Quick Design of Fuzzy Controllers With Good Interpretability in Mobile Robotics

Manuel Mucientes and Jorge Casillas

***Abstract*—This paper presents a methodology for the design of fuzzy controllers with good interpretability in mobile robotics. It is composed of a technique to automatically generate a training data set plus an efficient algorithm to learn fuzzy controllers. The proposed approach obtains a highly interpretable knowledge base in a very reduced time, and the designer only has to define the number of membership functions and the universe of discourse of each variable, together with a scoring function. In addition, the learned fuzzy controllers are general because the training set is composed of a number of automatically generated examples that cover the universe of discourse of each variable uniformly and with a predefined precision. The methodology has been applied to the design of a wall-following and moving object following behavior. Several tests in simulated environments using the Nomad 200 robot software and a comparison with another learning method show the performance and advantages of the proposed approach.**

***Index Terms*—Ant colony optimization, behavior design, fuzzy control, learning, mobile robot navigation.**

I. Introduction

AUTONOMOUS mobile robots are those robots that have the ability to move and perform tasks in real environments without human supervision. According to [1], control tasks on a robot can be of two different kinds: reactive and deliberative. Reactive control uses the current sensor information to select the control action, while deliberative control also uses previous sensor information and, as a result, plans about future positions and actions of the robot. Modern control architectures for this kind of robots are hybrid, thus they have several layers: at the lowest layer all the reactive control is grouped, while on the top layer the planning tasks are done. In that way, the robot is able to implement complex tasks and react to changes in the environment (obstacles moving, people appearing, etc.).

The reactive layer is usually implemented with behaviors (tasks like wall-following, go through a door, follow a person, avoid a moving object, etc.) that are coordinated by the planning layer. The environments in which an autonomous robot moves are unconstrained and have a high amount of uncertainty. Furthermore, information provided by robot sensors is noisy and unreliable. This problem becomes more important when data from ultrasound sensors are used because of low angular resolution and specular reflection. Fuzzy logic has shown to be a useful tool when dealing with this uncertainty and has been widely used for the design of behaviors in robotics [2], [3].

However, the design of fuzzy controllers has mainly two problems. On the one hand, the knowledge about the task to be controlled must be extracted, and sometimes this is very difficult. On the other, the designer usually has to spend a long time tuning the controller [4]. Due to these difficulties, the use of learning methods for the design of fuzzy controllers has been generalized [5]. There are different approaches: evolutionary algorithms [6]–[13], neural networks [14], [15], reinforcement learning [16]–[23], a combination of neural networks and evolutionary algorithms [24]–[27], etc. Evolutionary algorithms have some characteristics that make them especially useful for the design of fuzzy controllers: they are flexible to design different components of a controller, constraints can be easily included, and they let the designer decide the most adequate tradeoff between interpretability and accuracy for a specific controller.

The approaches that can be found in the bibliography have different shortcomings. Some of them take a long time to learn the behaviors [6], [7], [11], [17], [21]. Others need the definition of a lot of parameters that depend on the problem and/or environment, or a partial description of a knowledge base [9], which complicates the design of new behaviors since parameters must be tuned and expert knowledge has to be acquired. Besides, sometimes the learned behavior is not general [7], [9], [11], [27], [28]; thus the performance is adequate in some environments but poor in others. As a result, the learned behavior is not reliable and its implementation on the real robot is not adequate. Finally, the interpretability of some of the learned controllers is poor [6], [15], [21], [25], [27]–[30] and, as a consequence, it is difficult to detect and solve errors during the operation of the controller.

This paper presents a methodology to design behaviors in mobile robotics that tries to address the above mentioned drawbacks. This methodology consists of two parts: a *novel technique* to automatically generate data sets plus a *learning method* to automatically design fuzzy controllers from them.

The proposed technique automatically generates data sets for the design of behaviors in mobile robotics. It aims at providing a learned behavior totally general where the robot implements a valid control action in any situation. Therefore, the technique addresses two of the above-mentioned drawbacks since it *simplifies the design of fuzzy controllers* and they are built to

Manuscript received January 25, 2005; revised October 17, 2005 and January 19, 2006. This work was supported in part by the Spanish Ministry of Education and Science under Grants TIC2003-00877, TIN2005-08386-C05-01, TIN2005-08521, and TIN2005-03844 and by the DXID of the Xunta de Galicia under Grant PGIDIT04TIC206011PR.

M. Mucientes is with the Department of Electronics and Computer Science, University of Santiago de Compostela, E-15782 Santiago de Compostela, Spain (e-mail: manuel@dec.usc.es).

J. Casillas is with the Department of Computer Science and Artificial Intelligence, University of Granada, E-18071 Granada, Spain (e-mail: casillas@decsai.ugr.es).

Digital Object Identifier 10.1109/TFUZZ.2006.889889

1063-6706/$25.00 © 2007 IEEE

properly work in any environment. To illustrate its performance, we have chosen two different behaviors (wall-following and moving object following) in this paper, though the technique could be easily adapted to other behaviors.

The technique "opens the door" to the automatic design of fuzzy robot controllers for any data-driven learning method. The technique automates a number of tasks in the fuzzy controller design, thus leaving only a few components to be defined by the expert. With the aim of addressing some of the above-mentioned drawbacks by performing a *quick learning* and providing fuzzy controllers with *good interpretability,* we propose to use a simple but effective method: the cooperative rules (COR) methodology [31]–[33]. In this paper, we have adapted this methodology to our problem by considering several consequent variables and using a different fitness function to facilitate the rule base reduction. Besides, we use ant colony optimization (ACO) [34] as a search algorithm to decrease the learning time. Several experiments in different simulated environments (Nomad 200 simulator) have been done in order to test the performance of the proposed method.

This paper is organized as follows. Section II introduces the technique to generate data sets for the wall-following and moving object following behaviors. Section III presents the considered learning method. Section IV shows the obtained results and a detailed experimental analysis. Section V outlines some conclusions.

II. Automatic Design of Behaviors in Mobile Robotics

Our proposed technique for the generation of data sets for learning fuzzy controllers in mobile robotics has the following steps.

1) Definition of the input and output variables, and calculation of the input values using the information provided by the sensors and the robot's odometry.
2) Definition of the universe of discourse, the number of fuzzy sets, and the precision (p_n) of each variable n.
3) Definition of the scoring function (SF), a function that evaluates the action of the fuzzy controller over an example.
4) Definition of the objective function, the index that measures the global quality of the encoded rule set. This function is independent of the behavior.
5) Robot simulation, in order to reduce the time needed for learning. The robot is modeled with a set of equations that are valid for all behaviors.
6) Construction of the training set. The set of examples is created combining the values of the variables of the antecedent part. The values for the consequent part are calculated by testing all the possible combinations of the discrete output values.

In order to describe the steps of this technique in detail, a couple of behaviors are used as examples: wall-following and moving object following. Similar steps could be applied for learning other behaviors.

- The *wall-following behavior* is usually implemented when the robot is exploring an unknown area or when it is moving between two points in a map, generally in indoor environments. A good wall-following controller is characterized by three features: to maintain a suitable distance from the wall that is being followed, to move at a high velocity whenever possible, and to avoid sharp movements, making smooth and progressive turns and changes in velocity.
  The controller can be configured modifying the values of two parameters: the reference distance (d_{wall}), which is the desired distance between the robot and the selected wall, and the maximum velocity attainable by the robot $(v_{\max})$. In this paper, it is assumed that the robot is going to follow a contour that is on its right side. Of course, the robot could also follow the left-hand wall by simply interchanging the sensor inputs.
- A mobile robot can implement the *moving object following behavior* for pursuing a person, or when it is cooperating with other robots in the implementation of a task and one of the robots is guiding the other ones. A good implementation of the behavior has to place the robot at the objective point $(x_{\text{obj}}, y_{\text{obj}})$ (see Fig. 1). This point is defined using the desired distance (d_{ref}) between the robot and the moving object, and the reference deviation $(\text{dev}_{\text{ref}})$, which is an angle that indicates the position of the robot with respect to the advance direction of the moving object. If $\text{dev}_{\text{ref}} = 0$, the robot will follow the moving object exactly behind it, while positive values of dev_{ref} indicate that the robot will be placed at the right of the advance direction of the object, and negative values to the left. Furthermore, a good controller for this behavior must implement smooth changes in the velocity and angle of the robot.

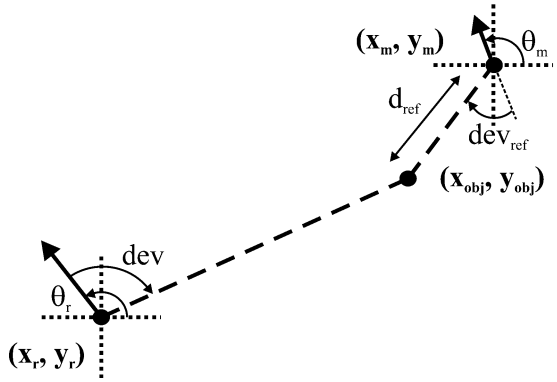


Fig. 1. Description of the points, distances, and angles needed for the calculation of the input variables for the moving object following behavior.

A. Preprocessing of the Variables

The first step in the design of the controller is the selection of the input and output variables.

1) Wall-Following: For the wall-following behavior, two of the input variables are the relative right-hand distance (RD) and the distances quotient (DQ), which are calculated as

$$\text{RD} = \frac{\text{right} - \text{hand distance}}{d_{\text{wall}}} \tag{1}$$

$$\text{DQ} = \frac{\text{left} - \text{hand distance}}{\text{right} - \text{hand distance}}. \tag{2}$$

DQ is the relative position of the robot inside a corridor, which provides information that is of greater relevance to the problem than simply using the left-hand distance. A high value for DQ means that the robot is closer to the right-hand wall, while a low value indicates that the closer wall is the left-hand one. In this case, the robot should approach to the right-hand wall, although the right-hand wall is lower than the reference distance. The other input variables are the relative linear velocity of the robot (LV)

$$\text{LV} = \frac{v_r}{v_{\max}} \tag{3}$$

where v_r is the real linear velocity of the robot and the orientation of the robot with respect to the wall it is following (θ_{wall}). A positive value of the orientation indicates that the robot is approaching the wall, while a negative value means the robot is moving away from the wall. The output variables are the relative linear acceleration (LA) and the relative angular velocity (AV)

$$\text{LA} = \frac{\text{linear acceleration}}{a_{\max}} \tag{4}$$

$$\text{AV} = \frac{\text{angular velocity}}{\omega_{\max}} \tag{5}$$

where $a_{\max}$ and $\omega_{\max}$ are the maximum linear acceleration and the maximum angular velocity attainable by the robot.

All the information used to calculate distances and orientations is obtained from the ultrasound sensors of the robot. The data are processed using *distributed perception* [35], and for this reason the sensors are grouped in different sets. Distances are measured as the minimum distance of a set of sensors (obviously, the set of sensors is different for RD and DQ). θ_{wall} will be a weighted sum of the orientation of each sensor in the set, giving more weight to those sensors that detect closer obstacles

$$\theta_{\text{wall}} = \frac{\sum_{i=1}^{n_u} \text{angle}_i \cdot \left(1 - \frac{d_i}{\max_{ud}}\right)}{\sum_{i=1}^{n_u} \left(1 - \frac{d_i}{\max_{ud}}\right)} \tag{6}$$

where n_u is the number of sensors in the set, angle_i is the angle of sensor i, d_i is the measured distance of this sensor, and $\max_{ud}$ is the maximum distance that a sensor can measure.

2) Moving Object Following: For this behavior, the input variables are the following.

- The distance between the robot and the objective point

$$d = \frac{\sqrt{(x_r - x_{\text{obj}})^2 + (y_r - y_{\text{obj}})^2}}{d_{\text{ref}}}. \tag{7}$$

- The deviation of the robot with respect to the objective point. A negative value indicates that the robot is moving in a direction to the left of the objective point, while a positive value means that it is moving to the right

$$\text{dev} = \arctan\left(\frac{y_{\text{obj}} - y_r}{x_{\text{obj}} - x_r}\right) - \theta_r. \tag{8}$$

- The difference of velocity between the robot and the object

$$\triangle v = \frac{v_r - v_m}{v_{\max}} \tag{9}$$

where v_r, v_m, and $v_{\max}$ are the linear velocities of the robot, the moving object, and the maximum velocity attainable by the robot (as has been previously defined for the other behavior).

