



CD-ROM

ISR 2006
37th International
Symposium on Robotics
and



ROBOTIK 2006
4th German Conference on Robotics

May 15 – 17, 2006



AUTOMATICA

VDI-BERICHTE

Herausgeber: VDI Wissensforum IWB GmbH

Copyright and Proceedings Permission: For all copying, reprint or republication permission write to VDI Wissensforum IWB GmbH, P.O. Box 10 11 39, D-40002 Duesseldorf, Germany. All rights reserved.

© Copyright 2006 by the VDI Wissensforum IWB GmbH.
Printed in Germany.

The manuscripts appearing in this documentation comprise the proceedings of the conference mentioned on the cover and title page. They reflect the author's opinions and are published without change, in the interest of timely dissemination. Their inclusion in this publication does not necessarily constitute endorsement by the editors.

ISBN 3-18-091956-6

Introduction

Automation is increasingly becoming a key technology, not only for high-wage countries. Highly complex products, which have to be manufactured in a very short time in modern processes, and increasingly shorter product life-cycles place increased demands on the availability of the production facilities, the flexibility of the entire process and the economic efficiency of entrepreneurial activity. Competitive production is inconceivable today without the use of automation technology in conjunction with organisational measures. The robot is and remains a central tool in these solutions.

The development of robot technology continues in all partial areas and applications. Alongside the classical industrial robots, the service robot sphere has not only established itself as an independent specialisation but is constantly increasing in importance.

Conference documentation

The conference proceedings for the joint conference of the 37th International Symposium on Robotics ISR 2006 and the 4th German Conference on Robotics ROBOTIK 2006 consist of a paper copy with the digests and a CDROM with the full papers.

ISR 2006

The International Symposium on Robotics (ISR) is the world's oldest international conference in the field of robotics, with a history that dates back to the dawn of modern industrial robotics, and a tradition and influence unmatched in the field. It is a forum for presenting state-of-the-art research in robot technologies and applications from around the world.

ROBOTIK 2006

The conference ROBOTIK is the largest national conference on robotics in Germany, organised by the German association on robotics (DGR). It was established as a series of events in 2000 in Berlin and is held every two years. ROBOTIK presents the state-of-the-art, new and well-engineered applications, and visions and trends of the German robot community.

Table of Contents

	Page
Keynotes	
A pioneer's viewpoint: 40 years development of robotics in Germany <i>Walter Reis, Reis Robotics GmbH & Co. KG, Obernburg (D)</i>	1
Location bound advantages and relocation of production <i>Joachim Rohwedder, Rohwedder AG, Bermatingen (D)</i>	
Challenges of Globalization in the Automotive Industry <i>Herbert Dreher, Continental Automotive Systems, Frankfurt (D)</i>	3
Robotic – Quo Vadis <i>Roland Siegwart, Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne (CH)</i>	5
Session A1: VR simulation techniques	
Cost effective robot simulation tools for SMEs <i>Craig Lybeck, Mika Anttila, Juha Renfors, Visual Components Oy, Helsinki (SF)</i>	9
Comprehensive use of simulation techniques to support new innovative robot applications <i>Volker Miegel, Christoph Winterhalter, ABB Automation GmbH, Friedberg (D)</i>	11
The Robocoaster simulation platform, path and video generation for an authentic mars flight simulation <i>Johann Heindl, Martin Otter, Heiko Hirschmueller, Mirko Frommberger, DLR Oberpfaffenhofen, Wessling, Florian Siegert, RSS GmbH, München, Harald Heinrich, KUKA Robotik GmbH, Augsburg (D)</i>	13
Taking space robotics technology into the forest and to construction sites: Control, simulation and training <i>Jürgen Roßmann, Dortmunder Initiative zur Rechnerintegrierten Fertigung (RIF) e.V., Dortmund (D)</i>	15
Session A2: Simulation techniques for system design	
MANDY: Consistent open source tool for design, programming, visualization and simulation of robot arms <i>Wolfgang Weber, FH Darmstadt, Stefan Rothenbücher, TU Darmstadt (D)</i>	17
Speeding up software development for mobile robots with flexible partial simulation environment <i>Ilkka Kauppi, Hannu Lehtinen, Petri Kaarmila, VTT, Espoo (SF)</i>	19
Simplified programming of robot assembly cells by using motion oriented elements <i>Markus Ehrmann, Marc Seckner, Detlef Zühlke, Technical University of Kaiserslautern (D)</i>	21
Fusing realities in human-robot social interaction <i>Mauro Dragone, University College Dublin (IRL), Brian R. Duffy, Institut Eurécom (F), Thomas Holz, Gregory M. P. O'Hare, University College Dublin (IRL)</i>	23
Session A3: Object and task modelling	
Automatic 2D and 3D models reconstruction in a pre-historical cave during robot inspection <i>Tommaso Gramigna, Grazia Cicirelli, Giovanni Attolico, Arcangelo Distante, National Research Council – ISSIA-CNR, Bari (I)</i>	25
Design and implementation of an interactive object modelling system <i>Regine Becher, Peter Steinhaus, Raoul Zöllner, Rüdiger Dillmann, University of Karlsruhe (D)</i>	27
A deterministic hierarchical local planner for probabilistic roadmap construction <i>Detlef Mages, Björn Hein, Heinz Wörn, University of Karlsruhe (D)</i>	29
Context-sensitive generation of goal-directed behavioral sequences based on neural attractor dynamics <i>Claudia Grote, Gregor Schöner, Ruhr-University of Bochum (D)</i>	31

	Page
Session A4: Control architectures	
Method transfer between applied robotics and general motion <i>Joachim Strobel, KUKA Roboter GmbH, Augsburg (D)</i>	33
A versatile modular robot control architecture for sensor integrated assembly <i>Jochen Maaß, Thomas Reisinger, Jürgen Hesselbach, Walter Schumacher, Technical University of Braunschweig (D)</i>	35
Robot integrated PLC-based process control for complex path applications <i>Alexander Meißner, Dürr Systems GmbH, Bietigheim-Bissingen (D)</i>	37
How MAS support distributed robot control <i>Bianca Innocenti, Beatriz López, Joaquim Salvi, University of Girona (E)</i>	39
Session A5: Robot motion control	
Study on the robust control technology for the flexible joint of industrial robots <i>Sang-Hun Lee, Jong-Sung Hur, Hyundai Heavy Industries Co., Ltd., Yongin-si, Gyeonggi-do, Jong-Guk Yim, Jong-Hyeon Park, Hanyang University, Seoul, (KR)</i>	41
A modular real-time coordinate transformation for a redundant 9-axis industrial robot <i>Peter Löber, Rashid Nawaz, Technical University Bergakademie Freiberg (D)</i>	43
Reaching with a redundant anthropomorphic robot arm using attractor dynamics <i>Ioannis Iossifidis, Gregor Schöner, Ruhr-University Bochum (D)</i>	45
Open real-time controller for PA10 robot arm <i>Raúl Armando Castillo-Cruces, Jürgen Wahrburg, University of Siegen (D)</i>	47
Session A6: Motion planning	
A new fast motion planning approach for dexterous manipulators in 3D-cartesian space <i>Darko Ojdic, University of Bremen, Oleg Ivlev, Friedrich-Wilhelm-Bessel-Institut Forschungsgesellschaft mbH, Bremen, Axel Graeser, University of Bremen (D)</i>	49
Fast hierarchical A* path planning for industrial robots based on efficient use of distance computation <i>Björn Hein, Heinz Wörn, University of Karlsruhe (D)</i>	51
Path planning and visualisation in high dimensional configuration space <i>Martin Ruehl, Hubert Roth, University of Siegen (D)</i>	53
Fast approximated conversion of workspace distances into a free regions in configuration space of robots <i>Marcos Adrian Salonia, Björn Hein, Heinz Wörn, University of Karlsruhe (D)</i>	55
Session A7: Robot programming	
Application of wireless technologies in automotive production systems <i>Renzo Calcagno, Fulvio Rusina, Franco Deregibus, Comau SPA, Torino (I), Alberto Sangiovanni Vincentelli, Alvise Bonivento, University of California, Berkeley (USA)</i>	57
Multi-modal graphical programming of automation components in a 3D-simulation environment <i>Jürgen Roßmann, Dortmunder Initiative zur Rechnerintegrierten Fertigung (RIF) e.V., Dortmund, Michael Schluse, EFR-Systems GmbH, Dortmund, Thomas Josef Jung, Dortmunder Initiative zur Rechnerintegrierten Fertigung (RIF) e.V., Dortmund (D)</i>	59
Discrete event based framework for controlling and programming robot manipulators and virtual humans <i>Jürgen Roßmann, Dortmunder Initiative zur Rechnerintegrierten Fertigung (RIF) e.V., Dortmund, Michael Schluse, Christian Schlette, EFR-Systems GmbH, Dortmund (D)</i>	61

