is shown in Figure 1.

There are three core components in a blackboard system: one is the blackboard for communication purpose and the other two are the independent knowledge source and its associating processor. By arranging these components in different configurations, unique blackboard architectures can be designed to suit specific needs. Distributed Blackboard architecture (DBB) [10], for example, is designed for a low-bandwidth distributed network environment. DBB systems have separate blackboards for each expert and information exchange is done between processors only. For faster operation, the Blackboard Server architecture (BBS) [8] was developed to eliminate multi access to the blackboard by designating one expert as the server to read and write to the blackboard and to channel information to other experts. Other architectures include the Virtual Blackboard (VBB) [1] and Shared Memory Blackboard (SMBB) [11]; each has a distinct architecture and advantages over others in different applications.

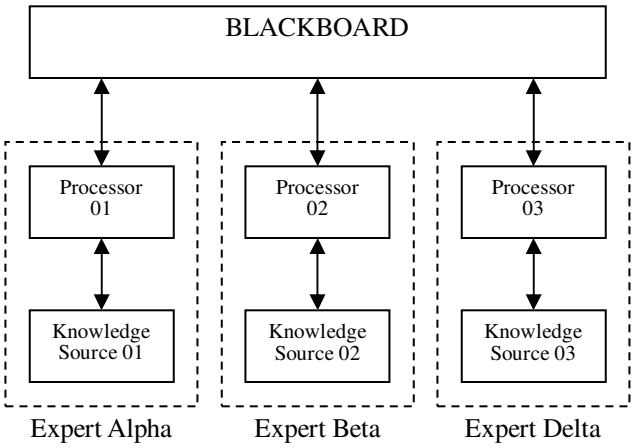

Figure 1: The basic architecture of a Blackboard System.

In the following section we will discuss why we have chosen the Distributed Blackboard (DBB) architecture as the control framework of our MSR robot and present the basic architecture of the system.

2.2 Distributed BBS for MSR Robots

When choosing a control system for our MSR robot, we firmly respect several criteria to ensure the proposed system can be easily implemented to our MSR robot under development and is general enough to be implemented on other decentralized MSR robots. These criteria are:

- Total Modularity – the system must be able to implement on modules that are absolutely identical in mechatronic design.
- High Expandability – the system must be able to evolve into more intelligent systems, i.e. to incorporate intelligent learning systems such as neural network.
- High Scalability – the system must be equally effective in control and communication regardless of the number of modules it is implementing on.

With respect to the above requirements, the Distributed Blackboard architecture (DBB) is the best option for our application because it is inherently divided into self-contained modules comprising all core components, the blackboard, the processor, and the knowledge source (total modularity). Like all other blackboard systems, DBB systems can be programmed to perform intelligent learning and reasoning (high expandability), and communication performance is not hindered by the number of modules within since DBB systems can support data parallelism very well [2]. The function of the three core components in each MSR robots' modules can be roughly outlined as follow:

The Blackboard can be treated as a synchronized temporary information storage that stores information created locally or mirrored from other module's blackboards through inter-module communication. The stored information is synchronized among all blackboards within the MSR robot, so each processor has exactly the same common information to work on.

The Knowledge Source stores common knowledge that is essential for the system to operate, such as rules for controlling the movement of the module and for evaluating the environment. This component is also responsible for storing unique knowledge acquired through learning during operations.

The Processor is the "brain" of each module, responsible for decision making and communication with other modules to update information on the blackboard(s). Together with the knowledge source, a processor can learn to acquire unique knowledge for problem solving; hence it is capable of shaping its corresponding module into an independent specialist.

The layout of a DBB system is illustrated in Figure 2. Arrow lines indicate information flow and dotted lines represent boundaries of modules. To understand how information flows in a DBB system, let us consider the sample system below.