- The difference of angle between the object and the robot

$$\triangle\theta = \theta_m - \theta_r. \tag{10}$$

The output variables are the LA (4) and the AV (5).

B. Universe of Discourse and Precision

The second step in the design is the definition of the universe of discourse, the number of fuzzy sets, and the precision (p_n) of each variable n. The universe of discourse is, for some variables (RD, DQ, θ_{wall} for wall-following, and $d, \triangle v, \triangle\theta$ for moving object following), a reduced version of the real universe of discourse, and it should contain those values of the variable that are meaningful for learning. For example, high values of distances are not useful during learning, because for all of them the robot will execute the same action. Therefore, it is enough to include only a few high values in the universe of discourse.

The same occurs with the precision of the variables. Precision is used to generate the examples. Very low values of p_n will generate a higher number of examples and, therefore, many of them will not be meaningful because there will be very similar examples. Selecting valid values for the universes of discourse and the precisions is not difficult for somebody who has defined the input and output variables, and it is always possible to select an extended universe of discourse or a lower precision. In the worst case, a higher number of examples will be generated (some of them useless) and learning will take more time.

For the wall-following behavior, different values for these parameters have been tested, giving a number of examples that range from 5070 to 23 085. As will be shown in the experimental section (Section IV), the learning algorithm can last from 4 m 5 s to 20 m in the worst case, which is also quite fast. These times include the time needed for the generation of the training set (about 1 s for the 5070 set). For this set, the following values have been used for the universes of discourse and precisions.

- universes of discourse: RD $\in [0, 3]$, DQ $\in [0, 2]$, $\theta_{\text{wall}} \in [-45, 45]$, LV $\in [0, 1]$, LA $\in [-1, 1]$, and AV $\in [-1, 1]$;
- precisions: $p_{\text{RD}} = 0.25$, $p_{\text{DQ}} = 0.5$, $p_{\theta_{\text{wall}}} = 7.5$, $p_{\text{LV}} = 0.2$, $p_{\text{LA}} = 0.125$, and $p_{\text{AV}} = 0.05$.

For the moving object following behavior, 6435 examples have been used and, as will be shown later, the controller was learnt in 7 m 19 s (includes 10 s for the generation of examples):

- universes of discourse: $d \in [0, 1]$, $\text{dev} \in [-180, 180]$, $\triangle v \in [-1, 1]$, $\triangle\theta \in [-20, 20]$, LA $\in [-1, 1]$, and AV $\in [-1, 1]$;
- precisions: $p_d = 0.1$, $p_{\text{dev}} = 30$, $p_{\triangle v} = 0.25$, $p_{\triangle\theta} = 10$, $p_{\text{LA}} = 0.125$, and $p_{\text{AV}} = 0.05$.

C. Scoring Function

An important aspect of the proposed data set generation technique is the definition of the SF (11), a function that evaluates the action of the fuzzy controller over an example. This function is behavior dependent. The role of the SF is to measure the deviation of each variable from the desired value (the one associated to the ideal state).

1) Wall-Following: For this behavior, SF is defined as

$$\mathrm{SF}(\mathrm{RB}(e^l)) = \alpha_1 + \alpha_2 + \alpha_3 \tag{11}$$

where e^l is the lth example and $\mathrm{SF}(\mathrm{RB}(e^l))$ is the score of the state reached by the robot starting at the state defined by e^l and applying the control action proposed by the fuzzy rule base RB. The values α_1, α_2, and α_3 are computed as follows:

$$\alpha_1 = 100 \cdot \frac{|\mathrm{RD} - 1|}{p_{\mathrm{RD}}} \tag{12}$$

$$\alpha_2 = 10 \cdot \frac{|\mathrm{LV} - 1|}{p_{\mathrm{LV}}} \tag{13}$$

$$\alpha_3 = \frac{|\theta_{\mathrm{wall}}|}{p_{\theta_{\mathrm{wall}}}}. \tag{14}$$

The parameters p_{RD}, p_{LV}, and $p_{\theta_{\mathrm{wall}}}$ are the precisions of the respective input variables. Precisions are used in these equations in order to evaluate the deviations of the values of the variables from the desired ones in a relative manner (the deviation of the value of variable n from the desired one is measured in units of p_n). This makes possible the comparison of the deviations of different variables and, as a consequence, the assignment of the weights for each one of the variables. These weights (100, 10, and 1 for (12)–(14), respectively) have been intuitively determined and indicate how much important the deviation in the value of a variable is with respect to the deviation of other variables. The highest weight has been assigned to the distance, as small variations of RD with respect to the reference distance should be highly penalized. An intermediate weight is associated to the velocity and, finally, the least important contribution to function SF is given by the orientation of the robot.

2) Moving Object Following: SF is defined for this behavior as follows:

$$\mathrm{SF}(\mathrm{RB}(e^l)) = \alpha_1 + \alpha_2 + \alpha_3 \tag{15}$$

where α_1, α_2, and α_3 are, respectively

$$\alpha_1 = 100 \cdot \frac{|d|}{p_d} \tag{16}$$

$$\alpha_2 = 10 \cdot \frac{|\mathrm{dev}|}{p_{\mathrm{dev}}} \tag{17}$$

$$\alpha_3 = \frac{|\triangle v|}{p_{\triangle v}}. \tag{18}$$

The parameters p_d, p_{dev}, and $p_{\triangle v}$ are the precisions of the respective input variables. The highest weight has been assigned to the distance, as the robot must be close to the objective point. An intermediate weight is associated to the deviation and, finally, the least important contribution to function SF is given by the difference in velocity.

D. Objective Function

The index that measures the global quality of the encoded rule set is independent of the behavior that is going to be learnt

$$f(\mathrm{RB}) = \frac{1}{2 \cdot \mathrm{NE}} \sum_{l=1}^{\mathrm{NE}} (g(e^l))^2 \tag{19}$$

where NE is the number of examples and $g(e^l)$ is defined as follows:

$$g(e^l) = \begin{cases} (1 - h(e^l)) \cdot \zeta + 1, & \text{if } h(e^l) \leq 1 \\ \exp(1 - h(e^l)), & \text{otherwise} \end{cases} \tag{20}$$

with ζ being a scaling factor that has been set to 1000 and

$$h(e^l) = \frac{\min(\mathrm{SF}(e^l)) + 1}{\mathrm{SF}(\mathrm{RB}(e^l)) + 1} \tag{21}$$

with $\min(\mathrm{SF}(e^l))$ being the minimum score that an action can obtain, for example, e^l (using the output values as described in Section II-F).

E. Robot Simulation

In order to reduce the time needed for learning, the simulation software of the Nomad 200 robot will only be used for testing the obtained controller. During learning, the movement of the robot will be modeled with the following set of equations (this model is valid for all behaviors, not only the two analyzed ones):

$$v_r(k) = v_r(k-1) + a_r(k) \triangle t \tag{22}$$

where $v_r(k)$ and $a_r(k)$ are the linear velocity and the linear acceleration of the robot at time k and $\triangle t$ is the time between two control cycles (a value of $\triangle t = 1/3s$ has been used)

$$\theta_r(k) = \theta_r(k-1) - \omega_r(k) \triangle t \tag{23}$$

where $\theta_r(k)$ and $\omega_r(k)$ are the orientation and the angular velocity of the robot at time k and

$$x_r(k) = x_r(k-1) + 2\, v_r(k)\, \triangle t \cos\left(\frac{\pi}{2} - \theta_r(k)\right) \tag{24}$$

$$y_r(k) = y_r(k-1) + 2\, v_r(k)\, \triangle t \sin\left(\frac{\pi}{2} - \theta_r(k)\right) \tag{25}$$

are the coordinates of the robot at time k.

The model assumes that the final v_r and θ_r are reached without time delay. To simulate the inertia of the robot in its movements, the new position is calculated as if there were two control cycles between orders [two in (24) and (25)], so the

selected accelerations and turnings are smoother (the robot will move a longer distance) and, on the contrary, decelerations must be harder.

F. Construction of the Training Set

The learning of the controller is done using a set of examples. As has been mentioned, depending on the selected values for the universes of discourse and the precisions, the number of examples can vary. In this paper, 5070 examples have been used for the wall-following behavior and 6435 for the moving object following behavior. The data set is automatically generated by starting from the minimum value of each variable and increasing the value in a quantity equal to p_n until the maximum value is reached; a number of different values for the variables is obtained. The set of examples is created combining these values for all the variables of the antecedent part.

The values of the variables of the consequent part for each example are determined by testing all the possible combinations of their discrete output values (according to the corresponding precision degrees) and selecting those which let the robot reach the state closest to the ideal state. For the wall-following behavior, the ideal state is that in which the robot is parallel, at the reference distance to the wall and traveling with the highest linear velocity. For the moving object following behavior, this state will be the one in which the robot is placed at the objective point and traveling with a linear velocity that is equal to the velocity of the moving object. The function that determines how close a state is from the ideal state is the SF as defined in Section II-C: the lower the value of SF, the closer the state is the ideal state.

III. Learning Methodology Based on COR

Once a data set has been generated to design a specific behavior as described in the previous section, a learning method is used to automatically generate a fuzzy controller. We propose to use a learning process based on the COR methodology (proposed in [31] and extended in [32]). We have selected this process due to its good properties to quickly obtain knowledge bases with a good interpretability. The three following sections describe the learning methodology, analyze its main properties to design behaviors in mobile robotics, and present the proposed algorithm based on COR.

A. COR Methodology

A family of efficient and simple methods to derive fuzzy rules guided by covering criteria of the data in the example set, called *ad hoc data-driven methods,* has been proposed in the literature in the last few years. Their simplicity, in addition to their quickness and easy understanding, make them very suitable for learning tasks. However, ad hoc data-driven methods usually look for the fuzzy rules with the best individual performance (e.g., see [36]) and therefore the global interaction among the rules of the rule base is not considered, thus involving knowledge bases with bad accuracy.

With the aim of addressing these drawbacks but keeping the interesting advantages of ad hoc data-driven methods, the COR methodology is proposed [31]. Instead of selecting the consequent with the highest performance in each subspace like these methods usually do, the COR methodology considers the possibility of using another consequent, different from the best one, when it allows the fuzzy rule-based system to be more accurate due to having a knowledge base with better cooperation.

COR consists of two stages.

1) *Search space construction*—It obtains a set of candidate consequents for each rule.
2) *Selection of the most cooperative fuzzy rule set*—It performs a combinatorial search among these sets looking for the combination of consequents with the best global accuracy.

A wider description of the COR-based rule generation process is shown in Fig. 2.

B. Advantages of the COR Methodology to Design Behaviors in Mobile Robotics

The above-mentioned methodology has some interesting advantages that make it very useful to learn fuzzy controllers, especially for the design of behaviors in mobile robotics. We can mainly highlight two characteristics: search space reduction and good interpretability.

1) Search Space Reduction: The COR methodology reduces the search space using heuristic information. This fact differentiates COR from other rule base learning methods [37] and allows it to be quicker and to make better exploration of the solutions. This is an important issue for the learning of fuzzy controllers, where a high number of examples are used. In the wall-following behavior presented in this paper, the methodology spends only 4 m 5 s to obtain the controller. As opposed to this, a solution based on genetic algorithms with the same number of examples could take several hours.

This search space reduction is performed by two constraints.

- *Maximum number of fuzzy input subspaces:* The maximum number of fuzzy input subspaces, and therefore the maximum number of fuzzy rules, is limited by the positive example sets. The constrains imposed to construct $E^+(S_s)$ [see (26) in Fig. 2] divides the input space with a crisp grid bounded by the cross-points between labels and, therefore, each example contributes to generate a single rule. It is a conservative subspace set selection that generates the least possible number of rules that guarantee a whole covering of the examples.

