Journal of Zhejiang University SCIENCE A
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

# Mechanical design and locomotion control of a homogenous lattice modular self-reconfigurable robot

XIA Ping (夏 平)[†1], ZHU Xin-jian (朱新坚)[1], FEI Yan-qiong (费燕琼)[2]

([1]*School of Electronics and Information Technology, Shanghai Jiao Tong University, Shanghai 200030, China*)

([2]*Research Institute of Robotics, Shanghai Jiao Tong University, Shanghai 200030, China*)

[†]E-mail: xiaping@sjtu.edu.cn

Received Sept. 12, 2005; revision accepted Dec. 23, 2005

**Abstract:** In this paper, we propose a novel, 3D, like cubic shape, modular self-reconfigurable (MSR) robot named M-Cubes. Its key mechanical components are analyzed in detail. By communicating with the neighboring modules, each unit employs its automatic lock device composed of a pin and a hole on each connection plane which can connect or disconnect with neighboring modules. The M-Cubes system consisting of many identical modules cooperates to change their connection, and then the whole structure transforms into arbitrary structure. Furthermore, we describe its locomotion control based on the driving function and the adjacency matrix which is effective for solving the computationally difficult problem and optimizing the system motion path during the self-reconfiguration process. Finally, a simulation experiment using java 3D technology, proved the new method for controlling modular robot is robust and useful.

**Key words:** Robot, Module, Self-reconfigurable, Locomotion control
**doi:**10.1631/jzus.2006.A0368 **Document code:** A **CLC number:** TP242

## INTRODUCTION

Modular self-reconfigurable (MSR) robots consisting of a set of standardized electromechanical modules which can independently and dynamically change their aggregate geometric structure to complete different task requirements. Compared with the traditional fixed architecture robot, this kind of robot has several potential advantages: versatility, adaptability, robustness, and low cost. MSR robots' ability in self-reconfiguration makes them particularly useful for applications in unstructured, unknown and hazardous environment, such as deep sea and space exploration, urban rescue, and military intelligence.

MSR robots are classified as homogeneous or heterogeneous depending on whether the robot system uses a single type of module or many. In a homogeneous robot, the position of the module in the robot defines its function. In a heterogeneous robot, the function of the module defines its position in the robot (Castano *et al*., 2002). They can also be classified as lattice-type and chain-type depending on whether their modules use substrate reconfiguration where the modules are placed on a lattice (in either 2D or 3D). Chain-type self-reconfigurable robots have a higher degree of mobility than lattice-type systems, because the degrees of freedom of chain-type robots often are less constrained than those of lattice-type systems. Lattice-type robots, on the other hand, can easily self-reconfigure and are suitable for forming various static configurations, but they have difficulty in generating motion.

Recently, MSR robots have been active researched. Several prototypes of MSR systems have been implemented by various researchers. Existing 2D systems include metamorphing hexagonal modules (Pamecha *et al*., 1997), self-repairing machine

[*]Project (No. 50305021) supported by the National Natural Science Foundation of China

(Yoshida *et al*., 1998) and Crystalline robot (Butler *et al*., 2002) moving in horizontal plane, and Inchworm (Kotay and Rus, 1997), self-organizing robot (Hosokawa *et al*., 1998) moving in vertical plane. Modular systems that can self-reconfigure and move in 3D space include Polypod/Polybot (Yim *et al*., 2000) and Conro (Shen *et al*., 2002) that combines different gaits, and robotic molecule (Kotay *et al*., 1998), self-reconfigurable structure (Yoshida *et al*., 1999), Proteo (Bojinov *et al*., 2000) and I-Cubes (Ünsal *et al*., 2001) that can change shape using neighboring modules. Most of these systems are homogeneous, with the exception of Polypod/Polybot and I-Cubes, which consist of two different types of modules.

In this paper, we propose a 3D, like cubic shape, homogeneous, lattice MSR robot called M-Cubes (Fig.1), and explain the basic design of module hardware. The topology description of M-Cubes system was referred to establish a new locomotion control method presented here based on driving function and the adjacency matrix. This method can reduce the system computation and generate a sequence of module motions which allows the M-Cubes system to trace a desired trajectory and build different 3D static structures in minimal steps.