HOW MAS SUPPORT DISTRIBUTED ROBOT CONTROL

Bianca Innocenti
IIIA- University of Girona
Spain

Beatriz López
IIIA- University of Girona
Spain

Joaquim Salvi
IIIA- University of Girona
Spain

Speaker: Bianca Innocenti, Institute of Informatics and Application (IIIA), U. of Girona, Girona, E-17071 Spain.
+34 972 41 88 84, bianca@eia.udg.es

Topic: Session A4: Control architectures

Keywords: multi-agent system, distributed control, mobile robotics, coordination mechanisms.

Introduction

According to the roadmap of the PLANET network, the ultimate goal for robotic systems is to obtain completely autonomous robots capable of modifying their performance in complex and changing environments. As it is technologically difficult and potentially dangerous to build complex systems that are controlled in a completely centralized way, a distributed control system should be used to develop the robot control architecture, in order to provide mechanisms to distribute, coordinate, adapt and extend the control system of a robot. Artificial Intelligence provides with a powerful tool to develop this kind of systems: Multi-Agent Systems (MAS). Most of the applications of MAS have been applied to the coordination of teams of robots (1 robot = 1 agent). But there are less examples in the application of MAS to implement distributed control system for a single robot (1 robot = n agents). In this latter case agents are used to maintain "independency" among the different behaviours of the robot, usually grouped in two main layers: reactive and deliberative, conforming an hybrid architecture. With the MAS approach, both layers can provide alternative actions at a given time (go, stop); even at the same layer, several agents can suggest different actions (go, avoid obstacles). So the key issue in MAS for distributed control systems is the coordination mechanism that arbitrates and selects the single action to be performed at a given time according to the possible set of actions provided by different agents. The most common approach is to have a hierarchical architecture in which agents at the reactive layer have priority over agents at the deliberative one. Other non hierarchical approaches, as [1] rely also on a centralized agent that determines which action must be carried out at a given time. In [2] the definition of the MAS system control architecture is based on organization theory and strategic alliances. This structure defines how to coordinate the activities of various actors and how they depend on each other. There is no negotiation to decide upon the action to perform, but all is predefined.

Summary

Our architecture pretends to take advantages of some of the above ideas. It has been designed as an organization, in which agents have been assigned different roles regarding their responsibility on the different tasks to perform in the community. These roles fix interactions and relationships among agents, making them to cooperate to achieve goals (as for example surveillance) and to compete for the use of the different resources (as for example the robot motors). Unlike previous architectures, ours distributes coordination: coordination is performed as a negotiation among agents.

Our architecture has perception, behavioural and actuator agents. Perception agents obtain information about the environment and about the internal conditions of the robot; behavioural agents carry out specific actions, as avoid obstacles; and actuator agents are in charge of controlling the linear and angular speed of the robot interacting directly with motors. As behavioural agents the avoid, the goThrough and the goto agents have been implemented. Avoid agent, based on the information received from the sonar agent calculates the linear and angular speeds needed to avoid the collision with the obstacle. On the other hand, the goto agent calculates the linear and angular speeds necessary to move the robot to the goal position, while the goThrough agent has to compute these speeds in order to drive de robot in narrow places. As these agents are trying to use the same resource at a given time, a coordination mechanism is needed. Particularly, a peer-to-peer coordination mechanism among the agents involved in one conflict is proposed. Coordination process is carried out based on *utility* values computed by the agents in conflict. All the agents in the architecture have their own utility function, known only by themselves, but all normalized between [0, 1] (they are comparable). Agent with the highest *utility* value takes the robot control. Several experiments have been performed in a commercial mobile robot, a Pioneer 2DX. The robot achieves the target position while avoiding obstacles.

- [1] D. Busquets, C. Sierra, and R López de Mántaras. *A multiagent approach to qualitative landmark-based navigation*. Autonomous Robots, 15, pages:129-154, 2003.
- [2] P. Giorgini, M. Kolp, and J. Mylopoulos. *Socio-intentional architectures for multi-agent systems: The mobile robot control case*. Proceedings of the Fourth International Bi- Conference Workshop on Agent-Oriented Information Systems (AOIS-02) at CAiSE2002, Toronto, Canada, 2002.





VDI

© VDI Verlag GmbH
Düsseldorf 2006

Proceedings of the
Joint Conference on Robotics

E AUTOMATICA

and

ROBOTIK 2006
4th German Conference
of Robotics

ISIRI 2006
37th International
Symposium on Robotics

ISIRI
International
Symposium
on Robotics
and
Intelligent
Systems

Proceedings of the Joint Conference on Robotics



ISR 2006
37th International
Symposium on Robotics
and



ROBOTIK 2006
4th German Conference on Robotics

May 15 – 17, 2006



AUTOMATICA





www.vdi.de

© Copyright VDI Verlag GmbH
Düsseldorf 2006

Subject: Control architectures

Control architectures

Conference: 1956/2006

Subject	Authors	Date	Conf./Page	A
 Method transfer between applied robotics and general motion Abstract	Joachim Strobel, KUKA Roboter GmbH, Augsburg (D)	15.05.2006	1956/0017	3:
 A versatile modular robot control architecture for sensor integrated assembly Abstract	Jochen Maaß, Thomas Reisinger, Jürgen Hesselbach, Walter Schumacher, Technical University of Braunschweig (D)	15.05.2006	1956/0018	3:
 Robot integrated PLC-based process control for complex path applications Abstract	Alexander Meißner, Dürr Systems GmbH, Bietigheim-Bissingen (D)	15.05.2006	1956/0019	3:
 How MAS support distributed robot control Abstract	Bianca Innocenti, Beatriz López, Joaquim Salvi, University of Girona (E)	15.05.2006	1956/0020	3:

HOW MAS SUPPORT DISTRIBUTED ROBOT CONTROL

Bianca Innocenti
IIIA- University of Girona
Spain

Beatriz López
IIIA- University of Girona
Spain

Joaquim Salvi
IIIA- University of Girona
Spain

Speaker: Bianca Innocenti, Institute of Informatics and Application (IIIA), U. of Girona, Girona, E-17071 Spain.
+34 972 41 88 84, bianca@eia.udg.es

Topic: Session A4: Control architectures

Keywords: multi-agent system, distributed control, mobile robotics, coordination mechanisms

Abstract

One of the current challenges of robotics is to make completely autonomous robots capable of modifying their performance in complex and changing environments. However, it is technologically difficult and potentially dangerous to build complex systems that are controlled in a completely centralized way. So, distributed systems should be used to develop the robot control architecture, in order to provide mechanisms to distribute, coordinate, adapt and extend the control system of the robots. In this paper we present a reactive multi-agent architecture to control a mobile robot. In this architecture, agents have fixed roles and they interact in order to achieve goals. One aspect of the architecture is that agents negotiate in order to take control of the robot and, hence, there is not a unique agent that implements action selection mechanisms. We have implemented this architecture and tested it on a real robot, a Pioneer 2 DX of ActivMedia Robotics.