  In our problem, since the example data are uniformly distributed in the whole input space (as described in Section II), no reduction of the number of fuzzy input subspaces is done. Nevertheless, the fact of assigning each example to only one subspace will decrease the number of candidate consequents, since the positive example sets are reduced.
- *Candidate rule set in each subspace:* Once the fuzzy input subspaces are defined, a second search space reduction is made by constraining the set of possible consequents for each antecedent combination, i.e., the candidate rules in each subspace. Again, we use a restrictive condition to construct $C(S_h)$ [see (27) in Fig. 2] that generates a low number of candidate rules.

To illustrate the effect of this search space reduction, from the example data set proposed in Section II-F for the wall-following

Inputs:

- *An input-output data set*—$E = \{e_1, \ldots, e_l, \ldots, e_N\}$, with $e_l = (x_1^l, \ldots, x_n^l, y_1^l, \ldots, y_m^l)$, $l \in \{1, \ldots, N\}$, N being the data set size, and n (m) being the number of input (output) variables—*representing the behavior of the problem being solved.*
- *A fuzzy partition of the variable spaces*. In our case, uniformly distributed fuzzy sets are regarded. Let $\mathcal{A}_i$ be the set of linguistic terms of the ith input variable, with $i \in \{1, \ldots, n\}$, and $\mathcal{B}_j$ be the set of linguistic terms of the jth output variable, with $j \in \{1, \ldots, m\}$, with $|\mathcal{A}_i|$ ($|\mathcal{B}_j|$) being the number of labels of the ith (jth) input (output) variable.

Algorithm:

1) *Search space construction*:

1.1. *Define the fuzzy input subspaces containing positive examples*: To do so, we should define the positive example set ($E^+(S_s)$) for each fuzzy input subspace $S_s = (A_1^s, \ldots, A_i^s, \ldots, A_n^s)$, with $A_i^s \in \mathcal{A}_i$ being a label, $s \in \{1, \ldots, N_S\}$, and $N_S = \prod_{i=1}^{n} |\mathcal{A}_i|$ being the number of fuzzy input subspaces. In this paper, we use the following:

$$E^+(S_s) = \{ \quad e_l \in E \mid \forall i \in \{1, \ldots, n\}, \; \forall A_i' \in \mathcal{A}_i, \mu_{A_i^s}(x_i^l) \geq \mu_{A_i'}(x_i^l) \; \} \tag{26}$$

with $\mu_{A_i^s}(\cdot)$ being the membership function associated with the label A_i^s.
Among all the N_S possible fuzzy input subspaces, consider only those containing at least one positive example. To do so, the set of subspaces with positive examples is defined as $S^+ = \{S_h \mid E^+(S_h) \neq \emptyset\}$.

1.2. *Generate the set of candidate rules in each subspace with positive examples*: Firstly, the candidate consequent set associated with each subspace containing at least an example, $S_h \in S^+$, is defined. In this paper, we use the following:

$$C(S_h) = \{ \quad (B_1^{k_h}, \ldots, B_m^{k_h}) \in \mathcal{B}_1 \times \cdots \times \mathcal{B}_m \mid \exists e_l \in E^+(S_h) \; where \; \forall j \in \{1, \ldots, m\}, \; \forall B_j' \in \mathcal{B}_j, \; \mu_{B_j^{k_h}}(y_j^l) \geq \mu_{B_j'}(y_j^l) \; \}. \tag{27}$$

Then, the candidate rule set for each subspace is defined as $CR(S_h) = \{R_{k_h} =$ [IF X_1 is A_1^h and ... and X_n is A_n^h THEN Y_1 is $B_1^{k_h}$ and ... and Y_m is $B_m^{k_h}$] such that $B^{k_h} = (B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)\}$.
To allow COR to reduce the initial number of fuzzy rules, the special element $R_\emptyset$ (which means "don't care") is added to each candidate rule set, i.e., $CR(S_h) = CR(S_h) \cup R_\emptyset$. If it is selected, no rules are used in the corresponding fuzzy input subspace.

2) *Selection of the most cooperative fuzzy rule set*—This stage is performed by running a combinatorial search algorithm to look for the combination $RB = \{R_1 \in CR(S_1), \ldots, R_h \in CR(S_h), \ldots, R_{|S^+|} \in CR(S_{|S^+|})\}$ with the best accuracy. Since the tackled search space is usually large, approximate search techniques should be used.
An index $f(RB)$ measuring the global quality of the encoded rule set (see eq. 19) is considered to evaluate the quality of each solution. In order to obtain solutions with a high interpretability, the original function is modified to penalize an excessive number of rules:

$$f'(RB) = f(RB) + \gamma \cdot f(RB_0) \cdot \frac{\#RB}{|S^+|} \tag{28}$$

with $\gamma \in [0, 1]$ being a parameter defined by the designer to regulate the importance of the number of rules, $\#RB$ being the number of rules used in the evaluated solution (i.e., $|S^+| - |\{R_h \in RB \; such \; that \; R_h = R_\emptyset\}|$), and RB_0 being the initial rule base considered by the search algorithm.

Fig. 2. COR algorithm.

behavior (with 5070 examples), and using the following number of linguistic terms for each input/output variable, $|\mathcal{A}_1| = 4$, $|\mathcal{A}_2| = 2$, $|\mathcal{A}_3| = 5$, $|\mathcal{A}_4| = 2$, $|\mathcal{B}_1| = 9$, $|\mathcal{B}_2| = 9$, our methodology generates a search space of $\prod_{S_h \in S^+} |C(S_h)| = 2.2\text{e}{+92}$ combinations, while the total of possible combinations (considering the $|S^+| = 80$ input subspaces analyzed) is $(|\mathcal{B}_1| \cdot |\mathcal{B}_2|)^{80} = 4.8\text{e} + 152$.

2) Interpretability Issues: The proposed methodology also has some interesting advantages from the point of view of the interpretability of the obtained fuzzy knowledge base. This is an important issue in fuzzy control for mobile robots navigation, since when the actions of the robot are easily understandable, it is easier to detect possible errors during the design or the learning process. Basically, we can remark the two following points.

- *Model structure and membership functions keep invariable to provide excellent interpretability:* The COR methodology is an effort to exploit the accuracy ability of linguistic fuzzy rule-based systems by exclusively focusing on the rule base design. In this case, the membership functions and the model structure keep invariable, thus resulting in the highest interpretability. Indeed, instead of improving the accuracy by deriving the shape of the membership functions [38] or by extending the model structure (weighted rules [39], linguistic hedges [38], [40], hierarchical knowledge bases [41], etc.), COR methodology improves the accuracy by inducing cooperation among linguistic fuzzy rules.
- *Rule base reduction to improve interpretability and accuracy:* A problem when defining a rule base is that one cannot be sure whether the rules are correctly defined, i.e., without redundant rules or rules that generate conflicts with others in certain situations. Moreover, a high number of rules is difficult to interpret, even when a linguistic fuzzy rule structure is considered.
  To face this problem, a *rule reduction postprocessing* is usually developed. When no restriction to the interpretability is considered, the rules can be merged [6], thus generating a scatter structure where each fuzzy rule uses different fuzzy sets for each variable.
  On the other hand, if we want to obtain linguistic fuzzy rules with good interpretability, a selection process can be developed to obtain a subset of the original rule base [41]. However, this approach does not seem to be appropriate to generate an accurate final rule set since the interdependency between the learning and reduction tasks is not considered. That happens because it is sure that after reducing the rule set, the new set of rules that best cooperate will be different.
  The COR methodology achieves the reduction process at the same time as the learning one with the aim of improving the accuracy (the cooperation among rules and thus the system performance can be improved by removing rules) and interpretability (a model with less number of rules is more interpretable) of the learned model.
  This process is performed by adding the null rule ($R_\emptyset$) to the candidate rule set corresponding to each subspace, as shown in step 1.2 of Fig. 2. In this way, if such an element is selected for a specific subspace, this will mean that no rule will select the corresponding antecedent combination.

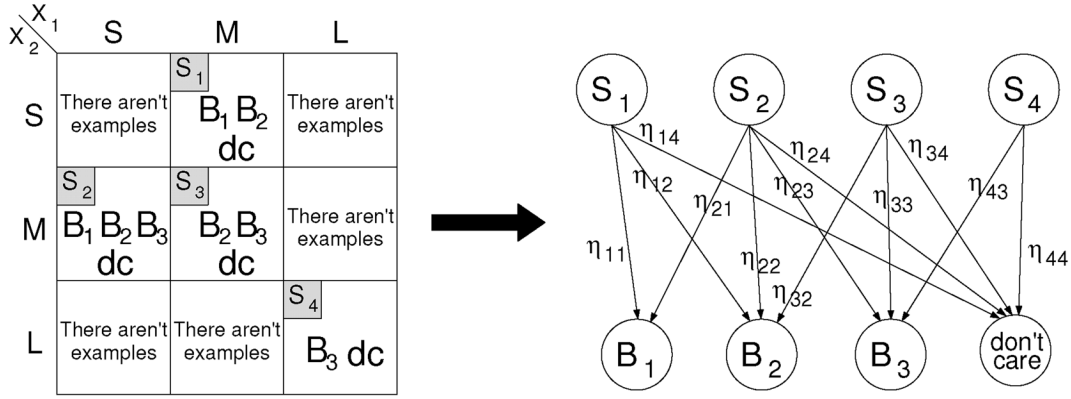


Fig. 3. Example of a graph built from sets of candidate rules generated by COR.

Notice that the addition of $R_\emptyset$ in each candidate rule set slightly increases the search space. Moreover, the objective function used to guide the search algorithm [see (28) in Fig. 2] is modified to penalize solutions with a high number of rules.

C. COR Methodology With Ant Colony Optimization

Since the search space tackled in step 2 of Fig. 2 is usually large, it is necessary to use approximate search techniques. In [31], accurate linguistic models have been obtained using simulated annealing. However, since one of our constrains is to deal with a computationally expensive evaluation function, in this paper the use of ACO [34], [42] is considered. It is a population search bioinspired technique that considers heuristic information to allow it to get good solutions quickly. Indeed, as was shown in [33] and [43], the use of ACO in COR, as opposed to other kinds of optimization techniques such as simulated annealing and genetic algorithms, performs a quick convergence obtaining accurate results. This section briefly describes the main components of the considered COR-based ACO algorithm [33].

1) Problem Representation for Learning Cooperative Fuzzy Rules: To apply ACO in the COR methodology, it is convenient to see it as a combinatorial optimization problem with the capability of being represented on a weighted graph. In this way, we can face the problem considering a fixed number of subspaces and interpreting the learning process as the way of assigning consequents vectors—i.e., labels of the output fuzzy partitions—to these subspaces with respect to an optimality criterion (i.e., following the COR methodology).

Therefore, according to the notation introduced in Fig. 2, each node $S_h \in S^+$ is assigned to each candidate consequent $(B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)$ and to the special symbol "don't care" $(R_\emptyset)$ that stands for the absence of rules in such a subspace. Fig. 3 shows the explored graph built from an example of candidate rule sets. To construct a complete solution, an ant iteratively goes over each rule and chooses a consequent with a probability that depends on the pheromone trail τ and the heuristic information η associated to each decision. The order of selection of the rules is irrelevant.

2) Heuristic Information: The heuristic information on the potential preference of selecting a specific consequent vector B^{k_h} in each antecedent combination (subspace) is determined as described in Fig. 4.

For each subspace $S_h \in S^+$ do:

1) Build the sets $E^+(S_h)$ and $C(S_h)$ as shown in Fig. 2.
2) For each $B^{k_h} = (B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)$, make use of an initialization function based on a covering criterion to give a heuristic preference degree to each choice. In this paper, we use the following:

$$\eta_{hk_h} = \max_{e_l \in E^+(S_h)} Min\left(\mu_{A^h}(x^l), \mu_{B_j^{k_h}}(y^l)\right). \quad (29)$$

3) For each $B^{k_h} \notin C(S_h)$, make $\eta_{hk_h} = 0$.
4) Finally, for the "don't care" symbol, make the following:

$$\eta_{h,|\mathcal{B}_1|\cdot\ldots\cdot|\mathcal{B}_m|+1} = \frac{1}{\max\limits_{k_h \in \{1,\ldots,|C(S_h)|\}} \eta_{hk_h}}. \quad (30)$$

Fig. 4. Heuristic assignment process.

