Learning Weighted Linguistic Rules for Mobile Robotics [*]

M. Mucientes[1] R. Alcalá[2] J. Alcalá-Fdez[2] J. Casillas[2]
[1] *Dept. Electronics and Computer Science. University of Santiago de Compostela. Spain, E-15782*
[2] *Dept. Computer Science and Artificial Intelligence. University of Granada. Spain, E-18071*
Email: manuel@dec.usc.es, alcala@decsai.ugr.es, jalcala@decsai.ugr.es, casillas@decsai.ugr.es

ABSTRACT:
A methodology for learning behaviors in mobile robotics has been developed. The algorithm is based on obtaining cooperative rules with weights, and uses a genetic algorithm to do the combinatorial search. The methodology has been employed to learn the wall-following behavior, and the obtained controller has been tested using the Nomad 200 simulation software in different environments.

Keywords: Mobile robotics, behaviors, fuzzy modeling, evolutionary algorithms.

1 Introduction

Control in mobile robotics requires different levels of action: planning (high level) and reacting (low level). Usually, the reactive layer is composed of behaviors that act directed by the planning level. These behaviors are implemented in different ways, being one of the most usual a fuzzy controller.

The characteristic that makes specially useful a fuzzy controller for the implementation of a behavior is the ability that fuzzy controllers have in order to cope with noisy inputs. This noise appears when the sensors of the robot detect the surrounding environment, and is particularly high when using ultrasound sensors (specular reflection, low angular resolution, etc.).

However, learning or tuning a fuzzy controller is a tedious task, and for this reason some learning techniques have been applied (evolutionary algorithms, neural networks, ...). Evolutionary algorithms have the advantage that can learn interpretable rules and, also, that the designer can select the most adequate tradeoff between interpretability and accuracy for the knowledge base that is going to be learned.

In [2], a new learning methodology to induce a better cooperation among the fuzzy rules was proposed: the Cooperative Rules (COR) methodology. The learning philosophy was based on the use of *ad hoc data-driven methods*[1] to determine the fuzzy input subspaces where a rule should exist and a set of candidate consequents assigned to each rule. After that, a combinatorial search was carried out in the set of candidate consequents to obtain a set of rules with good cooperation among them. In [3], different combinatorial search techniques were considered with this aim.

On the other hand, other technique to improve the rule cooperation is the use of weighted fuzzy rules [5, 8], in which modifying the linguistic model structure an importance factor (weight) is considered for each rule. By means of this technique, the way in which these rules interact with their neighbor ones could be indicated.

In [1], the Weighted COR (WCOR) methodology was presented to include the weight learning within the original COR methodology. In this way, both techniques were combined to obtain weighted cooperative fuzzy rules. Thus, the system accuracy is increased while the interpretability is maintained to an acceptable level.

In this paper, a methodology, based on WCOR, for the design of behaviors in mobile robotics is presented. The algorithm has been applied to the wall-following behavior. In order to show the performance of the obtained controller, it has been tested in several simulated environments, and it has also been compared with a previous approach based on COR [7]. The paper is structured as follows: section 2 introduces the wall-following behavior, section 3 describes the COR methodology, while in section 4 WCOR is presented. Finally, some results are shown, and conclusions are pointed out.

2 Learning the wall-following behavior

In order to evaluate the proposed methodology, we have selected the wall-following behavior, which is usually implemented when the robot is exploring an unknown area, or when it is moving between two points in a map. 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 finally 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, which is the desired distance between the robot and the selected wall, and the maximum velocity attainable by the robot. In what follows we assume 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, but this can be easily dealt with by simply interchanging the sensorial inputs.

The input variables of the control system are the right-hand distance (*RD*), the distances quotient (*DQ*), which is calculated as:

$$DQ = \frac{left - hand\,distance}{RD} \quad (1)$$

[*]This work was supported in part by the Spanish Ministry of Science and Technology under grants no. TIC2003-09400-C04-03 and TIC2002-04036-C05-01, and the DXID of the Xunta de Galicia under grant PGIDIT04TIC206011PR.

[1]A family of efficient and simple methods guided by covering criteria of the data in the example set.

