Available online at www.sciencedirect.com
ScienceDirect
Robotics and Autonomous Systems 55 (2007) 881–891
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

A multi-agent architecture with cooperative fuzzy control for a mobile robot[☆]

Bianca Innocenti[*], Beatriz López, Joaquim Salvi

The Institute of Informatics and Applications, University of Girona, Campus Montilivi, 17071-Girona, Spain

Available online 14 September 2007

Abstract

The challenges of robotics have led the researchers to develop control architectures composed of distributed, independent and asynchronous behaviors. One way to approach decentralization is through cooperative control, since it allows the development of complex behavior based on several controllers combined to achieve the desired result. Robots, however, require high-level cognitive capacities, and multi-agent architectures provide the appropriate level of abstraction to define them. This article describes a multi-agent architecture combined with cooperative control developed within the agent. The experiments were carried out on an ActivMedia Pioneer 2DX mobile robot.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Multi-agent architectures; Mobile robotics; Fuzzy logic; Distributed coordination

1. Introduction

One of the current challenges in the development of robot control systems is making them capable of intelligent and suitable responses to changing environments. Learning and adaptation methods, as well as decision-making techniques, help to achieve these objectives. Nevertheless, it is technologically difficult and potentially dangerous to build complex systems that are only centrally controlled [16], since the system will fail if the control system does not work. With decentralized control, it is possible for the system to continue working even when a part of it fails. Although centralized control allows multiple goals and constraints to be coordinated in a coherent manner, decentralization provides flexibility and robustness [12,25].

One of the first efforts to consider architectures as being composed of different distributed, independent and asynchronous behaviors coordinated by a central arbiter was DAMN [23]. The global behavior of Rosenblatt's architecture was rational, coherent, and goal-directed while preserving real-time responsiveness to its immediate physical environment. Since then, other architectures have been developed, such as, O2CA2 [21] which uses a voting scheme mechanism to coordinate different behaviors. Bryson, in [1], analyzed different architectures and emphasized the hierarchical organization of behaviors as one of the most advantageous ways of organizing them, since the combinatorial complexity of control is reduced (when a particular strategy is being executed, actions associated with alternative strategies should not interfere with actual behaviors). In these types of architectures, each behavior has been modeled as a module that can communicate with others.

On the other hand, recent work with multi-agent systems (MAS) has encouraged researchers to take another step forward in the design of control architectures and to replace modules with agents. Agents are independent, situated, self-contained systems with reflective and reflexive capacities (*self-awareness*) [15]. In [14], a multi-agent architecture for mobile robot control is presented and compared to other classical architectures (control-loop, layered, and task-trees) according to various measures: coordinability (the degree to which a system can coordinate by respecting interdependencies between agent actions, meeting global

[☆] This work was partially supported by the Spanish MEC Project TIN2004-06354-C02-02 and by the DURSI Automation Engineering and Distributed Systems Group, 00296.

[*] Corresponding address: The Institute of Informatics and Applications, University of Girona, Edifici P-IV-Escola Politecnica SupeCampus Montilivi, 17071-Girona, Spain. Tel.: +34 972 41 88 84; fax: +34 972 41 89 76.
E-mail address: bianca.innocenti@udg.edu (B. Innocenti).

0921-8890/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2007.07.007

constraints and optimizing its operations), predictability, reliability-tolerance, and adaptability. In their case study, the authors show that multi-agent architectures based on organizational principles (resulting from a design process, not from emergence) fit robot applications that require open and cooperative components better. [19] also emphasizes the use of a multi-agent architecture in order to build easily expandable and portable mobile robotic control programs. Different developers can implement agents to perform a specific task in different programming languages, and integrate them into the architecture, provided there is a common communication ontology.

In this kind of architecture, agents can be complex entities at the same time. For example, a *goto agent* in an obstacle-free path has to consider if the destination point is near to or far from the current position of the robot, and depending on this information, the robot must move slowly or quickly. One approach to agent design is to take advantage of the work already carried out in cooperative control to develop complex behavior at the agent level. Cooperative control consists of combinations of different controllers to obtain desired behaviors. Integrating these two lines of research, the multi-agent systems and the cooperative control approaches, results in a multi-agent architecture of cooperative controls, which we call a multiple cooperative control approach in a robot.

The entire architecture of the robot is based on a multi-agent system, while cooperative control is used to design and develop a single agent. The combination of the different controllers is achieved by using fuzzy logic concepts.

The advantages of our approach are two-fold, on account of the convergence of the benefits of both technologies: agents and collaborative control. First, using a multi-agent approach, the robot's architecture can be decomposed into flexible autonomous subsystems (agents). The architecture can then be described at a higher level, defining the agents that have to be in the system, the role of each of them, the interactions among them, the actions each of them performs, and the resources they need. Since the multi-agent system is inherently multi-threaded, each agent has its own thread of control; each agent decides whether or not to perform an action on request from another agent (autonomy); and each agent exhibits a reactive, pro-active and social behavior (flexibility) [28]. In addition, in our proposal, no centralized arbiter is defined, but the coordination of all the agents achieving a common goal (robot goals) is peer-to-peer based. Agents related to high cognitive capabilities like planning can be easily integrated with agents providing basic behaviors such as moving to a point, all running concurrently, sharing the robot resources and negotiating when conflicts arise.

Second, at the agent level, collaborative control allows the design and combination of simple controllers to achieve complex behaviors more easily than other approaches, such as predictive or adaptive control, in which a more accurate model of the robot is needed in order to implement the control laws. In addition, a fuzzy logic approach to combine controls allows the management of uncertainties involved in the decision-making process of choosing one or another controller.

Nevertheless, this approach also has some drawbacks too. Agents require interaction to get information and share resources, resulting in some communication overhead. This issue can be a problem when dealing with real physical systems, but in [26] Soh has demonstrated the capacity of multi-agent systems to respond in real time to changes in the environment. Nevertheless, keeping the number of communication needs to a minimum is good when dealing with real time systems. Along this line, the use of collaborative control at the agent level helps to achieve complex behaviors while keeping the number of agents to a certain amount, thereby reducing communication among them.

This article briefly describes the proposed multi-agent architecture [11] and provides a detailed explanation of an agent implemented with cooperative control based on fuzzy logic [13]. The aim of the article is to present the cooperative control carried out in a multi-agent architecture agent and to analyze the global behavior when an agent with cooperative control interacts with other agents of the architecture.

The remainder of the article is structured as follows. First the state-of-the-art of multi-agent and cooperative control architectures is detailed in Section 2. Section 3 presents our multiple cooperative control architecture and Section 4 introduces the cooperative control of the goto agent. Section 5 shows the implementation and experimental results. The article ends with conclusions.

2. Related work

It has already been established that multiple cooperative control in a single robot involves cooperative control and multi-agent systems. Cooperative control has a very general meaning and every time control algorithms are defined in a complex task, the idea of cooperation is implicitly introduced into the control. Therefore, any complex system developed with multi-agent architectures may be thought of as a cooperative approach.