1. Introduction

According to the roadmap of the PLANET network [1], the ultimate goal for robotic systems is to obtain a robot capable of improving their performance by autonomously adapting their control software to different tasks and environments, as well as by learning to perform novel tasks in an appropriate way. That means completely autonomous robots capable of modifying their performance in complex and changing environments. Artificial Intelligence provides learning and adaptation methods, as well as decision making techniques to achieve these control properties. However, it is technologically difficult and potentially dangerous to build complex systems that are controlled in a completely centralized way [2].

Recent advances in Multi-Agent Systems have inspired researchers to advance in implementing distributed architectures. There are many definitions of software agents, for example Murphy's definition [6]: A software agent is an autonomous program which can interact with and adapt to their world. An agent is self-contained, independent, situated and has self-awareness.

This is a global definition and many things can be thought as agents and certainly it does not involve "*intelligence*", so we prefer the definition of *intelligent agent* that is, an agent that is capable of *flexible* autonomous action in order to meet its design objectives, where flexibility means three things [7]-[9]:

- **Reactivity:** intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives. A *reactive* system is one that maintains an ongoing interaction with environment, and responds to changes that occur in it (in time for the response to be useful);
- **Pro-activeness:** intelligent agents are able to exhibit goal-directed behaviour by *taking the initiative* in order to satisfy their design objectives;
- **Social ability:** intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives. *Social ability* is the ability to interact with other agents (and possibly humans) via some kind of *agent-communication language*, and perhaps cooperate with others.

When problems become more complex, realistic and large-scaled, they are beyond the capabilities of a single agent. The only reasonable way to attack this type of problems is to create an organization of agents in which each agent is highly specialized at solving a particular problem aspect. This organization is known as Multi-Agent System. Formally, a **Multi-Agent System** is a distributed system that contains a collection of agents that work together in order to solve problems

[10], [7], [12]. Agents in MAS interact through communication. Therefore they provide more flexibility to the development of robot architectures. Regarding communication constraints due to agent interaction for coordination, recent works on real-time multi-agent systems have proved the responsiveness of these kinds of architectures to their environments [25].

Regarding robotics, most of the applications of MAS have been applied to the coordination of teams of robots (1 robot = 1 agent) [12]-[15]. But there are less examples in the application of MAS to implement distributed control system for a single robot (1 robot = n agents) [16]-[21]. In this latter case agents are used to maintain "*independency*" among the different behaviours of the robot, usually grouped in three main layers: reactive, intermediate and deliberative, conforming a hybrid architecture.

The present work develops a reactive multi-agent architecture. As we have designed the architecture as an organization, we have stated some different tasks to perform in the community, as for example, sensing, so agents have been assigned different roles regarding their responsibility on the tasks. These roles fix interactions and relationships among agents, making them to cooperate to achieve goals (as for example surveillance) and to compete for the use of the different resources (as for example the robot motors).

This paper is organized as follows. Next Section presents the related work. Section 3 describes the proposed architecture and Section 4 emphasises on the coordination mechanism developed. Empirical results are shown in Section 5. Finally, conclusions and future work are drawn in Section 6.

2. Related Work

Most robot architectures found in the literature rely on hybrid approaches that usually have three main layers: the reactive, the intermediate and the deliberative layers. Commonly, reactive layer contains a set of behaviours that are combined in order to give a rapid response over unexpected events while planning a task, or a mission, or a path is executed independently in the deliberative level. The intermediate layer usually is used as an interface between the reactive and the deliberative layers. According to these three layers, hybrid architectures are usually implemented as modules with a hierarchical organization, having lower level agents priority over higher level agents.

In this line, Rosenblatt's work was one of the pioneers of building an architecture composed of distributed, independent, asynchronous decision-making behaviours that are coordinated by a central arbiter [3]. The overall behaviour of the system is rational, coherent and goal-oriented while preserving real-time responsiveness to its immediate physical environment. The advantages of this architecture are the following: they facilitate their development and lead to the evolutionary creation of robust systems of incrementally greater capabilities [3]. Many more architectures have now been developed, as for example O2CA2 [4], in which the coordination is based on a voting scheme. In a more advanced work [5], several architectures are analyzed, and the advantages of hierarchical architectures are pointed out. In these architectures, each behaviour is implemented by a module with communication abilities.

Advances in Multi-Agent Systems (MAS) have inspired researchers to go one abstract level further in implementing the architectures, in which modules are replaced by agents [19], [20]. Using agents to deploy hybrid architectures instead of modules, the different layers can provide alternative actions at a given time (go, stop); even at the same layer, several agents can suggest different actions (go, avoid obstacles). So the key issue in MAS is the coordination mechanism that arbitrates and selects the single action to be performed at a given time according to the possible set of actions provided by different agents.

There are several architectures built as multi-agent systems to control a single robot. The MAS proposed by [17] has two kinds of agents at the reactive level: **elementary agents** that have basic abilities, and **high-level agents** responsible for the integration and coordination of several elementary agents.

In [20], there is a specific agent that based on an auction process, determines which action must be carried out. This architecture differs from typical hybrid ones in that all the components of the architecture coordinate among them obtaining a non hierarchical structure. In this case, bids represent the **urgency** of the system on having a service engaged. However final action decision depends on a centralized agent.

In [19] the definition of the MAS system control architecture is based on the organization theory and the strategic alliances. In this work, the MAS is considered as an organization and the robot architecture is designed as a joint of specialized social entities that interact and cooperate among each other in order to achieve common or private goals. The structure of the organization defines the social roles of various components, their responsibilities for tasks and goals, the

way in which the resources are allocated and the strategies that must be adopted. This structure defines how to coordinate the activities of various actors and how they depend on each other. There is no negotiation to decide upon the action to perform, but all is predefined.

Other interesting work related to the engineering of complex adaptive agents is [5], which proposes a behavioural-oriented design methodology in order to build complex systems. The author, however, implements a single agent with objects, because the author believes that objects presumably operate more quickly. Even that some communication and coordination issues can waste time, recent works as [25] corroborate that negotiation among different agents can be achieved in real-time.

Our architecture pretends to take advantages of some of the above ideas. Particularly, we propose a non-hierarchical architecture, conceived as an organization in which agents have their own roles. As it is a Multi-agent system we give flexibility to the architecture being easier to add agents to the community and to distribute the system in different computers. Unlike the presented architectures, ours distributes coordination: coordination is performed as a negotiation among agents.

3. Reactive MAS Architecture

The reactive multi-agent architecture compounds several specific agents related to different aspects of reactive control: perception agents, actuator agents, and behavioural agents. Perception agents obtain information about the environment and about the internal conditions of the robot; actuator agents are in charge of controlling the linear and angular speed of the robot interacting directly with motors; and behavioural agents carry out specific actions, as avoiding obstacles.

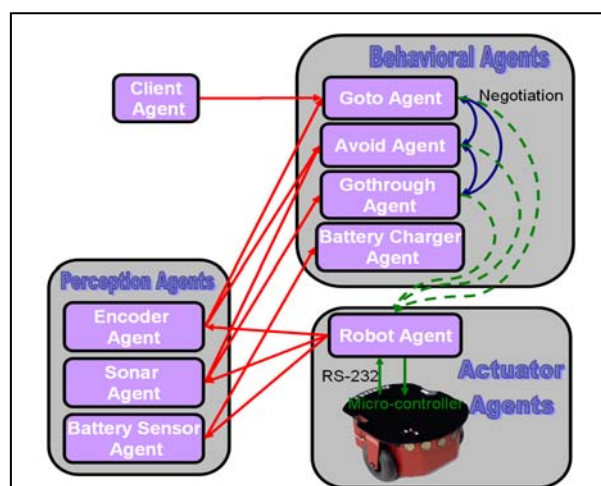


Figure 1: MAS reactive architecture

There is a perception agent per each sensor present on the robot. These agents extract information about the environment and the robot and process it in order to communicate it, when required, in an appropriate way to other agents in the community.

