Expert Systems with Applications 37 (2010) 1471–1493

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A case study for learning behaviors in mobile robotics by evolutionary fuzzy systems ☆

M. Mucientes [a,*], J. Alcalá-Fdez [b], R. Alcalá [b], J. Casillas [b]

[a] Department of Electronics and Computer Science, University of Santiago de Compostela, E-15782, Spain
[b] Department of Computer Science and Artificial Intelligence, University of Granada, E-18071, Spain

ARTICLE INFO

Keywords:
Service robots
Data-driven learning algorithms
Evolutionary algorithms
Fuzzy controllers
Design of behaviors

ABSTRACT

Service robots will play an increasing and more important role in the society in the next years. One of the main challenges is to endow robots with enough autonomy to operate on real environments. To reach that goal, the design of controllers to solve simple tasks must be automatized. Engineers look for learning algorithms that are general, robust, require low expertise knowledge, and generate controllers that can run on the real robot without any tuning stage. In this paper, a framework to learn behaviors (controllers) in mobile robotics, fulfilling the previous requirements, has been used. The framework is based on two modules: dataset generation and a data-driven evolutionary-based learning algorithm to obtain fuzzy controllers. Nevertheless, the design of a fuzzy controller still requires the selection of the type of learning algorithm, and also to choose the value of some design parameters. In this paper we present an exhaustive study on a set of evolutionary-based data-driven learning algorithms, for learning fuzzy controllers in mobile robotics, that cover a wide range of the accuracy/interpretability trade-off. The study has also evaluated the influence of the values of all the design parameters over accuracy and interpretability. The objective is to analyze the performance of the different algorithms for the design of behaviors in mobile robotics, and to extract some general rules that can help in the process to design new behaviors. The analysis comprises two different behaviors (wall-following and moving object following) and more than 450 tests, both in simulation and on a *Pioneer II AT* robot. Results have shown very good performances in complex and realistic conditions for the different combinations of algorithms and parameters.

1. Introduction

Nowadays service robots are awaking a greater interest, not only in the robotics community, but also in the society and the companies. In the last years, it is not infrequent to read sentences like "In 25 years,... more than 75% of homes will have at least one robot regularly performing routine chores" (Angle, 2008), and these kind of ideas are starting to be considered as plausible. A service robot is a robot that can assist people, generally, for performing a job or task that is dangerous, dull, repetitive, etc. Although service robots can be autonomous and/or manually operated, a high degree of autonomy is desirable. Autonomy has two great advantages: the robot can operate without human supervision and in unstructured environments. These two characteristics, together with the fact that some companies are starting to sell service robots ready to operate without any modification, are making service robots popular. Some examples of tasks that are successfully done by service robots are: cleaning and housekeeping (pool cleaner, vacuum cleaner, window cleaner), lawn mowing, entertainment, surveillance or transportation. Also, another field that is generating the attention of the robotics research community is the assistance to people at home, by helping elderly and handicapped people to move and communicate.

To develop this kind of tasks, the robot must be endowed with a control system. The control of an autonomous robot can be generally described at two levels: the reactive and the deliberative (or planning) layers. On the one hand, at the deliberative layer the robot solves those tasks that require knowledge of current and previous sensor readings, like planning a route, or selecting the set of actions that are going to be implemented to fulfill a goal. On the other hand, the reactive layer is usually composed of a set of behaviors. A behavior is a controller that is able to solve a specific task on real time, like crossing a door, following a moving object, avoiding a collision, following a contour, etc.