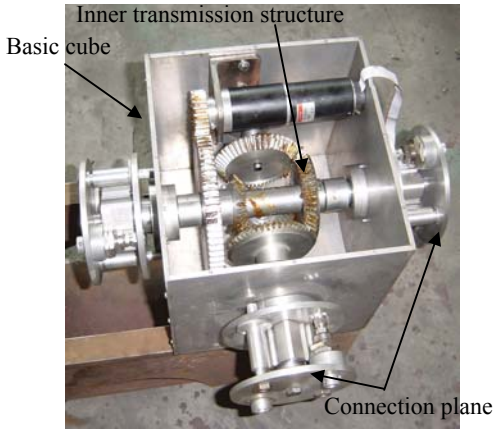


**Fig.1 Mechanical prototype of M-Cubes**

## HARDWARE DESIGN

The M-Cubes module has a basic cube at the center and six connection planes attached to its surfaces. The module can independently rotate each connection plane which has a connection mechanism by which the module can connect to or disconnect from its adjacent modules. Considering limitations of physical structure, motor power and gravitational effect, we limit use of at most one movable module in a basic motion. So there are two basic motions: a linear motion and a plane motion as shown in Fig.2. The M-Cubes module cannot change its position by itself, so it needs at least three modules to cooperate together to do the self-reconfiguration or motion in 3D space. By changing the interconnection between modules, an arbitrary cubic structure like a jungle-gym is realized. A module has 12 degrees of freedom (DOF) motion by using 12 electromagnetism clutches, 6 DOF for connection plane rotation and 6 DOF for connection plane connection.

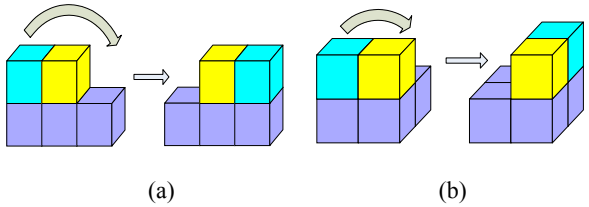


**Fig.2 Basic motions. (a) linear motion; (b) plane motion**

### Basic cube

In 3D system design, each module whose connection planes are nonsexual must have spatial symmetry. In other words, the structure and physical characteristics must be the same for output shaft of the basic cube which is directly connected to the connection plane. There are some other requirements in hardware design, such as compact and simple structure, convenient assembling, and motion transmission efficiency. Considering all the above requirements, an inner actuator and torque transmission system was designed as illustrated in Fig.3.

The details of inner actuator and torque transmission route can be described as follows: when the torque is actuated by the combination of a Maxon DC motor and a planetary gear reducer, whose output is delivered to the main shaft by two different diameters cone-shaped gears mounted at the main shaft. When the main shaft rotates, the six cone-shaped gears and four drive shafts will simultaneously rotate so it can control the output shaft to rotate or stop respectively by using an electromagnetism clutch. This transmis-

sion system can ensure that the six cone-shaped gears are free from interference and have the same rotary speed and torque of the six output shafts.

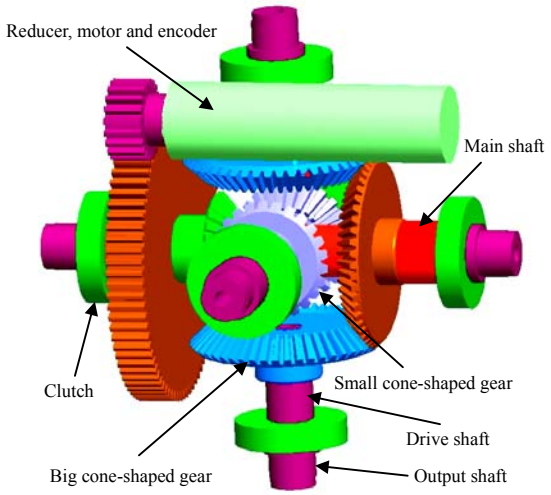


**Fig.3 Inner transmission structure**

## Connection plane

Connection plane is responsible for connecting, disconnecting and communication with the neighboring modules. Each connection plane consists of a clutch, a screw and nut, automatic lock devices composed of a pin and hole, and some IR sensors, etc.

Fig.4 shows the structure of the pin composed of a slider, a cylinder, and some balls. There are two contrary polarity permanent magnets which are fixed on different contact surfaces respectively (contact surface A or B). The slider slides back and forth along a screw shaft connected to the main shaft or the drive shaft through a small electromagnetic clutch.