There is a behavioural agent per each specific behaviour to perform. Based on the information received from perception agents, they calculate the linear and angular speed the robot must have in order to achieve the desired action. As they cannot send the desired linear and angular speeds all together to the actuator agents, some coordination must be done. Instead of centralizing the behaviour selection in a coordinator agent (to select the winning action as in [20]), behavioural agents negotiate to take control.

Figure 1 shows the complete multi-agent architecture used to control the robot. As the robot has encoders, ultrasonic sensors and a battery charge sensor, there are three perception agents: the encoder agent, the sonar agent and the battery sensor agent. More agents can be added in case of addition of new sensors. Regarding to behavioural agents, we have defined the goto agent, the avoid agent, the goThrough agent and the battery charge agent. Analogously, it is possible to introduce more behaviours if required. There are two additional agents in this architecture that are the client and the robot agents. We explain next, the roles of the different agents that form the reactive architecture. We begin the explanation with

the robot agent because it represents the physical constraints of our robot that determines some design decisions in our MAS architecture.

3.1 Robot Agent

The robot agent is the interface between the multi-agent architecture and the robot micro-controller. He represents the real robot in the architecture. The commercial robot Pioneer 2DX of ActivMedia Robotics has two driving wheels, motorized with DC motors, fixed at the front of the robot's body, and a free wheel at the back. Each front wheel has an encoder sensor attached. There are also 8 ultrasonic sensors placed as a ring in the front of the robot. The dynamic model and a detailed description of the robot can be found in [22]. It is also possible to measure the battery charge during operation. The robot has a micro-controller and a PC-104 on-board. The micro-controller obtains the sensor readings and controls the motors. Inside, there are two controllers, a linear speed controller and an angular speed controller. Communication with the PC-104 (where the MAS architecture is running) is made via RS-232. The commercial library ARIA from ActivMedia is used to communicate the local server (micro-controller) to the agents.

The micro-controller acts as a server and only allows one client connection at a time. Thus, it provides the information of the different sensors and actuators. It computes the position and heading of the robot in local coordinates, reads all the ranges provided by sonars and gives the battery charge when required.

Particularly, the robot agent communicates, each 100 ms, with the robot and gets the actual position, sonar readings and battery charge, and it distribute this information to the different sensor agents when required. It also gets the desired angular and linear speed from behavioural agents and sends them to the micro-controller.

3.2 Encoder Agent

This agent is responsible of getting the position and heading of the robot in local coordinates, related to a frame attached to the centre of mass of the robot and translate them to global coordinates, in an earth fixed frame.

It gets the position and heading from the robot agent, as the latter is the only agent that can interact with the micro-controller.

3.3 Sonar Agent

This agent is in charge of creating a local map to locate obstacles based on the reading of the 8 ultrasonic sensors. It gets the different measurements from the robot agent and treats them in order to find the obstacles in the path of the robot. It also has to update this map as the robot moves on.

To create the map, a zone around the robot is divided in cells as shown in figure 2. Cells are obtained dividing the circle around the robot in 18 circular sectors that represents the ultrasonic sensor visibility zones and the circular sectors in 10 parts representing different distances from the robot.

This organization in cells is useful for dealing with noise and fictitious obstacles detected by ultrasonic sensors. If an object is detected in a cell several times, then the probability associated to the cell increments, indicating the presence of the object. So, each cell is labelled by a probability regarding the fact that an obstacle has been detected inside. Probabilities are incremented when an obstacle is sensed in the cell and decremented otherwise. In this way, we introduce some "memory" to sensors.

At each sample time, this agent first applies the movement to the map (to move objects according to the robot motion), then update the sonar information and set it to the map. After that, and using probabilities, finds the closest point (centre of the cell where an object has been detected) to the robot. This point must be eluded, so its coordinates are sent to the avoid agent in order to be used for speed calculation.

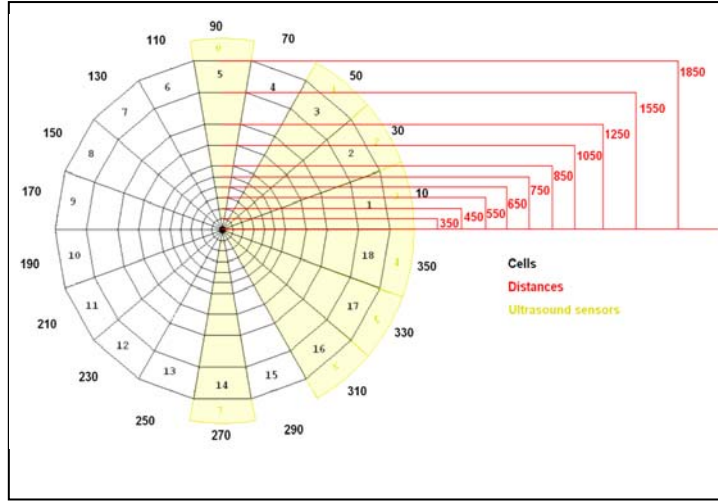


Figure 2: Local map

3.4 Battery Sensor Agent

It obtains the battery charge level from the robot, asking this parameter to the robot agent. It is necessary to check this value time to time because the robot should recharge batteries when its voltages are around 10V. This is due to the fact that batteries can be permanently damaged when this level goes below this value. Therefore this agent senses the battery level at a given frequency. The frequency is dynamically increased as the level value tends to 10V.

3.5 Goto Agent

This agent is responsible of driving the robot to the goal position, obtained from the client agent, based on the information provided by the encoder agent.

Given a desired position (x, y) and an orientation θ , and according to the actual position and heading, this agent calculates the linear and angular speeds to drive the robot to the target position (goal). There are two separate PID controllers that obtain these speeds depending on distance and heading to the desired position, which are combined following a fuzzy approach (see [26] for further details).

In addition, this agent calculates a parameter that we call **utility**. This parameter is computed based on the distance left to the goal (d_i) in millimetres:

$$pGoto = \alpha \cdot d_i + \beta \quad (1)$$

Being:

$$\alpha = \frac{P_{\max} - P_{\min}}{u_L - u_H} \quad \beta = \frac{u_L \cdot P_{\min} - u_H \cdot P_{\max}}{u_L - u_H} \quad (2)$$

Where p_{\max} and p_{\min} are the maximum and minimum value respectively of the $pGoto$ parameter ($p_{\min} \leq pGoto \leq p_{\max}$), and u_L and u_H are distance thresholds.

Figure 3 shows the shape of $pGoto$ as well as the parameters used to calculate α and β . p_{\min} and p_{\max} allow the modification of the overall relevance of the behaviour, while u_L and u_H are set according to the loss of the degree of relevance.

Once the $pGoto$ utility is calculated, the goto agent sends this parameter to the avoid and to the goThrough agents in order to coordinate their behaviours.