Fuzzy logic has been widely used in robotics (Saffiotti, 1997), and particularly for the implementation of behaviors (Hoffmann, 2003; Mucientes, Iglesias, Regueiro, Bugarín, & Barro, 2003;

☆ Manuel Mucientes is supported by the *Ramón y Cajal* program of the Spanish Ministry of Science and Innovation.
* Corresponding author.
E-mail addresses: manuel.mucientes@usc.es (M. Mucientes), jalcala@decsai.ugr.es (J. Alcalá-Fdez), alcala@decsai.ugr.es (R. Alcalá), casillas@decsai.ugr.es (J. Casillas).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.06.095

Mucientes et al., 2001). The main reasons are, firstly, the inherent uncertainty of real environments in which a robot operates: there are people moving in the surrounding of the robot, the obstacles can be moved (a chair can be displaced, a door can be opened or closed, etc.). In second place, the information of the environment is provided by sensors that are not ideal: there is lack of reliability in the data provided by a sensor, and this uncertainty is different for each kind of sensor. Finally, the actuators of the robot are also imperfect, thus the control orders are never performed with precision. Fuzzy controllers are able to cope with this unreliability, uncertainty and lack of robustness, and this is the reason of their success in robotics.

In general, the design of a fuzzy controller is a difficult and highly time consuming task, as there are lots of values of parameters to define. If the design does not incorporate previous/expertise knowledge of the process to be controlled, then the designer has to define the universe of discourse of each variable, the number of labels, the description of each label, and the consequent of each rule. Thus, it seems adequate to use a learning algorithm to design a fuzzy controller.

The implementation of behaviors in robotics is specially difficult, as the controller must be run on the real robot but, generally, the learning algorithm cannot. A learning process usually lasts for a long time, trying thousands of different situations. On the real robot this would require the supervision of a human along the learning time. Moreover, the learning methodologies try the same situation with different outputs to evaluate the performance, but on the real robot it is very difficult to reproduce the same conditions several times, and to place the robot in the same initial position. Also, during the learning stage, control actions that take the robot into hazardous situations could be tried. On the real robot these situations can provoke damage on the robot due to a collision. In summary, learning on the real robot presents several drawbacks, and from our point of view it should only be considered for tuning.

Learning of fuzzy controllers in mobile robotics is usually solved learning off-line, using a simulator of the real robot. Depending on the quality of the learning methodology and its interaction with the simulator, the learned controller can be directly used on the real robot, or it will require some kind of tuning. The increase of popularity of service robots demands the existence of good and easy to use algorithms to learn behaviors. With good algorithms, we mean methodologies that do not require any learning/tuning stage on the real robot. Thus, a behavior is learned off-line, and directly implemented on the real robot. Also, we want easy to use algorithms: methodologies that require the definition of a low number of parameters, and low expertise knowledge. In that way, it would be easy to guide an inexpert user through the process for learning a new behavior.

At this point, two questions arise: which methodology should be used to learn a fuzzy controller? And, what kind of fuzzy controller is the most adequate? The most popular groups of learning methodologies for fuzzy controllers in robotics are evolutionary algorithms (Dahl & Giraud-Carrier, 2004; Gu, Hu, Reynolds, & Tsang, 2003; Hagras, Callaghan, & Collin, 2004; Izumi, Watanabe, & Jin, 1999; Katagami & Yamada, 2000; Mucientes, Moreno, Bugarín, & Barro, 2006; Mucientes, Moreno, Bugarín, & Barro, 2007; Oh & Barlow, 2004; Yamada, 2005), neural networks (Hui, Mahendar, & Pratihar, 2006; Shiah & Young, 2004) and reinforcement learning (Beom & Cho, 1995; Bonarini, 1997; Gu, Hu, & Spacek, 2003; Kalmár, Szepesvári, & Lörincz, 1998; Lin, 2003; Takahashi & Asada, 2003; Thongchai, 2002; Wang, Huber, Papudesi, & Cook, 2003; Zhou, 2002). Also, combinations of them, like neural networks and evolutionary algorithms (Berlanga, Sanchis, Isasi, & Molina, 2000; Chen & Chiang, 2004; Floreano & Mondada, 1998; Lee & Zhang, 2000; Miglino, Lund, & Nolfi, 1995; Nelson, Grant, Barlow, & White, 2003; Tuci, Quinn, & Harvey, 2003), have been successfully applied.

