Agent-Oriented Design of a Multi-Robot System

Bram Gruneir[1], Ben Miners[2], Alaa Khamis[2], Hamada Ghenniwa[3] and Mohamed Kamel[2]

[1] Department of Systems Design Engineering, University of Waterloo,
200 University Avenue West, Waterloo, Ontario, Canada, N2L 3G1

[2] Department of Electrical and Computer Engineering, University of Waterloo
[3] Department of Electrical and Computer Engineering, The University of Western Ontario,
1151 Richmond Street, Suite 2, London, Ontario, Canada, N6A 5B8

Abstract. This paper presents a new architecture for multiple robot systems using an agent oriented design methodology. The proposed architecture combines the hierarchical and the decentralized approaches. It splits all processes into two layers: the cognitive layer, where the higher brain functions take place and the action layer, where the low level functions take place. It also addresses the use of cooperative software agents organized in hardware or software components. Each of these agents independently handles a small set of specialized tasks and cooperates to achieve system-level goals. The overall system behaviour emerges from the autonomous behaviours of the individual agents. Advantages include support for multiple robots with different specifications to communicate with each other and perform meaningful tasks. Experiments using mobile, autonomous robots equipped with a vision system demonstrate the usefulness of the proposed architecture in the development of multi-robot cooperative behaviours.

1 Introduction

Intense research activities have been conducted over the last decade to develop multi-robot systems capable of performing robust cooperative work. In the context of multi-robot systems, cooperation is defined as the situation in which several robots operate together to perform a global task that either cannot be achieved by a single robot, or whose execution can be improved by using more than one robot, thus obtaining higher performance [1].

Agent technology is one of the most important paradigms that can be used to facilitate the development of multi-robot systems. Agents represent a fairly new way to conceptualize and implement software applications. Linguistically, an agent is an autonomous entity with an ontological commitment and agenda of its own. A software agent is a computational entity that is not limited to react to external stimuli, but is also able to start new communicative acts autonomously to accomplish a given task. A software agent is perceptive; it is able to perceive and respond to changes in its environment. A software agent is autonomous; the agent is capable of operating as a standalone process and performing actions without user intervention.

Wooldrige and Jennings define a software agent as a computer system that is capable of autonomous action in its environment to meet its design objectives [2]. They describe

Gruneir B., Miners B., Khamis A., Ghenniwa H. and Kamel M. (2005).
Agent-Oriented Design of a Multi-Robot System.
In *Proceedings of the 1st International Workshop on Multi-Agent Robotic Systems*, pages 132-143
DOI: 10.5220/0001196601320143

autonomy, reactivity, pro-activeness and social ability as essential properties of this computer system [2, 3]. This conceptual definition shows why these four features help explain the recent popularity and research in using agent technology in cooperative robotics.

Cooperative agents are a popular candidate for dealing with the size and complexity of multi-robot systems. This paradigm provides modularity, distribution and abstraction for the system. Problem solving techniques can also be used for adding intelligence into agents for critical tasks such as providing cognitive behaviours and learning from experience. An important question that must be answered by cooperative agents' strategies is whether to use centralized, hierarchical or distributed architectures for managing the cooperation between agents.

A centralized approach maintains a central controller that is responsible for all other individual entities. The main drawback to this approach is the large communication load because all the computational entities information have to be transferred to the central processor. The central processor also may have a high computational load as it has to handle all of the information. This approach proves not to be very reliable as a single central processor is critical to the performance of the entire system.

The hierarchical or multilayer approach distributes the computational load and possibly lowers the communication load but the system is still not robust due to the presence of a single coordinator. The distributed or decentralized approach can be used to increase the robustness of the system to component failure and decrease the computation and communication load. The most important advantage is survivability. When one node or computational entity fails, the entire network can continue to function by reallocating processing load to the remaining nodes. A second advantage is reduced computational complexity on a per processor basis.