3) Pheromone Initialization: The initial pheromone value of each assignment is obtained as follows:

$$\tau_0 = \frac{1}{|S^+|} \sum_{S_h \in S^+} \max_{B^{k_h} \in C(S_h)} \eta_{hk_h}. \quad (31)$$

In this way, the initial pheromone will be the mean value of the path constructed taking the best consequent in each rule according to the heuristic information (a greedy assignment).

4) Fitness Function: The fitness function will be the said objective function, defined in (28) in Fig. 2.

5) Ant Colony Optimization Scheme, the Best–Worst Ant System Algorithm: Once the previous components have been defined, an ACO algorithm has to be given to solve the problem. In [32], the ant colony system [44] was applied. Opposite to that, in this paper an advanced ACO algorithm, the best–worst ant system (BWAS) [45], is considered in order to improve the search process. Its global scheme is shown in Fig. 5.

In the original BWAS, local search is applied to every generated solution. However, in our fuzzy rule learning problem, opposite to other applications such as traveling salesman or quadratic assignment problems, it is not possible to optimize the evaluation of neighbor solutions. Therefore, in order to keep a high speed of the learning process, we apply the local search

1) Give an initial pheromone value, τ_0, to each edge.
2) While (*termination_condition* is not satisfied) do:
   a) Perform the track of each ant by the **solution construction process**.
   b) Apply the **pheromone evaporation mechanism**.
   c) Apply the **local search process** on the current-best solution.
   d) Update $S_{global\ best}$ and $S_{current\ worst}$.
   e) Apply the **Best-Worst pheromone trail update rule**.
   f) Apply the **pheromone trail mutation**.
   g) If (*stuck_condition* is satisfied) then apply **restart**.

Fig. 5. BWAS algorithm.

TABLE I
SUMMARY OF THE LEARNING PROCESS PERFORMED FOR THE TWO ANALYZED BEHAVIORS

Behavior	*#Examp.*	t_{examp}	t_{learn}	*Fitness*
Wall-following	5,070	1 s	4 m 5s	181,569
Moving obj. fol.	6,435	10 s	7m 19s	162,661

only to the current best solution (step 2.c in Fig. 5). The local search is a simple hill-climbing algorithm.

IV. EXPERIMENTAL RESULTS

The methodology for the design of behaviors in mobile robotics has been tested with two different behaviors: wall-following and moving object following. Table I collects a summary of the results obtained by each of them. It shows the number of examples of the training set, the time needed for the generation of that training set (t_{examp}), the time expended by the learning method (t_{learn}) to design the fuzzy controller (t_{examp} is included in this time), and the number of rules and fitness values (19) of the obtained fuzzy controllers.

A. Wall-Following Behavior

1) Experiment Setup: The learned fuzzy controller has been tested in six simulated environments using the Nomad 200 simulation software. These environments include very different situations that the robot usually faces during navigation: straight walls of different lengths, followed and/or preceded by a number of concave and convex corners, thus covering a wide range of contours to follow and truly defining very complex test environments. It is important to remark that these environments have not been used during training. The training set is only composed of a list of examples (5070) that have been chosen covering the input space with an adequate precision. These conditions guarantee that the quality of the learned behavior does not depend on the environment, and also that the robot will be capable to face any situation.

We have used the following parameter values for the COR-based ACO algorithm: 50 iterations, 30 ants, $\rho = 0.8$, $\alpha = 2$, $\beta = 2$, probability of mutation $P_m = 0.3$, mutation rate $\sigma = 4$, iterations of the local search $\mathrm{LSi} = 10$, neighbor size in the local search $\mathrm{LSn} = 30$, and number of iterations before restart $R = 5$. We would like to highlight that these values have been intuitively chosen according to experimentations usually done with ACO [34] and that, in our experimental study, they are fixed regardless the problem or the environment. Therefore, an expert that would like to apply the proposed methodology to design a new behavior could keep the same values. Anyway, no experiments were made with different values for these parameters; therefore, the results shown below maybe could be improved with a more exhaustive parameter value selection.

2) Obtained Results: Fig. 6 shows the robot path along four of the environments used for testing. The robot trajectory is represented by circular marks. A higher concentration of marks indicates lower velocity. The learned controller (Table I, row 1) has 52 linguistic rules and has been obtained in only 4 m 5 s (with an Intel Pentium 4 CPU 3.20 GHz processor) using a value of $\gamma = 0.20$ (28). This time includes the time needed for the generation of the training set, which is of 1 s. If for any situation no rule is fired, then a null linear acceleration and angular velocity are selected. The maximum velocity the robot can reach is 61 cm/s, and the reference distance at which the robot should follow the right wall is 51 cm. Ten tests have been done for each one of the analyzed environments. The mean values and the standard deviations measured for some parameters that reflect the controller performance are shown in Table II. These parameters are the average distance to the right wall (the wall that is being followed), the average linear velocity, the time spent by the robot along the path, and the average velocity change. The latter parameter measures the change in the linear velocity between two consecutive cycles, reflecting the smoothness of the behavior (a low value indicates a smooth behavior).

3) Analysis of Efficiency and Accuracy: In order to show the quality of the controller, the path of the robot in environment *F* [Fig. 6(a)] will be described in detail. This environment is quite complex, with ten concave corners and six convex corners in a circuit of a length of 48 m. Also, the walls have several gaps, which increase the noise in the measurement of the distances. Convex corners are truly difficult situations, because the robot's sensors may cease to correctly detect the wall at some given moments, even though some of them may occasionally detect it. The controller must also significantly reduce velocity at corners. In spite of these difficulties, the obtained average velocity has been quite high, and the distance at which the robot should follow the wall is near the desired reference distance. The difference between both distances is caused by the high number of corners, in which the orientation of the robot is very bad (at concave corners the robot is detecting two perpendicular walls, and sometimes at convex corners it detects no wall), and a fast turning is prioritized over a correct distance.

During learning, the reference distance to the wall (d_{wall}) and the maximum linear velocity (v_{max}) must have a fixed value (51 cm and 61 cm/s, respectively, in this case). It is interesting to discuss how the accuracy is affected when these parameters are changed in the learned controller. Table III shows the accuracy of the learned controller for different values of the reference distance and the maximum velocity for environment *A*.

The knowledge base should be learnt using the values for the two parameters that represent the worst conditions (low reference distances and high velocities). If the value of the reference distance is increased or the maximum velocity is reduced, the accuracy of the controller is not affected, as can be seen for the first row of Table III. If the maximum velocity is maintained

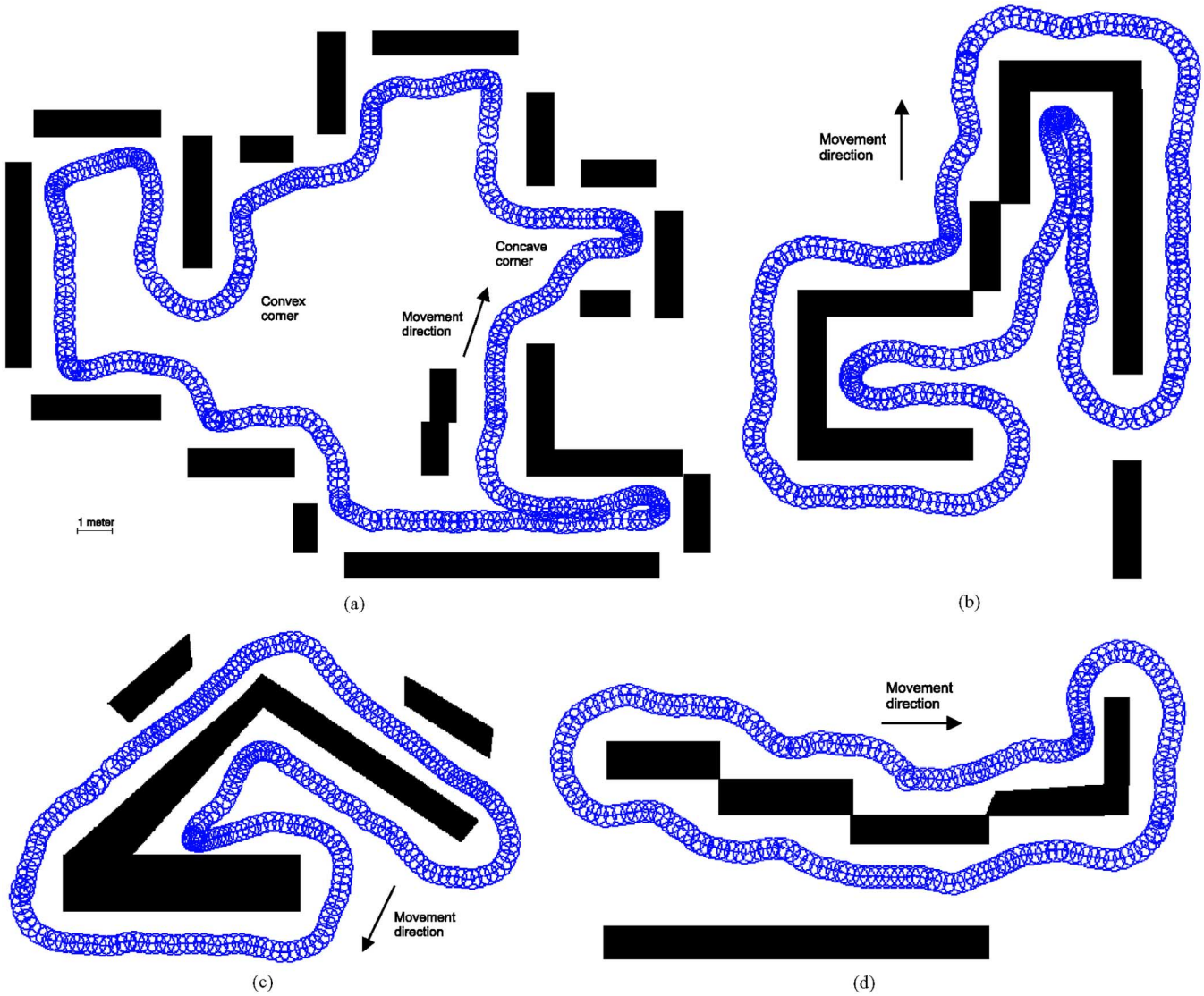


Fig. 6. Path of the robot along some of the different simulated environments for the wall-following behavior. (a) Environment F, (b) Environment A, (c) Environment C, and (d) Environment B.

TABLE II
RESULTS ($\bar{x} \pm \sigma$) FOR THE WALL-FOLLOWING BEHAVIOR (52 LINGUISTIC FUZZY RULES)

Env.	*Dist. (cm)*	*Vel. (cm/s)*	*Vel. ch. (cm/s)*	*Time (s)*
A	53.47 ± 0.69	50.63 ± 0.62	10.39 ± 0.94	106.60 ± 1.90
B	53.47 ± 0.67	51.47 ± 0.91	10.17 ± 1.57	65.60 ± 1.26
C	49.88 ± 0.65	49.02 ± 0.63	9.40 ± 0.78	80.50 ± 1.27
D	53.35 ± 0.63	51.71 ± 0.71	10.08 ± 1.04	116.00 ± 2.00
E	53.45 ± 0.59	51.03 ± 0.76	10.29 ± 1.65	103.50 ± 1.43
F	49.29 ± 0.56	46.12 ± 0.38	11.22 ± 0.70	122.80 ± 1.55

TABLE III
RESULTS ($\bar{x} \pm \sigma$) FOR DIFFERENT VALUES OF THE REFERENCE DISTANCE (CM) AND THE MAXIMUM VELOCITY (CM/S) FOR ENVIRONMENT A

d_{wall} / v_{max}	*Dist. (cm)*	*Vel. (cm/s)*	*V. ch. (cm/s)*	*Time (s)*
75 / 61	74.60 ± 0.62	48.08 ± 0.41	12.33 ± 0.69	113.20 ± 1.03
51 / 61	53.47 ± 0.69	50.63 ± 0.62	10.39 ± 0.94	106.60 ± 1.90
40 / 61	41.97 ± 0.62	49.96 ± 0.37	10.62 ± 1.18	108.00 ± 1.15
30 / 50	30.83 ± 0.39	40.45 ± 0.31	11.33 ± 1.22	131.90 ± 2.08

and the reference distance is reduced by 20% (row 3), the behavior still has a high accuracy. Finally, if the reference distance is drastically reduced to 30 cm (a 40% reduction), the controller is not able to implement the wall-following behavior. To follow the wall for this reference distance, the maximum velocity has to be reduced to 50 cm/s (row 4).

