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

Self-organized architecture for outdoor mobile robot navigation

ZHANG Huan-cheng (张焕成)[†], ZHU Miao-liang (朱淼良)
(*School of Computer Science, Zhejiang University, Hangzhou 310027, China*)
[†]E-mail: amr@cs.zju.edu.cn

Received Oct. 1, 2004; revision accepted Dec. 15, 2004

Abstract: This paper proposed a multi-agent based architecture for outdoor mobile robot navigation where event-driven control is used to handle the dynamically changing of the environment. With the support of a distributed communication infrastructure and an event-driven situation evaluation agent, the robot can initiate action adaptive to the dynamical changes in the environment through reorganize its internal architecture. Adaptiveness and feasibility of the proposed architecture is validated through navigation experiments on the robot in a variety of natural outdoor environments.

Key words: Mobile robot, Architecture, Multi-agent
doi:10.1631/jzus.2005.A0583 **Document code:** A **CLC number:** TP242

INTRODUCTION

Autonomous mobile robot is a machine able to extract information from its environment and use knowledge about its world to move safely in a meaningful and purposive manner, it can operate on its own without a human directly controlling it (Ehlert, 1999). The primary task of a mobile robot is environmental navigation as basis for more useful tasks. In contrast to indoor mobile robots and industrial manipulation robots both working in relatively static and structured environment, outdoor mobile robot works in dynamic and unstructured environment, to navigate which, the robot must be able to perceive its surroundings through different kinds of sensors and initiate appropriate actions in that environment through actuators to achieve its designed goals. The robot must also deal with the uncertain and incomplete knowledge of its environment and the effects of its own actions and has the ability to respond to potentially dangerous situation in real-time while maintaining enough safety and performance. This requires that an architecture is proposed to integrate the sensing, planning, driving and other functional parts of the robot to form a coherent system.

The robot architecture is a key issue in the design of a mobile robot. Architecture defines the principles involving organizing hardware and software function modules, integration methods and supporting tools (Alami *et al*., 2000). Under certain given resources, robot architecture design must decide whether the mobile robot can react timely and correctly to undefined, unstructured dynamically changing environment and to achieve the designed purposes.

Since mobile robots were invented in the 1960s, the mobile robotics research community had paid much attention to the development of different control architectures. Three main classes of control architectures have been developed. One is the functional decomposition based architecture; the other is behavioral decomposition based architecture; and the third is the hybrid architecture that combines the former two.

Functional decomposition based architecture, also called sense-plan-act architecture, is a top-down approach to build robotic systems based on the assumption that on the highest level, an abstract model of the world exists. Sensory processing initializes and maintains this world model through combination and integration of information from different sources. Decision is made based on the world model, and then is translated to commands to executors through different layers. The entire control task of a mobile robot is divided into subtasks, which are then completed by separate functional modules. The flow of information is used as the main guideline for the decomposition of the robot system. Information flows from the sensors to a series of perception and modeling processes, via a reasoning or decision making process, through a series of forward control processes, such as navigation and motor control, to the actuators. Earlier mobile robots, such as Shakey from SRI (Nillson, 1984) and Navlab from CMU (Shafter *et al*., 1986; Goto and Stantz, 1987; Goto, 1993) were based on this approach. The advantages of functional decomposition based architecture are that prior knowledge and global knowledge can be used to produce reasonable intelligence. It also facilitates decomposition of the whole design task and assignment of the subtasks to different research groups that are specialized in their specific domains, such as sensory processing, motor control, etc. Using multiple sensors has the following advantages: Redundant information can be used to decrease the uncertainty about the environment; complementary information can be used to extract features not available by a single/type sensor. While the disadvantages are that all the sensory data must be fused and all tasks must be coordinated before any action can be taken, and thus introduces unnecessary delays, the system tends to be as slow as its slowest sensing process, a troublesome property for a real-time system such as an autonomous robot. The system based on this architecture does make out a plan, but before it can be used, the environment has changed dramatically. Such decomposition is also unreliable if any part of the system is functioning incorrectly. The interfaces between adjacent layers are restricted and must be defined first in the design time, but it is not always possible.