DQ shows the relative position of the robot inside a corridor, which provides with information that is more relevant 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, whilst a low value indicates that the closer wall is the left-hand one. The other input variables are the linear velocity of the robot (LV) and the orientation of the robot with respect to the wall it is following. A positive value of the orientation indicates that the robot is approaching to the wall, whilst a negative value means the robot is moving away from the wall. The output variables are the linear acceleration and the angular velocity.

The values for the distances and the orientation are obtained from the distances measured by the ultrasound sensors of the robot. We use the *distributed perception* [10]: distance is measured as the minimum distance of a set of sensors, and the orientation will be a weighted sum of the orientation of each sensor in the set, giving more weight to those sensors that detect closer obstacles.

A set of examples (1638) has been chosen for learning the knowledge base. These examples cover the universe of discourse of all the variables in the antecedent part of the rule. The universes of discourse have been discretized, in order to minimize the search space, with a step or precision p_n, where n is the variable. Function SF, that scores the action of the rule base over an example, is defined as:

$$SF\left(RB(e^l)\right) = \alpha_1 + \alpha_2 + \alpha_3 \tag{2}$$

where e^l is an example, and α_1, α_2, and α_3 are respectively:

$$\alpha_1 = 100 \cdot \frac{|RD - reference\,distance|}{p_{RD}} \tag{3}$$

$$\alpha_2 = 10 \cdot \frac{|maximum\,velocity - LV|}{p_{LV}} \tag{4}$$

$$\alpha_3 = \frac{|orientation|}{p_{orientation}} \tag{5}$$

Thus, low values of SF indicate a good control action (a score of 0 means that the robot has reached the best state). p_{RD}, p_{LV}, and $p_{orientation}$ 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 (3), (4), and (5) respectively) have been heuristically 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 velocity and, finally, the least important contribution to function SF is for the orientation of the robot.

The index that measures the global quality of the encoded rule set is:

$$f(RB) = \frac{1}{2 \cdot NE} \sum_{l=1}^{NE} \left(g(e^l)\right)^2 \tag{6}$$

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

$$g(e^l) = \begin{Bmatrix} \left(1 - h(e^l)\right) \cdot \omega + 1 & if\; h(e^l) \le 1 \\ \exp\left(1 - h(e^l)\right) & if\; h(e^l) > 1 \end{Bmatrix} \tag{7}$$

being ω a scaling factor that has been set to 1000, and $h(e^l)$:

$$h(e^l) = \frac{\min\left(SF(e^l)\right) + 1}{SF\left(RB(e^l)\right) + 1} \tag{8}$$

where $\min\left(SF(e^l)\right)$ is the minimum score that an action can obtain for example e^l (using only the discrete values of the output variables). These values are obtained before the beginning of the algorithm, trying and scoring all the possible actions for each example.

3 The COR Methodology

The process followed to learn the fuzzy controller is based on the COR methodology (proposed in [2] and extended in [4]). The COR methodology is guided by example covering criteria to obtain antecedents (fuzzy input subspaces) and candidate consequents [3]. Depending on the combination of this technique with different *ad hoc data-driven methods*, different learning approaches arise. In this work, we will consider the Wang and Mendel's method [11] (WM) for this purpose —approach guided by examples—. The COR methodology following this approach 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. 1.

The above mentioned methodology has some interesting advantages that make it very useful to learn fuzzy controllers in mobile robots navigation. We can mainly highlight two characteristics:

1. *Search space reduction* — The COR methodology reduces the search space respect to other rule base learning methods [9] and allows it to be quicker and to make a better solution exploration. It is due to two main reasons:

   - The fact of assigning each example to only one subspace will involve to reduce the number of candidate consequents, since the positive example sets are reduced.
   - The use of a restrictive condition to construct $C(S_h)$ (see eq. (10) in Fig. 1) that generates a low number of candidate rules.

   This is an important issue for the learning of fuzzy controllers, where a high number of examples is used. In the wall-following behavior presented in this paper, 1638 examples have been used, and the employed methodology spends around one hour (with an Intel Pentium 4 2.4 GHz processor) in order to obtain the controller.