The proposed architecture allows for a hierarchical and a decentralized approach. It splits all processes into two layers: the cognitive layer, where the higher brain functions takes place and the action layer, where the low level functions take place. It also addresses the use of cooperative agents organized in hardware or software components, such that each one independently handles a small set of specialized tasks and cooperate to achieve system-level goals. The overall system behaviours emerge from the autonomous behaviours of the individual agents.

This architecture is designed to standardize both the internal and external interfaces for diverse robots. Standardizing the inner design of each robot, as well the basic way in which robots interact with each other, simplifies development and implementation. Although robots may be extremely dissimilar in both function and programming, the layered architecture minimizes the effort required to implement a variety of techniques.

The remainder of the paper is structured as follows: Section 2 presents a brief survey on agent-based architectures for multi-robot systems. Section 3 describes the proposed architecture, addressing its two layers and modules, followed by its implementation in section 4. Section 5 explains the experiments conducted to verify the usefulness of the proposed architecture. Finally, conclusions are summarized in section 6.

2 Background: Agent-Based Approaches for Multi-Robot Systems

Many agent-based architectures are proposed to facilitate the development of multi-robot systems capable of performing robust cooperative work. A reusable framework is proposed for coordinating a team of mobile robots that can accomplish high level or tightly coupled missions that could not be easily achieved using single robot solutions [7]. Their proposed framework is based on the behaviour-based architecture model for the basic control layer and agent-oriented software engineering paradigm for coordinating the team of autonomous mobile robots. Infantino et al propose a method, implemented using a FIPA compliant platform, for the design of multi-agent robot architectures including vision agents [8]. This method extends the classical behaviour-based approach and is used in the design and implementation of a robot vision system based on agent inserted in a generic multilevel architecture.

A collective robot framework is presented in [9], where a team of heterogeneous communicating mobile robots, operating without a supervisor and without a centralized control of their behaviours, adapt the collective behaviour during runtime in the presence of a changing environment. The innovative aspect in this approach rests on a system integrating communication as an active and dynamic component in the adaptation, and not only as a static part of the robot's interactions. While several research groups are investigating the development of multi-agent based architectures for multi-robot systems, as mentioned above, the proposed approach presented in this paper is unique by its focus on the inner design of each robot. This is done by making both the design of each robot, as well the way in which the robots interact with each other, a standardized system. This standardization allows robust designs even though the robots may be extremely dissimilar in both function and programming. The proposed architecture will allow a variety of techniques to be implemented with minimal difficultly.

3 Architecture Description

Many issues arise when designing a multi-robot system such as autonomy, cooperation, communication structure and coordination. Collective autonomy refers to the ability of the robots to work autonomously without human intervention. Cooperation is the ability of the robots to work with each other and requires communication whenever the robots' actions depend critically on knowledge that is accessible only from another agent. Coordination addresses the interdependency management among the cooperative robots to achieve a common goal.

All of these issues can be addressed by using an agent oriented approach. Taking into account that the system deals with physical robots, not simulated ones, a completely agent-based solution is difficult due to the lack of low level control in agent-based languages. In addition, if the agents do not exist within the robot, having the low level controls reside in the agent would not be practical. Based on the Physical Robot Agent (PRA) concept described in [10], a new architecture has been developed as shown in Fig. 1. The two layers of the proposed architecture are the Action Layer, which handles all the sensory and movement functions; and the Cognitive Layer, which handles the decision making.