The accuracy of the obtained controller is, obviously, very dependent on the definition of the scoring function (11). As has been explained, the idea of the SF is to measure the deviation of each variable from the desired value. Equations (12)–(14) show these deviations for the different variables. Each α_i has an associated weight to indicate how important the deviation in the value of a variable is with respect to the other variables. Table IV shows a comparison of the learned controller (first row) with other learned controllers using different weights in the scoring function. The tests have been done in environment *A*. Results show very similar values for the average velocity and quite close values also for the average right distance (the difference between the best and the worst is lower than 7%). The number of rules

TABLE IV
RESULTS ($\bar{x} \pm \sigma$) FOR DIFFERENT VALUES OF THE WEIGHTS IN THE SCORING FUNCTION (11)–(14) FOR ENVIRONMENT A

Weights	*#R*	*Dist. (cm)*	*Vel. (cm/s)*	*V. ch. (cm/s)*	*Time (s)*
100, 10, 1	52	53.47 ± 0.69	50.63 ± 0.62	10.39 ± 0.94	106.60 ± 1.90
4, 2, 1	66	57.08 ± 0.68	49.96 ± 0.44	1.74 ± 0.17	111.50 ± 0.97
10, 10, 1	62	57.15 ± 1.59	49.56 ± 0.70	3.87 ± 0.85	112.70 ± 1.77
225, 15, 1	52	56.65 ± 0.31	47.99 ± 0.28	6.49 ± 0.37	110.20 ± 0.63

TABLE V
RESULTS ($\bar{x} \pm \sigma$) FOR THE WALL-FOLLOWING BEHAVIOR WITH $\gamma = 0.5$ (40 LINGUISTIC FUZZY RULES)

Env.	*Dist. (cm)*	*Vel. (cm/s)*	*Vel. ch. (cm/s)*	*Time (s)*
A	62.00 ± 0.90	51.02 ± 0.69	2.10 ± 0.18	110.20 ± 1.81
B	54.80 ± 1.13	46.23 ± 1.10	5.02 ± 0.46	74.40 ± 1.90
C	58.71 ± 1.63	49.83 ± 1.51	2.88 ± 0.31	81.20 ± 2.53
D	60.01 ± 1.45	51.58 ± 1.19	2.05 ± 0.21	119.10 ± 3.21
E	53.11 ± 1.21	44.19 ± 0.96	4.08 ± 0.13	122.50 ± 2.64
F	50.62 ± 0.96	41.50 ± 1.00	5.62 ± 0.27	136.90 ± 4.82

decreases as the difference in the weights increases, because one or two variables dominate the scoring function and, then, more general rules are learnt.

Parameter γ lets the designer adjust the desired tradeoff between accuracy and the size of the rule base: a higher value of γ means lower accuracy, but also a lower number of rules. As an example, a controller has been learnt for $\gamma = 0.5$, obtaining only 40 rules. The mean values of some parameters for the test environments are presented in Table V. Due to the lower number of rules, the accuracy of the controller is worse than that learned for $\gamma = 0.20$; the values of the average velocity are much lower in three of the environments, and the average distance is worse in four of the environments. Finally, the average velocity change is much lower because, due to the lower number of rules, few rules are fired (there is less interaction among rules) or no rule is fired in some control cycles (the robot maintains its velocity and steering).

4) Comparison With the Method Proposed in [6]: We have tried to compare the obtained controller with another one learnt using genetic algorithms [6]. The design in [6] consisted of two stages: learning of the data base and a general rule base (Pittsburgh-style genetic algorithm) and reduction of the generated rule base merging adjacent membership functions. This reduction produces loss of interpretability in the final knowledge base since a different fuzzy set is built for each fuzzy rule, thus losing the legibility provided by the use of linguistic variables with global semantics. Apart from the different methodologies, the method proposed in [6] and the COR-based approach also differ in the following ways.

- *Size of the training set:* 23 085 examples were used for training in [6]. This number arises when using a lower value for the precision (p_n) of the different variables. In the COR-based approach we have only used 5070 examples.
- *Generation of the data base:* in both approaches, the designer has to select the universe of discourse and the number of labels for each variable. The difference is that in [6], taking these parameters into account, the data base is learnt using a genetic algorithm, while in the COR-based, approach the data base is automatically generated using uniformly distributed fuzzy sets.
- The output variables for [6] are the objective velocity (V_{obj}) and the objective orientation (OR_{obj})—tenths of degree—but they can be easily transformed in LA and AV, and the performance of the system is not affected.

The COR-based approach has been compared with different modified versions of the controller introduced in [6]:

1) training set with 5070 examples;
2) training set with 5070 examples and COR-based approach data base (the first stage of [6] is suppressed);
3) training set with 23 085 examples and COR-based approach data base (the first stage of [6] is suppressed);
4) training set with 23 085 examples (original [6] approach).

For the first approach, the obtained knowledge base has too few rules, and the controller is not able to complete the path in any of the environments. The second and third approaches point out the same problem: the rules have been learnt individually, so there is a lack of cooperation between the rules and the controller fails in some of the environments, leading the robot to a crash or to a halt.

Finally, for the fourth approach, the obtained controller has 46 fuzzy rules (the knowledge base is shown in Fig. 7), and the mean values of some parameters for the test environments are presented in Table VI. The time spent to learn the system is very high (about 8 h) compared with 4 m 5 s for the controller presented in this paper.

The controller described in this paper has a higher number of rules (52 versus 46) but clearly improves the results of the controller presented in [6]. The average velocity is higher in all the environments, reducing the time spent by the robot in the circuit. This improvement ranges from the 9% of increase in the average velocity in environment E to 36% in C. Our proposed controller not only reaches higher velocity compared to [6] but also obtains slightly better values of the average right-hand distance in four of the environments and clearly outperforms the method of [6] in another one (B). Finally, the average velocity change shows higher values in our case because the controller described in this paper has to implement many changes in velocity due to the shape of these environments to get such high average velocity values.

5) Analysis on Interpretability: Fig. 8 shows the data and rule bases of the learned controller. As can be noticed, the interpretability of the obtained rules is very high, as opposed to [6] (see Fig. 7). This makes it easier to understand the actions taken by the robot.

For a better understanding of the fuzzy controller (Fig. 8), rules have been grouped taking into account the labels for the right-hand distance (RD) and the distances quotient (DQ). Blank lines represent rules corresponding to that group that have been eliminated during the learning process.

1) The first group of rules $(\text{RD} = \text{low}, \text{DQ} = \text{low})$ is fired in those situations when both of the walls are quite close to the robot. The controller tries to get the robot parallel to the walls and reduce the velocity, except in those cases when it is already parallel and the velocity is low, so the robot can maintain its speed.
2) The second group of rules $(\text{RD} = \text{low}, \text{DQ} = \text{high})$ is very similar to the previous one. But now, due to the left wall's being farther, the speed can be higher (if linear acceleration

Rule	*RD*	*DQ*	θ_{wall}	*V*	V_{obj}	OR_{obj}
R_1					0.625	800
R_2					1.000	800
R_3					0.000	-100
R_4					0.000	150
R_5					0.375	-50
R_6					0.625	-50
R_7					1.000	100
R_8					0.250	0
R_9					0.625	450
R_{10}					0.250	-150
R_{11}					1.000	150
R_{12}					1.000	350
R_{13}					1.000	0
R_{14}					0.125	0
R_{15}					1.000	50
R_{16}					0.625	-200
R_{17}					1.000	-150
R_{18}					0.000	0
R_{19}					0.625	-50
R_{20}					0.625	0
R_{21}					1.000	150
R_{22}					1.000	50
R_{23}					0.250	0
R_{24}					1.000	200
R_{25}					0.625	750
R_{26}					0.000	0
R_{27}					0.625	50
R_{28}					0.625	-50
R_{29}					0.625	-100
R_{30}					1.000	350
R_{31}					1.000	100
R_{32}					1.000	-50
R_{33}					1.000	-50
R_{34}					1.000	-100
R_{35}					1.000	150
R_{36}					1.000	50
R_{37}					0.625	100
R_{38}					1.000	200
R_{39}					1.000	0
R_{40}					1.000	50
R_{41}					1.000	200
R_{42}					1.000	-200
R_{43}					0.625	0
R_{44}					1.000	0
R_{45}					0.625	50
R_{46}					1.000	150

Fig. 7. Fuzzy rule set generated by the method proposed in [6].

was *null*, now it can be *soft acceleration*, etc.) and also a higher turn to the left can be selected in order to go away from the right wall.

3) These rules $(\text{RD} = \text{medium}, \text{DQ} = \text{low})$ are designed to approach the robot softly to the right wall, because the left wall is closer. The linear acceleration is selected taking into account the current orientation and the angular velocity that is going to be applied.

TABLE VI
RESULTS ($\bar{x} \pm \sigma$) OBTAINED BY THE METHOD PROPOSED IN [6] FOR THE WALL-FOLLOWING BEHAVIOR (46 FUZZY RULES)

Env.	*Dist. (cm)*	*Vel. (cm/s)*	*Vel. ch. (cm/s)*	*Time (s)*
A	54.48 ± 1.26	43.22 ± 0.83	9.70 ± 0.67	120.20 ± 2.04
B	59.22 ± 1.31	46.15 ± 0.58	8.69 ± 0.58	72.00 ± 1.15
C	53.79 ± 1.44	36.05 ± 0.60	7.75 ± 0.61	113.80 ± 2.30
D	53.04 ± 2.14	41.28 ± 3.33	8.35 ± 0.54	126.40 ± 5.76
E	53.51 ± 1.13	46.83 ± 0.74	7.14 ± 0.59	125.80 ± 2.35
F	47.89 ± 1.11	38.47 ± 0.82	9.99 ± 0.47	133.00 ± 2.98

4) This group of rules $(\text{RD} = \text{medium}, \text{DQ} = \text{high})$ is fired in situations in which the distances take adequate values, so the objective is to improve the orientation and accelerate, except in those cases where the robot has a wrong orientation.
5) The rules of the fifth group $(\text{RD} = \text{high}, \text{DQ} = \text{low})$ try to approach the robot to the right-hand wall. Control is not as soft as for the rules in the third group, because distance is higher.
6) The sixth group of rules $(\text{RD} = \text{high}, \text{DQ} = \text{high})$ is similar to the previous one, but control is softer, due to a higher distance to the left wall.
7) This group of rules $(\text{RD} = \text{very high}, \text{DQ} = \text{low})$ is fired when the robot is quite far from the right-hand wall, so the objective is to turn the robot to the wall and accelerate (only if the previous orientation was not to the left and the velocity high).
8) The last group of rules $(\text{RD} = \text{very high}, \text{DQ} = \text{high})$ implements very similar control actions, but control is very abrupt because the walls are far from the robot.

In summary, the objective is first to place the robot at an adequate distance to the right-hand wall, then to select a high velocity if possible, and finally to orient it parallel to the wall.