Evolutionary algorithms have some characteristics that make them specially adequate for learning fuzzy controllers. The flexibility in the representation of the solutions is very high. A chromosome can codify a complete knowledge base, a rule, some parameters of the fuzzy sets, etc. For this reason, this kind of optimization techniques can easily handle any type of fuzzy systems. Indeed, the combination of fuzzy logic with the learning capabilities of evolutionary algorithms is well known as evolutionary fuzzy systems (Cordón, Herrera, Hoffmann, & Magdalena, 2001). Thus, depending on the characteristics of the problem and the demands of the final user, the designer has the possibility to determine the most appropriate trade-off between accuracy and interpretability by selecting different kinds of fuzzy controllers. This property of evolutionary algorithms links with the second of the questions that must be decided to learn a fuzzy controller: the type of fuzzy controller that should be learned. The controller can be Mamdani or TSK, it can have weighted rules, it can be defined with global or local semantics, etc. The decision of the type of controller will be based on the desired trade-off between accuracy and interpretability.

In summary, a designer has the possibility to choose different combinations of fuzzy controllers and learning algorithms. As far as we know, no exhaustive studies about the performance of all these choices has been done from the point of view of the design of behaviors in mobile robotics. Exhaustive means that several types of evolutionary fuzzy systems are analyzed and compared, for different behaviors, in many different situations (both simulated and on the real robot), and with a deep study about the influence of the different design parameters.

In this paper, we present a deep experimental study on the performance of different evolutionary fuzzy systems for the implementation of behaviors in robotics. The selected methodologies cover a wide range of the accuracy/interpretability trade-off. All of them are based on the application of a technique for the generation of datasets, proposed in Mucientes and Casillas (2007), to learn behaviors in robotics. Two totally different behaviors have been considered for evaluation purposes: wall-following and moving object following. For both behaviors, an exhaustive analysis has been performed with a robot simulator, together with the evaluation of the best controllers on the real robot. The analysis includes a study on several parameters like the number of labels, number of examples, importance of rule base reduction in the fitness function, weights in the fitness function, and other parameters that are behavior dependent.

The paper is structured as follows: Section 2 presents in short the framework to learn behaviors in mobile robotics, and Section 3 describes the four data-driven learning algorithms. Section 4 describes the exhaustive studies that have been performed on two different behaviors and, finally, Section 5 points out the conclusions.

2. A framework to learn behaviors in mobile robotics

Our approach to learn behaviors for autonomous mobile robots comprises two different modules: the dataset generation and the learning algorithm. The objective of the dataset generation is to provide a methodology that enables any behavior to be learned by any general purpose data-driven learning algorithm. The advantages of the dataset generation methodology are:

- Provides the designer with a set of simple steps to design any behavior.
- Generates a dataset that covers all the meaningful situations that the controller could face when it is running on the real robot.

- Obtains a behavior that can be directly implemented on the real robot without any additional learning/tuning stage.
- Gives the designer freedom to choose any data-driven learning methodology, as the dataset generation is independent.

The second of the modules of the framework is a data-driven learning algorithm. In this paper we have used four different algorithms, covering a wide spectrum of the accuracy/interpretability trade-off. These are: COR (Section 3.1), WCOR (Section 3.2), HSWLR (Section 3.3) and TSK (Section 3.4). In the remaining of the section, a short introduction to the dataset generation methodology is given (see Mucientes & Casillas (2007) for a more detailed description).

2.1. Dataset generation methodology

The technique used for the generation of datasets has the following steps:

(1) Definition of the input and output variables, and the 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 study on the evolutionary-based methodologies has been done for two different behaviors: wall-following and moving object following. The wall-following behavior is typically used in indoor environments for exploring unknown areas, or for moving between two positions of the environment. The main objectives of a controller for this behavior are: to keep a suitable distance to the wall, to move at the highest possible velocity, and to implement smooth control actions. The controller can be configured with two parameters: the reference distance (d_{wall}), which is the desired distance between the robot and the wall, and the maximum velocity of the robot (v_{max}).