2.1. Multi-agent robot system architectures

The majority of multi-agent robot system architectures, rather than being built to control single robot systems, are meant for multi-robot systems [10,27,4,3]. The mapping from a robot to an agent seems straightforward, as long as each robot represents a physical entity with a specific task. Such mapping, however, can also be extended inside a robot, carrying out different activities in order to respond in time to the environment. For example some activities might require input from a variety of sensors, to control the motion of its wheels, and plan paths. In addition the robot should avoid obstacles and deal with uncertainty due to sensor constraints, power levels and other events that it may encounter. Such activities should be organized to achieve the robot's goals, and multi-agent methodologies offer an appropriate background.

For example, the multi-agent architecture proposed for the reactive level by [17] has two types of agents: elemental, with basic skills, and high-level, responsible for integrating and coordinating various elemental agents. In [2], a multi-agent system is proposed for the navigation system, in which five agents (map manager, target tracker, risk manager, rescuer, and communicator) are coordinated by means of a bidding mechanism to determine the action to be carried out. In [14], two MAS control system architectures are defined based on organization theory and strategic alliances: structure-in-5 and joint venture. In the former, five subsystems are placed following a start structure, where the navigator is in the middle as the central intermediate agent coordinating the movements of the robot to assume reliability-tolerance and adaptability management. In the latter, the robot architecture is organized around a joint manager assuming coordination. Our approach differs from [14] in which we have no central arbiter; according to the organizational terminology introduced by the authors, we are using an arm's-length style, in which competitive and independent agents establish agreements between themselves, while keeping their autonomy and acting and putting their knowledge together to accomplish specific common goals.

Ref. [22] proposes a multi-agent system that improves autonomous robot navigation in unknown, semi-structured environments. The novelty of this work is the use of case-based reasoning (CBR) techniques to deal with problematic situations, such as dead ends or the location of obstacles the robot cannot avoid. In [19], a multi-agent architecture is also proposed to deploy an intelligent wheelchair. The agents considered in this architecture are the sensor handler, the collision detector, the corridor recognizer and the drive controller. The behaviors implemented in the system are obstacle avoidance, door passage and wall following. Specifically, the collision detector, responsible for the safety of the robot, is fuzzy-based. The input of the agent is the linear distance, and the velocity and turn-angle are the outputs. Note however, that in this case fuzzy logic is not used to combine controllers in a collaborative way, as we are proposing.

2.2. *[Non-multi-agent] cooperative control*

Cooperative control is applied to three types of systems: sensor fusion, where knowledge is obtained by processing information gathered from many sources; control of multiple processes, in which various controllers share control of the system; and multiple human operators, in which various operators share and negotiate the control. Our research is focused on the branch of multiple process control.

Various researchers have worked with mobile robots in this line of research. In [9], various sources control each wheel of the robot. The controllers are modeled as finite automata whose inputs are the location of the robot. The desired increase in movement is based on the average of the total vote of the outputs of the automata. In [6], high-level knowledge is used to select the appropriate controller to carry out the desired action. The controller is selected by choosing, from a case-base of controllers, the one that has similar specifications to those sent by the high-level subsystem. Other approaches, like that of [8], show how cooperative control (called concurrent control) presents several desirable properties for the control of mobile robots like fault tolerance, distribution and scalability. This approach is based on alternating orders coming from different controllers in the motors. The resulting movement is achieved by overlapping various control signals.

In [24] a high-level decision-making procedure, instead of four overlapping signals, is proposed to coordinate the various controllers. In particular, the authors use fuzzy logic to model the control actions coming from the heterogeneous controllers and to decide, in accordance with the dynamic model of the robot, what combination of control actions should be carried out at any given moment. The main difference between our work and that of Saffiotti is that a fuzzy controller is not implemented by fuzzy rules; rather, fuzzy sets are used to model the relevance of the different controllers in the aggregation procedure and in this way expand on the work presented by [8]. In addition, in our approach the result of the cooperative controller is the output of a single behavior, instead of being the direct command of the actuators. The output of each behavior (agent) is coordinated in the multi-agent architecture to decide on the next actions of the robot.

As far as we know, no previous work has proved the validity of the integration of both approaches, multi-agent and collaborative control, in a mobile robot control architecture.

3. Multiple cooperative control architecture

The multi-agent system proposed as the robot control architecture can be divided into four subsystems of agents: **perception, behavior, deliberative** and **actuator** (see Fig. 1). In addition to these agents, there are the **client** agent that is the user interface (through which goals are specified) and the **platform** agents that are in charge of providing all the services necessary to guarantee the correct operation of the multi-agent platform.

Fig. 1 also depicts the flow of information among different agents. Solid lines represent no message flow restrictions while dashed lines indicate that only one agent communicates with the *robot* agent or the *path planning* agent at a time (after coordination).

The perception subsystem agents obtain information about the environment and about the internal conditions of the robot (for example the level of the battery charge). They collect data from the sensors and adapt them to provide the information requested by the other agents of the system. There are as many perception agents as there are sensors or groups of them in the robot. The **encoder agent** is in charge of obtaining the coordinates (x, y) of the robot and its orientation, with reference to a fixed axis. The **sonar agent**, collects all the sonar readings of the robot and creates a local map of obstacles. And the **battery sensor agent** monitors the battery, to prevent it from discharging too much and remaining permanently damaged.

The behavior subsystem carries out specific actions, such as avoiding obstacles, going to a point, etc. The information coming from the perception agents is used to react or respond to the changes produced in the robot itself or in the environment. The following agents have been defined: the **goto agent**, which is in charge of taking the robot from the initial to the final

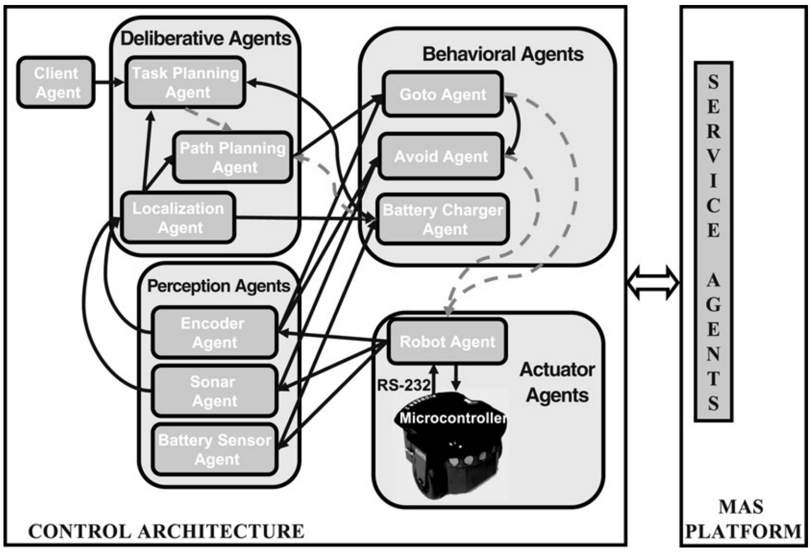

Fig. 1. Multiple cooperative control architecture: the deliberative, behavioral, perception and actuator subsystems and the relationships among the different agents.

coordinates without considering obstacles; the **avoid agent**, which must go around the obstacles when they are found in the path of the robot; and the **battery charger agent** which requests path replanning to take the robot to the charging area when the battery is close to the lowest allowed limit.