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 i-th input variable, with $i \in \{1, \ldots, n\}$, and $\mathcal{B}_j$ be the set of linguistic terms of the j-th output variable, with $j \in \{1, \ldots, m\}$, with $|\mathcal{A}_i|$ ($|\mathcal{B}_j|$) being the number of labels of the i-th (j-th) 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:

   $$E^+(S_s) = \{ \quad e_l \in E \mid \forall i \in \{1, \ldots, n\}, \\ \forall A_i' \in \mathcal{A}_i, \mu_{A_i^s}(x_i^l) \geq \mu_{A_i'}(x_i^l) \quad \} \tag{9}$$

   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:

   $$C(S_h) = \{ \quad (B_1^{k_h}, \ldots, B_m^{k_h}) \in \mathcal{B}_1 \times \cdots \times \mathcal{B}_m \mid \\ \exists e_l \in E^+(S_h) \text{ where } \forall j \in \{1, \ldots, m\}, \\ \forall B_j' \in \mathcal{B}_j,\ \mu_{B_j^{k_h}}(y_j^l) \geq \mu_{B_j'}(y_j^l) \quad \}. \tag{10}$$

   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 $(B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)\}$.

   To allow COR to reduce the initial number for fuzzy rules, the special element $R_\emptyset$ (which means "do not 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 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 excessive number of rules:

   $$f'(RB) = f(RB) + \beta \cdot f(RB_0) \cdot \frac{\#RB}{|S^+|} \tag{11}$$

   with $\beta \in [0,1]$ 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., $|S^+| - |\{R_h \in RB \text{ such that } R_h = R_\emptyset\}|$), and RB_0 being the initial rule base considered by the search algorithm.

Figure 1: COR algorithm

2. *Interpretability issues* — The proposed methodology has also some interesting advantages from the interpretability of the obtained fuzzy knowledge point of view. In this case, the membership functions and the model structure keep invariable, since the COR methodology improves the accuracy by only inducing cooperation among linguistic fuzzy rules. Furthermore, the COR methodology achieves a rule 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 rules is more interpretable) of the learned model. These are important issues in fuzzy control for mobile robot navigation, as the actions of the robot are easily understandable.

Finally, since the search space tackled in step 2. is usually large, it is necessary to use approximate search techniques. In [3] four different well-known techniques were proposed for this purpose. One of them, the ant colony optimization (ACO) [6], was first applied to the fuzzy control for mobile robot navigation in [7]. We will consider this approach for comparison in the experiments section.

4 Learning Weighted Linguistic Rules based on COR

In this section we present an extension of the COR methodology to obtain a cooperative set of weighted linguistic rules. In the following, we present the use of the weighted linguistic rules and the said WCOR methodology (learning scheme and evolutionary algorithm).

4.1 The Use of Weighted Linguistic Rules

Using rule weights [5, 8] has been usually considered to improve the way in which rules interact, improving the accuracy of the learned model. In this way, rule weights suppose an effective extension of the conventional fuzzy reasoning system that allow the tuning of the system to be developed at the rule level [5, 8].

When weights are applied to complete rules, the corresponding weight is used to modulate the firing strength of a rule in the process of computing the defuzzified value. From human beings, it is very near to consider this weight as an importance degree associated to the rule, determining how this rule interacts with its neighbor ones. We will follow this approach, since the interpretability of the system is appropriately maintained. In addition, we will only consider weight values in $[0,1]$ since it preserves the model readability. In this way, the use of rule weights represents an ideal framework for extended linguistic fuzzy modeling when we search for a trade-off between accuracy and interpretability. In order to do so, we will follow the weighted rule structure and the inference system proposed in [8] extended for multiple output variables:

$$\text{IF } X_1 \text{ is } \mathcal{A}_1 \text{ and } \ldots \text{ and } X_n \text{ is } \mathcal{A}_n \\ \text{THEN } Y_1 \text{ is } \mathcal{B}_1 \text{ and } \ldots \text{ and } Y_m \text{ is } \mathcal{B}_m \text{ with } [w], \tag{12}$$

where X_i (Y_j) are the linguistic input (output) variables, A_i (B_j) are the linguistic labels used in the input (output) variables, w

is the real-valued rule weight, and *with* is the operator modeling the weighting of a rule.