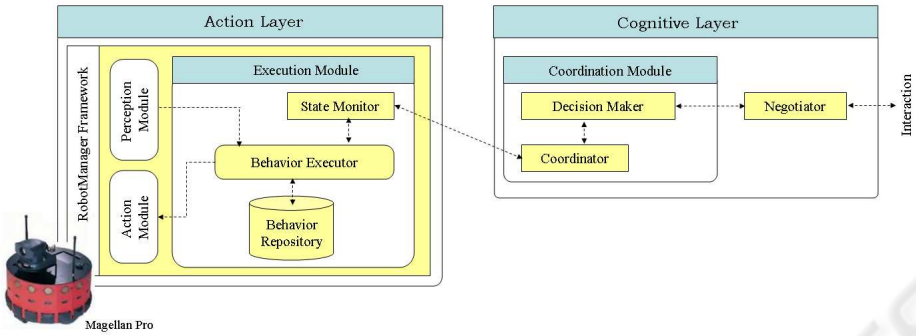

Fig. 1. Proposed Architecture

This layering system is inspired from ethology, the science of studying animal behaviours, where the cognitive layer represents the conscious brain and the action layer represents a combination of both the body and the unconscious brain. For example, when controlling a limb, the action layer understands the inner workings of the movement as well as the touch and heat sensors, but the overall goal is described by the cognitive layer. Although these layers are only abstractions, the way they are implemented affects the overall structure of how the robot functions. The following subsections describe these two layers and the inter-layer communication.

3.1 Action Layer

The action layer is where the actual physical actions of the robot are controlled. In this layer, tasks or reactions are executed. These tasks are simple programs that are controlled from the cognitive layer. The action layer consists of a three key elements: the Executor, the Repository and the State Monitor as shown in Fig. 1.

The Executor controls every aspect of the physical operation of the PRA. It tells the action modules what to do and receives feedback from the perception modules. The State Monitor tells the executor what tasks it should be performing and the Repository tells it how to perform these tasks. In return, the Executor informs the State Monitor of all updated variables. The Executor is the only element in the PRA that has access to sensors and actuators through the perception and action modules as shown in Fig. 1. This part of the abstraction is critical, as it ensures that all physical manifestations are controlled in the same element. Without the Executor, a robot will not be able to interact with its surrounding world. The types of tasks that are required to implement the Executor include:

- Initializes which are run once, at start-up, to initialize some aspect of the robot, e.g. the pan and tilt of a camera;
- Actions or steps which are run as part of a state machine;
- Alerts which are run and sit idle until a specific condition occurs; and

– Reflexes which quickly react to a situation without consulting the cognitive layer, such as crash avoidance.

The State Monitor is the Action Layer's communication channel. It is the only way in which the Action Layer can communicate with the Cognitive Layer. The complex nature of all the interactions between the layers gives rise to an element in both layers designed to handle the intricacies of these communications. For the Executor to run tasks, it must know which ones are required. And in order for the Cognitive Layer to have a full picture of the current status of the PRA, the State Monitor must both package the data and inform it of the updates.

As the State Monitor is the element in the Action Layer that relays information between the two layers, it must be able to both send and receive data from the Cognitive Layer. The State Monitor must be able to inform the Executor of the tasks that need to be run, all of the variables and constraints for those tasks and how and when to run them. It might also need to find the index or location of tasks found in the Repository, or even update or alter the stored tasks. It must also send the results of these tasks (if any) and updates on a list of internal variables as requested by the Cognitive Layer.

3.2 Cognitive Layer

In the Cognitive Layer all high level decisions are made. It is the Cognitive Layer that makes the PRA autonomous. The Cognitive Layer receives status updates from the Action Layer and uses these updates to determine which course of action to pursue. Even with the vastly different requirements for different systems, three main elements are always required in the Cognitive Layer: the Decision Maker, the Negotiator and the Coordinator as shown in Fig. 1. The Cognitive Layer is where the agents reside as two of these key elements are already included within an agent (by definition), the Decision Maker and the Negotiator. Due to these pre-existing components, an agent oriented solution is the most likely course of action.

The Decision Maker is the main and the most important element of the PRA. It is the thinker, the problem solver, the higher brain. Everything that occurs within the PRA of any consequence must go through the Decision Maker. Without this element, the robot would not be autonomous, would not be able to adapt to new situations and would not be able to form any consensus with other robots. From the Decision Maker, commands are sent to the Action Layer via the Coordinator and all inter-robot communications are facilitated through the Negotiator.

The Negotiator enables communication between robots and helps them interpret responses. Assuming an agent oriented design is already being used, this element is already included in every agent, based in the definition of an agent. The Negotiator must have the ability to communicate with other robots and to understand what is being communicated. The Negotiator does not only relay messages, but can also perform any bidding or handshaking required for decisions and consensus building.