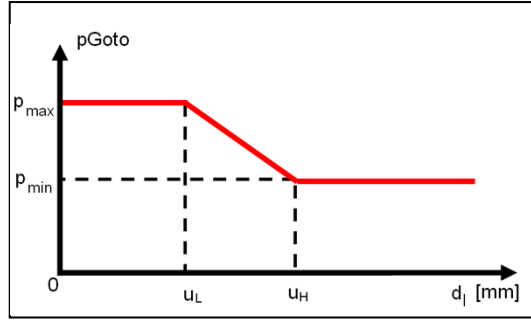


Figure 3: Urgency parameter of goto agent

3.6 Avoid Agent

This agent is responsible of avoiding the obstacles that could be in the robot path to the goal. He asks the sonar agent for the closest obstacle point and calculates the suitable linear and angular speeds for avoiding collisions with the object. In order to calculate the speeds, the agent has three different zones around the object: the caution zone, the danger zone and the stop zone. When the robot is in the **caution zone** (see figure 4) the avoid agent calculates a lower linear speed, because it's far from the object and there is a low probability of colliding. In the **danger zone**, the avoid agent changes the linear speed as well as the angular speed, heading the robot out of this zone. If the robot enters in the **stop zone**, the avoid agent stops the linear movement and rotates the robot 180 degrees. These zones vary according with the kind of environment the robot is moving. If there are plenty of objects the zones are smaller than if the surroundings are wide. Moreover maximum speeds can be greater in the second case.

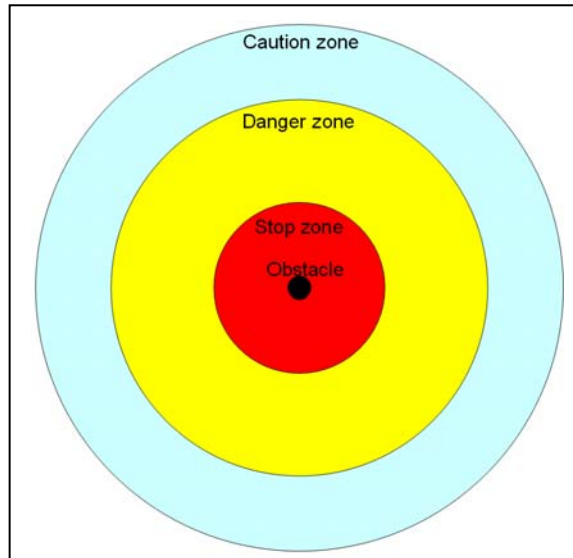


Figure 4: Zones around the obstacle point

Once the linear and angular speed are computed, the avoid agent computes the *utility* parameter. This parameter is calculated in function to the distance left to the obstacle (p_d) and the heading of the robot respect to the obstacle (p_h):

$$p_{Avoid} = p_h \cdot p_d \quad (3)$$

Both, p_h and p_d are defined in the $[0,1]$ interval and computed according to the following expression:

$$\begin{aligned} p_h &= \alpha_1 \cdot h_o + \beta_1 \\ p_d &= \alpha_2 \cdot d_o + \beta_2 \end{aligned} \quad (4)$$

Where α_i and β_i with $i=1,2$ are defined as in (2), being in this case p_{min} and p_{max} the minimum and maximum value of the p_h value ($p_{hmin} \leq p_h \leq p_{hmax}$) and the p_d value ($p_{dmin} \leq p_d \leq p_{dmax}$) respectively; d_o represents the distance left to the object and h_o is

the compounding angle as shown in figure 5. The *pAvoid utility* parameter is sent to the goto agent and to the goThrough agent, to coordinate their behaviours.

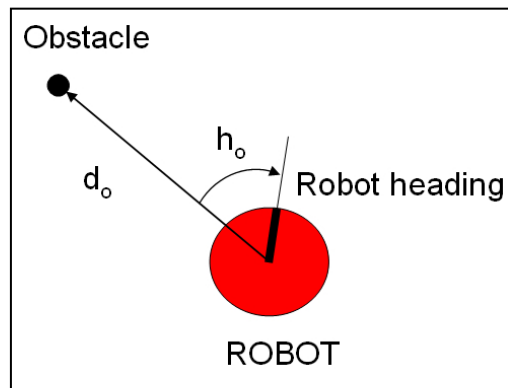


Figure 5: Angle (h_o) and distance (d_o) involved in the pAvoid calculation

3.7 GoThrough Agent

Based on the information provided by the sonar agents, the goThrough agent calculates the linear and the angular speeds of the robot in order to drive the robot through narrow places like doors.

The *utility* function of the goThrough agent is computed considering the distance to the door in a similar way to the goto agent (equation (1)). The pGoThrough *utility* value is sent to the goto and to the avoid agents in order to coordinate their behaviours.

3.8 Client Agent

This agent is the user interface of the architecture and allows commanding the robot goals (desired position and orientation). So, the user can send a goal position and orientation to the architecture through the client agent.

4. Agents' Coordination

Coordination among agents is necessary when there are several agents trying to use the same resource at a given time. In the presented architecture, these conflicts can arise among the avoid, the goto and the goThrough agents when trying to send conflicting actions to the robot, as shown in figure 6. Here, circles represent the agents, and the arrows, the communication flow between two agents.

A possible solution to this problem is to define a central coordinator agent that imposes a decision among the conflicting agents based on some knowledge. However, we believe that such centralized coordination mechanism can be a bottleneck when dealing with architectures with a lot of agents. Conversely, we think that a distributed coordination approach can be more appropriated to deal with them.

In the proposed architecture each agent computes a normalized *utility* value (between [0,1]) only known by the agent itself. In order to solve conflicting decisions, the *utility* value is used to obtain the control over the resources. This *utility* value is sent to the other agents in conflict in order to negotiate the best action. Thus, the agent who has a higher value of *utility* wins the decision. This agent is getting the control regarding to the conflicting activity. For example, suppose that the goto agent has a *utility* value of 0.5, the goThrough agent of 0.3 and the avoid agent of 0.7; being 0.7 the higher value. The avoid agent takes the control of the situation, and it is the only one that sends speeds to the robot agent.

To reduce communication among agents using this decentralized approach, the agent who has the control, broadcasts its *utility* value. If there is no response, meaning that it has the higher value, the agent uses the resource. On the other hand, if there is an agent with a higher *utility* value, then it informs all the agents with this value, indicating that it is going to use the resource. In this way, communication process is reduced and centralization of coordination is avoided.

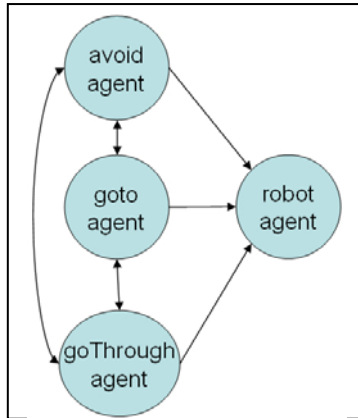


Figure 6: Conflicts among the avoid, the goto and the goThrough agents

Figure 7 shows the negotiation state diagram for the conflicting action depicted in figure 6. Circles means the states and arrows are the messages between the agents. The parameters of the message refers to the emitter agent and to the receiver agent, as for example, $send_utility(goto, avoid)$ means that the goto agent sends its *utility* value to the avoid agent. The goto agent changes from the initial state to the state number 1 when it receives a message from the client agent, indicating the desired position. It remains in state 1 until the encoder agent sends the current position. At this moment, changes to state number 2. Here the goto agent computes the desired angular and linear speeds and the *utility* value and sends the later to the other agents (conflicting agents), and changes its state to number 3. At this state, it computes continuously the desired speeds based on the information given by the encoder agent and computes its *utility* value. Each time, it sends the *utility* value to the conflicting agents (in this case, the avoid and the goThrough agents) and waits a short time for an answer. If there is no message, the cycle continues and the agent sends the desired speeds to the robot agent. If a higher *utility* value is received, the goto agent changes its state to number 4. Based on the information received from the encoder agent, it computes the desired speeds and the *utility*, but in this new state it does not send the *utility* to the other agents in conflict. It remains listening to the agent with the highest utility, which has taken the robot control. When the utility is the highest, the state changes to state number 3. Changes from state 3 to 4 and conversely are performed until in a given moment, the desired position is reached. This can only happens from state 3, when the goto agent has the robot control. Then a null speed message is sent to the robot, ending the whole process.