With this structure, the fuzzy reasoning must be extended. The classical approach is to infer with the FITA (First Infer, Then Aggregate) scheme and compute the defuzzified output of the j-th variable as the following *weighted sum*:

$$y(j) = \frac{\sum_h m_h \cdot w_h \cdot P_h(j)}{\sum_h m_h \cdot w_h}, \qquad (13)$$

with m_h being the matching degree of the h-th rule, w_h being the weight associated to the h-th rule, and P_{hj} being the characteristic value of the output fuzzy set corresponding to that rule in the j-th variable. In this contribution, the center of gravity will be considered as characteristic value and the *minimum t-norm* will play the role of the implication and conjunctive operators.

A simple approximation for weighted rule learning would consist in considering an optimization technique to derive the associated weights of the previously obtained rules (e.g., by means of ad hoc data-driven methods as WM, or even COR).

4.2 The WCOR Methodology

This methodology involves an extension of the original COR methodology. Therefore, WCOR [1] consists of the following steps:

1. *Obtain the subspaces with positive examples $S_h \in S^+$ and a set of candidate consequents $C(S_h)$ associated to them.*
2. *Problem representation*. For each rule R_h we have: S_h, $C(S_h)$, and $w_h \in [0,1]$.

   Since S_h is kept fixed, the problem will consist of determining the consequents and the weight associated to each rule. Two vectors, c_1 and c_2, of size $|S^+|$ (number of rules finally obtained) are defined to represent this information, where,

   $$c_1[h] = k_h \mid R_{k_h} \in CR(S_h) \qquad (14)$$

   $$c_2[h] = w_h,\ \forall h \in \{1,\dots,|S^+|\} \qquad (15)$$

   In this way, the c_1 part is an integer-valued vector in which each cell represents the index of the consequents used to build the corresponding rule. The c_2 part is a real-valued vector in which each cell represents the weight associated to this rule. Finally, a problem solution is represented as follows:

   $$c = c_1\ c_2 \qquad (16)$$

3. *Perform a search on the c vector, looking for the combination of consequents and weights with the best cooperation.* To do that, we consider the use of a simple Genetic Algorithm (GA).

4.3 Genetic Algorithm Applied to the WCOR Methodology

The proposed GA performs an approximate search among the candidate rules with the main aim of selecting the set of consequents with the best cooperation and simultaneously learning the weights associated to the obtained rules. The main characteristics of the said algorithm are presented in the following:

- *Genetic Approach* — An elitist generational GA with the Baker's stochastic universal sampling procedure.
- *Initial Pool* — The initial pool is obtained by generating a possible combination at random for the c_1 part of each individual in the population. And for the c_2 part, it is obtained with an individual having all the genes with value '1', and the remaining individuals generated at random in $[0,1]$.
- *Fitness Function* — The fitness function will be the said objective function, defined in eq. (11) in Fig. 1.
- *Crossover* — The standard two-point crossover in the c_1 part combined with the max-min-arithmetical crossover in the c_2 part. By using the max-min-arithmetical crossover, if $c_2^v = (c[1],\dots,c[k],\dots,c[n])$ and $c_2^w = (c'[1],\dots,c'[k],\dots,c'[n])$ are crossed, the next four offspring are obtained:

  $$c_2^1 = ac_2^w + (1-a)c_2^v \qquad (17)$$

  $$c_2^2 = ac_2^v + (1-a)c_2^w \qquad (18)$$

  $$c_2^3 \text{ with } c_3[k] = \min\{c[k],c'[k]\} \qquad (19)$$

  $$c_2^4 \text{ with } c_4[k] = \max\{c[k],c'[k]\} \qquad (20)$$

  with $a \in [0,1]$ being a parameter chosen by the GA designer.

  In this case, eight offspring are generated by combining the two ones from the c_1 part (two-point crossover) with the four ones from the c_2 part (max-min-arithmetical crossover). The two best offspring so obtained replace the two corresponding parents in the population.
- *Mutation* — The operator considered in the c_1 part randomly selects a gene ($h \in \{1,\dots,|S^+|\}$) and changes at random the current consequent B_{k_h} by other consequent $B_{k_h'}$ such that $R_{k_h'} \in CR(S_h)$. On the other hand, the selected gene in the c_2 part takes a value at random within the interval $[0,1]$.

5 Results

The learned controller has been tested using the Nomad 200 simulation software. Six environments have been chosen which include very different situations that the robot usually faces during navigation: straight walls of different lengths, followed and/or preceded of a number of concave and convex corners, gaps, ... thus covering a wide range of contours to follow and truly defining very complex test environments. It is important to remark that none of these environments have been used during the learning process. Thus, the training set has been a set of 1638 examples that uniformly cover the universe of discourse of the input variables.