When an external stimulus comes through the sensor (e.g. a tactile sensor returning a contact signal) of a module of the MSR robot that reaches processor_02 as a digital signal, processor_02 will carry out two different procedures for control and communication purposes. For control purpose, processor_02 will evaluate the input signal and execute the program command stored in the knowledge source to direct the movement of the module or to evaluate the environment. For communication purpose, processor_02 will post the signal on blackboard_02 as a new piece of information and pass the same information to all directly connected neighbors, i.e. processor_01 and processor_03. Processor_01 and processor_03 will post the received information on their corresponded blackboards and execute program command in respond to the signal. In addition, processor_01 and processor_03 will pass again the information to their neighboring processors, and similarly, the neighboring processors will pass the information to its neighbors. The information transmission process continues until all modules have the same information on their blackboard.

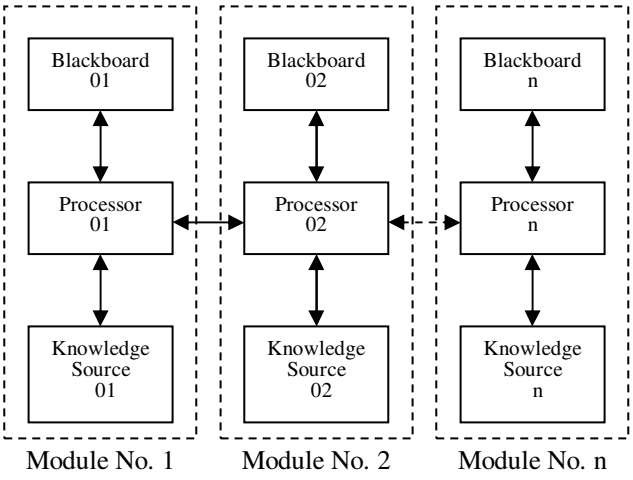

Figure 2: Distributed Blackboard system (DBB).

3. Locomotion Control

In this section, a distributed control system based on the DBB system for controlling the locomotion of the 5-module MSR robot is presented. The five modules are arranged in a looped chain that resembles the shape of a pentagon and can perform planar rolling like the tractor belt of a tank. The purpose of this experiment is to demonstrate how a simple rule-based blackboard control system can coordinate a group of self-contained modules to perform a common task using purely local control rules executed at every module and through communication between neighboring modules alone.

3.1 The Simulation

The mechatronic module we want to simulate in this experiment is a two link mechanism connected by a hinge joint. The simulated environment is assume to be absolutely flat and has no obstacles in the robot's line of motion. The simulation model was developed using MATLAB. Figure 3 shows the basic structure of a single module.

The links of the module are represented by 2 vectors expressed in polar coordinates. The tail of the module (the end of the rear link) is marked with a little square, the origin of the module (where the two links join) is marked with a circle, and the head of the module (front end of the head link) is left unmarked to avoid confusion when connected to another module. The two links of the module are labeled as *Head Link* and *Rear Link* respectively. The

angle between the rear link and the x-axis is called the *global angle*; it increases in counter clockwise direction with zero starting from the positive x-axis. The *local angle* is the external angle between the rear link and the head link, when this angle equals zero, the two links form a straight line.

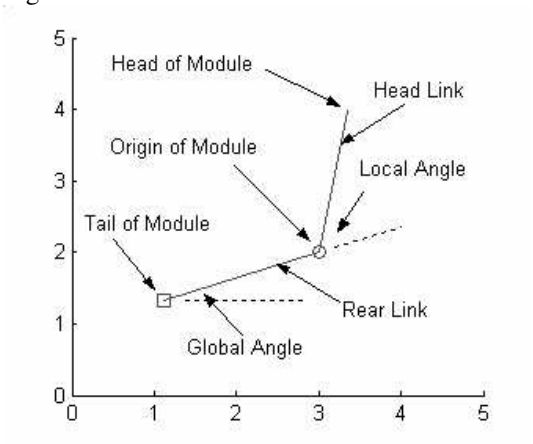

Figure 3: Computer simulated module.

The initial stage of the simulation begins with all 5 modules connecting head to tail to form a closed loop. All modules' local angles are adjusted to equal value, hence forming a perfect pentagon. Figure 4: shows the initial stage of the model simulated in MATLAB.