On the other hand, the moving object following behavior, tries to maintain the robot at the desired distance (d_{ref}) and angle (dev_{ref}) to the object being followed. This controller is very useful to pursue a person, or to follow other robots for the cooperation in the fulfillment of other tasks.

2.1.1. Selection of the variables

The goal of any behavior is to place the robot from its current state to the objective state. If we are able to define the characteristics of this objective state, then most of the work in the dataset generation module is done. The methodology is described using as examples the two mentioned behaviors. For the wall-following behavior the objective state is the state that has a right distance equal to d_{wall},[1] the speed is v_{max}, the robot is parallel to the wall, and the left distance is higher than the right distance (as the robot is following the right wall). With this information, we can establish that the input variables for this behavior are the right (RD) and left (DQ) distances to the wall, the velocity of the robot (LV), and the orientation of the robot to the right wall (θ_{wall}). Moreover, the output variables are the linear acceleration (LA) and the angular velocity (AV). These output variables are common to all the behaviors.

For the moving object following behavior the goal is to place the robot at the objective point: a point placed at a distance equal to d_{ref} and with an angle of dev_{ref} with respect to the moving object. Also, the speed and angle of the robot should be equal to those of the moving object. Thus, the input variables are the distance (d) and angle (dev) from the robot to the objective point, and the differences in speed (Δv) and angle ($\Delta\theta$) between the robot and the moving object.

2.2. 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 must contain all the values that are meaningful from the point of view of the learning algorithm. For example, if the robot has to follow the wall at 0.5 m, a possible universe of discourse could be $RD \in [0,3]$, but also $RD \in [0,5]$. However, control actions for distances between 3 and 5 m should be equal for all these situations. The selection of the second of the universes of discourse would generate a larger dataset, with examples that are irrelevant for learning. Although both universes of discourse are valid for the learning algorithm module, the second one slows down the learning time. Therefore, the designer should keep the universe of discourse as reduced as possible, but taking into account that the cost of choosing a universe of discourse reduced over the limit decreases the quality of the learned behavior, while an oversized universe of discourse only penalizes the learning time.

The same occurs with the precision of the variables. Precision is used to generate the examples (Section 2.6). Very low values of p_i generate a higher number of examples and, therefore, many of them are not meaningful for the learning algorithm. The rules for the selection of the universe of discourse can be applied again for the selection of the precisions of the variables, but in this case, a higher precision means a lower number of examples.

Finally, the fuzzy sets of each variable are uniformly distributed in the universe of discourse, with triangular shape, and following a Ruspini's strong fuzzy partition.

2.3. Scoring function

The key point of the dataset generation methodology is the definition of the scoring function, *SF*, for a behavior. Given a state of the robot (an example of the training set), *SF* has to evaluate the state reached by the robot after applying the control action proposed by the fuzzy controller. The evaluation of a state consists in measuring its similarity to the objective state. Thus, independently of the behavior, *SF* can be defined as:

$$SF(RB(e^l)) = \sum_i \alpha_i, \tag{1}$$

where e^l is the lth example and $SF(RB(e^l))$ 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*. Moreover, α_i measures the similarity in the value of variable i between the reached state and the objective state:

$$\alpha_i = w_i \cdot \frac{|val_i - obj_i|}{maxDiff_i}, \tag{2}$$

where val_i is the value of variable i, obj_i is the objective value of that variable and w_i is a weight that measures the importance of the contribution of variable i to *SF*. $maxDiff_i$ is the maximum of all the possible values of $|val_i - obj_i|$. In that way, each α_i takes values in $[0, w_i]$.

[1] We assume that the robot is following the right wall.

Table 1
Main characteristics of the analyzed learning methods.