Figure 2 shows the robot path along one of the test environments. The robot trajectory is represented by circular marks. A higher concentration of marks indicates lower velocity. 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.

Several controllers have been learned for different values of β (equation 11), a parameter to regulate the importance of the

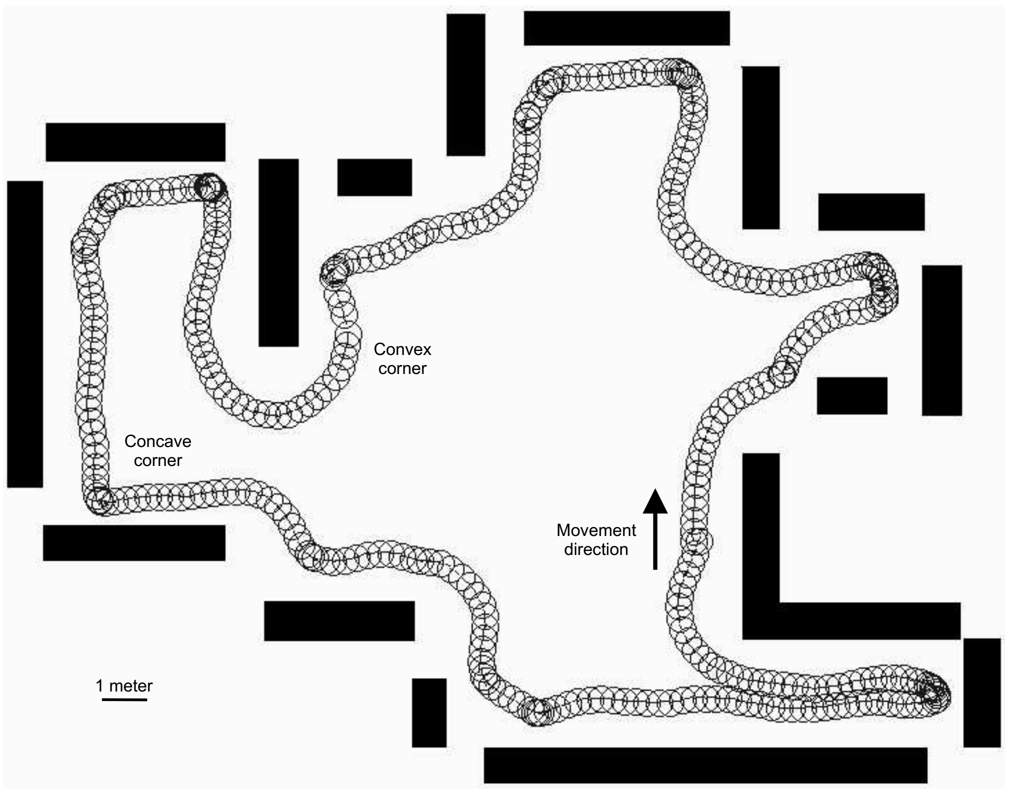

Figure 2: Path of the robot along environment E.

number of rules. Finally, the controller with $\beta = 0.2$ has been selected. This controller has also been compared with another one obtained using the methodology presented in [7]. In [7] the COR methodology without weights had been applied for the same purpose, but in that case the search technique was the ant colony optimization algorithm, while in this approach a genetic algorithm has been used.

The controllers have 46 rules (WCOR) and 55 rules (COR) [7] respectively (both learned for $\beta = 0.2$). Ten tests have been done for each one of the analyzed environments. The average values measured for some parameters that reflect the controllers performance are shown in table 1, while table 2 shows the values of these parameters for the COR controller [7]. The 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.

Table 1: Average values of some parameters for WCOR controller (46 rules).

Env.	RD (cm)	Vel. (cm/s)	ΔVel. (cm/s)	Time (s)
A	57	54	3.87	101
B	58	56	3.96	61
C	51	54	3.09	74
D	55	56	3.60	108
E	49	50	6.17	114
F	55	54	4.13	101

As can be seen, the controller learned using WCOR has increased the average velocity in most of the environments around a 10% without getting worse the average right-hand distance. Also the smoothness of the behavior has been improved in all of the environments. The improvement of these parameters due to the use of weights for the rules has been obtained without loosing interpretability of the knowledge base.