In contrast, behavioral decomposition based architecture (Rosenblatt and Payton, 1989; Sowmya, 1992; Watanabe *et al*., 1992; Fayek *et al*., 1993; Eustace *et al*., 1994; Neves and Oliveira, 1997; Hwang and Ju, 1998) got popular since Rodney Brooks published has subsumption architecture in 1986 (Brooks, 1986). It is based on the assumption that "On the lowest-level algorithms can be found that are able to combine and integrate the steering commands from different sources." The module containing the algorithms is commonly called "arbiter". Controllers are active at the same time. Behavior is a module that accepts inputs from sensors and generates control commands to executors. Therefore, each of them is capable of producing meaningful action, which in turn can be composed to form levels of competence. It is a bottom-up approach to build a system gradually from a very low level. Successive levels can be added incrementally in order to enhance the functionality of the robot. The research on behavior-based architectures mainly focuses on how to design robust behavior-producing modules and how to coordinate activities of several modules. One coordination method is competition, such as Brooks' hardwired subsumption. It can be viewed as a winner-takes-all strategy: the single response in the winning behavior suppresses all the others and is directed to the robot's actuators for execution. The second is cooperative method, i.e. behavior fusion; it provides the ability to use the output of more than a single behavior at a time (Hwang and Ju, 1998). The main problem in behavior control is that it is hard to design the arbitration mechanism between primitive behaviors in order to execute a complex task. Besides, it cannot make better use of a priori knowledge and global knowledge. Behavior-based architectures only rely on the feedback to immediate sensory inputs, they do not implement fusions of the information from different sensors, not integrate the sensor feedback with a priori knowledge. Since multiple behaviors are involved and each behavior only makes use of part of the sensor data, potential collision between these behaviors arises. Behavior-based system lacks global reasoning and planning. They only work well in a specific environment with limited set of behaviors. As the environment is changed, the fixed arbiter can hardly adjust to these behaviors. Extended behavior-based robot systems were developed to deal with this problem. In these kinds of system, reasoning and planning are used to choose the most appropriate behavior for carrying out the current task. Instead of activating all the behaviors, only relevant behaviors are used and command fusion is still needed. Based on the environment's current state, dynamic arbitration is used to decide the criteria for selecting the right commands to executors. Robots design based on this architecture are responsive especially to emergent situations. Controllers in the lowest level can operate independently of one another, which make this approach robust and easy to extend. A single controller's malfunction only degrades the overall system performance a little. Addition or removal of a controller only affects the arbiter.

Some researchers try to combine the advantages of functional decomposition based and behavior-based architectures and propose the hybrid architecture (Pons *et al*., 1993; Ollero *et al*., 1994; Chung *et al*., 1998; Park *et al*., 1999; Cai *et al*., 2000; Albus, 2002; Low *et al*., 2002). The original intention of hybrid is to allow the abstract symbolic plan to modify the execution of low-level behaviors based on the goal and changes of the environment. In recent years, agent technology was introduced to the robotics area for developing the multi-agent based architecture for mobile robot navigation (Shyu *et al*., 1998; Brzykcy *et al*., 2001).

We introduce a multi-agent based self-organized control architecture based on our former research on outdoor mobile robot navigation (Zhu *et al*., 2000). The architecture is based on the hypothesis that clear awareness of the environmental situation is important for initiating intelligent behavior. Based on this hypothesis, we developed a distributed cooperative multi-agent based control architecture, in which agents cooperate to initiate different actions according to different environments. A priori knowledge and global knowledge can be exploited through a special agent called Bulletinboard (detailed in Section 4), which collects the internal and external events to make a situation evaluation, which will then guide the cooperation of agents.

The control architecture is illustrated in Section 2. ROBIX–the infrastructure for agents' communication is described in Section 3. Section 4, self-organized control is explained. Simulation and experimental results are presented in Section 5 and conclusions are given in Section 6.

MULTI-AGENT BASED ARCHITECTURE

Agent modelling

In the field of robotics, agent can be viewed as a software and/or hardware entity that can perceive its environment via sensors and act upon its environment through actuators. In a multi-agent based architecture, the robot carries out its missions through the internal agents' communication and cooperation. In the proposed architecture, an agent is defined as a software and/or hardware entity that has inputs, outputs, and certain capabilities to solve certain problems. A simple agent is composed of three parts: input/output interfaces and internal data and functionalities, as shown in Fig.1.