Method	Rule type	Weights	Hierarchical	MF type	MF tuning	RB reduction	Flexibility	Interpretability
COR	Mamdani	No	No	Grid	No	Embedded	Weak	Excellent
WCOR	Mamdani	Yes	No	Grid	No	Embedded	Fair	Good
HSWLR	Mamdani	Yes	Yes	Grid	No	*A posteriori*	Good	fair
TSK	TS	No	No	Scatter	Yes	*A posteriori*	Excellent	Poor

The definitions of *SF* (Eq. (1)) and α_i (Eq. (2)) are independent of the behavior. For an specific behavior, the designer has to define w_i and obj_i. Values of obj_i represent the objective state. Thus, for the wall-following behavior these values are: $obj_{RD} = d_{wall}$, $obj_{LV} = v_{max}$, and $obj_{\theta_{wall}} = 0$. These values are the representation of our definition of objective state: the robot must be at the reference distance, with the maximum velocity and parallel to the wall.[2] In the same way, we can define obj_i for the moving object following behavior: $obj_d = d_{ref}$, $obj_{dev} = dev_{ref}$, $obj_{\Delta v} = 0$, and $obj_{\Delta\theta} = 0$.

2.4. Objective function

The index that measures the global quality of the encoded rule set is independent of the behavior, and can be defined as:

$$f(RB) = \frac{1}{2 \cdot NE} \sum_{l=1}^{NE} (g(e^l))^2, \tag{3}$$

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

$$g(e^l) = \begin{cases} (1 - h(e^l)) \cdot \zeta + 1, & \text{if } h(e^l) \leqslant 1, \\ \exp(1 - h(e^l)), & \text{otherwise}, \end{cases} \tag{4}$$

with ζ being a scaling factor that has been set to 1000 for all the behaviors, and

$$h(e^l) = \frac{\min(SF(e^l)) + 1}{SF(RB(e^l)) + 1}, \tag{5}$$

with $\min(SF(e^l))$ being the minimum score that an action can obtain for example e^l (using the output values as described in Section 2.6).

2.5. Robot simulation

The use of a robot simulation software facilitates the learning and test stages, as compared with the real robot. Nevertheless, trying thousands of control orders with the simulation software is very time consuming. Therefore, in order to reduce the time needed for learning, the simulation software has not been used. Instead, the movement of the robot has been modeled with a standard set of kinematic equations: the equations of a uniformly accelerated motion in two dimensions.

2.6. Construction of the training set

Controllers are learned 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. The dataset is automatically generated by: starting from the minimum value of each variable and increasing the value in a quantity equal to p_i 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. The function that determines how close a state is from the ideal state is *SF* as defined in Section 2.3: the lower the value of *SF*, the closer the state is to the ideal state.

3. Learning fuzzy models with different accuracy/interpretability trade-offs

After introducing the strategy to provide data sets for automatic learning of fuzzy controllers, this section introduces several learning methods that could be applied on these data. Indeed, one of the most important problems in the applications of fuzzy logic is the automatic derivation of fuzzy models from numerical information (input–output data pairs) representing the behavior of the real system. Numerous automatic methods—based on *ad hoc* data-driven approaches (Nozaki, Ishibuchi, & Tanaka, 1997; Wang & Mendel, 1992) or on different techniques such as neural networks (Fuller, 2000; Nauck, Klawonn, & Kruse, 1997) and evolutionary algorithms (GAs) (Cordón et al., 2001)—have been developed to perform this task.

One of the most promising research topics in fuzzy modeling relates with the quest of a good trade-off between interpretability and accuracy (Casillas, Cordón, Herrera, & Magdalena, 2003a, 2003b). To do so, different mechanisms to improve the accuracy and the interpretability are used and, moreover, they are properly gathered to regulate the desired trade-off. Although the degree of accuracy and interpretability is mainly related with the user preferences, depending on the problem being solved (Bonissone, Subbu, Eklund, & Kiehl, 2006) an adequate trade-off should be ensured in order to prevent overfitting. In fact, the degree of complexity or flexibility of the learned models is usually increased in order to improve the system accuracy. The derivation of too complex or flexible models can lead to very specific system behaviors with very good performance in training data but inconsistent performance in the real system. Therefore, a study on different kinds of learning algorithms and on the influence of the involved parameters is required in our case in order to ensure a proper behavior of the real robot.