The Coordinator is the element that is in charge of communications with the Action Layer. This is the Cognitive Layer's version of the Action Layer's State Monitor. It only receives communications from the Action Layer and the Decision Maker. It maintains this single pathway of communication; it ensures that the Decision Maker is always in

charge of all aspects of the PRA. The primary reason that a separate element is required for the Coordinator is to emphasize that this pathway is very strict.

3.3 Communication

Cooperation requires communication whenever a robot's actions depend critically on knowledge that is accessible only from another robot. The communication structure of a group determines the possible modes of inter-agent interaction. These modes of interaction are sometimes classified into Interaction via Environment, Interaction via Sensing and Interaction via Communications [11]. In the proposed design, the Interaction via Communications is adopted. As shown in Fig. 2, there are two types of communications in the proposed system: inter-layer communication and inter-agent communication.

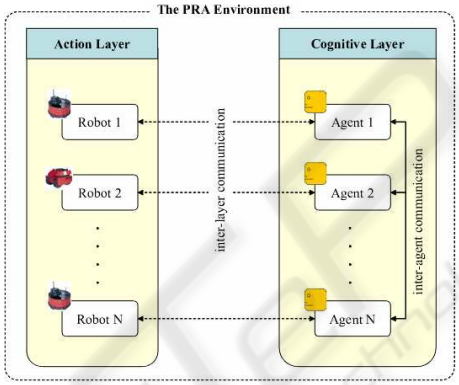

Fig. 2. Communication Links

The communication between layers is important if this architecture is to be used for real-time applications. The performance of the robots will be limited by either communication or processing speed. If a bottleneck exists within the communications, there are only two locations where this can occur. The first is within a PRA between the two layers, where all communication should be minimized. The second location is between PRAs, in which network speed and traffic are the main culprits for slower speeds.

Communication is essential for multiple agents to work together. Agents must be able to communicate between one another to locate other agents and talk to them. For this, two main components are needed, a communication protocol for inter-agent communication and a network protocol in which to send the messages. The most important aspect of inter-agent communications is that it should be limited to as few and as small messages as possible. If there are too much data being sent or there are too many messages being sent, the load they create can severely slow down a system. It is here that network constraints define the total amount of communication that can occur. Inter-agent communications should be concise yet meaningful. This will help reduce the amount of network traffic.

4 Architecture Implementation

Ideally, if a large number of known action layer tasks exist, then it is possible to write the *brain* behind a robot without having to re-write any of the lower level functionality. This simplified development process is supported with a Repository of useful tasks. The cognitive layer can only ask the action layer to do the following: initiate a task; cancel a task; watch a variable; stop watching a variable and assign a variable. This intentionally limits the abilities of the cognitive layer to help keep the abstraction between high and low level tasks. To enable the cognitive level to communicate with other robots yet still be independent, autonomous and cooperative, an agent-oriented solution is the natural course.

To create the cognitive layer, a number of software agents are used. Each robot can have as many agents as required acting collectively as the cognitive layer of a particular robot. Currently, only two agents per robot are used. The first agent is the main *brain* of the cognitive layer. It encompasses both the Decision Maker and the Negotiator. A second agent acts as the Coordinator. This Coordinator agent acts as the intermediary between the Decision Maker and the action layer of the robot. The reasoning behind this two agent system is simply that every robot requires a Coordinator and incorporating it into the main agent unnecessarily complicates the matter. In this manner, every robot can has exact same Coordinator agent, no matter how the rest of the cognitive layer is designed.

Each robot may have completely different agents or they may all be the same. This depends entirely on the project. For the most part, the Coordinator should be independent from the project. All of these agents, including the required infrastructure can be placed on one or multiple servers and they need not actually reside on the robots themselves. This enables robots with less processing power to still take advantage of the proposed multi-level architecture. This architecture is designed to be independent of any specific goal so that it can be tailored to satisfy each project's requirements.