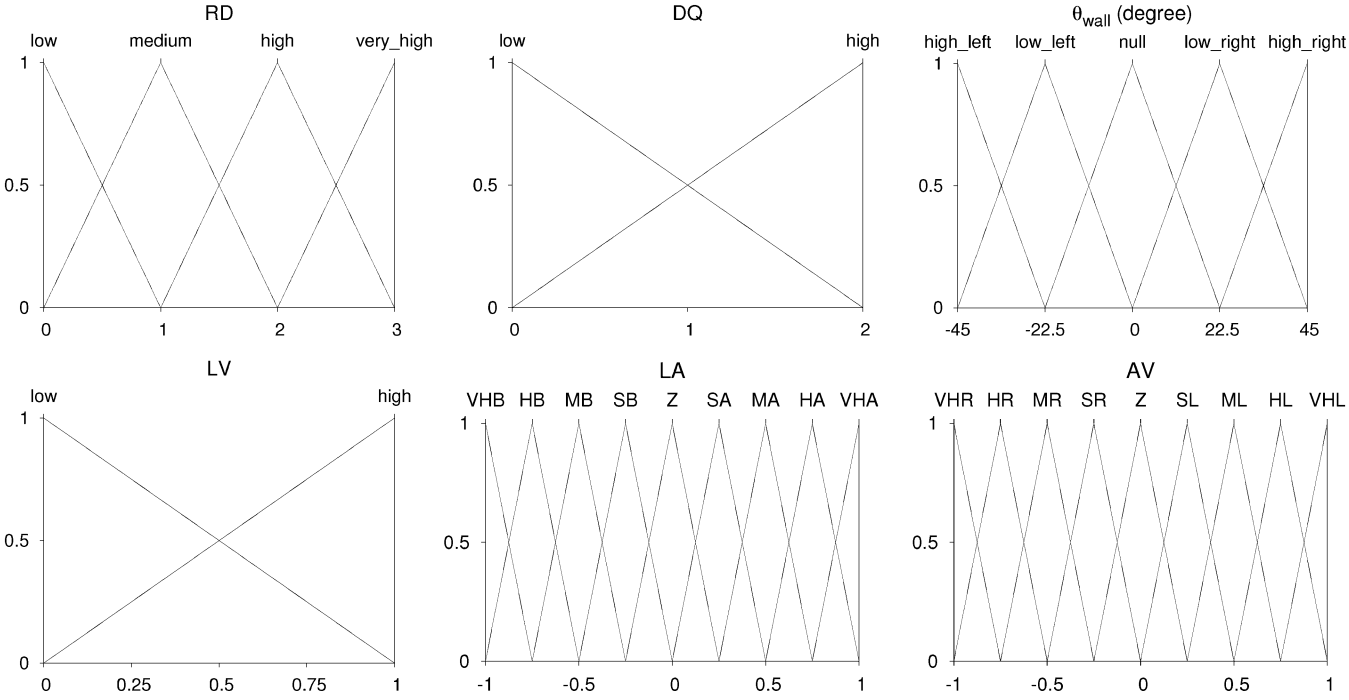
B. Moving Object Following Behavior

We have tested the learned fuzzy controller for this behavior using the Nomad 200 simulation software. The position, velocity, and advance direction of the moving object were directly obtained from the simulation software and passed to the control system in order to calculate the input variables. Tests have been carefully chosen, trying the controller with a wide range of situations of velocity and turning of the moving object. The parameter values for the COR-based ACO algorithm have been the same as in the previous behavior.

The learned controller (Table I, row 2) has 83 linguistic rules (shown in Fig. 9) and has been learnt in only 7 m 19 s (with an Intel Pentium 4 CPU 3.20 GHz processor) using a value of $\gamma = 0$ (28). If for any situation no rule is fired, then a null linear acceleration and angular velocity are selected. The maximum linear velocity the robot can reach is 61 cm/s, and the maximum angular velocity is 45°/s.

Fig. 10 shows some trajectories of the robot when it is following different moving objects at a reference distance of 1.5 m and with a reference deviation of 0°. The trajectories are represented by circular marks. In order to visualize adequately both trajectories, in Fig. 10, the trajectory of the moving object has been shifted in the y-axis direction. Thus, at the beginning

(a)



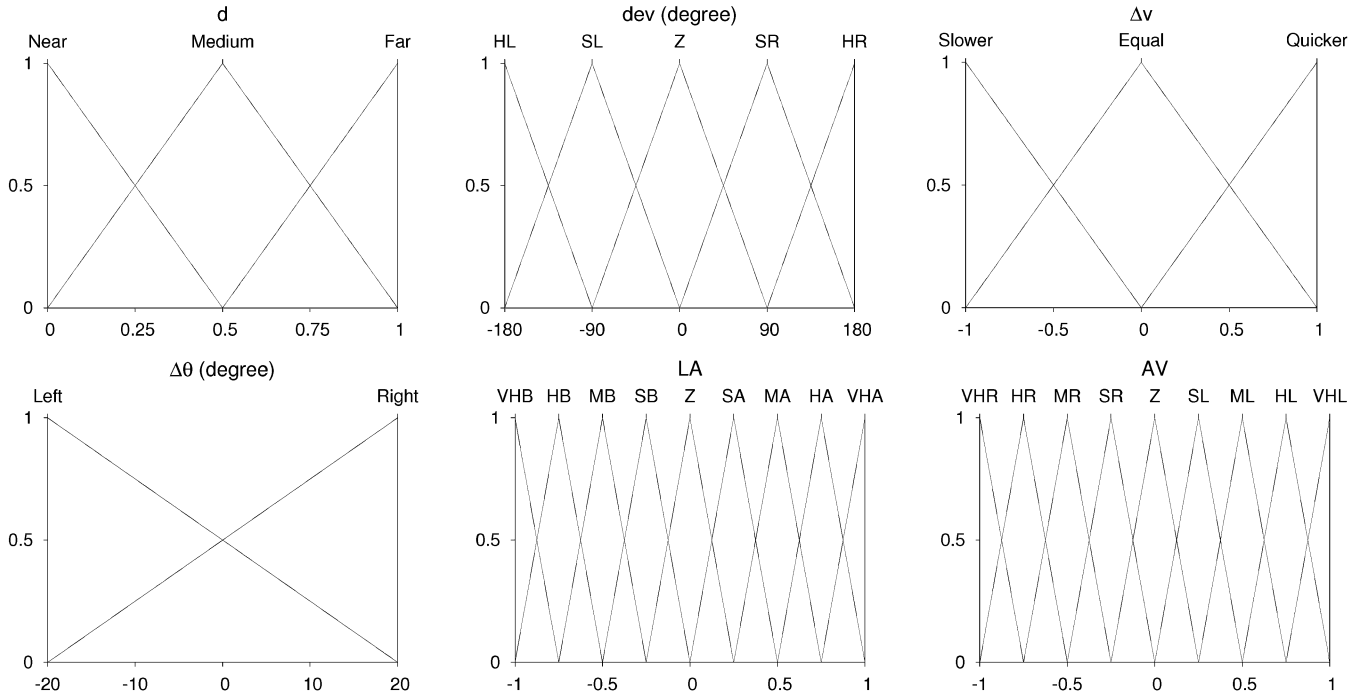
(b)

Rule	*RD*	*DQ*	θ_{wall}	*LV*	*LA*	*AV*	*Rule*	*RD*	*DQ*	θ_{wall}	*LV*	*LA*	*AV*
R_1	low	low	high_left	low	very_hard_brak.	very_hard_right	R_{29}	high	low	high_left	low	very_hard_brak.	very_hard_right
R_2	low	low	low_left	low	null	very_hard_right							
R_3	low	low	low_left	high	very_hard_brak.	mdm_right	R_{30}	high	low	low_left	high	soft_braking	very_hard_right
R_4	low	low	null	low	null	soft_right	R_{31}	high	low	null	low	hard_accel.	very_hard_right
R_5	low	low	null	high	hard_braking	hard_right	R_{32}	high	low	null	high	soft_accel.	very_hard_right
							R_{33}	high	low	low_right	low	hard_accel.	soft_right
R_6	low	low	low_right	high	hard_braking	hard_left							
							R_{34}	high	low	high_right	low	hard_accel.	soft_left
							R_{35}	high	low	high_right	high	null	very_hard_left
R_7	low	high	high_left	low	very_hard_brak.	very_hard_right	R_{36}	high	high	high_left	low	very_hard_brak.	very_hard_right
R_8	low	high	low_left	low	soft_accel.	hard_right	R_{37}	high	high	low_left	low	mdm_accel.	very_hard_right
R_9	low	high	low_left	high	hard_braking	mdm_right	R_{38}	high	high	low_left	high	soft_accel.	very_hard_right
R_{10}	low	high	null	low	soft_accel.	mdm_left	R_{39}	high	high	null	low	hard_accel.	very_hard_right
R_{11}	low	high	null	high	hard_braking	mdm_left	R_{40}	high	high	null	high	soft_accel.	very_hard_right
R_{12}	low	high	low_right	low	null	very_hard_left							
R_{13}	low	high	low_right	high	hard_braking	hard_left	R_{41}	high	high	low_right	high	soft_accel.	mdm_right
R_{14}	low	high	high_right	low	very_hard_brak.	very_hard_left	R_{42}	high	high	high_right	low	mdm_accel.	mdm_left
R_{15}	low	high	high_right	high	very_hard_brak.	very_hard_left	R_{43}	high	high	high_right	high	soft_accel.	very_hard_left
R_{16}	medium	low	high_left	high	very_hard_brak.	very_hard_right							
							R_{44}	very_high	low	low_left	low	hard_accel.	very_hard_right
R_{17}	medium	low	low_left	high	hard_braking	very_hard_right	R_{45}	very_high	low	low_left	high	soft_braking	very_hard_right
R_{18}	medium	low	null	low	mdm_accel.	very_hard_right							
R_{19}	medium	low	low_right	low	null	soft_left	R_{46}	very_high	low	low_right	low	hard_accel.	very_hard_right
R_{20}	medium	low	low_right	high	soft_braking	mdm_left	R_{47}	very_high	low	low_right	high	soft_accel.	very_hard_right
R_{21}	medium	low	high_right	low	null	very_hard_left	R_{48}	very_high	low	high_right	low	hard_accel.	hard_right
R_{22}	medium	low	high_right	high	mdm_braking	hard_left	R_{49}	very_high	low	high_right	high	soft_accel.	hard_right
R_{23}	medium	high	high_left	low	very_hard_brak.	very_hard_right							
R_{24}	medium	high	low_left	low	hard_accel.	very_hard_right							
R_{25}	medium	high	null	low	hard_accel.	mdm_left	R_{50}	very_high	high	null	low	hard_accel.	very_hard_right
R_{26}	medium	high	low_right	low	hard_accel.	very_hard_left	R_{51}	very_high	high	low_right	low	hard_accel.	very_hard_right
R_{27}	medium	high	low_right	high	soft_braking	very_hard_left							
							R_{52}	very_high	high	high_right	low	hard_accel.	very_hard_right
R_{28}	medium	high	high_right	high	very_hard_brak.	very_hard_left							

Fig. 8. Knowledge base generated by the COR-based algorithm for the wall-following behavior. (a) Data base and (b) rule base.

(points A_r for the robot and A_m for the moving object), both the robot and the object have the same y coordinate, and their x coordinate is the one represented in the figure (the robot is placed 1.5 m to the left of the moving object). The labels that have been

(a)



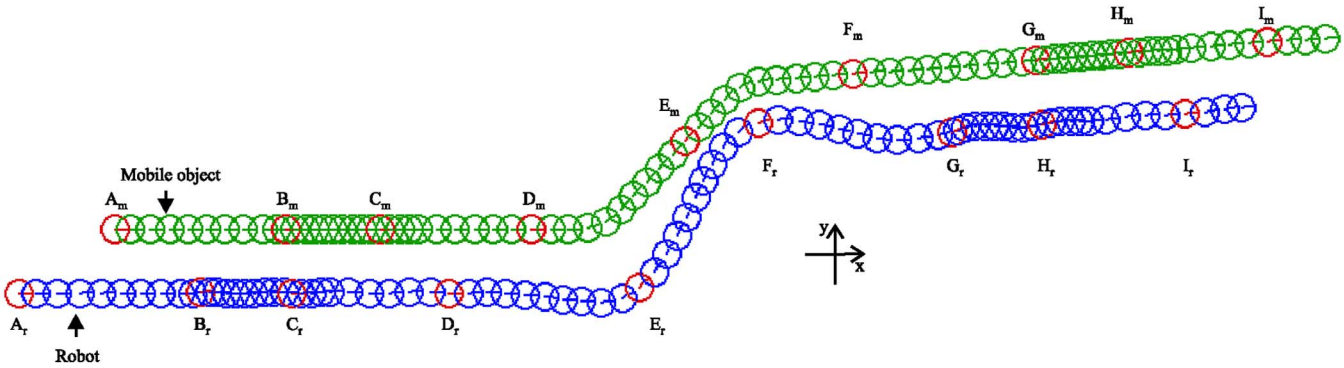
(b)