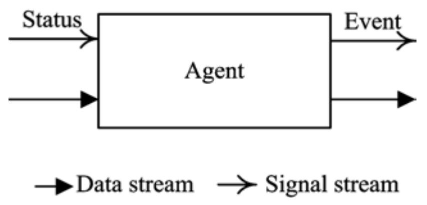

Fig.1 Simple agent

Input/output interfaces get information and system status from the environment or other agents, through processing via internal functionalities, output information usable to other agents and report events found during data processing. The running cycle of an agent is as the follows. First, an agent gets system status from the Bulletinboard agent (detailed in Section 4), based on the system status, it checks an internal look-up table or FSM (finite state machine) to decide which functionality to choose to operate on the input information and which agents to communicate with. It then processes information coming from the input interface and output results via output interface to the destination agent, and at the same time reports events found during the processing to the Bulletinboard. An agent has three states: running, sleeping, waiting. When an agent is running, it gets system status and processing data information, reports events; when it is sleeping, it only receives system status and awakes on demand; when it is waiting, it receives status and reports events, but the data outputs are suppressed. For example, the LPL agent's look-up table is illustrated in Table 1 (System states in the table are defined in Section 4).

Agents list

In the proposed architecture, agents are divided according to their functionalities as follows:

(1) Sensory agents

RFCCD (Road Following CCD): using both long and short range CCD cameras to detect the white line of structured road and edges of semi-structured

Table 1 Lookup table of LPL agent

System state	Function utilities called	Utility description
RS	*plrs*()	Straight road plan
RT	*plrs*()	Turn road plan
RB	*plrs*()	Branch road plan
RO	*plrs*()	Avoid obstacle plan on road
CO	*plrs*()	Avoiding obstacle plan cross-country
…	…	

road, then provides this extracted data of white lines and road edges to the FUSION agent.

STCCD (Stereo CCD): using stereo CCD cameras to detect obstacles on road and intraversable area off-road and provides the circumscribed polygon of the obstacle or intraversable area to the Fusion agent.

LADAR: using LADAR to detect positive obstacles on road or off-road, provides the circumscribed polygon and height of the obstacle to the Fusion agent.

Each sensory agent gets the environment information and outputs description of the environment valuable for next-step processing. A direct mapping from sensed environment to actuator may also be implemented through artificial neural network.

(2) Actuator agents

Drive: using the path given by LPL to generate locomotion and steering commands to the robot.

Sensory agents may also have actuators to pan and tilt themselves.

(3) Processing agents

Fusion: integrates information from the sensory agents as needed and gives a unique description of the environment in the field of view of the robot and provides this information to LPL agent.

GPL (Global Path Planning): based on a priori known low resolution map database, plans a global path from the start-out position to the goal position for a mission, the global path may need re-planning whenever the current planned path is found infeasible.

LPL (Local Path Planning): according to the global path given by GPL and the current description from Fusion (or directly from sensory agents when the environment is simple), plans a smoothing trajectory for the robot to follow. Depending on overall system status, it may take straight road plan, curved road plan, branch road plan or cross-country plan.

Bulletinboard: based on the reasoning rules and events from the sensory agents, actuator agents and processing agents, deduces the current state of the environment and the mobile robot.

Localization: using GPS/INS and odometer together for combined localization and orientation, to provide real-time high accuracy position (x, y, z) and stance (roll, pitch, yaw) and velocity, acceleration of the robot needed by other agents.

(4) Human-machine interface and assistant agents

Supervisor: initializes a mission, displays current environment obtained by sensors in real-time, the position and internal status of the robot. Under emergent situation dangerous to the robot or situation that the robot will get duck, gives command to the robot to a stop or awakes another agent.

TeleControl: takes control of the robot seamlessly from the robot's on-board computers. When the robot leaves the area beyond its on-board system's capabilities, TeleControl will give the control back to the on-board system seamlessly.

Recorder: records the operating data of each agent that are valuable for off-line analysis into a database.

Mapper: maintains and updates the a priori global maps used by the GPL and Fusion agents.

Agent communication system

We developed a multi-agent communication infrastructure named ROBIX, described in Section 3.