The action layer serves two important purposes; to abstract variations in physical hardware from the cognitive layer and to carry out local time-critical tasks. Abstracting the hardware in this layer is an approach to allow the same action logic to be carried out on several different hardware platforms. Latency is minimized using an event-driven approach to ensure appropriate tasks or reactions are carried out for each external stimulus.

The action layer communicates with physical robot hardware through an abstraction interface. This interface maps each received sensor value to a specific location and orientation and translates generic motion control commands to hardware specific values. All action logic is defined using a set of simple concurrent tasks. Each of these tasks can be in one of two states as decided by the cognitive layer; passive or active. Active tasks can carry out their actions when triggered, while passive tasks do nothing until activated from the cognitive layer. Activation and deactivation of tasks is the primary method of control from the cognitive layer.

A specific precondition based on external stimuli is defined for each task. Examples of these conditions include the arrival of new a video frame, a sonar measurement, or a change in robot position. Including these preconditions outside task logic helps to keep internal logic simple and allows for a single task to easily respond to different trig-

gers or external stimuli. As soon as a task's precondition is met, the task is executed. During execution, tasks can process sensor data, control robot movement and sensor parameters in addition to exposing high-level task state as feedback to the cognitive layer. Processing sensor data locally in the action layer eliminates unnecessary communication of low-level data while ensuring relevant high-level information is available to the cognitive layer.

5 Experiments

An experiment has been conducted to validate the proposed architecture. The overall goal of this experiment is to validate the architecture, and to do so, a task was defined that could be performed using a differing number of Magellan Pro [13] robots.

The defined task requires the robots to encircle (or surround) a target. A coloured can on top of a basketball was chosen as the target of choice, as it is easy to spot from afar and allowed a good estimate of distance when up close. The can has four colours, green, blue, yellow and magenta, evenly placed along its surface. These colours are used by the robots to determine which angle they are viewing the target from. Thus, the robots have a common form of perception of the target and can discuss which one should move to where to corner it. The method with which this is achieved is dependent on the number of robots participating. If there is only one robot, it should find the front of the target. The front of the target is an arbitrary point, in this case, the line between the yellow and blue colours was chosen. If there are two robots, then they both should be 180 degrees apart from each other. With three robots, they should all be 120 degrees apart.

One important aspect of this experiment is that it must be possible to add or remove robots from the system. If one robot is attempting to look at the front of the target when another joins, the system should adjust and both robots should attempt to place themselves at 180 degrees apart. Similarly, if there are three robots encircling the target, when one is removed, the remaining two should compensate.

5.1 Coordinator Agent

For the purposes of this experiment, a Coordinator agent is implemented. This agent acts as an intermediary between the agents representing the cognitive level and the low level functions of the action layer. It accepts messages and based on a specified format, performs a variety of operations. The most important operation is the one which sends the action layer a command to initiate a task. This command is sent to the action layer through an open port that the action layer is sensitive to. This agent also can request that a variable be monitored and it can also cancel a task. Whenever a variable being monitored is altered, the action layer sends an update through a Java stub. The Coordinator sees this and sends an update to the subscribed agent. Only the variables to which an agent is subscribed are sent through the Java stub. This means that even though a task may make a variety of variables available, only the ones that are needed in the cognitive layer are actually sent. Thus, the library of action layer tasks can be used in not only a single experiment, but for future ones as well. The Coordinator agent needs

to be quick to allow for seamless operation of the robot, however, this agent should only need to be created once and then it can be used on all future projects.

5.2 Single-Robot Scenario

A single robot agent was created to test the communication between the cognitive layer and the action layer. As it is designed only for a single robot, only the Decision Maker element was included.