Rule	*d*	*dev*	$\triangle v$	$\triangle\theta$	*LA*	*AV*	*Rule*	*d*	*dev*	$\triangle v$	$\triangle\theta$	*LA*	*AV*
R_1	Near	HL	Slower	Left	MA	MR	R_{43}	Medium	Z	Equal	Right	HA	SL
R_2	Near	HL	Slower	Right	HA	ML	R_{44}	Medium	Z	Quicker	Left	SB	SR
R_3	Near	HL	Equal	Left	SB	MR	R_{45}	Medium	Z	Quicker	Right	SB	SL
R_4	Near	HL	Equal	Right	Z	ML	R_{46}	Medium	SR	Slower	Left	MA	HL
R_5	Near	HL	Quicker	Left	VHB	SR	R_{47}	Medium	SR	Slower	Right	VHA	VHL
R_6	Near	HL	Quicker	Right	VHB	HL	R_{48}	Medium	SR	Equal	Left	Z	HL
R_7	Near	SL	Slower	Left	MA	SR	R_{49}	Medium	SR	Equal	Right	MA	VHL
R_8	Near	SL	Slower	Right	HA	ML	R_{50}	Medium	SR	Quicker	Left	VHB	HL
R_9	Near	SL	Equal	Left	Z	HR	R_{51}	Medium	SR	Quicker	Right	MB	VHL
R_{10}	Near	SL	Equal	Right	Z	SL	R_{52}	Medium	HR	Slower	Left	Z	VHR
R_{11}	Near	SL	Quicker	Right	HB	ML	R_{53}	Medium	HR	Slower	Right	MA	VHL
R_{12}	Near	Z	Slower	Left	HA	SR	R_{54}	Medium	HR	Equal	Left	VHB	VHL
R_{13}	Near	Z	Slower	Right	HA	SL	R_{55}	Medium	HR	Equal	Right	MB	HL
R_{14}	Near	Z	Equal	Left	Z	MR	R_{56}	Medium	HR	Quicker	Left	VHB	Z
R_{15}	Near	Z	Equal	Right	Z	ML	R_{57}	Medium	HR	Quicker	Right	VHB	HL
R_{16}	Near	Z	Quicker	Left	HB	SR	R_{58}	Far	HL	Slower	Right	SB	HR
R_{17}	Near	Z	Quicker	Right	HB	ML	R_{59}	Far	HL	Equal	Left	VHB	SR
R_{18}	Near	SR	Slower	Left	HA	SR	R_{60}	Far	HL	Quicker	Left	VHB	VHR
R_{19}	Near	SR	Slower	Right	HA	SL	R_{61}	Far	HL	Quicker	Right	VHB	VHL
R_{20}	Near	SR	Equal	Left	Z	SR	R_{62}	Far	SL	Slower	Left	VHA	VHR
R_{21}	Near	SR	Equal	Right	Z	HL	R_{63}	Far	SL	Equal	Left	VHA	VHR
R_{22}	Near	SR	Quicker	Left	VHB	MR	R_{64}	Far	SL	Equal	Right	SA	VHR
R_{23}	Near	SR	Quicker	Right	HB	ML	R_{65}	Far	SL	Quicker	Left	Z	VHR
R_{24}	Near	HR	Slower	Left	MA	Z	R_{66}	Far	SL	Quicker	Right	VHB	VHR
R_{25}	Near	HR	Slower	Right	HA	ML	R_{67}	Far	Z	Slower	Left	VHA	SR
R_{26}	Near	HR	Equal	Left	Z	MR	R_{68}	Far	Z	Equal	Left	VHA	SR
R_{27}	Near	HR	Equal	Right	Z	ML	R_{69}	Far	Z	Equal	Right	VHA	SL
R_{28}	Near	HR	Quicker	Right	VHB	SL	R_{70}	Far	Z	Quicker	Left	Z	SR
R_{29}	Medium	HL	Slower	Left	SB	HR	R_{71}	Far	Z	Quicker	Right	Z	ML
R_{30}	Medium	HL	Slower	Right	SA	VHR	R_{72}	Far	SR	Slower	Left	VHA	HL
R_{31}	Medium	HL	Equal	Right	HB	VHL	R_{73}	Far	SR	Slower	Right	VHA	VHL
R_{32}	Medium	HL	Quicker	Left	VHB	MR	R_{74}	Far	SR	Equal	Left	SA	VHL
R_{33}	Medium	HL	Quicker	Right	VHB	SL	R_{75}	Far	SR	Equal	Right	HA	VHL
R_{34}	Medium	SL	Slower	Left	VHA	VHR	R_{76}	Far	SR	Quicker	Left	MB	VHL
R_{35}	Medium	SL	Slower	Right	HA	HR	R_{77}	Far	SR	Quicker	Right	SA	VHL
R_{36}	Medium	SL	Equal	Left	SA	VHR	R_{78}	Far	HR	Slower	Left	SB	HL
R_{37}	Medium	SL	Equal	Right	SB	HR	R_{79}	Far	HR	Slower	Right	SB	MR
R_{38}	Medium	SL	Quicker	Left	HB	VHR	R_{80}	Far	HR	Equal	Left	VHB	HL
R_{39}	Medium	SL	Quicker	Right	VHB	VHR	R_{81}	Far	HR	Equal	Right	VHB	MR
R_{40}	Medium	Z	Slower	Left	VHA	SR	R_{82}	Far	HR	Quicker	Left	VHB	VHR
R_{41}	Medium	Z	Slower	Right	VHA	ML	R_{83}	Far	HR	Quicker	Right	VHB	VHL
R_{42}	Medium	Z	Equal	Left	HA	MR							

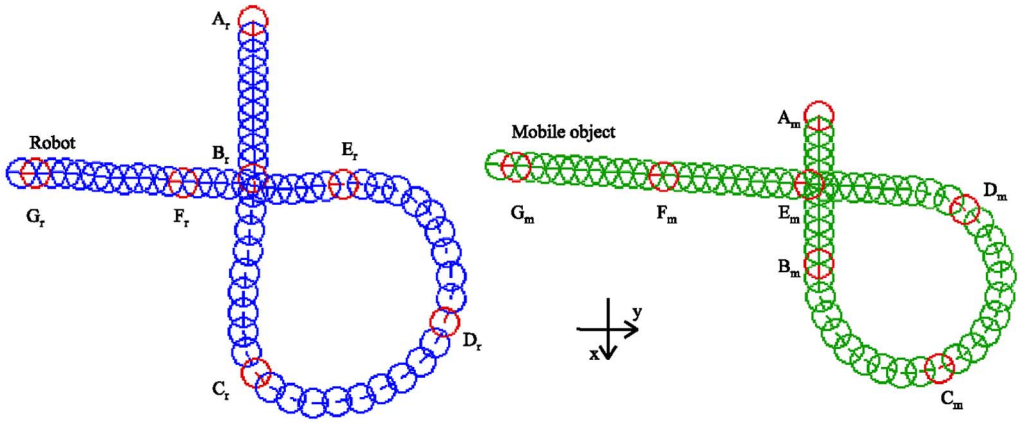
Fig. 9. Knowledge base generated by the COR-based algorithm for the moving object following behavior. (a) Data base and (b) rule base.

placed along the trajectories represent a time gap between them of 3.3 s (ten control cycles).

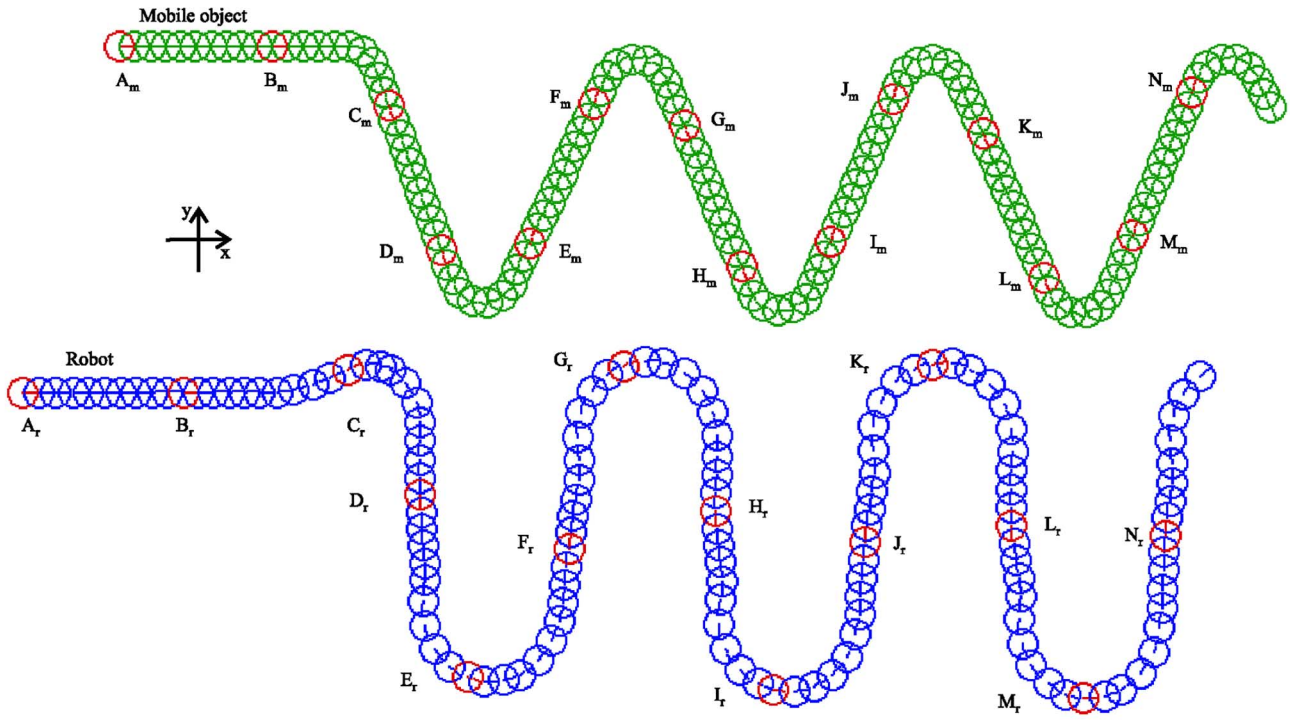
Ten tests have been done for each one of the three analyzed types of trajectories (Fig. 10). The mean values measured



(a)



(b)



(c)

Fig. 10. Trajectories of the robot following different moving objects.

TABLE VII
RESULTS ($\bar{x} \pm \sigma$) FOR THE MOVING OBJECT FOLLOWING BEHAVIOR

Figure	*δ error*	*δdev (degrees)*	*Rvc*
Fig. 10(a)	9.56 ± 0.18	4.23 ± 0.15	7.28 ± 0.19
Fig. 10(b)	19.36 ± 0.43	9.27 ± 0.20	7.03 ± 0.47
Fig. 10(c)	21.97 ± 1.57	20.39 ± 0.67	6.12 ± 0.31

for some parameters that reflect the controller performance are shown in Table VII. These parameters are the average distance error ($\delta d = |d - d_{\text{ref}}|$), the average deviation error $d_{\text{error}} = d \cdot d_{\text{ref}}$, and the average relative velocity change, defined as

$$R_{vc} = \frac{vc_r + 1}{vc_m + 1} \tag{32}$$

where vc_r and vc_m are the average velocity changes for the robot and the moving object, respectively.

In order to show the accuracy of the controller, the three trajectories of Fig. 10 are going to be described.

- Fig. 10(a): this is the easiest example, as there are few turns of the moving object. The object moves with a high velocity (51 cm/s) except between points $B_m - C_m$ and $G_m - H_m$, where velocity is decreased to 25 cm/s in order to test the controller. Moreover, turnings are implemented with an angular velocity of 30°/s. With these conditions, the errors in distance and deviation are low.
- Fig. 10(b): the second type of trajectory is quite difficult because the moving object is changing its movement direction for a long time. Between points A_m-C_m and G_m-L_m, the object moves straight and at 38 cm/s, but between C_m-G_m, the speed is increased to 51 cm/s, and a continuous turning at 20°/s is implemented. Due to this continuous change in the heading of the moving object, errors (Table VII) take a value higher than the previous type of trajectory.
- Fig. 10(c): this example shows a behavior of the moving object that makes very difficult to implement the moving object following task. At the beginning the robot is placed at A_r and the object is placed at A_m (remember that really the y coordinate is the same for both points). The moving object has a linear velocity of 38 cm/s along all the path and implements turnings with the maximum angular velocity (45°/s). These sudden and very sharp changes in direction make it very difficult for the robot to be at the right reference distance and with the adequate reference deviation in the next control cycles. As a result, the errors are the highest ones of the three types of trajectories (Table VII).