System architecture

The overall system architecture is depicted in Fig.2.

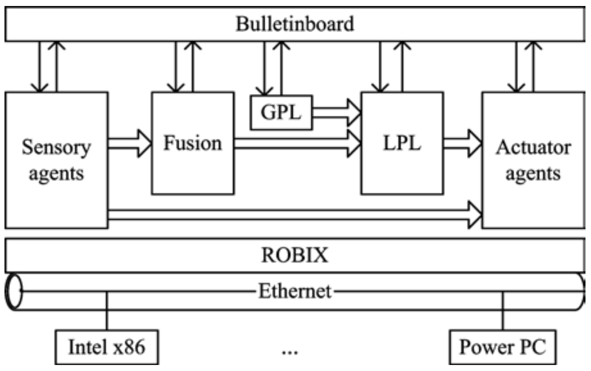

Fig.2 System architecture

In Fig.2, the data flows between agents in fact are through ROBIX, and for the sake of briefness, not all the agents are drawn in the figure. The hardware platforms include several Intel Xeons, Power PC single board computers connected with 100Base-T LAN, and the operating systems include Microsoft Windows 2000, Red Hat Linux 9.0 and VxWorks. ROBIX hides the hardware, software and location dependencies, providing transparent distributed communication for the agents. There are three types of messages in the system. One is the data information describing the images, maps, paths, etc., which is successively processed by sensory agents, fusion agent, plan agent and actuator agents. The other is event from the evaluation of an agent to its processing results, which is coded and reported to the Bulletin-board agent. The third is the current system status, which is the evaluation result of the Bulletinboard to the robot's current situation, it is then broadcasted to all other agents. So there exist two different data/control streams, making up the circuit of sense-plan-act, while the events/status stream makes up state-decision-action circuit, the latter is a super-vising circuit to the former. Through separation of data stream and events/status stream, there is no communication bottleneck in the architecture.

DISTRIBUTED COMMUNICATION INFRASTRUCTURE

Multi-agent system needs agents to communicate and cooperate in order to achieve the system goal. As our agents are distributed on different computers running different operating systems, a communication infrastructure is needed to enable the agents to make transparent, i.e. hardware and operating system and location independent communication. ROBIX–a real-time communication infrastructure based on TCP/IP protocol, has been implemented, as shown in Fig.3.

The communication infrastructure is based on a postoffice model, mainly includes another two agents deployed on each computer, namely Sndagent to send message out and Recvagent to get message in for agents in the host computer. A sender agent need not care where another agent localizes; it just puts its message into its postbox. The Sndagent collects the message from the postbox, checks the name of the destination agents, then looks up the IP address of the destination agents in the processes table shown in Table 2, and obtains the corresponding socket in the hosts table, shown in Table 3, then sends the message to the receiver agent's host through the socket. Recvagent on the receiver's host computer will pickup the message and put it in the letterbox for the receiver agent. If the message is an urgent one, the Recvagent can also interrupt the receiver agent to ask him to handle the message immediately. The Supervisor agent reads the processes table and hosts table in from the configuration file when the system starts running. The processes table and hosts' table are illustrated as follows.

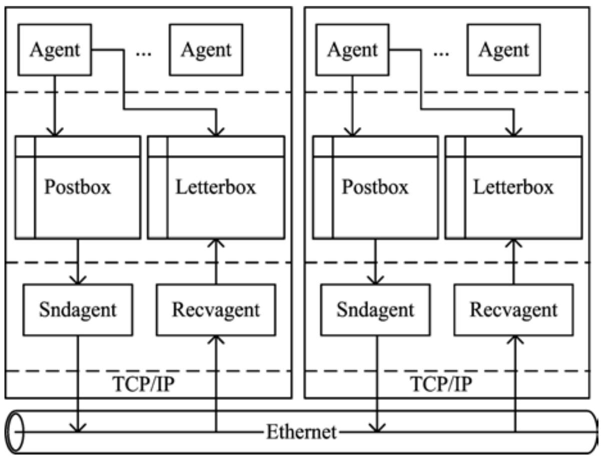

Fig.3 ROBIX communication system

Table 2 Processes table

Indicate bit	Process name	Host IP	PID
1	Fusion	192.168.0.2	1247
0	…	…	…

Table 3 Hosts table

Indicate bit	Host name	Host IP	Socket	Local
1	AMR	192.168.0.2	4	1
2	…	…		…

EVENT-DRIVEN SITUATION EVALUATION AGENT

System state

The states are defined as follows:

RS–road straight; RT–road turn; RB–road

branch; RO–road with obstacle;

CN–cross-country without obstacle; CO–cross-country with obstacle;

IT–system initializing; RD–system gets ready; GL–goal achieved;

EM–emergency; SP–stop.