The deliberative subsystem is composed of agents in charge of carrying out high-level complex tasks which require a certain amount of time. As for deliberative agents, there is the **localization agent**, which locates the robot on the global map, the **path planning agent**, which must search for obstacle-free paths, and the **task planning agent**, which is in charge of breaking down the missions into tasks.

The actuator subsystem is responsible for directly using the robot's various performance motion systems. There can be as many agents as there are actuators or sets of them in the robot. In our case, only one actuator agent is defined: the **robot agent** which functions as an interface between the control architecture and the microprocessor of the robot. This is due to the limitations of the Pioneer 2DX operating system, which only allows one connection at a time with the microprocessor. Through this link the sensors must be read and the actuators written. In addition, the robot carries linear and angular velocity controllers recommended by the manufacturer, and their use impedes the commands from being sent directly to the motors.

Finally, the service agents, who have been designed following the Foundation for Intelligent Physical Agents (FIPA-IEEE) standards [7], are accountable for providing the basic services of the multi-agent platform that include the way agents communicate among themselves. In our proposal, we consider the **directory facilitator agent** (DFA), which keeps track of the agents currently active in the architecture, their location in the net (host name and port name), the services they provide and the resources they need. Each time an agent joins the community, it has to register with the DFA which then informs the rest of the community about the new agent, the services that it offers and the resources that it uses. Thus, this agent is a key element of the platform regarding interoperating issues such as the use of various software programs from different providers.

Once all the agents of the control architecture are running, the user or other external agent can request a task through the *client* agent which sends the new robot goal to the *path planning* agent. This agent decomposes the task into a trajectory and sends the first target position to the *goto* agent. Based on this information and the actual position (obtained from the *encoder* agent), the *goto* agent calculates the best linear and angular speeds to reach the target. On the other hand, based on the information provided by the *encoder* and *sonar* agents, the *avoid* agent calculates the linear and angular speeds needed to dodge the obstacle. At this point both agents (*goto* and *avoid*) negotiate in order to decide who uses the motors. The one that wins sends the desired speeds to the *robot* agent who sends the values to the robot micro-processor. The latter agent obtains the sonar and the encoder readings and sends them to the *sonar* agent and the *encoder* agent, correspondingly. With this new information all the agents update their internal state and new decisions can be taken. If the target position sent by the *path planning* agent is reached, it sends another target position to the *goto* agent. In the other case, the *goto* and *avoid* agents calculate new speeds to send to the *robot* agent, and so on.

3.1. Cooperation between agents

Given that all the agents compete for the available resources of the robot, a minimum of coordination is required between them. Generally, this problem comes up in the robot control architectures when various behaviors want to access a certain resource at the same time. Coordination between them is therefore necessary to achieve suitable global behavior. To coordinate behaviors, different alternative mechanisms are studied. These can be divided into two large groups [20]: arbitration and command fusion. Those of the first group choose a behavior from among those that are competing for the resource, and they give it control until the next selection cycle. Those of the second group combine the recommendations from the different behaviors to form a single control action representing their consensus (all the behaviors contribute simultaneously to the resource control).

The majority of the robot control architectures use these coordination mechanisms in a centralized way. That is, there is a central module that knows what actions are in conflict and imposes a decision.

Centralizing the decision can be a problem when there are many behaviors in the architecture, and a distributed coordination approach might be more suitable. We propose a *peer-to-peer* coordination mechanism between the agents in conflict. In our architecture, conflicts can appear between the *avoid* and *goto* behavior agents when they try to send contradictory actions to the *robot* agent and between the *battery charger* and the *task planning* agents when they request from the *path planning* agent a specific path, as depicted in Fig. 1. The present work, however, is focused only on the conflicts between the *goto* and the *avoid* agents. However, if new agents are added, they can know other conflicting agents when they register with the *directory facilitator agent*.

Negotiations to obtain control over the conflicting resource are based on the value calculation of the *utility* of the action required by each agent.

3.1.1. Utility function

The *utility* of an agent is a function that represents the benefit the system will receive if it carries out the proposed action from the point of view of the agent. The value of the utility varies in the interval [0, 1], 1 being the maximum value.

Both the *goto* agent and the *avoid* agent have defined utility functions, whose value they use to negotiate the sending of linear and angular velocities to the *robot* agent.

Hence, the *goto* agent calculates its utility as a function of the distance that remains to reach the destination location as:

$$p\text{Goto} = \alpha * d_l + \beta \tag{1}$$

being:

$$\alpha = \frac{p_{\max} - p_{\min}}{u_L - u_H} \qquad \beta = \frac{u_L * p_{\min} - u_H * p_{\max}}{u_L - u_H} \tag{2}$$

where $p_{\max}$ and $p_{\min}$ are the maximum and minimum values, respectively, of pGoto ($p_{\min} \le p\text{Goto} \le p_{\max}$); and u_L and u_H represent distance thresholds, which are found empirically.

Fig. 2(a) shows the generic utility function of the goto agent, while Fig. 2(b) shows the utility function with the numerical values of the different parameters used in the experiments.

For its part, the avoid agent calculates its utility as a function of the distance that remains before a collision with the obstacle (d_o) and the orientation of the robot with respect to the obstacle (h_o) (see Fig. 3), as:

$$p\text{Avoid} = p_h \cdot p_d \tag{3}$$

with:

$$p_h = \alpha_1 * h_o + \beta_1$$
$$p_d = \alpha_2 * d_o + \beta_2 \tag{4}$$

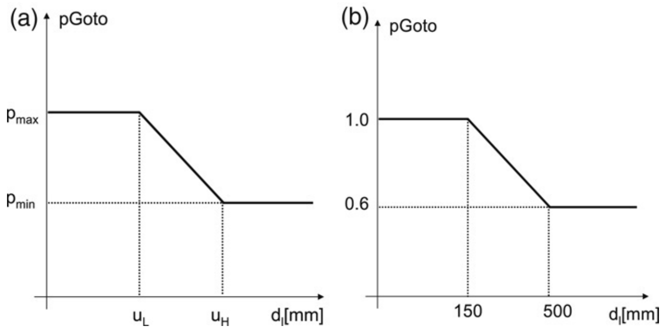

Fig. 2. Utility function of the *goto agent*: (a) the generic function definition and (b) utility function with numeric values.

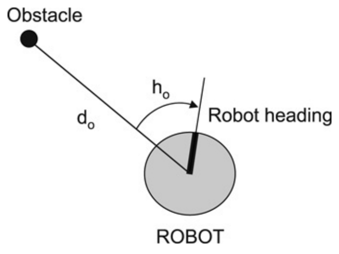

Fig. 3. Parameters involved in the calculation of the utility of the *avoid agent*.