The single robot followed a state machine with a series of tasks. The first task is that of finding the target. The robot rotates until the target is located. The second task is for the robot to approach and then centre in on the target. Once completed, the robot prepares for pivoting by panning the camera 90 degrees while simultaneously backing off from the target. This preparation is required because the robot only has two functional wheels, so to allow a pivot around the target, it must be pointing in a tangential trajectory from the target. As well, the camera is rotated to keep the target in view. Finally, the robot calculates the fastest route to get to the 0 degree mark, the front of the target, and pivots accordingly. Once there, the robot remains on alert for any new commands or changes to the target. If at any time the robot loses site of the can, it will return to the first state and start searching for the can anew.

This agent performed well. The main problem with the single robot scenario was in the action layer where there were complications with colour detection under different lighting conditions.

5.3 Multi-Robot Scenario

The multi-robot agent encompasses both the Decision Maker element and the Negotiator element. This software agent is used on multiple robots that not only communicate with the action layer through the Coordinator agent, but also communicate with other agents on other robots. This agent builds upon the single robot version however it now coordinates its actions with the other software agents running in the cognitive layers of the other robots. The state machine of this agent can be seen in Fig. 3. This state machine is very similar to that of the single robot version. Unlike the single robot state machine, there are new steps that are designed for negotiation with other PRAs. Each of these new steps is a cognitive layer task and not an action layer one. These steps include: greeting the other agents; reporting to the other agent that this machine is ready to pivot; getting and sending the cost to travel to the different locations around the can; and telling the other robots that an error has occurred and the whole process needs to be restarted. It is important to note that none of the action layer tasks had to be altered for this new agent. For both signal and multiple robot implementations, the action layer performs the exact same tasks.

This scenario again suffered from the same lighting problems as the single robot scenario; nevertheless, the robots were able to function well as a team. A new challenge was as occasionally, a PRA saw the red color of another robot and mistook it for the basketball. This does not seem to adversely affect the final outcome of the program, as this only occurs at far distances. The robots all locate the target and encircle it while staying in sync with one another. When one robot's path is interrupted, all the other

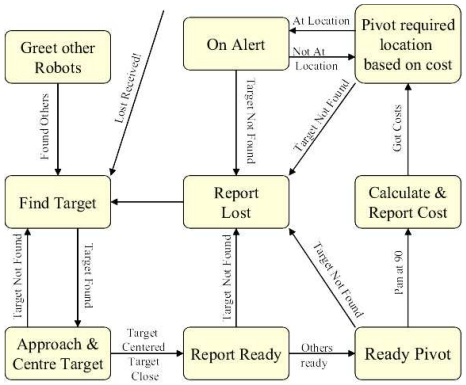

Fig. 3. Multiple Robot State Machine

robots correctly react almost instantly and reset to searching for the target. Table 1 summarizes the results obtained during this experiment and Fig. 4 is a photograph of the robots in action.

The communication between agents functioned flawlessly with low latency and the robots performed as expected. There were an acceptable number failures when the robots were operating during all of the tests, but none of these are attributed to the architecture. Failures arose due to errors in detection of the distance from the target and a robot hitting a piece of furniture or wall (as there is no detection or avoidance currently for other objects). When a failure occurred, the robots would typically reset, as previously defined in the state machine.

Table 1. Best-View Demonstration Experiment Scenario

Scenario	Number of Robots	Results
Encircle Target	1 (find front of target only)	The robots performed as expected. All three robots were tested.
	2	Any two robots performed as expected.
	3	The robots performed as expected.
Remove Robot	3 down to 2	The robots performed as expected. Choice of robot did not affect the results.
	2 down to 1	The robots performed as expected. Choice of robot did not affect the results.
Add Robot	1 to 2	The robots performed as expected.
	2 to 3	The robots performed as expected.