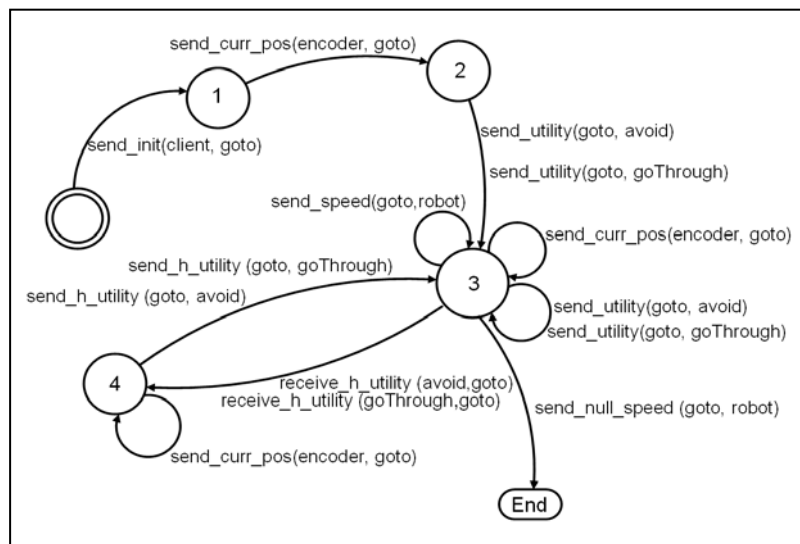


Figure 7: Negotiation state diagram

5. Experiments

As stated before, we have implemented this architecture and tested it on a real robot. The robot used to test the architecture is the commercial robot Pioneer 2DX of ActivMedia Robotics.

In order to develop the architecture, an ad-hoc MAS platform has been developed using the C++ programming language for the Linux operating system. The decision of building an ad-hoc platform stands on the fact that all the existing commercial multi-agent platforms cannot operate in real-time and in most of them, there is an agent (as for example the

facilitator in Open Agent Architecture [28]) that centralizes all the communication, becoming the bottleneck of the whole architecture.

Several experiments have been carried out with our proposed architecture. Next, we select one of those experiments to illustrate the performance of the architecture.

Figure 8 shows the trajectory described by the robot when asked to arrive to the goal point of $(x,y) = (5,0)$ meters (m) and a heading of $\theta = 0^\circ$, starting at $(x,y)_i = (0,0)$ m and $\theta_i = 0^\circ$. At approximately 2.36 m there is an object, a rectangular column of 0.50×0.50 m. Remind that obstacles are considered as a point (the closest point to the robot) while the robot is travelling so, the different zones of the avoid agent are moving according to the detected point. In the scenario of figure 8, the first obstacle is detected at $(2.0, 0.0)$, when the robot is in position $(1.3, 0.0)$ and is moving forward.

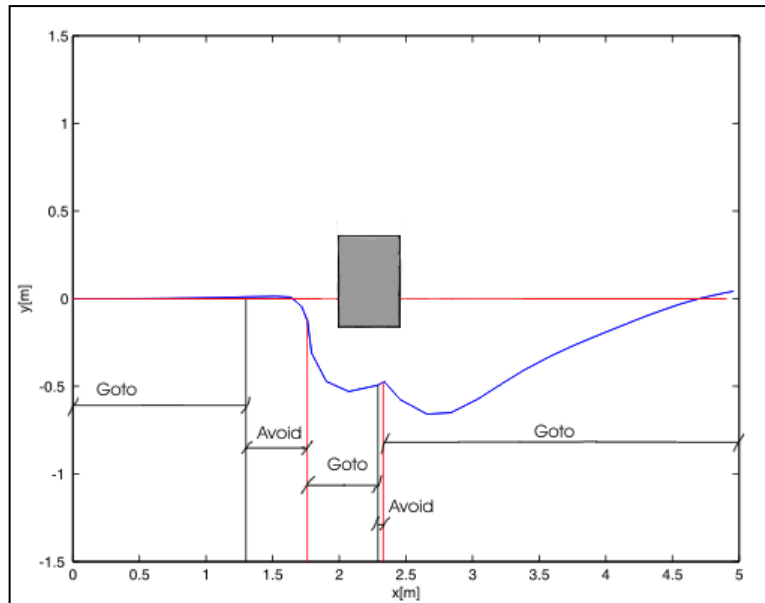


Figure 8: Trajectory of the robot until goal position

Also, it can be seen in figure 8 when each agent takes control of the robot. First, as the obstacle is not detected, the goto agent is sending his calculated speeds to the robot, because its *utility* value is higher than the others. As the goal is right in front of it, the angular speed is zero and the linear speed is maximum (see figure 9). At approximately at 1.3 m the obstacle is detected by the sonar agent. The *utility* value of the avoid agent starts to increase until its value is greater than the *utility* of the goto agent, and at this moment, the avoid agent takes control over the robot. Since the robot position is inside the **caution zone**, the avoid agent reduces the robot's linear speed and increases the angular speed in order to rotate the robot out of the path of the obstacle. When the obstacle is not detected anymore in the trajectory, the *utility* value of the avoid agent goes below the *utility* value of the goto agent, so the later gets control again. As it turns the robot towards the goal, the obstacle is found again and the avoid agent wins back the robot control. As soon as the obstacle is not on the path of the robot, the goto agent takes control again to drive the robot to the goal position. The experiment shows how the negotiation protocol is valid when dealing with the robot movement while maintaining an adequate responsiveness to the obstacles found in the path to the goal.

It can be observed that there is only one agent acting at a time. This causes non smooth trajectories to be obtained due to the change of behaviours. Smoother trajectories can be achieved by other methods as for example [23] and [24]) but they are centralized. How to improve trajectory smoothness by maintaining a distributed coordination mechanisms has to be analyzed in the future.

Figure 9 shows the different linear and angular speeds applied to the robot during the entire trajectory described in figure 8. Maximum linear speed is limited to 20 cm/s while maximum angular speed is $80^\circ/s$, even though this value is never reached.

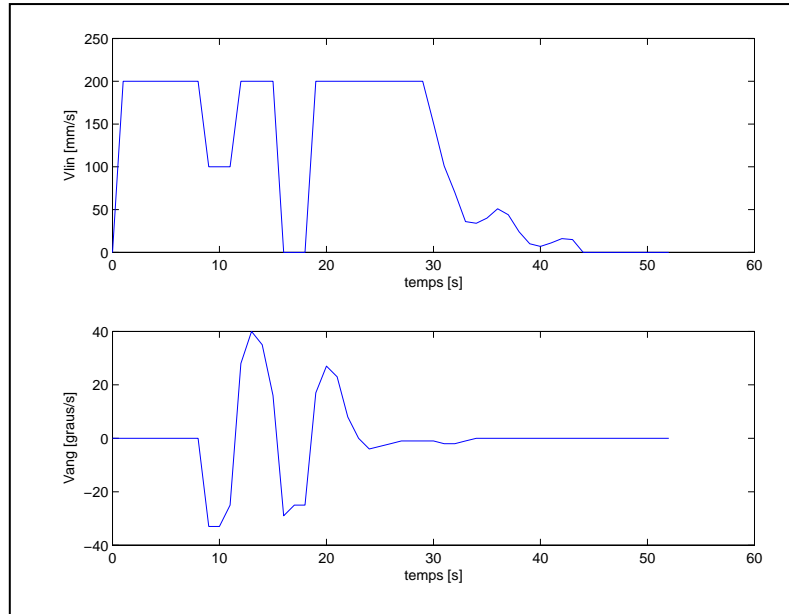


Figure 9: Speeds sent to the robot

Finally in figure 10 it is depicted the evolution of variables x, y and θ along time. It can be seen that the x coordinate is reached while there is a little error (less than 5 cm) in the y coordinate. Something similar happens with the orientation of the robot, where the error is less than 7 degrees. These errors are due to the fact that a zone around the goal is defined in order to avoid that the robot starts to oscillate trying to reach the goal point. The same happens with the angle. In this case, the maxim distance at which the robot can stop is 10 cm with a $\pm 10^\circ$ error in the heading.