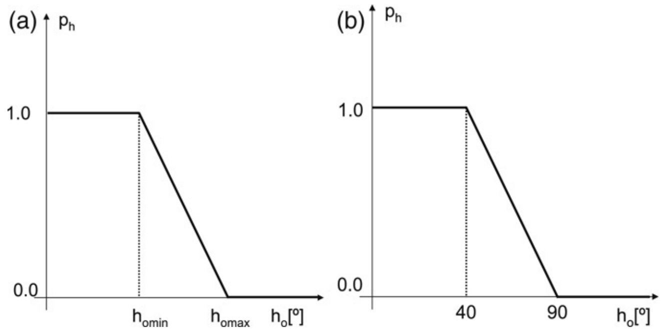

Fig. 4. Utility function corresponding to the orientation in the *avoid agent*.

and α_i and β_i defined by (2), in this case $p_{\max}$ and $p_{\min}$ being the maximum and minimum values of p_h ($p_{h\min} \le p_h \le p_{h\max}$) and those of p_d ($p_{d\min} \le p_d \le p_{d\max}$), respectively.

Fig. 4(a) shows the generic function corresponding to the orientation while Fig. 4(b) shows the numerical parameters used in the experiments. In the case of p_d (distance to the obstacle), there are two generic parameters $d_{\min}$ and $d_{\max}$ that change over time. We have made them proportional to the velocity in such a way that the value of p_d actually reflects the time that would remain before the robot collided with the obstacle if it were going directly toward it. Applying Eq. (3), the time until collision, assuming that the robot is headed directly to the obstacle, is taken into account, as is the orientation.

Once the values of the utility are calculated, the agent controlling the robot sends a message to the other agent in conflict to inform it of the value of its utility. If this value is greater than that calculated by the receiving agent, the second agent does not respond to the message, and the initial agent continues controlling the robot. If, on the other hand, the utility of the receiving agent is greater, it then answers the message

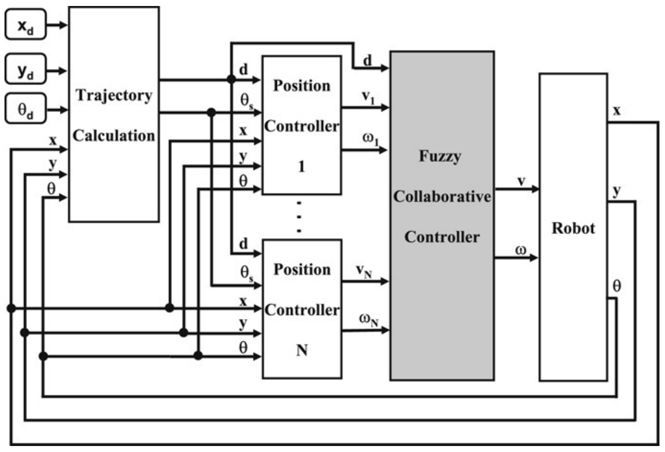

Fig. 5. Block diagram of the cooperative control of the *goto* agent.

assuring itself that the other agent knows that it will take control of the robot. This last agent will have control until its utility is smaller than that of the other agent.

The way to select the behavior that takes control is contained in the arbitration methods defined by [20], but the decision about which agent takes control is made by the agent itself, to prevent only one agent from centralizing the entire architecture.

4. Cooperative control of the *goto* agent

This section presents the cooperative control method, based on fuzzy logic, used to combine multiple controllers in the behavior of the *goto* agent. Instead of developing only one very elaborate controller, we have designed several simple controllers aimed at treating different aspects of the control separately and unifying their actions to obtain a final complex behavior. The Sugeno fuzzy inference methodology provides the adequate way of modelling the relevance of the various controllers.

The control loop of cooperative control proposed for the *goto* agent is shown in Fig. 5. The fuzzy collaborative controller block is in charge of the mix of the desired controller velocities to convert progressively from one controller to another.

4.1. Fuzzy collaborative controller

The starting point is the equation proposed by [8], which is extended by adding the weights corresponding to the relevance of each controller, in accordance with the current context. Thus, the calculation of the desired velocity is defined as a weighted average of the commands provided by the simpler controllers, according to the following formulae:

$$v = \frac{\sum_{i=1}^{n} v_i \cdot \mu_i}{\sum_{i=1}^{n} \mu_i} \qquad \omega = \frac{\sum_{i=1}^{n} \omega_i \cdot \mu_i}{\sum_{i=1}^{n} \mu_i} \tag{5}$$

where v_i is the linear velocity and ω_i is the angular velocity desired for the robot by the ith controller at any given moment of time, v the real linear velocity of the robot, ω the real angular

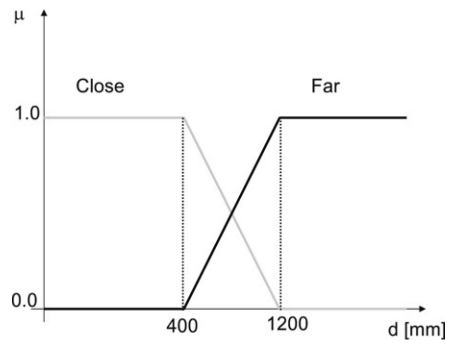

Fig. 6. Fuzzy sets.

velocity of the robot, n the number of controllers and μ_i the weights of each controller. These weights satisfy the condition $\sum_{i=1}^{n} \mu_i = 1$.

It is possible, with the weights, to give more or less importance to all of the controllers. In order to introduce this adjustment of velocities expressed by equation Eq. (5), a Sugeno functional fuzzy system is proposed [5], since its output can be a linear function and the implementation of equation Eq. (5) is straightforward.

In a Sugeno approach, rules have the following structure:

IF **d** IS dlabel1 AND C_1 IS C1label1 AND... C_N IS CNlabeln
THEN **v** IS vlabel1 AND ω IS ωlabel1

where $d, C_1, \ldots, C_N$ are the input variables and v and ω are the output variables.

4.2. Input variables

According to Fig. 5 the inputs of the collaborative block are the distance (d) and the linear (v) and the angular (ω) speeds of the different controllers. For each input variable several labels are defined (for example, dlabel1), and are modeled by membership functions. Inputs are fuzzified by applying the membership functions to their current values.

d is defined as a function of the distance that the robot has already covered d_{rec} and the distance from the initial position of the robot to the destination point $d_{\max}$, i.e. $d = d_{\max} - d_{\text{rec}}$. In this case, two possible values for d have been defined: *near* and *far*. The terms *near* and *far* are modeled with fuzzy sets and defined as:

$$\mu_{\text{near}}(d) = \left\{ \begin{array}{ll} 1 & d \le \min \\ \dfrac{(-d + \max)}{(\max - \min)} & \min < d < \max \\ 0 & d \ge \max \end{array} \right\} \tag{6}$$
$$\mu_{\text{far}}(d) = 1 - \mu_{\text{near}}(d)$$

where min and max are parameters that have been empirically tuned. Fig. 6 shows these parameters for the current implementation.