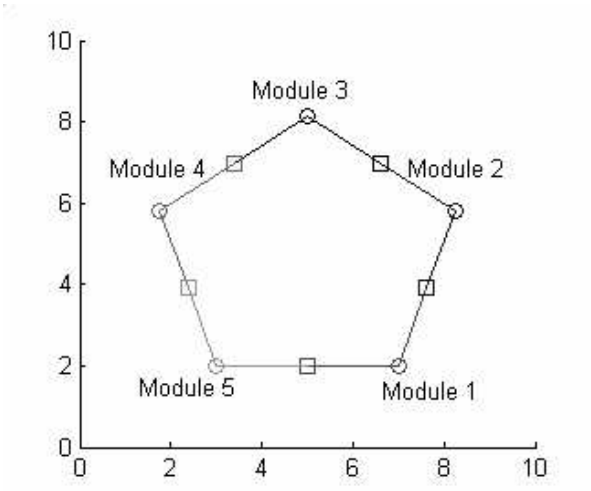

Figure 4: Initial stage of the simulated MSR robot.

To avoid the model from overlapping with the x-axis during simulation, we have arbitrarily chosen y = 2 as the ground and set initial position of the origin of module_1 at (7, 2). Before any module can initiate a move, there are certain information that all modules must submit to the blackboard and cross evaluate. These information are:

Call-signs of the two neighboring modules –

In this experiment, all modules are assigned with a *call-sign* and are capable of reading the neighboring modules' call-sign. These simple pieces of information enable all modules to understand if the robot is forming a close loop or an open-end structure.

Local angle –

By analyzing the value of all local angles together with the order of each module's call-sign, the shape of the robot can be determined by the processors.

Ground contact signal –

All modules are assumed to have a touch sensor at their origin and can return a *ground_contact* signal if the origin of the module is in contact with the ground. In the simulation this is done by comparing the y-value of each module's origin to the arbitrarily defined ground at y=2.

3.2 Control Strategy

A pentagon resting on ground has 2 *ground_contact* points and an upright trapezoid has three. If we continuously transforming the robot between these two stages in the same direction, we can produce a very simple planar locomotion scheme to mobilize the simulated robot. Figure 5 illustrates the main steps in the transformation process.

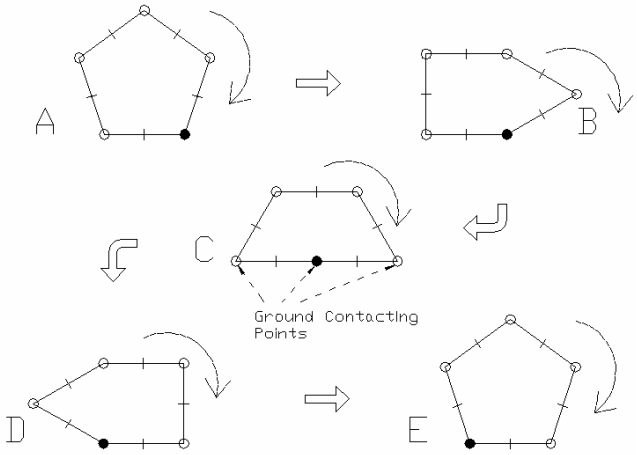

Figure 5: Transformation between a perfect pentagon and an upright trapezoid.

From stage A to B:

Starting from a perfect pentagon with 2 *ground_contact* points, the robot shifts its center of gravity to the right by moving the three modules without a *ground_contact* signal to the right.

From stage B to C:

To increase *ground_contact* points to three, the robot lowers down the front tip to form an upright trapezoid.

From stage C to D:

In order to keeps the robot rolling to the right, stage C

must transform again into a perfect pentagon. The robot will give up one *ground_contact* at the back, adjust its *local_angle* and shift center of gravity towards the front.

From stage D to E:

While keeping the two *ground_contact* points unmoved, all modules adjust their *local_angle* to equal value (i.e. 72 degrees) to become a perfect pentagon again.

One complete cycle of the above transformation process is considered as one step.

3.3 Blackboard Communication

Following Section 3.1 that describes the basic design of the simulated robot and the setup of the simulation environment; and Section 3.2 that describes the control strategy, this section concentrates on how DBB system is used in our simulation to control the movement of the robot and to facilitate communication between modules. In contrast to the board overview of the entire system in section 3.1, and 3.2, here we will focus on the operation at local level to see what exactly each module is doing and what information is being exchanged.