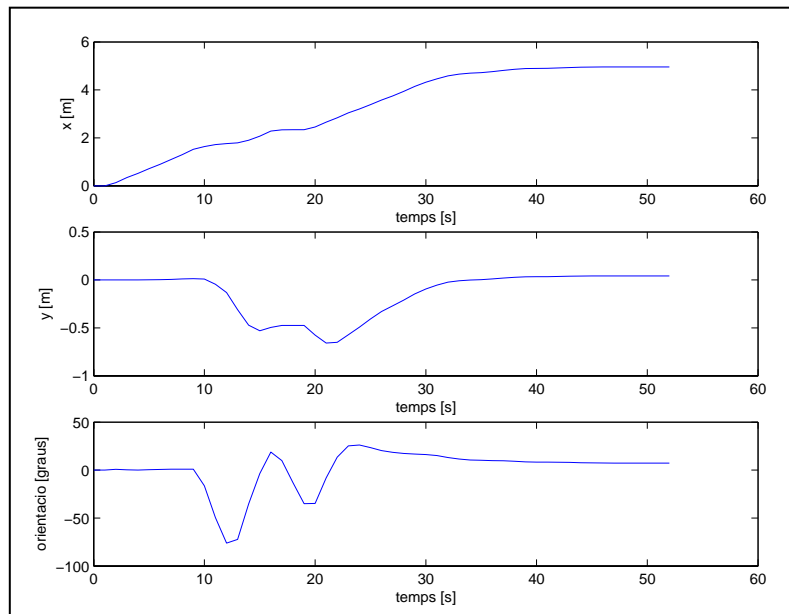


Figure 10: Evolution of x, y and θ along time

Among the different experiments performed, we have adjusted the parameters of the utility functions. In figure 11 are shown the best parameters for the utility function of the goto agent. They are $p_{\min}=0.7$, $p_{\max}=1.0$, $u_L=100\text{mm}$ and $u_H=300\text{mm}$. The best parameters for the utility function of the avoid agent are depicted in figure 12. For the p_d function they are $p_{\min}=0$, $p_{\max}=1.0$, $u_L=650\text{mm}$ and $u_H=850\text{mm}$ and for the p_h function, $p_{\min}=0$, $p_{\max}=1.0$, $u_L=40^\circ$ and $u_H=90^\circ$.

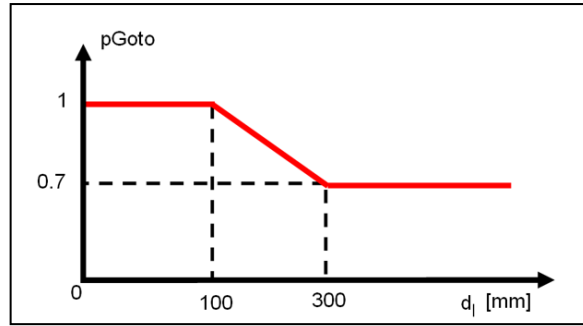


Figure 11: Calculation of urgency of the goto agent

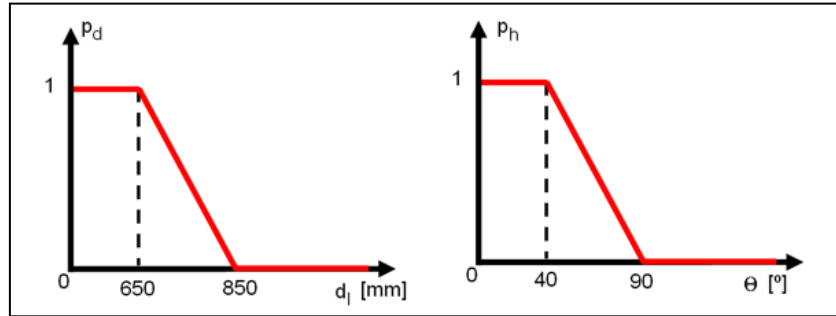


Figure 12: Calculation of urgency of the avoid agent

We have performed several experiments starting at different positions and headings, with different goals and with the presence of more than one obstacle and we have obtained the results summarized in table I (see [27] for detailed information).

Table I: Experimental results

N° obstacles	Succeed	Failure	Collision on failure
0	98%	2%	0%
1	80%	20%	75% (of 20%)
2	60%	40%	90% (of 40%)

Success is considered when the robot stops inside a circle of 10 cm radius and $\pm 10^\circ$ heading of the goal position; otherwise the robot fails in achieving its target position. The causes of failure can be analyzed according to the percentage of collisions, as shown in the third column of table I. When there are no obstacles the robot achieves the goal position the 98% of the times. When there is only one obstacle, the robot succeeds 80% of the times. Of the 20% of failing times 75% the robot collides with the obstacle. When there are two objects, the robot arrives to the goal the 60% of the times. Of the 40% of failing times, 90% the robot collides. We have analyzed that collisions are due to ultrasonic sensors. Most of the times, sonars do not detect the object, especially convex angles as columns or corridor corner's.

6. Conclusion

In order to develop completely autonomous robots capable of modifying their performance in complex and changing environments, distributed systems should be used to develop the robot control architecture. Multi-Agent Systems is a powerful Artificial Intelligence tool that allows to develop this kind of systems. In this paper a multi-agent based architecture is presented. This architecture has been conceived as an organization in which its members have an assigned role to accomplish, so, in this way it is defined how several agents coordinate their activities.

The agents that constitute the architecture are perception agents, behavioural agents and actuator agents. As perception agents we have implemented the encoder and the sonar agents according to the sensors available in our robot. The encoder agent besides calculating the movement related to the robot's attached frame, it translates them to global coordinates. The sonar agent aside from taking the sonar readings, it constructs a local map around the robot pointing out the zones with possible presence of obstacles and finds the object point nearest to the robot. If some new sensor is added to the robot, new agents can be implemented in order to treat new sensor information.

As behavioural agents we have implemented the avoid agent, the goto agent and the goThrough agent. The avoid agent, based on the information received from the sonar agent calculates the linear and angular speed needed to avoid the collision with the obstacle. He also calculates a *utility* value (pAvoid) representing the degree to fulfil the desired action. The goto agent calculates the linear and angular speeds necessary to move the robot to the goal position. This agent also calculates a *utility* parameter (pGoto). Finally, the goThrough agent calculates the linear and angular speeds necessary to cross over narrow places. It also computes a *utility* value (pGoThrough). A negotiation protocol based on the *utility* values has been presented among the behavioural agents in order to coordinate commands to be sent to the robot.

In addition to these agents, there are the client agent and the robot agent. The former is an interface that allows user to send target points to the robot. The latter represents the robot in the architecture in order to guarantee that only one connection at a time to the micro-controller is performed, according to the robot constraints.

We have performed several experiments obtaining encouraging results, that we expect to improve with the addition of localization and navigation agents, that support higher deliberative behaviours, as planning. Particularly, we expect to resolve the arisen problems due to the ultrasound readings that do not detect the obstacles with the addition of a localization and a navigation agents. Non-moving obstacles can be detected using a global map computed by these new agents, instead of a single point as done in the present implementation.

Finally we want to stress the fact that our distributed architecture can run in different computers if needed. This fact strengthens the computational properties of the system, so if new agents are added to the system, providing richer and more complex behaviours, their execution can be distributed in different computers. As a consequence, complex behaviours could be achieved without significant computational cost but probably with some communication throughput cost. Recent works as [25], in which a MAS approach has been proved useful for real-time robotics, make us believe that we are working in the right direction.