According to this definition, the distance to the destination point can vary in a nonlinear way in time, depending on the movement of the robot, causing the fuzzy weights also

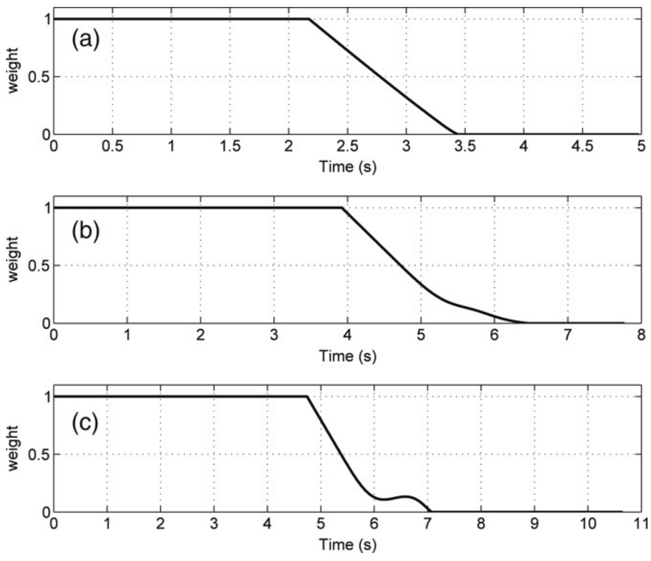

Fig. 7. Fuzzy weights resulting after applying the near fuzzy set.

to be nonlinear. Fig. 7 shows different results for the fuzzy weights described by (6) over time. In Fig. 7(a) a linear case is presented, while Fig. 7(b) and (c) reveal the nonlinearity of the equation, which also depends on the min and max parameters.

The controller's inputs (linear and angular speeds) have not been fuzzified. Currently we have designed two different controllers to setup the collaborative controller. One of them is very fast (but rough) and the other is precise (but slow); so the total amount of input variables is currently 5.

4.3. Output variables

As output variables (conclusion of the rule) we have the final linear and angular velocities. The values of the output variables are determined as linear combinations of the inputs (Sugeno's system with $n+1$ variables). In other words, each value x of the output variable s is computed as follows:

$$\begin{aligned} u_x(s) &= a_{n+1}\cdot v_{n+1}+\cdots+a_1\cdot v_1+a_0 \\ &= (a_{n+1},\ldots,a_1,a_0) \end{aligned} \tag{7}$$

where a_i is the coefficient of the input variable v_i and a_0 is a constant. The final output of the system is the sum of the individual outputs averaged by the inputs (that is, using the average function as the defuzzification method).

The following output vectors have been defined: $u_{\text{slow}}(v) = (0,0,1,0)$ and $u_{\text{slow}}(\omega) = (0,0,1,0)$ and $u_{\text{fast}}(v) = (0,1,0,0)$ and $u_{\text{fast}}(\omega) = (0,1,0,0)$ where variable $n+1=3$ corresponds to the coefficient of the distance d to the objective, $n=2$ to the fast controller (v_2,ω_2) and $n-1=1$ to the slow controller (v_1,ω_1). For the constant a_0, 0 has been chosen.

4.4. Rules

The rules that describe the fuzzy system are:

If **d** is near then **v** is slow and ω is slow
If **d** is far then **v** is fast and ω is fast.

The function chosen for evidence propagation (from premises to conclusions) is the product. As a consequence, if the first rule is activated, $\mu_{\text{near}}(d)\cdot v_2$ and $\mu_{\text{near}}(d)\cdot\omega_2$ are obtained as results for the linear and the angular speeds respectively. If the second rule is activated, it gives $\mu_{\text{far}}(d)\cdot v_1$ and $\mu_{\text{far}}(d)\cdot\omega_1$. The combination of both rules is the result expressed in Eq. (5).

5. Implementation and results

To test the new method, a multi-agent architecture has been developed on an ad hoc multi-agent platform, programmed in C++ on Linux because the majority of commercial platforms are not capable of responding in real time and normally have an agent that centralizes the entire platform. The experiments are carried out in an ActivMedia Pioneer 2DX robot. Included in the architecture are the perception, actuator and behavior agents and a very simple *path planning* agent. Note that the *battery charger* agent does not intervene in the experiments because, without no conflicts with the *goto* and *avoid* agents, it does not act directly on the velocities but rather asks the *path planning* agent to search for the fastest path to reach the battery charging point.

For the *goto* agent, it was decided to implement two different controllers and to mix the two resulting control vectors using fuzzy weight factors, as explained in Section 4.1. There is a PID to control the distance (for v) and another one for the orientation (for ω) of the robot. The controllers are independent thanks to a trajectory calculation module (see Fig. 5) which calculates the distance and the orientation the robot has to follow to arrive at the destination point, based on the information the *goto* agent receives from other agents of the architecture (*path planning, client* and *encoder* agents). This block computes the orientation and the distance the robot has to follow, instant-by-instant, to arrive at the goal. The distance and orientation values this module provides are passed on to the PID controllers.

One particular aspect of these controllers is that they use S-approximation to obtain physically possible control vectors **u(t)** instead of using T-approximation, as explained in [18]. These approximations have to be used when one or more of the components of the velocity vector go beyond the limit imposed by the actuator. The first proportionally modifies the two components, while the second only modifies the affected component (reducing its maximum value). As a result, with S-approximation, the final velocity is smaller than desired, but the orientation continues to be what is required, in this way avoiding the errors produced in the T-approximation.

In the case of the fast controller, the linear velocity is limited to $v = 1000$ mm/s and the angular velocity to $\omega = 100°$/s, while in the precise controller, the limitations are $v = 200$ mm/s and $\omega = 45°$/s.

5.1. Experimental results

The main objective is to obtain a robot control architecture that provides coherent and rational conduct aimed at achieving the goal that was set and, at the same time, preserving the real-time response to the immediate physical environment [23]. In