Assuming all modules has already come to a conclusion that they are connected in a closed loop. If stage A (Figure 5) is to roll in a clockwise direction to the right and transform itself into stage C; all modules with a *ground_contact* signal will post a message on the blackboard to see which module is the most appropriate to initiate the move. The knowledge source of all eligible modules will inform their associating processors to take the initiative if their module is the closest to the intended direction of motion. In this case module_1 (see Figure 4) will take the initiative. Processor_1 of module_1 will retrieve parameters of stage C (i.e. the *ground_contact* signal, *local_angle*, and relative position within the system) from the knowledge source and compare these parameters with module_1's current parameters. By substituting these parameters into a set of preliminary rules, processor_1 will be able to list what module_1 need to do to initiate the transformation. These information will be broadcast to other processors to update their blackboards.

When module_1 begins to adjust local_angle_1, processor_1 will continuously post the latest reading of local_angle_1 and alert all other modules when the adjustment is done. Simultaneously, processor_2 of module_2 acknowledges the intention of module_1 from the blackboard, it will consult with its own knowledge source to retrieve the corresponding duty for its position in relation to module_1. While executing rules from the knowledge source (e.g. closing up local_angle_2); processor_2 will broadcast updated parameters of module_2 to other processors in turn to update their blackboards.

This process of evaluating information, posting information, and adapting to new information will keep on continuing in the remaining modules. The transformation process from stage A to C will stop when all parameters from each module are identical to the specific parameters of stage C.

The control schemes for transforming from stage A to C and stage C to E are basically identical. The only difference is in gaining or losing one contact point after the transformation. The reason to start the transformation cycle from a perfect pentagon is to provide an obvious visual check point during simulation. Any arbitrary shape such as an inversed trapezoid will work just as fine. Figure 6 lists all information broadcast on the blackboard during a transformation from stage A to C, then from C to E.

M	Information broadcast to the blackboard	Descriptions
1 5	Ground_contact_1 Ground_contact_5	Transform from A to C Bidding for initiative
1 1 1	Module_1 will Initiate Decreasing local_angle_1 Wait for Module_2 ground_contact signal	Module_1 is the front most modules in the direction of action.
1 2 3 4 5	Adjust local_angle_1 Adjust local_angle_2 Adjust local_angle_3 Adjust local_angle_4 Adjust local_angle_5	All modules adjust local_angle to facilitate Module_1's motion.
2	Ground_contact_2	Achieved 3 ground_contact Transformation from A to C completed
1 1 2	Fix local_angle_1 Module_1 give up initiative Module_2 will initiate the next transformation	Transform from C to E Pass on initiative role to module_2
5 1 2 3 4 5	Giving up ground_contact_5 Adjust until local_angle_1 = 72 Adjust until local_angle_2 = 72 Adjust until local_angle_3 = 72 Adjust until local_angle_4 = 72 Adjust until local_angle_5 = 72	Transform from 3 ground_contact (trapezoid) to 2 ground_contact (pentagon) Adjust all local_angle to 72 to form a perfect pentagon
		Transformation complete when all local_angle = 72

Figure 6: The far left column indicates to which module the information belongs to. Each blocked roll represents one round of information exchange.

4. Conclusion

Before MSR robots can realize all of its compelling promises, a flexible decentralized control system must first be available. Multiagent systems have gained much attention in recent years and have shown promising result when applied to some lattice type MSR robots. The use of distributed blackboard system to control the proposed 5-module, chain type MSR robot has successfully

demonstrated the feasibility of using a simple rule-based blackboard control system to execute simple locomotion control such as traversing of level terrain.

With the design of the particular MSR robot shown in this paper and the application of the blackboard-based control scheme, the ability of coordinating a group of agents (the module of the MSR robot) to carry out a common task by executing purely local control rules (the updating of the blackboards and performing corresponding procedures for motion commands according to the current state and constraints) and through communication between neighboring modules alone is demonstrated through MATLAB simulation.

Most of our work devoted till now has been focusing on the demonstration of using distributed blackboard control system to coordinate a small group of modules to perform simple tasks. Our next step is to exploit the exceptional learning and reasoning capacity of the blackboard systems and to integrate feedbacks from distributed sensors to detect obstacles and uneven terrain.