Table 2: Average values of some parameters for COR controller (55 rules) [7].

Env.	RD (cm)	Vel. (cm/s)	ΔVel. (cm/s)	Time (s)
A	55	49	5.57	113
B	57	50	7.52	69
C	52	41	5.65	97
D	54	54	5.59	112
E	50	48	7.00	121
F	53	49	7.19	111

The Data Base and the Rule Base of the model obtained by WCOR are respectively presented in Figures 3 and 4. In Figure 4, each row of the table represents a fuzzy subspace and contains its associated output consequents, i.e., the correspondent labels together with its respective rounded rule weight. Moreover, these weights have been graphically showed by means of the gray scale, from black (1.0) to white (0.0).

In order to show the quality of the controller we are going to describe in detail the path of the robot in environment E (figure 2). As it has been said, this environment is quite complex, with ten concave corners and six convex corners in a circuit of a length of 57 meters. The measurements of the ultrasound sensors are quite noisy due to the gaps present in the wall, and also because of the convex corners. These are truly difficult situations, because the robot's sensors may cease to correctly detect the wall at some given moments. The controller must also significantly reduce velocity at corners. All these situations provoke a reduction in the average distance and velocity. As can be seen, the robot follows the wall with a high precision, except at the corners, where it approaches to the wall (concave corners) or gets away from it (convex corners) in order to turn.

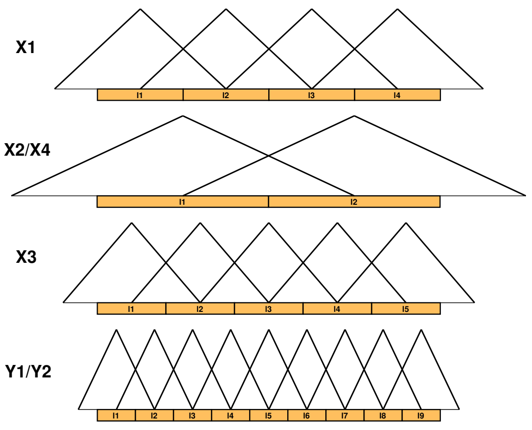

Figure 3: Data Base of the obtained model.

#R: 46

	X1	X2	X3	X4	Y1,Y2	*with*
▓	l1	l1	l2	l2	l1,l2	[0.660]
░	l1	l1	l3	l1	l5,l5	[0.297]
░	l1	l1	l3	l2	l4,l3	[0.360]
░	l1	l1	l4	l2	l8,l2	[0.333]
░	l1	l2	l1	l2	l1,l2	[0.334]
■	l1	l2	l2	l1	l3,l6	[0.932]
░	l1	l2	l2	l2	l4,l3	[0.382]
▓	l1	l2	l3	l1	l7,l6	[0.743]
▓	l1	l2	l3	l2	l8,l3	[0.610]
■	l1	l2	l4	l1	l9,l6	[0.986]
■	l1	l2	l4	l2	l9,l2	[0.947]
■	l1	l2	l5	l1	l9,l1	[0.873]
▒	l2	l1	l1	l1	l1,l1	[0.417]
■	l2	l1	l1	l2	l1,l1	[0.965]
▓	l2	l1	l2	l1	l1,l8	[0.771]
▒	l2	l1	l3	l1	l1,l7	[0.408]
▒	l2	l1	l3	l2	l3,l4	[0.440]
▓	l2	l1	l4	l1	l7,l7	[0.729]
▒	l2	l1	l4	l2	l7,l4	[0.466]
▒	l2	l1	l5	l1	l9,l6	[0.519]
▓	l2	l1	l5	l2	l9,l3	[0.698]
■	l2	l2	l2	l1	l1,l8	[0.976]
■	l2	l2	l2	l2	l1,l6	[0.957]
▓	l2	l2	l3	l1	l8,l8	[0.751]
▒	l2	l2	l3	l2	l7,l6	[0.559]
▒	l2	l2	l5	l2	l9,l1	[0.465]
■	l3	l1	l1	l2	l1,l1	[0.981]
■	l3	l1	l2	l2	l1,l6	[0.981]
■	l3	l1	l3	l1	l2,l8	[0.879]
░	l3	l1	l3	l2	l1,l4	[0.273]
▒	l3	l1	l4	l1	l8,l8	[0.498]
▒	l3	l1	l4	l2	l7,l6	[0.425]
▓	l3	l1	l5	l1	l9,l8	[0.765]
▓	l3	l1	l5	l2	l9,l4	[0.683]
▓	l3	l2	l3	l1	l1,l8	[0.647]
▒	l3	l2	l3	l2	l1,l6	[0.426]
░	l3	l2	l4	l2	l7,l6	[0.368]
■	l4	l1	l1	l1	l1,l1	[0.929]
░	l4	l1	l2	l2	l1,l6	[0.195]
■	l4	l1	l3	l1	l1,l8	[0.982]
■	l4	l1	l3	l2	l1,l6	[0.960]
▒	l4	l1	l4	l1	l1,l8	[0.585]
░	l4	l1	l4	l2	l1,l6	[0.374]
▒	l4	l1	l5	l1	l5,l8	[0.528]
░	l4	l1	l5	l2	l6,l6	[0.365]
▓	l4	l2	l4	l1	l2,l8	[0.600]