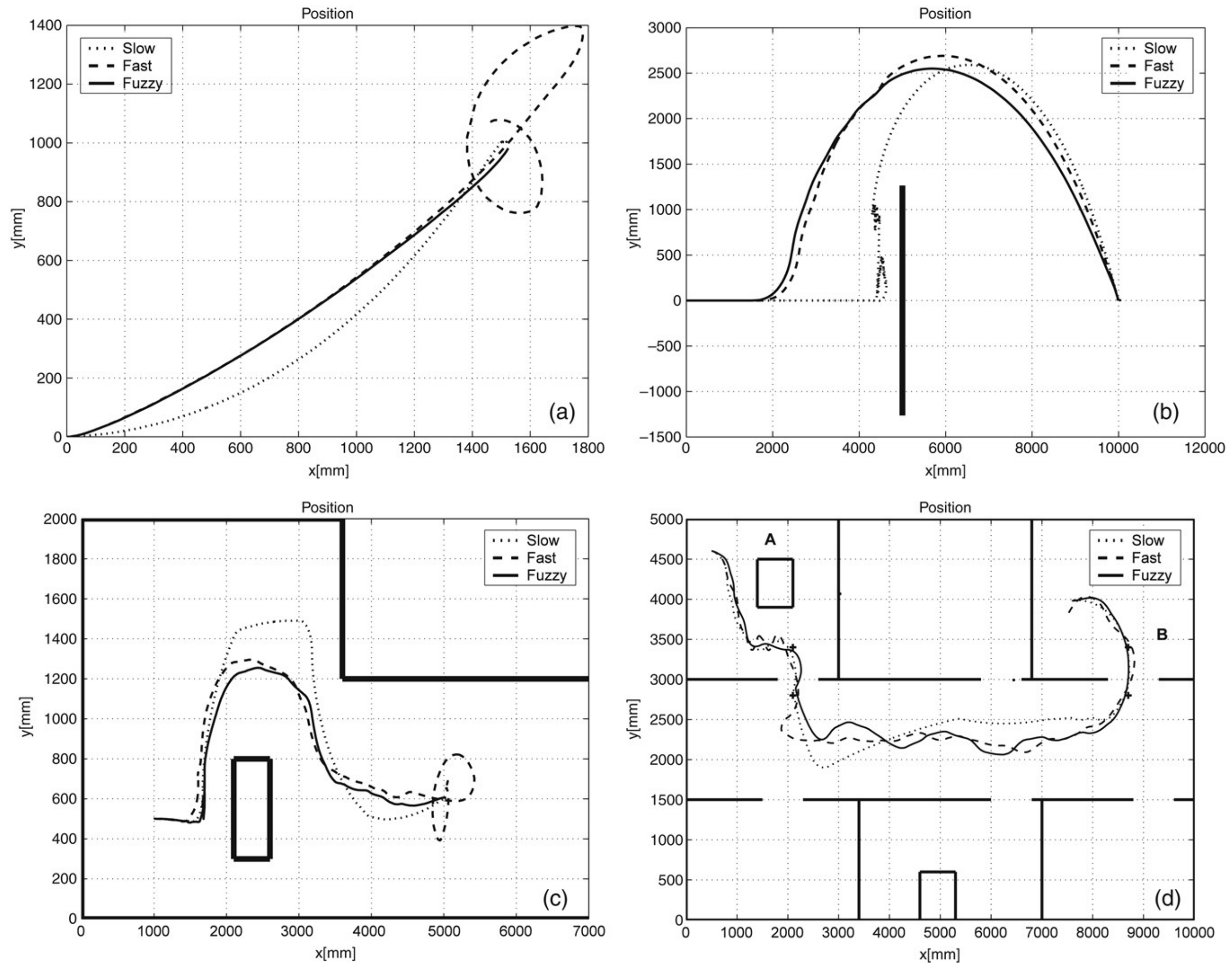

Fig. 8. Scenarios with an example of the trajectory for the three *goto* agent implementations: (a) Scenario 1, (b) Scenario 2, (c) Scenario 3 and (d) Scenario 4.

this sense, multi-agent systems provide the basic architecture, while cooperative control makes the *goto* agent, which is in charge of leading the robot to the desired position with the required orientation, fast and precise in all situations.

This section presents four scenarios used to test the performance of the multi-agent architecture with collaborative control.

Scenario 1 (*Sc*1): In this scenario, the robot must go from the initial position $(x_0, y_0, \theta_0) = (0.0, 0.0, 0.0)$ to a target position $(x_f, y_f, \theta_f) = (1.5, 1.0, 0.0)$, the position being expressed in meters and the orientation in degrees, as can be seen in Fig. 8(a). In this scenario there are no obstacles present in the path of the robot, so the *avoid* agent does not intervene.

Scenario 2 (*Sc*2): In this setting, an obstacle is introduced into the path of the robot. So, the *avoid* agent intervenes in the robot guidance. The initial position is $(x_0, y_0, \theta_0) = (0.0, 0.0, 0.0)$ and the final is $(x_f, y_f, \theta_f) = (10.0, 0.0, 0.0)$ as can be seen in Fig. 8(b). There is a wall with a length of 2.56 m located 5 m from the robot.

Scenario 3 (*Sc*3): In this example, the robot moves through a corridor with a column in the middle. The initial position of the robot is $(x_0, y_0, \theta_0) = (1.0, 0.5, 0.0)$, the final position is at $(x_f, y_f, \theta_f) = (5.0, 0.6, 0.0)$, as can be seen in Fig. 8(c). Again, the *avoid* agent intervenes in the robot guidance.

Scenario 4 (*Sc*4): In this scenario, the robot must go from room A to room B. In this case, the *path planning* agent intervenes in the calculation of the trajectory, giving the *goto* agent the points that are necessary to pass through the doors, as can be seen in Fig. 8(d).

To carry out the experiments, three different versions have been implemented for the *goto* agent: the first uses only the fast controller, the second uses the precise controller and the third version uses fuzzy cooperative control. The experiments are carried out separately for each implementation of the *goto* agent and the results of each of them are compared according to the following measures:

- *Travelled distance (TD)*: the distance travelled by the robot to reach the goal.
- *Final orientation (FO)*: the heading of the robot at the goal position.
- *Total time (TT)*: the total amount of time the robot needs to achieve the goal.
- *Precision (P)*: how closed to the goal position is the center of mass of the robot.

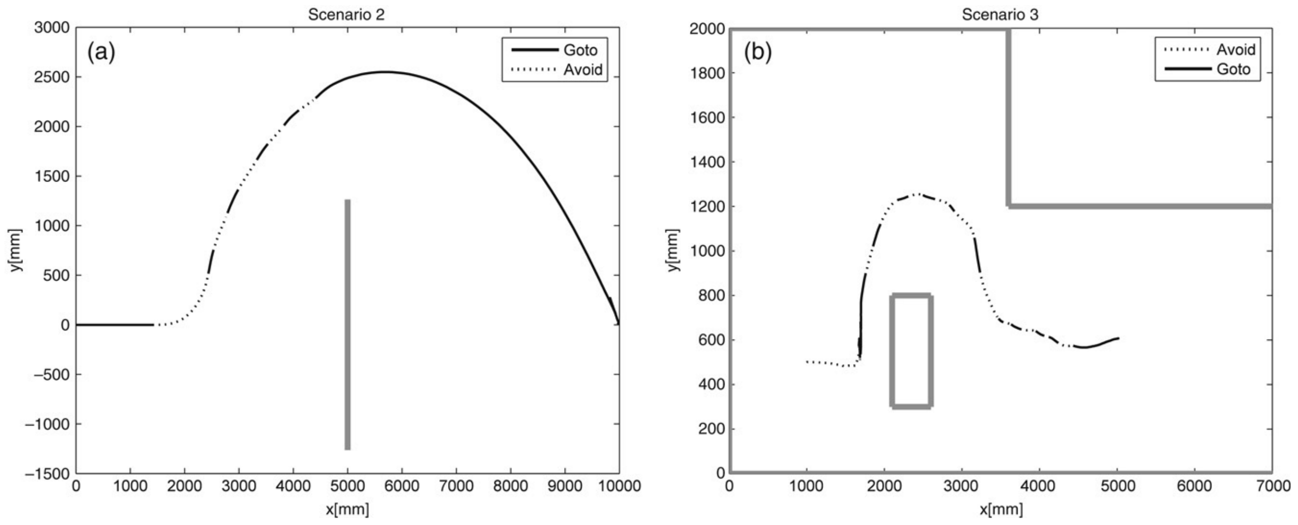

Fig. 9. Interaction of the *goto* and *avoid* agents.