Events from agents

Each agent can report the following events. The events name is denoted by abbreviation of the agent's name followed by the event's name:

RFRS, RFRT, RFRB, RFRN–road straight, road turn, road branch, and road none events from RFCCD;

LAOBS, LAOBN–obstacle, obstacle none events from LADAR;

STOBS, STOBN–obstacle, obstacle none events from STCCD;

FURS, FURT, FURB, FURO, FUCN, FUCO–road straight, road turn, road branch, road with obstacle, cross-country, cross-country with obstacle events from Fusion;

GPRS, GPRT, GPRB, GPCC–road straight, road turn, road branch, cross-country events from GPL;

PLRS, PLRT, PLRB, PLRO, PLCN, PLCO, PLF–road straight, road turn, road branch, road with obstacle, cross-country, cross-country with obstacle, failure events from LPL.

The above events reported by agents will cause the changes of system status.

Bulletinboard–the event-driven situation evaluation agent

The mobile robot achieves self-organized control by the internal agents' dynamically changing its working status and links to other agents based on the system status obtained by the Bulletinboard agent. The Bulletinboard agent deduces the system status based on events received from each agent and then broadcasts the system status to each agent through the communication infrastructure ROBIX. Then each agent can adjust its working status and input/output links to other agents according to the system status and internal event-state transition rules. Thus, the robot architecture is dynamically reorganized and the agents are combined to a final one to act purposively and coherently.

The internal structure of the Bulletinboard agent composed of a real-time reasoning system based on DES (Discrete Event System) model and events Monitor, as shown in Fig.4. The reasoning system includes a Rule base, a Facts base, and an Inference machine based on improved Rete algorithm. The events Monitor gets the events from the sensorial agents, processing agents, and actuator agents, etc., and puts them into the Facts base. Upon arrival of new facts, the Inference machine gets activated to deduce the new system status based on the rules in the Rule base. As soon as the new system status is available, events Monitor broadcasts it to all the agents in the system. Then the agents receive this broadcast and update the system status in their memory, then make a decision on what to do based on the internal looking-up table or FSM. That is, whether to work or sleep, send message to whom and receive message from whom. Thus, dynamic links between agents can be established, and the system architecture is reorganized according to the changes of the environment to implement adaptive actions.

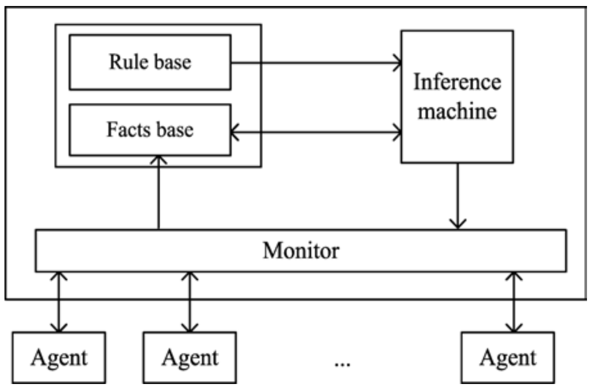

Fig.4 Bulletinboard agent

Example for self-organized control

For example, when the robot moves along a road, and the LADAR agent reports that there are not any obstacles for a certain times. The Bulletinboard agent collects these events and deduces that the system status is RS (road straight), and then broadcasts this status to other agents. Under such simple situations, sensorial fusion is not needed and the Fusion agent gets asleep, RFCCD agent directly sends results to LPL agent instead of Fusion for the robot to accelerate to high vehicle speed, as depicted in Fig.5a. Whenever LADAR agent reports there are suspected obstacles again, the Bulletinboard deduces the new system status RO {road driving with obstacle}, this new status broadcast will wake up Fusion. Fusion then requests outputs from RFCCD and LADAR to make a unique description about the environments and sends processing results to LPL, depicted in Fig.5b, thus relatively safe driving is achieved.