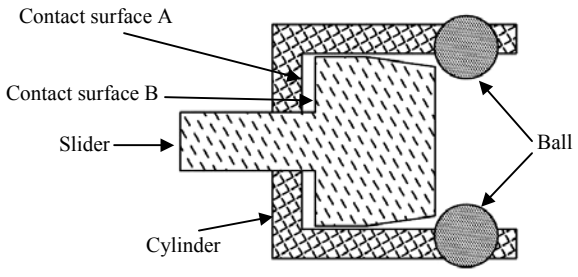


**Fig.4 Mechanical sketch of pin**

Fig.5 illustrates the motion process of the pin when two neighboring modules connect with each other. When the slider begins to slide forward, attraction between the permanent magnets draws the cylinder to slide forward together. As the front plate of the cylinder touches the hole bottom, the motion of the cylinder is blocked and stops. Then the slider will overcome the attraction between the permanent magnets and continue to slide forward. Connection completes when the slider stops its sliding and presses the balls mounted in the cylinder on the chamfer of the hole.

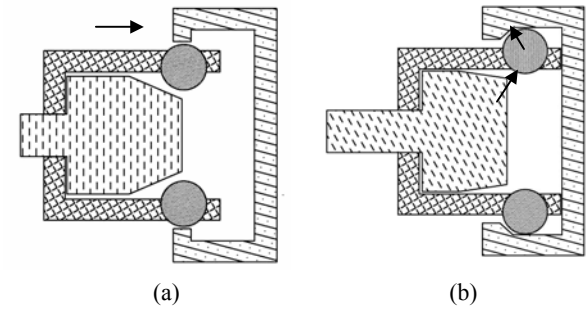


**Fig.5 Example of motion process of the pin**
(a) Slider entering into hole; (b) Lock between slider and hole

The motion of the pin is reversed when the module is disconnected. After the slider slides back, the pressure between slider and balls disappears. Then the balls go back to their default location. Disconnection completes when the back plate of the slider pushes the cylinder slide back and leaves the hole of another module's connection plane.

## DESCRIPTIVE MODEL OF THE M-CUBES ROBOT

It is important to find an unambiguous, valid and accurate module representation to reduce the M-Cubes system computation and simplify the complicacy of the control algorithm in the reconfiguration process. The representation of a modular robot must take into account that a module might have multiple connection ports so that there might be multiple ways to connect two same modules. There are some key problems in the selection of a suitable representation, for example, how to differentiate and express each connection way. Many researchers successfully apply topology theory to solve the module representation. A remarkable effort to formalize this representation was made by Castano and Will (1998). They discovered

and identified a robot configuration by using graphs and digraphs, and then applied this methodology to the Conro robot. Saidani (2004) proposed a distributed algorithm which transforms a quadruped robot into a chain by defining the notion of the graph topodynamics.

In M-Cubes system, a module is represented as a vertex, and a connection between modules as an edge. To express clearly the position of each module, a global coordinate system $\Sigma O$ must be built up. A unit length of the lattice is defined as the length between the centers of two modules. $G_i^t(x_i^t, y_i^t, z_i^t)$ is utilized to express the center coordinate of module $i$ at time $t$. We depict 6 neighboring cube area information of module $i$ by using 6 eigenvector $\boldsymbol{L}_i$ ($s_1$, $s_2$, $s_3$, $s_4$, $s_5$, $s_6$) ($s_j$ denoting eigenvalue). $s_j$ is used to define the neighboring lattice information on module $i$ in one of the directions. If the lattice is empty, $s_j$ is 0. If the lattice is an obstacle, $s_j$ is 1. If there is a module which is connected with module $i$, $s_j$ is 2. If there is a module which is disconnected with module $i$, $s_j$ is 3. If there is an invalid module which is connected with module $i$, $s_j$ is 4. If there is an invalid module which is disconnected with module $i$, $s_j$ is 5. $\lambda_i^t$ is utilized to express the number of the neighborhood module connecting with module $i$ at time $t$, called the degree of module $i$ at time $t$. In other words, $\lambda_i^t$ indicates the edge number of the node $m_i$ connecting with the other nodes.