- *Time goto (TG)*: the total time the *goto* agent has the robot control.

The experiments consisted of five executions in each scenario for each implementation of the *goto* agent. Table 1 shows the average and standard deviation of each evaluation measure and scenario. Regarding the statistical significance of the obtained results, the *p* values (with $t_{0.05}$) are the expected ones.

The travelled distance (*TD*) has the lower value for the fuzzy collaborative controller except for scenario 4. To explain this behavior, we can look at the particular trajectories described by the robot shown in Fig. 8. In Fig. 8(a) it can be observed that the cooperative control behavior follows the fast controller at the beginning and the precise one at the end, obtaining a curve which comes closer to the ideal (which would be a straight line between the initial and the final points). This happens in the other scenarios too, but the interaction with the *avoid* agent hides it. In Scenario 4, the *path planning* agent gives several points between the initial position and the desired goal. These points, marked with an *x* in Fig. 8(d), are situated near the doors. In this case, the collaborative controller follows both controllers several times during the execution of the trajectory.

The total time (*TT*) consumed by the precise controller to reach the goal position is the highest. The collaborative controller times are similar to those of the fast controller, so getting the benefits of it.

Even though the precision (*P*) of all the controllers is above 90%, the travelled distance (*TD*) approximates more to the minimum distance (i.e. a straight line to the goal position) in all the scenarios with the collaborative controller. On the other hand, the final orientation (*FO*) of the collaborative controller (as well as for the precise controller) approximates the desired orientation better.

Regarding the time goto (*TG*), its value is higher with the precise controller because, as it is slow, the *avoid* agent does not need to take control of the robot very often, only when the obstacle is very near the robot. This is not true for scenario 2 where the obstacle is so long that the *avoid* agent needs more time to dodge it. With the fast controller it is just the contrary: the *avoid* agent needs to intervene more often. The exception is again scenario 2; as this controller is very fast, both agents interacting dodge the obstacle faster, so the *goto* agent has the control more often. For the collaborative controller this measurement is between both values, the *TG* of the fast controller and the *TG* of the precise controller, as expected. Fig. 9 shows the interaction between the *goto* and *avoid* agents for the scenarios 2 and 3 with the fuzzy collaborative controller. When it has control, the *goto* agent is represented by a continuous line, while the avoid *agent* is represented by a dotted one.

Summarizing, we can conclude that the *goto* agent with the fuzzy collaborative controller is capable of reaching the goal precisely, with the desired heading and in a reasonable time. On the other hand the interaction with the *path planning* agent and coordination with the *avoid* agent let the robot perform complex trajectories with obstacles on them, using a distributed coordination mechanism.

6. Conclusions

The challenges of robotics have led researchers to develop control architectures composed of distributed, independent and asynchronous behaviors. The appearance and evolution of multi-agent systems have allowed modules to be transformed into agents and the properties of this kind of system to be taken advantage of in robot control architectures. But the majority of these architectures centralize the coordination of the behaviors in a single module or agent, which represents a problem when there is more than one behavior competing for the available resources.

On the other hand, some previous work has taken advantage of the progress made in the field of cooperative control to develop complex behaviors, in which various controllers share control of the system.

This work presents a robot with an architecture based on the integration of both research lines, multi-agent systems and cooperative control. To the best of our knowledge, this paper is

Table 1
Comparison of the *goto agents* for the three different controllers

Controller	Parameters	Sc1	Sc2	Sc3	Sc4
	TD	5.55 ± 0.43 m	13.37 ± 0.40 m	6.10 ± 1.38 m	15.71 ± 2.85 m
	FO	7.58 ± 4.71°	49.18 ± 44.56°	25.96 ± 29.76°	56.75 ± 67.98°
Fast	*TT*	14.28 ±1.29 s	27.78±3.80 s	28.20 ± 6.67 s	64.66 ± 13.5 s
	P	99.46 ± 0.56%	99.36 ± 0.25%	97.83 ± 0.49%	94.40 ± 5.78%
	TG	100 ± 0%	90.19 ± 2.82%	50.07 ± 5.23%	50.22 ± 4.89%
	TD	5.17 ± 0.05 m	17.92 ± 2.84 m	5.78 ± 0.38 m	12.26 ± 0.10 m
	FO	1.37 ± 1.13°	1.77 ± 2.93°	2.58 ± 0.90°	2.97 ± 2.93°
Precise	*TT*	34.50 ± 6.25 s	164.40 ± 20.91 s	40.62 ± 4.48 s	79.00 ± 2.77 s
	P	99.81 ± 0.16%	99.85 ± 0.07%	99.83 ± 0.10%	99.79 ± 0.11%
	TG	100 ± 0%	80.82 ± 5.06%	81.95 ± 3.20%	85.99 ± 1.53%
	TD	5.11 ± 0.01 m	12.46 ± 0.66 m	5.08 ± 0.30 m	12.55 ± 0.44 m
	FO	1.95 ± 0.71°	1.76 ± 0.84°	1.97 ± 1.01°	1.77 ± 1.63°
Collaborative	*TT*	16.56 ± 0.44 s	27.76 ± 1.22 s	27.90 ± 1.35 s	66.06 ± 8.79 s
	P	99.41 ± 0.22%	99.67 ± 0.06%	99.55 ± 0.16%	99.78 ± 0.13%
	TG	100 ± 0%	88.40 ± 2.29%	58.68 ± 4.27%	66.79 ± 2.31%

the first integrating multi-agent and cooperative control architectures. In particular, attention has been focused on the design of the *goto* agent, which uses fuzzy tuning of two different position controllers, and on using the utility functions in the *goto* and *avoid* agents to negotiate the control of the robot.

The multi-agent control architecture with cooperative control in the agents presents several desirable features such as flexibility, adaptability and easy design, since complex behaviors can be achieved using simple controllers and the whole architecture can be described at a higher level. As a consequence of having multi-agent systems, communication overhead has to be considered in order to ensure real-time response of the system so, a distributed coordination mechanisms with minimum communication requirements has been defined.

In order to test the method, a multi-agent architecture has been implemented with three different *goto* agents. Two different basic controllers, one fast and the other precise, and a fuzzy cooperative controller combining the control actions were developed to set up the different implementations of the *goto* agent. Afterwards, various experiments were carried out to compare the efficiency of the simple controllers with that of the cooperative controller, especially in situations in which negotiation with the *avoid* agent and interaction with the *path planning* agent are required to perform complex trajectories.