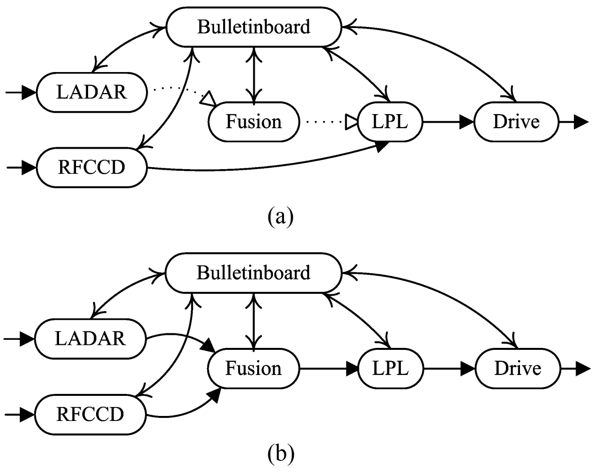

Fig.5 Self-organized control example. (a) Architecture in road following without obstacle; (b) Architecture in road following with obstacle

EXPERIMENTAL RESULTS

Our mobile robot is a refurnished automobile equipped with both long-range and short-range CCD cameras, binocular CCD cameras, two LADARs, and a GPS/INS and odometer integrated localization and orientation system. The architecture proposed in this paper has been tested by computer simulation and then outdoor field experiments for both on road and off road navigation under different weather conditions. The field environments included structured roads with clear makings, semi-structured roads paved with gravels and moderate undulation cross-country roads with sparse vegetation. Experimental results showed that the system architecture is dynamically reorganized according to the changes in the environment. In simple environments, a few agents are involved and the robot achieves high driving speed, while in complicated environments, more agents will be involved and the robot drives at relatively low but safe speed. Satisfactorily robust performance in robustness, responsiveness, and speed, compared to the autonomous mobility capabilities of the experimental unmanned vehicle DEMO III (Shoemaker and Bornstein, 2000), are achieved through this architecture.

The experimental results also showed that, through the dynamic reorganization according the dynamically changing environment, the architecture proposed in this paper have the advantages of both behavior-based architecture's responsiveness and functional decomposition based architecture's abilities to reason about the world by making use of a priori known global knowledge. These characteristics of the architecture give rise to a responsive, adaptive and intelligent outdoor mobile robot.

CONCLUSION

In this paper, we proposed a self-organized multi-agent based architecture for outdoor mobile robot navigation. According to the environmental situation and the internal status, the robot can dynamically reorganize the links between its internal agents to initiate adaptive action to the changing environment and to achieve the best performance. Simulation and field experiments results have shown that the architecture is suitable to outdoor mobile robots navigation in unknown, unstructured and dynamic environments.

References

Alami, R., Herrb, M., Morisset, B., Chatila, R., Ingrand, F., Moutarlier, P., Fleury, S., Khatib, M., Simeon, T., 2000. Around the Lab in 40 Days [Indoor Robot Navigation]. Proceedings of IEEE International Conference on Robotics and Automation, San Francisco, CA, **3**:88-94.

Albus, J.S., 2002. 4D/RCS–A Reference Model Architecture for Intelligent Unmanned Ground Vehicles. Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, **4715**:303-310.

Brooks, R.A., 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, **2**:14-23.

Brzykcy, G., Martinek, J., Meissner, A., Skrzypczynski P., 2001. Multiagent Blackboard Architecture for A Mobile Robot. Proceedings of 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, HI, **4**:2369-2374.

Cai, Z., Zhou, X., Li, M., Lei, M., 2000. Evolutionary control architecture for autonomous mobile robot based on function/behavior integration. *Jiqiren/Robot*, **22**(3):169-175 (in Chinese).

Chung, J., Ryu, B.S., Yang, H.S., 1998. Integrated control architecture based on behavior and plan for mobile robot navigation. *Robotica*, **16**(4):387-399.