In this paper, we have analyzed four learning methods (COR, WCOR, HSWLR and TSK) that consider a large number of accuracy and interpretability improvements and represent different degrees of complexity and flexibility. Table 1 summarizes their main characteristics. In that table, *TS* represents Takagi–Sugeno type systems (Takagi & Sugeno, 1985), *MF type* distinguishes between membership functions with *grid partition* (i.e., using linguistic variables or global semantics) or *scatter partition* (i.e., fuzzy variables without associated semantics or with local semantics), *MF tuning* refers whether the initial membership functions are tuned or not, *RB reduction* distinguishes between *embedded* (where a rule selection is performed at the same time that they are defined) or *a posteriori* (where the selection is performed once a preliminary fuzzy rule set is defined), and *Flexibility* gives an idea of the potential approximation ability provided by the obtained fuzzy model according to the linguistic scale {*excellent*, *good*, *fair*, *weak*, *poor*}. The following subsections briefly describe the considered learning methods by

[2] There is not an objective left distance.

extending them for multiple output variables (they were proposed for single output based problems). For more detailed descriptions, please refer to the corresponding papers.

3.1. COR: the COR methodology

COR methodology (Casillas, Cordón, & Herrera, 2002) is guided by example covering criteria to obtain antecedents (fuzzy input subspaces) and candidate consequents. Following the Wang and Mendel's method (1992) (WM) approach this methodology presents the following learning scheme:

Let $E = \{e_1, \ldots, e_l, \ldots, e_N\}$ be an input–output data set representing the behavior of the problem being solved—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. And let $\mathscr{A}_i$ be the set of linguistic terms of the ith input variable—with $i \in \{1, \ldots, n\}$—and $\mathscr{B}_j$ be the set of linguistic terms of the jth output variable—with $j \in \{1, \ldots, m\}$.

(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 $A_i^s \in \mathscr{A}_i$ being a label, $s \in \{1, \ldots, N_S\}$, and $N_S = \prod_{i=1}^n |\mathscr{A}_i|$ being the number of fuzzy input subspaces. In this paper, we use the following:

$$E^+(S_s) = \left\{ e_l \in E | \forall i \in \{1, \ldots, n\}, \forall A_i' \in \mathscr{A}_i, \mu_{A_i^s}(x_i^l) \geqslant \mu_{A_i'}(x_i^l) \right\}, \tag{6}$$

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 $S^+ = \{S_h | E^+(S_h) \neq \emptyset\}$.

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:

$$C(S_h) = \left\{ (B_1^{k_h}, \ldots, B_m^{k_h}) \in \mathscr{B}_1 \times \cdots \times \mathscr{B}_m | \exists e_l \in E^+(S_h) \right.$$
$$\left. \text{where } \forall j \in \{1, \ldots, m\}, \forall B_j' \in \mathscr{B}_j, \mu_{B_j^{k_h}}(y_j^l) \geqslant \mu_{B_j'}(y_j^l) \right\}. \tag{7}$$

Then, the candidate rule set for each subspace is defined as $CR(S_h) = \left\{ R_{k_h} = \left[\text{IF } X_1 \text{ is } A_1^h \text{ and} \ldots \text{and } X_n \text{ is } A_n^h \text{ THEN } Y_1 \text{ is } B_1^{k_h} \text{ and} \ldots \text{and } Y_m \text{ is } B_m^{k_h} \right] \text{ such that } \left(B_1^{k_h}, \ldots, B_m^{k_h} \right) \in C(S_h) \right\}$. To allow COR to reduce the initial number of 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) *Perform a combinatorial search among the sets $CR(S_h)$ looking for the combination of consequents with the best cooperation*. Since the search space tackled in step 3 of the algorithm is usually large, it is necessary to use approximate search techniques. In Casillas, Cordón, and Herrera (2002), different search paradigms were proposed for this purpose. In this work, we consider the Ant Colony Optimization method described in Casillas, Cordón, de Viana, and Herrera (2005) as search technique. In order to assess this combinatorial search technique, the following global index has been defined:

$$f'(RB) = f(RB) + \phi \cdot f(RB_0) \cdot \frac{\#RB}{|S^+|} \tag{8}$$

with $f(\,)$ being the function defined in Eq. (3), $\phi \in [0, 1]$ 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, and RB_0 being the initial rule base considered by the search algorithm.