With the cooperative fuzzy control, better results have been obtained with respect to the time necessary to reach the goal and to the precision with which it is achieved, in terms of position as well as orientation.

As for future work, it is expected that the proposed solution will be extended to the case of *n* controllers. Also, new behavioral agents will be included in the architecture to analyze the scalability of our approach.

References

[1] J. Bryson, Intelligence by design: Principles of modularity and coordination for engineering complex adaptive agents, Ph.D. Thesis, Massachusetts Institute of Technology, 2001.

[2] D. Busquets, C. Sierra, R. López de Màntaras, A multi-agent approach to qualitative landmark-based navigation, Autonomous Robots 15 (2003) 129–154.

[3] J.L. de la Rosa, et al., Rogi team real: Research on physical agents, in: Veloso, Pagello, Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, 1999, pp. 434–438.

[4] M. Dorigo, et al., Evolving self-organizing behaviors for a swarm-bot, Autonomous Robots 17 (2004) 223–245.

[5] D. Driankov, H. Hellnoorn, M. Reinfrank, An Introduction to Fuzzy Control, Springer, 1991.

[6] A. Figueras, J. Colomer, J. de la Rosa, Supervision of heterogeneous controllers for a mobile robot, in: The XV World Congress IFAC, 2002.

[7] FIPA: Foundation for Intelligent Physical Agents. On-line: http://www.fipa.org/.

[8] B. Gerkey, M. Mataric, G. Sukhatme, Exploiting physical dynamics for concurrent control of a mobile robot, in: Proceedings ICRA '02. IEEE International Conference on Robotics and Automation, vol. 4, 2002, pp. 3467–3472.

[9] K. Goldberg, B. Chen, Collaborative control of robot motion: Robustness to error, in: Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001, pp. 655–660.

[10] H. Hu, D. Gu, A multi-agent system for cooperative quadruped walking robots, in: Proceedings of the IASTED International Conference Robotics and Applications, 2000, pp. 1–5.

[11] B. Innocenti, B. López, J. Salvi, How MAS support distributed robot control, in: International Symposium of Robotics, ISR, 2006. Conf./Page: 1956/0020 Archive Number: 330319560020.

[12] N. Jennings, S. Bussmann, Agent-based control systems: Why are they suited to engineering complex systems? IEEE Control Systems Magazine 23 (3) (2003) 61–73.

[13] G.J. Klir, T.A. Folger, Fuzzy Sets, Uncertainty and Information, Prentice Hall, 1992.

[14] M. Kolp, P. Giorgini, J. Mylopoulos, Multi-agent architectures as organizational structures, Autonomous Agents and Multi-Agent Systems 13 (2006) 3–25.

[15] R.R. Murphy, Introduction to AI Robotics, The MIT Press, 2000.

[16] R. Murray, K. Åström, S. Boyd, R. Brockett, G. Stein, Future directions in control in an information-rich world, IEEE Control Systems Magazine 23 (2) (2003) 20–33.

[17] M.C. Neves, E. Oliveira, A multi-agent approach for a mobile robot control system, in: Proceedings of Workshop on Multi-Agent Systems: Theory and Applications (MASTA'97 - EPPIA'97), Coimbra, Portugal, 1997, pp. 1–14.

[18] D. Omerdic, G. Roberts, Thruster fault diagnosis and accommodation for open-frame underwater vehicles, Engineering Practice 12 (2004) 1575–1598.

[19] Y. Ono, H. Uchiyama, W. Potter, A mobile robot for corridor navigation: A multi-agent approach, in: ACMSE'04: ACM Southeast Regional Conference, ACM Press, 2004, pp. 379–384.

[20] P. Pirjanian, Behavior coordination mechanisms — state-of-the-art, Tech. Rep. IRIS-99-375, Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California, 1999.

[21] P. Ridao, M. Carreras, J. Batlle, O2CA2, a new object oriented control architecture for autonomy: The reactive layer, Control Engineering Practice 10 (8) (2002) 857–873.

[22] R. Ros, R. López de Màntaras, C. Sierra, J.L. Arcos, A CBR system for autonomous robot navigation, in: Frontiers in Artificial Intelligence and Applications, vol. 131, IOS Press, 2005, pp. 299–306.

[23] J.K. Rosenblatt, DAMN: A distributed architecture for mobile navigation, Ph.D. Thesis, Robotics Institute at Carnegie Mellon University, 1997.

[24] A. Saffiotti, The uses of fuzzy logic in autonomous robot navigation, Soft Computing 1 (4) (1997) 180–197.

[25] O. Sauer, G. Sutschet, Agent-based control, IET Computing & Control Engineering (2006) 32–37.

[26] L.-K. Soh, C. Tsatsoulis, A real-time negotiation model and a multi-agent sensor network implementation, Autonomous Agents and Multi-Agent Systems (November) (2005) 215–271.

[27] W. Spears, D. Spears, J. Hamann, R. Heil, Distributed, physics-based control of swarms of vehicles, Autonomous Robots 17 (2004) 137–162.

[28] M. Wooldridge, Intelligent agents, in: Gerhard Weiss (Ed.), Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press, 1999, pp. 27–78.

Bianca Innocenti graduated in Electronic Engineering from the National University of San Juan (Rep. Argentina) in 1997 and graduated in Automation and Industrial Electronic Engineering at 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 Ph.D. program Industrial Informatics and Advanced Control Technologies in October 2000. At present, she is an assistant professor in the Department of Electronics, Computer Science and Systems Engineering at the University of Girona. Her current research interests are in the field of mobile robotics and multi-agent systems.

Beatriz López graduated in Computer Science from the Autonomous University of Barcelona (UAB, Barcelona, Spain) in 1986. Two years later she joined the Artificial Intelligence Research Institute of the Spanish Scientific Research Council (CSIC, then located at the Centre of Advanced Studies of Blanes, CEAB) where she received the Ph.D. degree in Computer Science from the Technical University of Catalonia (UPC, Barcelona, Spain) in 1993. She was associate professor from 1992–1995 and 1998–2000 at the Rovira Virgili University. She has also served 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 in the University of Girona, Spain. Her research interests include multi-agent systems, planning and scheduling, and case-based reasoning. She is a member of the Catalan Association for Artificial Intelligence which belongs to the European Coordination Committee on Artificial Intelligence (ECCAI).

Joaquim Salvi graduated in Computer Science from the Technical University of Catalonia, Catalonia, Spain in 1993, and received the D.E.A. degree in computer science in July 1996, and the Ph.D. degree in industrial engineering in 1998, from the Computer Vision and Robotics Group, University of Girona, Girona, Spain. He is currently an associate professor at the Electronics, Computer Engineering and Automation Department, University of Girona. He is involved in some governmental projects and technology transfer to industrial environments. His current interests are in the field of computer vision and mobile robotics, focused on structured light, stereovision, and camera calibration. He is the leader of the 3D Perception Lab. Dr. Salvi received the Best Thesis Award in engineering for his Ph.D. Dissertation.