Reference

[1] L.S. Baum, V. Jagannathan, and R.T. Dodhiawala, (1989). The Erasmus System. In V. Jagannathan, R.T. Dodhiawala and L.S. Baum, editors. Blackboard Architectures and Applications, pages 347-370. Academic Press, San Diego, CA, 1989.

[2] R. Bisiani, and A. Forin (1989). Parallelization of Blackboard Architectures and the Agora System. In V. Jagannathan, R.T. Dodhiawala and L.S. Baum, editors. Blackboard Architectures and Applications, pages 153-178. Academic Press, San Diego, CA, 1989.

[3] H. Bojinov, A. Casal, and T. Hogg. Multiagent Control of Self-reconfigurable Robots. In Proc. of the *Intl. Conf. on Multiagent Systems.* IEEE, 2000.

[4] H. Bojinov, A. Casal, and T. Hogg. Emergent Structures in Modular Self-reconfigurable Robots. In Proc. of the *Intl. Conf. on Robotics & Automation.* IEEE, 2000.

[5] Z. Butler, R. Fitch, D. Rus, and Y. Wang. Distributed Goal Recognition Algorithms for Modular Robots. In Proc. of the *Intl. Conf. on Robotics & Automation.* IEEE, 2002.

[6] A. Castano, and P. Will. Mechanical Design of a Module for Reconfigurable Robots. In Proc. of the *Intl. Conf. on Intelligent Robotics & Systems*. IEEE, 2000.

[7] R.S. Engelmore and A.J. Morgan. Blackboard Systems. Addison Wesley, Reading, UK, 1988.

[8] V. Jagannathan (1989). Realizing the Concurrent Blackboard Model. In V. Jagannathan, R.T. Dodhiawala and L.S. Baum, editors. Blackboard Architectures and Applications, pages 85-97. Academic Press, San Diego, CA, 1989.

[9] A. Kamimura, S. Murata, E. Yoshida et al. Self-Reconfigurable Modular Robot – Experiments on Reconfiguration and Locomotion. In Proc. of the *Intl. Conf. on Intelligent Robotics & Systems.*. IEEE, 2001.

[10] V. R. Lesser, and D. D. Corkill, (1983). The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. In R.S. Engelmore and A.J. Morgan, editors. Blackboard Systems, pages 353-386. Addison Wesley, Reading, UK, 1988.

[11] J. Rice, N. Aiello, and H.P. Nii (1989). See How They Run…The Architecture and Performance of Two Concurrent Blackboard Systems. In V. Jagannathan, R.T. Dodhiawala and L.S. Baum, editors. Blackboard Architectures and Applications, pages 153-178. Academic Press, San Diego, CA, 1989.

[12] D. Rus, and M. Vona. A Physical Implementation of the Self-reconfiguring Crystalline Robot. In Proc. of the *Intl. Conf. on Robotics & Automation.* IEEE, 2000.

[13] D.G. Schwartz. Cooperating Heterogeneous Systems. Kluwer Academic Publishers, Dordrecht, Nethlands, 1995.

[14] C. Unsal, and P.K. Khosla. Mechatronic Design of a Modular Self-Reconfiguring Robotic System. In Proc. of the *Intl. Conf. on Robotics & Automation.* IEEE, 2000.

[15] C. Unsal, and P.K. Khosla. A Multi-Layered Planner for Self-Reconfiguration of a Uniform Group of I-Cube Modules. In Proc. of the *Intl. Conf. on intelligent Robots and Systems*. IEEE 2001.

[16] S. Vassilvitskii, J. Kubica, and E. Rieffel. General Reconfiguration Problem for Expanding Cube Style Modular Robots. In Proc. of the *Intl. Conf. on Robotics & Automation.* IEEE, 2002.

[17] M.J. Wooldridge. An Introduction to multiagent systems. John Wiley & Sons, West Sussex, England, 2002.

[18] M. Yim, D.G. Duff, and K.D. Roufas. PolyBot: a Modular Reconfigurable Robot. In Proc. of the *Intl. Conf. on Robotics & Automation*. IEEE, 2000.