3.2. WCOR: the weighted COR methodology

A good technique to improve the cooperation of a set of rules is the use of weighted fuzzy rules (Alcalá, Alcalá-Fdez, Casillas, Cordón, & Herrera, 2006; Cho & Park, 2000; Pal & Pal, 1999). This approach consists of modifying the linguistic model structure to include an importance factor for each rule (in our case, a value in $[0, 1]$). In order to do so, we will use the weighted rule structure and the inference system proposed in Pal and Pal (1999) extended for multiple output variables:

$$\begin{array}{l} \text{IF } X_1 \text{ is } \mathscr{A}_1 \text{ and } \ldots \text{ and } X_n \text{ is } \mathscr{A}_n \\ \text{THEN } Y_1 \text{ is } \mathscr{B}_1 \text{ and } \ldots \text{ and } Y_m \text{ is } \mathscr{B}_m \textit{ with } [w], \end{array} \tag{9}$$

where $X_i(Y_j)$ are the linguistic input (output) variables, $\mathscr{A}_i(\mathscr{B}_j)$ 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 by inferring with the FITA (First Infer, Then Aggregate) scheme and by computing the defuzzified output of the jth variable as the following *weighted sum*:

$$y(j) = \frac{\sum_h m_h \cdot w_h \cdot P_h(j)}{\sum_h m_h \cdot w_h}, \tag{10}$$

with m_h being the matching degree of the hth rule, w_h being the weight associated to the hth rule, and $P_h(j)$ being the characteristic value of the output fuzzy set corresponding to that rule in the jth 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.

In Alcalá, Casillas, Cordón, and Herrera (2002), the Weighted COR (WCOR) methodology was presented to include weight learning within the COR methodology (Casillas et al., 2002). In this way, a Genetic Algorithm (GA) could automatically learn the best consequent label and its associated weight for each possible antecedent combination in the problem input space. WCOR (Alcalá et al., 2002) involves an extension of the original COR methodology and consists of:

(1) *Search space construction:* Obtain the subspaces with positive examples $S_h \in S^+$ and a set of candidate consequents $C(S_h)$ associated to them using the set of candidate rules $CR(S_h)$ based on WM (Wang & Mendel, 1992) (see Section 3.1).

(2) *Selection of the most cooperative fuzzy rule set and learning of weights:*

- *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 initial subspaces/rules obtained) are defined to represent this information, where,

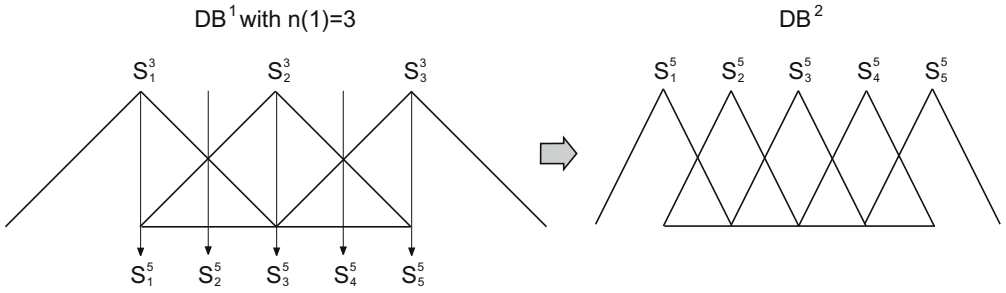


Fig. 1. Two layers of linguistic partitions.