The whole system must be kept connected in the self-reconfiguration process. Thus we can use depth first search algorithm to find the system topology structure as follow:

Step 1: When no modules are searched, system chooses one module at random to start anticlockwise searching for other modules, records 6 pieces of neighborhood lattice information, gives the module a mark and places the module in stack according to the sequence of its being searched.

Step 2: Module $i$ tries to search whether or not all its neighboring modules had been marked, if one of neighboring modules had not marked, module $i$ will give it a mark and repeat Step 2. If all neighboring modules had been marked, the algorithm will start from Step 3.

Step 3: If module $i$ finds that all its neighboring modules had already been marked, it will go out of the stack and the other modules in the stack start to search their neighboring modules, the algorithm then switches to Step 2. If the stack is empty, that is, all modules had already searched their neighboring modules, the algorithm finishes.

## MOTION PLANNING OF M-CUBES ROBOT

M-Cubes system is nonlinear system with multiple degrees of freedom. As more modules are added to the system, the possible reconfiguration of a group of modules from an initial configuration to a final configuration based on some constraints increases exponentially. This leads to computational complexity in determining an optimal or near-optimal method. So a motion planning method based on an appraisal function and the adjacency matrix is selected to give a near optimal solution. In the initial reconfiguration phase, an appraisal function is utilized to drive modules to the geometric center of the final configuration based on the local rules. When a module reaches the vicinity of the final configuration's geometric center, the appraisal function dies and the adjacency matrix will be used to compute the steps from each module position to the present goal position, and then minimal steps are selected. By repeatedly adjusting modules coincidence with unoccupied objective position, the final configuration can be completed. The geometric center $G_g$ of the final configuration and the distance $S_i^t$ between the center of module $i$ and $G_g$ at time $t$ can be expressed as Eqs.(1) and (2) respectively:

$$G_g(x_g \quad y_g \quad z_g) = \sum_{i=1}^{n} G_i^f(x_g \quad y_g \quad z_g)/n, \tag{1}$$

$$S_i^t = |x_i^t - x_g| + |y_i^t - y_g| + |z_i^t - z_g|. \tag{2}$$

The difference between $S_i^t$ and $S_i^{t+1}$ can be used as the appraisal function $f_i = S_i^t - S_i^{t+1}$. When module $i$ moves to one direction so that the value of $f_i$ is bigger, this motion direction will be more encouraged. If the value of $f_i$ is negative, this motion direction will be discouraged. In short, modules tend to move in the direction producing bigger plus value of $f_i$.

The motion priority in reconfiguration process is shown in Eq.(3)

$$\delta_i(t) = \alpha S_i^t + \beta/\lambda_i^t. \tag{3}$$

The motion priority of a module is decided by two factors: $S_i^t$ and $\lambda_i^t$. Obviously, if a module is farther from the center of the final configuration or the number of $\lambda_i^t$ for this module is smaller, this module has more motion priority. In order to avoid invalid motions, the number of $\lambda_i^t$ for movable modules is limited to at most 3. Different value of coefficients $\alpha$, $\beta$ will influence different reconfiguration sequence between modules, and then bring into being different possible motion paths and steps when the system completes the final configuration. Selection of improper value of coefficients $\alpha$, $\beta$ will induce unsuccessful configuration. As a result of our experiments, the right value of coefficients $\alpha$, $\beta$ are decided by the module number of the robot system and the distance between the start location and the goal location.

In the start phase, control of the reconfiguration consists of the following three steps. Each module (1) tests if it is movable and, if so, calculates the value of $\delta_i(t)$; (2) compares the value of $\delta_i(t)$ with neighboring modules by local communication. If the value of $\delta_i(t)$ is biggest in the neighboring modules, module $i$ calculates all the reachable positions and prepares for moving. If the value of $\delta_i(t)$ is equal to that of some neighboring modules but bigger than that of the others, the system randomly selects one of modules having biggest priority value and prepares for moving this module; (3) calculates the value of $f_i$ in all the reachable positions. If all values are negative numbers, this module stops moving. If some values are positive numbers, module $i$ moves to the position producing bigger value of $f_i$.