7. References

- [1] M. Beetz: “*A roadmap for research in robot planning*”. PLANET: European Network of Excellence in AI Planning. Technological Roadmap on AI Planning and Scheduling., pp 89-118,2003.
- [2] R. Murray, K. Aström, S. Boyd, R. Brockett, and G. Stein: “*Future directions in control in an information-rich world*”. IEEE Control Systems Magazine, 23, issue 2 pp 20-33, 2003.
- [3] J. Rosenblatt: “*DAMN: A Distributed Architecture for Mobile Navigation*”. PhD thesis, Robotics Institute at Carnegie Mellon University, 1997.
- [4] P. Ridao, J. Batlle, M. Carreras: “*O2CA2: a new object oriented control architecture for autonomy: the reactive layer*”. Control Engineering Practice, 10(8) pp: 857-873, 2002.
- [5] J. Bryson: “*Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents*”. PhD thesis, Massachusetts Institute of Technology. 2001.
- [6] R. Murphy: “*Introduction to AI Robotics*”. The MIT Press, 2000.
- [7] M. Wooldridge: “*An Introduction to Multiagent Systems*”. John Wiley & Sons, LTD, 2002.
- [8] G. Weiss: “*Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*”. The MIT Press, Cambridge, Massachusetts, 1999.
- [9] N. R. Jennings, K. Sycara, and M. Wooldridge: “*A roadmap of agent research and development.*” Autonomous Agents and Multi-Agents Systems. Editorial Kluwer Academic Publishers, pp. 7-38, 1998.
- [10] B. Chaib-Draa, B. Moulin, and I. Jarras: “*Systèmes multi-agents: Principes généraux et applications*”. Principes et architecture des systèmes multi-agents, JP. Briot et Y Demazeau (eds) (Hermes, Lavoisier). 2001.
- [11] B. Moulin and B. Chaib-Draa: “*A review of distributed artificial intelligence*”. Foundations of Distributed Artificial Intelligence. O'Hare, G. and Jennings, N. (eds), Wiley, pp. 3-55, 1996.

- [12] H. Hu and D. Gu: “*A multi-agent system for cooperative quadruped walking robots*”. Proceedings of the IASTED International Conference Robotics and Applications. pp. 1 -- 5, 2000.
- [13] W. Spears, D. Spears, J. Hamann, and R. Heil: “*Distributed, physics based control of swarms of vehicles*”. Autonomous Robots 17, pages: 137-162, 2004.
- [14] M. Dorigo and et al.: “*Evolving self-organizing behaviours for a swarm-bot*”. Autonomous Robots 17, pages: 223-245, 2004.
- [15] J. L. De la Rosa and et al.: “*Rogi team real: Research on physical agents*”. RoboCup-99: Robot Soccer World Cup III. Veloso, Pagello, Kitano (eds), pp. 434 --438, 1999.
- [16] M. C. Garcia-Alegre and F. Recio: “*Basic agents for Visual/Motor coordination of a mobile robot*”. Conference Proceedings of the First International Conference on Autonomous Agents, pp. 429-434, 1997.
- [17] M. C. Neves and E. Oliveira: “*A multi-agent approach for a mobile robot control system*”. Proceedings of Workshop on "Multi-Agent Systems:Theory and Applications" (MASTA'97 - EPPIA'97) - Coimbra -Portugal, pages 1-14, 1997.
- [18] R. J. Ross: “*Research proposal: Development of a MAS based robot control architecture and the investigation of speech priming on robots*”. University College Dublin, 2002.
- [19] P. Giorgini, M. Kolp, and J. Mylopoulos: “*Socio-intentional architectures for multi-agent systems: The mobile robot control case*”. Proceedings of the Fourth International Bi- Conference Workshop on Agent-Oriented Information Systems (AOIS-02) at CAiSE2002, Toronto, Canada, 2002.
- [20] D. Busquets, C. Sierra, and R. López de Mántaras: “*A multiagent approach to qualitative landmark-based navigation*”. Autonomous Robots, 15, pages: 129-154, 2003.
- [21] R. Manzotti and V. Tagliasco: “*From behaviour-based robots to motivation-based robots*”. Robotics and Autonomous Systems, vol.~51, pp. 175-190, 2005.
- [22] B. Innocenti, P. Ridao, N. Gascons, A. El-Fakdi, B. López and J. Salvi: “*Dynamical model parameters identification of a wheeled mobile robot*”. 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (preprints), 2004.
- [23] B. Gerkey, M. Mataric, and G. Sukhatme: “*Exploiting physical dynamics for concurrent control of a mobile robot*”. Proceedings ICRA '02. IEEE International Conference on Robotics and Automation, vol. 4, pp. 3467-3472, 2002.
- [24] A. Saffiotti: “*The uses of fuzzy logic in autonomous robot navigation*”. Soft Computing, vol. 1, no. 4, pp. 180-197, 1997.
- [25] L. Soh, C. Tsatsoulis: “*A real-time negotiation model and a multi-agent sensor network implementation*”. Autonomous Agents and Multi-Agent Systems, 12(3) pp: 215-271, 2005.
- [26] B. Innocenti, B. López, Q. Salvi: “*Fuzzy Concurrent Position Control Adjustment for a Mobile Robot*”. Technical report: IliA 06-02-RR. Institute of Informatics and Applications. 2006.
- [27] S. Garcia: “*Implementation of the control reactive architecture of the Grill robot*”. BsC. Thesis: submitted to University of Girona, 2005.
- [28] SRI International: “*Open agent architecture (OAA): Developer's guide v. 2.3.0*”. On-line: <http://www.ai.sri.com/oaa/distribution/v2.3/>

BIANCA INNOCENTI graduated in Electronic Engineering in the National University of San Juan (Rep. Argentina) in 1997 and graduated in Automation and Industrial Electronic Engineering in the Technical University of Catalonia (Spain) in 2005. She joined the Control Engineering and Intelligent Systems Group in the University of Girona (Spain), where she obtained the DEA degree in the PhD program Industrial Informatics and Advanced Control Technologies in October 2000. At present, she is an assistant professor in the Electronics, Computer Engineering and Automation Department of the University of Girona. Her current research interests are in the field of mobile robotics and multi-agent systems.

BEATRIZ LOPEZ graduated in Computer Science from the Autonomous University of Barcelona (UAB, Barcelona, Spain) in 1986. She joined the Artificial Intelligence Research Institute of the Spanish Scientific Research Council (CSIC, then placed at the Centre of Advanced Studies of Blanes, CEAB) in 1988 where she received the PhD. degree in Computer Science from the Technical University of Catalonia (UPC, Barcelona, Spain) in 1993. She has been associate professor from 1992-1995 and 1998-2000 at the Rovira Virgili University. She has served also as a Computer Science Engineer in several private companies. Currently, she is an associate professor at the Department of Electronics, Computer Science and Systems Engineering at the University of Girona, Spain. Her research interests include multi-agent systems, planning and scheduling, and case-based reasoning. She is member of the Catalan Association for Artificial Intelligence that belongs to the European Coordination Committee on Artificial Intelligence (ECCAI).

JOAQUIM SALVI graduated in Computer Science in the Technical University of Catalonia in 1993. He joined the Computer Vision and Robotics Group in the University of Girona, where he received the DEA degree in Computer science in July 1996 and the PhD in Industrial Engineering in January 1998 which was rewarded with the best thesis award in engineering in the University of Girona. At present, he is an associate professor in the Electronics, Computer Engineering and Automation Department of the University of Girona. His current interests are in the field of computer vision and mobile robotics, focused on structured light, stereovision and camera calibration.