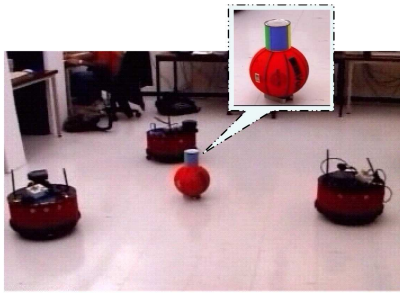

Fig. 4. Robots in Action

6 Conclusion and Future Work

A new architecture using an agent oriented design method is proposed for Multiple Robot Systems. This architecture combines the hierarchical and the decentralized approaches. It splits all processes into two layers, the cognitive layer where the higher brain functions take place and the action layer where the low level functions take place. It also addresses the use of cooperative agents organized in hardware or software components that each independently handle a small set of specialized tasks and cooperate to achieve system-level goals. The overall system behaviour emerges from the autonomous behaviours of the individual agents. The architecture is designed to make the inner design of each robot, as well the way in which the robots interact with each other, a standardized system. Although the robots may be extremely dissimilar in both function and programming, this layered architecture will allow for a variety of techniques to be implemented with almost no difficultly.

The conducted experiments show the usefulness of the proposed architecture in facilitating the development of cooperative behaviours between multiple robots. Further testing of the system presented above is being performed. Furthermore, a robot soccer team is currently being developed in which this architecture is being used. Another project is currently using the proposed architecture to examine multi-agent based remote interaction with multi-robot system. In future work, more cooperative behaviours will be developed using the proposed architecture.

References

1. F. Heylighen, ”Collective Intelligence and its Implementation on the Web: Algorithms to Develop a Collective Mental Map”, Computational and Mathematical Theory of Organizations 5(3), 253-280, 1999.
2. M. Wooldrige and N. Jennings. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 1995.
3. M. Wooldridge and P. Ciancarini. Agent-oriented software engineering: The state of the art. In: Agent-Oriented Software Engineering. Springer-Verlag, Berlin, 2001.
4. J. Altman, F. Gruber, L. Klug, W. Stockner, and E. Weippl, ”Using Mobile Agents in Real World: A Survey and Evaluation of Agent Platforms,” Proceedings of the 2nd International

Workshop on Infrastructure for Agents, MAS, and Scalable MAS at the 5th International Conference on Autonomous Agents, pp. 33-39, Montreal, Canada, ACM press, June 2001.
5. F. Bellifemine, A. Poggi and G. Rimassa, ”JADE- A FIPA-compliant Agent Framework”, CSELT internal technical report. Part of this report has been also published in Proceedings of PAAM’99, London, April 1999, pp.97-108.
6. FIPA - Foundation for Intelligent Physical Agents, http://www.fipa.org , April, 2005.
7. Lucia Vacariu, B. P. Csaba, I. A. Letia, G. A. Fodor, O. Cret, ”A Multi-Agent Cooperative Mobile Robotics Approach for Search and Rescue Missions”, IFAC Symposium on Intelligent Autonomous Vehicles, 5-7 July, Lisbon, Portugal, 2004.
8. I. Infantino, M. Cossentino, A. Chella - ”An Agent Based Multilevel Architecture for robotics vision systems” - The 2002 International Conference on Artificial Intelligence (IC-AI’02) - June 24 - 27, 2002 - Las Vegas (NV), USA.
9. S. Mostefaoui and M. Courant, ”Collective Adaptation of a Heterogeneous Communicating Multi-Robot System”, In Proceedings of the International Arab Conference on Information Technology, AICT’02, P. University of Qatar, Doha-Qatar, 16 - 19 December 2002. pp. 1038-1044.
10. J. Eze, H. Ghenniwa, and W. Shen, ”Distributed Control Architecture for Collaborative Physical Robot Agents”, In 2003 IEEE International Conference on Systems, Man & Cybernetics, Washington DC, 2977-2982. 2003.
11. Y. Cao, A. Fukunaga, and A. Kahng, ”Cooperative Mobile Robotics: Antecedents and Directions”, Autonomous Robots, 4, 1-23, Kluwer Academic Publishers, 1997.
12. J. Eze, H. Ghenniwa, and W. Shen, ”Integration Framework for Collaborative Remote Physical Agents” in proceedings of the 3rd International Symposium on Robotics and Automation, ISRA’2002, Toluca, Mexico, 2002.
13. iRobot Corporation - http://www.irobot.com/ , April, 2005.