To sum up, the accuracy of the controller is good, but when the moving object implements continuous or sharp changes in direction, the controller needs a few control cycles to reach the reference distance and deviation. On the other hand, the interpretability of the obtained rules is good, as all the linguistic labels have the same shape (triangular), are uniformly distributed along the universes of discourse, and have the same meaning for all the rules.

V. CONCLUSION

A methodology for the design of behaviors in mobile robotics has been presented. It consists of both a strategy to formulate the problem as a data-driven learning process and an efficient algorithm to accurately learn interpretable fuzzy rules from them. The learning algorithm is based on COR methodology and extended using an ACO algorithm.

The main characteristics of the approach are as follows. First, the designer only needs to define the universe of discourse, number of labels, and precision of each variable, together with the scoring function. Secondly, learning is done using a set of training examples that have been automatically generated covering the whole universe of discourse of each one of the variables. This makes the learned behavior very general, so the robot will be capable of facing any situation. Thirdly, the learning process is very fast compared with other methodologies (genetic algorithms, neural networks, etc.). Finally, the obtained knowledge bases have a good interpretability, which makes it easy to detect possible errors during the design or the learning process.

This methodology has been applied to the design of two behaviors: wall-following and moving object following. For the first behavior, a controller with 52 linguistic rules has been obtained very quickly (4 m 5 s). It has been tested in a number of simulated environments showing good results in the mean values of some parameters that reflect the quality of the behavior. The system has also been compared with a previous design based on genetic algorithms, increasing the accuracy and interpretability of it. For the moving object following behavior, the obtained controller has 83 rules and has been obtained also quite quickly (7 m 19 s). The knowledge base was tested with three different types of trajectories for the moving object, obtaining good values for the parameters that measure the quality of the behavior.

REFERENCES

[1] R. R. Murphy, *Introduction to AI Robotics*. Cambridge, MA: MIT Press, 2000.
[2] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Comput.*, vol. 1, no. 4, pp. 180–197, 1997.
[3] F. Hoffmann, "An overview on soft computing in behavior based robotics," in *Proc. IFSA 2003 World Congr.*, 2003, pp. 544–551.
[4] M. Mucientes, R. Iglesias, C. V. Regueiro, A. Bugarini, and S. Barro, "A fuzzy temporal rule-based velocity controller for mobile robotics," *Fuzzy Sets Syst.*, vol. 134, pp. 83–99, 2003.
[5] A. Bonarini, "Evolutionary learning of fuzzy rules: Competition and cooperation," in *Fuzzy Modelling: Paradigms and Practice*, W. Pedrycz, Ed. Norwell, MA: Kluwer Academic, 1996, pp. 265–284.
[6] M. Mucientes, D. L. Moreno, C. V. Regueiro, A. Bugarini, and S. Barro, "Design of a fuzzy controller for the wall-following behavior in mobile robotics with evolutionary algorithms," in *Proc. Int. Conf. Inf. Process. Manage. Uncertainty Knowledge-Based Syst. (IPMU'2004)*, Perugia, Italy, 2004, pp. 175–182.
[7] H. Hagras, V. Callaghan, and M. Collin, "Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online fuzzy-genetic system," *Fuzzy Sets Syst.*, vol. 141, pp. 107–160, 2004.
[8] K. Izumi, K. Watanabe, and S.-H. Jin, "Obstacle avoidance of mobile robot using fuzzy behavior-based control with module learning," in *Proc. 1999 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1999, pp. 454–459.
[9] D. Gu, H. Hu, J. Reynolds, and E. Tsang, "GA-based learning in behaviour based robotics," in *Proc. 2003 IEEE Int. Symp. Comp. Intell. Robot. Automat.*, Kobe, Japan, 2003, pp. 1521–1526.
[10] D. Katagami and S. Yamada, "Interactive classifier system for real robot learning," in *Proc. IEEE Int. Workshop Robot-Human Interaction (ROMAN-2000)*, Osaka, Japan, 2000, pp. 258–263.
[11] S. Yamada, "Evolutionary behavior learning for action-based environment modeling by a mobile robot," *Appl. Soft Computat.*, pp. 245–257, 2005.

[12] C. K. Oh and G. J. Barlow, "Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming," in *Proc. Congr. Evol. Computat.*, Portland, OR, 2004, pp. 1538–1545.
[13] T. Dahl and C. Giraud-Carrier, "Evolution-inspired incremental development of complex autonomous intelligence," in *Proc. 8th Int. Conf. Intell. Auton. Syst. (IAS'04)*, Amsterdam, The Netherlands, 2004, pp. 395–402.
[14] D. Floreano and F. Mondada, "Evolutionary neurocontrollers for autonomous mobile robots," *Neural Netw.*, vol. 11, pp. 1461–1478, 1998.
[15] A. Berlanga, A. Sanchis, P. Isasi, and J. M. Molina, "A general learning co-evolution method to generalize autonomous robot navigation behavior," in *Proc. 2000 Congr. Evol. Computat.*, La Jolla, CA, 2000, pp. 769–776.
[16] C. K. Lin, "A reinforcement learning adaptative fuzzy controller for robots," *Fuzzy Sets Syst.*, vol. 137, pp. 339–352, 2003.
[17] C. Zhou, "Robot learning with Ga-based fuzzy reinforcement learning agents," *Inform. Sci.*, vol. 145, pp. 45–68, 2002.
[18] D. Gu, H. Hu, and L. Spacek, "Learning fuzzy logic controller for reactive robot behaviours," in *Proc. 2003 IEEE/ASME Int. Conf. Adv. Intell. Mechatron. (AIM 2003)*, 2003, pp. 46–51.
[19] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 3, pp. 464–477, 1995.
[20] Z. Kalmar, C. Szepesvári, and A. Lörincz, "Module-based reinforcement learning: Experiments with a real robot," *Machine Learn.*, vol. 31, pp. 55–85, 1998.
[21] S. Thongchai, "Behavior-based learning fuzzy rules for mobile robots," in *Proc. Amer. Contr. Conf.*, Anchorage, AK, 2002, pp. 995–1000.
[22] Y. Takahashi and M. Asada, "Multi-layered learning systems for vision-based behavior acquisition of a real mobile robot," in *Proc. SICE Annu. Conf. 2003*, 2003, pp. 2937–2942.
[23] Y. Wang, M. Huber, V. Papudesi, and D. Cook, "User-guided reinforcement learning of robot assistive tasks for an intelligent environment," in *Proc. IEEE/RJS Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, 2003, pp. 424–429.
[24] E. Tuci, M. Quinn, and I. Harvey, "An evolutionary ecological approach to the study of learning behaviour using a robot based model," *Adapt. Behav.*, vol. 10, no. 3/4, pp. 201–221, 2003.
[25] O. Miglino, H. H. Lund, and S. Nolfi, "Evolving mobile robots in simulated and real environments," *Artif. Life*, vol. 2, no. 4, pp. 417–434, 1995.
[26] A. L. Nelson, E. Grant, G. Barlow, and M. White, "Evolution of complex autonomous robot behaviors using competitive fitness," in *Proc. IEEE Int. Conf. Integr. Knowl. Intensive Multi-Agent Syst.*, Boston, MA, 2003, pp. 145–150.
[27] K. J. Lee and B. T. Zhang, "Learning robot behaviors by evolving genetic programs," in *Proc. 26th Int. Conf. Ind. Electron., Contr. Instrum. (IECON-2000)*, 2000, vol. 4, pp. 2867–2872.
[28] O. Fuentes, R. Rao, and M. V. Wie, "Hierarchical learning of reactive behaviors in an autonomous mobile robot," in *IEEE Int. Conf. Syst., Man, Cybern.*, 1995, pp. 4691–4695.
[29] K. Ward, "Rapid simultaneous learning of multiple behaviours with a mobile robot," in *Proc. 2001 Austral. Conf. Robot. Automat.*, Sydney, Australia, 2001, pp. 1–6.
[30] M. Hülse, B. Lara, F. Pasemann, and U. Steinmetz, G. Dorffner, H. Bischof, and K. Hornik, Eds., "Evolving neural behavior control for autonomous robots," *Proc. Int. Conf. Artif. Neural Netw. 2001*, ser. Lecture Notes in Computer Science, vol. 2130, pp. 957–962, 2001.
[31] J. Casillas, O. Cordón, and F. Herrera, "COR: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 4, pp. 526–537, 2002.
[32] ——, "COR methodology: A simple way to obtain linguistic fuzzy models with good interpretability and accuracy," in *Accuracy Improvements in Linguistic Fuzzy Modeling,* J. Casillas, O. Cordon, F. Herrera, and L. Magdalena, Eds. Heidelberg, Germany: Springer, 2003.
[33] J. Casillas, O. Cordón, I. F. de Viana, and F. Herrera, "Learning cooperative linguistic fuzzy rules using the best-worst ant system algorithm," *Int. J. Intell. Syst.*, vol. 20, pp. 433–452, 2005.
[34] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.
[35] J. Urzelai, J. P. Uribe, and M. Ezkerra, "Fuzzy controller for wall-following with a non-holonomous mobile robot," in *Proc. 6th IEEE Int. Conf. Fuzzy Syst. (Fuzz-IEEE'97)*, Barcelona, Spain, 1997, pp. 1361–1368.
[36] L.-X. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, 1992.
[37] P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. Belew and L. Booker, Eds., San Mateo, CA, 1991, pp. 509–513, Morgan Kaufmann.
[38] J. Casillas, O. Cordón, M. del Jesus, and F. Herrera, "Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 1, pp. 13–29, 2005.
[39] N. Pal and K. Pal, "Handling of inconsistent rules with an extended model of fuzzy reasoning," *J. Intell. Fuzzy Syst.*, vol. 7, pp. 55–73, 1999.
[40] A. González and R. Pérez, "A study about the inclusion of linguistic hedges in a fuzzy rule learning algorithm," *Int. J. Uncertain., Fuzziness Knowl.-Based Syst.*, vol. 7, no. 3, pp. 257–266, 1999.
[41] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 260–270, 1995.
[42] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.
[43] J. Casillas, O. Cordón, and F. Herrera, "Learning fuzzy rules using ant colony optimization algorithms," in *Proc. 2nd Int. Workshop Ant Algorithms*, Brussels, Belgium, 2000, pp. 13–21.
[44] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Computat.*, vol. 1, no. 1, pp. 53–66, 1997.
[45] O. Cordón, F. Herrera, I. F. de Viana, and L. Moreno, "A new AGO model integrating evolutionary computation concepts: The best-worst ant system," in *Proc. 2nd Int. Workshop Ant Algorithms*, Brussels, Belgium, 2000, pp. 22–29.

Manuel Mucientes received the M.Sc. and Ph.D. degrees in physics from the University of Santiago de Compostela, Santiago de Compostela, Spain, in 1997 and 2002, respectively.

He is currently an Assistant Professor with the Department of Electronics and Computer Science, University of Santiago de Compostela. His research interests include autonomous mobile robotics, machine learning, evolving fuzzy systems, and pattern recognition.

Jorge Casillas received the B.Sc., M.Sc., and Ph.D. graduate degrees in computer science from the University of Granada, Granada, Spain, in 1996, 1998, and 2001, respectively.

He is an Associate Professor with the Department of Computer Science and Artificial Intelligence, University of Granada, where he is a member of the Soft Computing and Intelligent Information Systems research group. He has worked on several research projects supported by the Spanish government and the European Union. He has coedited two international books and an international special issue. He has organized several special sessions in international conferences. He is coauthor of more than 50 publications in journals, book chapters, and conferences. His research interests include fuzzy modeling, intelligent robotics, knowledge discovery, and metaheuristics.