There will be at least one module motion at one time based on the above rules. Parallelism insures the stabilization and scalability of the system. When module $i$ reaches at most one unit distance from the center of the final configuration ($S_i^t \le 1$), the system changes the control algorithm and starts another control algorithm to complete the final configuration. In order to select minimal steps to reach the desired location, we define the following lattices space set:

(1) $S_m^t$ is the lattice space set that the configuration at time $t$ is occupying.

(2) $S_p^t$ is the unoccupied lattice space set at time $t$ that movable modules in the present configuration can reach in only one reconfiguration process. Obviously, $S_m^t \cap S_p^t = \varnothing$.

(3) $S_f$ is the lattice space set that the final configuration possesses.

Based on the above three lattice space sets, four intersections are defined as follows:

$$S_n^t = S_p^t \cap S_f,\ \ S_o^t = S_m^t \cap S_f,\ \ S_q^t = S_m^t \cap \overline{S}_f,$$
$$S_j^t = S_m^t \cup S_p^t = (S_o^t + S_q^t) \cup S_p^t.$$

Depth first search algorithm can be used to find the topology structure of $S_j^t$. The adjacency matrix $\boldsymbol{A}^t$ can be expressed as Eq.(4):

$$\boldsymbol{A}^t = (a_{ij}^t)_{K\times K}, \tag{4}$$

$K$ is the element number of set $S_j^t$. $V_i$ is the lattice space of set $S_j^t$. At time $t$, if a module at lattice space $V_i$ can reach the lattice space $V_j$ in only one step, $a_{ij}^t$ is equal to 1. If it cannot, $a_{ij}^t$ is 0. For example $V_i$ is too far from $V_j$ or there is a module in $V_j$ so that the module in $V_i$ cannot reach $V_j$ only in one step. If a module at lattice space $V_i$ cannot reach the lattice space $V_j$ under any circumstances, $a_{ij}^t$ is $\infty$. For example there is some obstacle in $V_j$. The adjacency matrix $\boldsymbol{A}^t$ is utilized to represent in how many steps a module can reach another lattice space of set $S_j^t$. The system planner only moves non-overlapped modules in set $S_q^t$ to non-overlapped module positions in set $S_f$ until the reconfiguration is complete. If $a_{ij}^t$ is equal to 1 and $V_i \in S_q^t$, $V_j \in S_n^t$, the system planner moves module $i$ to the position $V_j$. If the above condition does not exist, the system planner computes the square of $\boldsymbol{A}^t$ as follows.

$$(\boldsymbol{A}^t)^2 = (\boldsymbol{A}^t)\cdot(\boldsymbol{A}^t) = ((a_{ij}^t)^2)_{K\times K}, \tag{5}$$

where

$$(a_{ij}^t)^2 = \sum_{l=1}^{K} a_{il}^t a_{lj}^t. \tag{6}$$

It can be seen by Eq.(6) that $a_{il}^t a_{lj}^t$ is not equal to 0 under condition that only $a_{il}^t$ and $a_{lj}^t$ are 1. It means if $a_{il}^t a_{lj}^t$ is not equal to 0, the module can start from position $V_i$, via position $V_l$, then to position $V_j$. So the value of $(a_{ij}^t)^2$ denotes how many paths the module can start from position $V_i$ to position $V_j$ in two steps. If modules in set $S_q^t$ cannot reach the position in set $S_n^t$ in one step, the system planner computes $(\boldsymbol{A}^t)^2$ and tries to find if there is a path to finish the task. If there

is no such path, the system planner continues to compute $(\boldsymbol{A}^t)^3$ and so on until the module in position $V_i$ moves to the goal position $V_j$. The final configuration will be completed by repeating above reconfiguration process.

A motion simulation is shown in Fig.6 under condition that gravity, friction, inertia, etc. are negligible. The initial configuration is couch-shaped (Fig.6a). Fig.6b shows snapshots of the first reconfiguration phase. Fig.6c is a snapshot showing that a module reaches at most one unit distance from the center of the final configuration. Fig.6d shows the final configuration. Simulation experiment results proved that above control algorithm is highly efficient.