Ehlert, P.A.M., 1999. The Use of Artificial Intelligence in Autonomous Mobile Robots. Research Report, Knowledge Based Systems Group, Delft University of Technology.

Eustace, D., Barnes, D., Gray, J., 1994. Behaviour Synthesis Architecture for Co-operant Mobile Robot Control. Proceedings of the International Conference on CONTROL'94, Part 1 (of 2). Coventry, UK, **1**:549-554.

Fayek, R.E., Liscano, R., Karam, G.M., 1993. System Architecture for A Mobile Robot Based on Activities and A Blackboard Control Unit. Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA, **2**:267-274.

Goto, Y., 1993. Mobile robot architecture with parallelism between and within layers. *Transactions of the Japan Society of Mechanical Engineers, Part C*, **59**(563):232-239.

Goto, Y., Stentz, A., 1987. The CMU System for Mobile Robot Navigation. Proceedings of IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, **4**:99-105.

Hwang, K.S., Ju, M.Y., 1998. Auto-agent: A Behavior-based Architecture for Mobile Navigation. SPIE Conf. Sensor Fusion and Decentralized Control in Robotic Systems, Boston, Massachusetts, **3523**:48-56.

Low, K.H., Leow, W.K., Ang, Jr.M.H., 2002. A Hybrid Mobile Robot Architecture with Integrated Planning and Control. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, p.219-226.

Neves, M.C., Oliveira, E., 1997. Control Architecture for An Autonomous Mobile Robot. Proceedings of the 1997 1st International Conference on Autonomous Agents, Marina del Rey, CA, p.193-200.

Nillson, N.J., 1984. Shakey the Robot. Technical Report 323, AI Center, SRI International, Menlo Park, CA.

Ollero, A., Mandow, A., Munoz, V., Gomez De Gabriel, J., 1994. Control architecture for mobile robot operation and navigation. *Robotics and Computer-Integrated Manufacturing*, **11**(4):259-269.

Park, J.M., Song, I., Cho, Y.J., Oh, S.R., 1999. Hybrid Control Architecture Using A Reactive Sequencing Strategy for Mobile Robot Navigation. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'99): Human and Environment Friendly Robots with High Intelligence and Emotional Quotients, Kyongju, South Korea, **3**:1279-1284.

Pons, N., Delaplace, S., Rabit, J., 1993. Mobile Robot Architecture Dedicated to Asynchronous Events Management. Proceedings of the 8th International Conference on Applications of Artificial Intelligence in Engineering, Toulouse, Fr, **2**:547-560.

Rosenblatt, J.K., Payton, D.W., 1989. Fine-grained Alternative to the Subsumption Architecture for Mobile Robot Control. Proceedings of IJCNN International Joint Conference on Neural Networks, Washington DC, p.317-323.

Shafter, S.A., Stentz, A., Thorpe, C.E., 1986. Architecture for Sensor Fusion in A Mobile Robot. Proceedings of 1986 IEEE International Conference on Robotics and Automation, San Francisco, CA, p.2002-2011.

Shoemaker, C.M., Bornstein, J.A., 2000. Overview and Update of the Demo III Experimental Unmanned Vehicle Program. Proceedings of SPIE in Unmanned Ground Vehicle Technology II, 4**024**:212-220.

Shyu, J.H., Liu, A., Hwang, K.S., 1998. Multi-agent Architecture for Mobile Robot Navigation Control. Proceedings of the 1998 IEEE 10th International Conference on Tools with Artificial Intelligence, Taipei, China, p.50-57.

Sowmya, A., 1992. Real-time Reactive Model for Mobile Robot Architecture. Proceedings of SPIE Conference on Applications of Artificial Intelligence X: Machine Vision and Robotics, Orlando, FL, **1708**:713-721.

Watanabe, M., Onoguchi, K., Kweon, I., Kuno, Y., 1992. Architecture of Behavior-based Mobile Robot in Dynamic Environment. Proceedings of IEEE International Conference on Robotics and Automation, Nice, Fr, **3**:2711-2718.

Zhu, M., Zhang, X., Wang, X., Tang, W., 2000. Computer integration system of autonomous intelligent robot self-organization structure IRASO. *Pattern Recognition and Artificial Intelligence*, **13**(1):36-41 (in Chinese).