Figure 4: Rule Base of the obtained model.

6 Conclusions

A methodology for the design of behaviors in mobile robotics has been presented. It is based on COR (Cooperative Rules) with weights (WCOR), and uses a genetic algorithm to perform the search. Using rule weights improves the accuracy of the knowledge base (the way rules interact), while maintaining a good interpretability.

The system has been tested learning the wall-following behavior in mobile robotics using the Nomad 200 software, showing a good performance in the different environments in which it has been tested. Also, the learned controller has been compared with another one obtained using the COR methodology [7]. The WCOR controller improves the average linear velocity and the smoothness of the behavior maintaining an adequate distance to the right-hand wall.

REFERENCES

[1] R. Alcalá, J. Casillas, O. Cordón, and F. Herrera. Improving simple linguistic fuzzy models by means of the weighted cor methodology. In *Advances in Artificial Intelligence - IBERAMIA 2002*, number 2527 in Lecture Notes in Artificial Intelligence, pp. 294–302, Sevilla (Spain), 2002. Springer-Verlag.

[2] 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*, 32(4):526–537, 2002.

[3] J. Casillas, O. Cordón, and F. Herrera. Different approaches to induce cooperation in fuzzy linguistic models under the cor methodology. In B. Bouchon-Meunier, J. Gutiérrez-Ríos, L. Magdalena, and R. R. Yager, editors, *Techniques for Constructing Intelligent Systems*, pp. 321–334. Springer-Verlag, 2002.

[4] J. Casillas, O. Cordón, and F. Herrera. COR methodology: a simple way to obtain linguistic fuzzy models with good interpretability and accuracy. In J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors, *Accuracy improvements in linguistic fuzzy modeling*. Springer, Heidelberg, Germany, 2003.

[5] J. S. Cho and D. J. Park. Novel fuzzy logic control based on weighting of partially inconsistent rules using neural network. *J. Intell. Fuzzy Systems*, 8:99–110, 2000.

[6] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst., Man, Cybern. B*, 26(1):29–41, 1996.

[7] M. Mucientes and J. Casillas. Obtaining a fuzzy controller with high interpretability in mobile robots navigation. In *Proc. Fuzz-IEEE 2004*, pp. 1637–1642, Budapest (Hungary), 2004.

[8] N. R. Pal and K. Pal. Handling of inconsistent rules with an extended model of fuzzy reasoning. *J. Intell. Fuzzy Systems*, 7:55–73, 1999.

[9] P. Thrift. Fuzzy logic synthesis with genetic algorithms. In R.K. Belew and L.B. Booker, editors, *Proc. 4th Int. Conf. on Genetic Algorithms*, pp. 509–513, San Mateo, CA, USA, 1991. Morgan Kaufmann Publishers.

[10] J. Urzelai, J. P. Uribe, and M. Ezkerra. Fuzzy controller for wall-following with a non-holonomous mobile robot. In *Proc. Fuzz-IEEE'97*, pp. 1361–1368, Barcelona (Spain), 1997.

[11] L.-X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. Syst., Man, Cybern.*, 22(6):1414–1427, 1992.