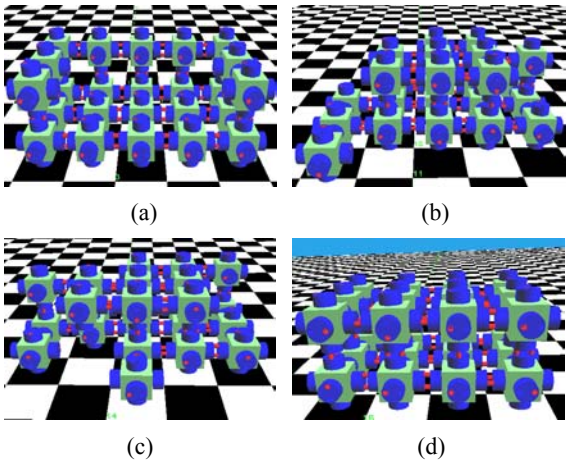


**Fig.6 Simulation of motion**
(a) The initial configuration; (b) The first motion phase; (c) The second motion phase; (d) The final configuration

## CONCLUSION

This paper presents the novel self-reconfigurable robot here. The inner structure and basic motion of M-Cubes are presented. We describe the global topology of system by local sensing information on each component module. The model can express information on modules position, connecting relationship and failure modules. We show a new locomotion control method based on driving function and the relevant adjacency matrix. The method provides a shortest path from initial configuration to final configuration based on the local rules under various surrounding environment and the module's physical structure limit.

## References

Bojinov, H., Casal, A., Hogg, T., 2000. Emergent Structures in Modular Self-reconfigurable Robots. Proceedings of the IEEE International Conference of Robotic and Automation, San Francisco, p.1734-1741.

Butler, Z., Fitch, R., Rus, D., 2002. Distributed control for unit-compressible robots: goal-recognition, locomotion, and splitting. *IEEE/ASME Trans. on Mechatronics*, **7**(4):418-430. [doi:10.1109/TMECH.2002.806230]

Castano, A., Will, P.M., 1998. Representing and Discovering the Configuration of Conro Robots. Proc. of the 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea, p.860-865.

Castano, A., Behar, A., Will, P.M., 2002. The Conro modules for reconfigurable robots. *IEEE/ASME Trans. on Mechatronics*, **7**(4):403-409. [doi:10.1109/TMECH.2002.806233]

Hosokawa, K., Tsujimori, T., Fujii, T., Kaetsu, H., Asama, H., Kuroda, Y., Endo, I., 1998. Mechanisms for self-organizing robots which reconfigure in a vertical plane. *Distributed Autonomous Robotic Systems*, **3**:111-118.

Kotay, K., Rus, D., 1997. Self-reconfigurable Robots for Navigation and Manipulation. Proc. Intl. Symp. on Experimental Robots, p.621-632

Kotay, K., Rus, D., Vona, M., McGray, C., 1998. The Self-reconfiguring Molecule: Design and Control Algorithms. Proc. of the Workshop on the Algorithmic Foundations of Robotics, p.376-386.

Pamecha, A., Ebert-Uphoff, I., Chirikjian, G.S., 1997. Useful metrics for modular robot motion planning. *IEEE Trans. on Robotics and Automation*, **13**(4):531-545. [doi:10.1109/70.611311]

Saidani, S., 2004. Self-Reconfigurable Robots Topodynamic. Proc. of the 2004 IEEE International Conference on Robotics and Automation, New Orieans, p.2883-2887.

Shen, W.M., Salemi, B., Will, P., 2002. Hormone-inspired adaptive communication and distributed control for Conro self-reconfigurable robots. *IEEE Trans. on Robotics and Automation*, **18**(5):700-712. [doi:10.1109/TRA.2002.804502]

Ünsal, C., Kiliççöte, H., Khosla, P.K., 2001. A modular self-reconfigurable bipartite robotic system: implementation and motion planning. *Autonomous Robots*, **10**(1):23-40. [doi:10.1023/A:1026592302259]

Yim, M., Duff, D., Roufas, K.D., 2000. PolyBot: a Modular Reconfigurable Robot. Proc. IEEE Conf. on Rob. & Auto., San Francisco, p.514-520.

Yoshida, E., Murata, S, Tomita, K., Kurokawa, H., Kokaji, S., 1998. Experiment of self-repairing modular machine. *Distributed Autonomous Robotic Systems*, **3**:119-128.

Yoshida, E., Murata, S., Tomita, K., Kurokawa, H., Kokaji, S., 1999. An experimental study on a self-repairing modular machine. *Robotics and Autonomous Systems*, **29**:79-89. [doi:10.1016/S0921-8890(99)00040-8]