$$c_1[h] = \emptyset \text{ or } k_h | R_{c_1[h]} \in CR(S_h), \tag{11}$$

$$c_2[h] = w_h, \quad \forall h \in \{1, \ldots, |S^+|\}. \tag{12}$$

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:

$$c = c_1 c_2. \tag{13}$$

- *Perform a search on the c vector, looking for the combination of consequents and weights with the best cooperation*. To do that, we consider an elitist generational GA with the Baker's stochastic universal sampling procedure with the same fitness function $f'(RB)$ defined in Eq. (8). The initial pool is generated at random but including an individual having all the genes with value '1' in the c_2 part and with valid values for each $c_1[h]$ ($\emptyset$ or k_h values). 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. As regards the *mutation* operator, it applies classical mutation to c_1 and takes a value at random within the interval $[0, 1]$ for the corresponding gene in c_2. See Alcalá et al. (2002) for a complete description of these operators.

3.3. HSWLR: Hierarchical Systems of Weighted Linguistic Rules

In Alcalá et al. (2003), the hybridization of both hierarchical and weighted linguistic fuzzy rules (see Eq. (9)) to derive Hierarchical Systems of Weighted Linguistic Rules (HSWLRs) is presented. A Hierarchical Knowledge Base (HKB) is composed of a set of layers, and each layer is defined by a data base DB^t containing the linguistic partitions of layer t (t-linguistic partitions), and a rule base RB^t formed by those linguistic rules whose linguistic variables take values in the former partitions (t-linguistic rules). The number of linguistic terms in the partitions of layer t will be defined in the following way (by using strong fuzzy partitions):

$$n(t) = (n(1) - 1) \cdot 2^{t-1} + 1, \tag{14}$$

with $n(1)$ being the granularity of the initial fuzzy partitions. Fig. 1 graphically depicts the way in which a linguistic partition in $DB^1 (n(1) = 3)$ becomes a linguistic partition in DB^2.

In this subsection, we describe the two-level HSWLR learning methodology proposed in Alcalá et al. (2003) to generate two-layer Weighted HKBs (WHKBs). To do so, WM (Wang & Mendel, 1992) is used based on the existence of a previously defined DB^1 and a set of input–output training data E with N being the data set size. The measure of error used in the algorithm will be called $F(E, RB)$, that is the mentioned objective function $f(RB)$, defined in Eq. (3) and applied on the training set E. The algorithm consists of the steps:

1. RB^1 *Generation by means of the WM method*: $RB^1 = WM(DB^1, E)$.
2. RB^2 *Generation*: Generate RB^2 from RB^1, DB^1 and DB^2.
   (a) *Calculate the error of* $RB^1 : F(E, RB^1)$.
   (b) *Calculate for each 1-linguistic rule:* $F(E_i, R_i^{n(1)})$, with E_i being the set of examples matching the ith rule antecedents to degree $\tau \in (0, 1]$.
   (c) *Select the 1-linguistic rules with bad performance to be expanded*[3]: *If* $F(E_i, R_i^{n(1)}) \geqslant \alpha \cdot F(E, RB^1)$ *Then* $R_i^{n(1)} \in RB^1_{bad}$ *Else* $R_i^{n(1)} \in RB^1_{good}$.
   (d) *Create* DB^2.
   (e) *Select the 2-linguistic partition terms from* DB^2 *that* δ*-intersect the ones of the bad performance 1-linguistic rules:* $I(R_i^{n(1)}),\ \forall\ R_i^{n(1)} \in RB^1_{bad}$, where $\delta \in [0, 1]$ is a cross level of "*significant intersection*".
   (f) *Extract a candidate set of L 2-linguistic rules:*

$$CLR(R_i^{n(1)}) = WM(I(R_i^{n(1)}), E_i) = \{R_{i_1}^{2 \cdot n(1)-1}, \ldots, R_{i_L}^{2 \cdot n(1)-1}\}.$$

3. *Summarization*: Obtain a joined set of candidate linguistic rules (JCLR) with the new generated *2-linguistic rules* and the good *1-linguistic rules*,

$$JCLR = RB^1_{good} \cup (\bigcup_i CLR(R_i^{n(1)})), R_i^{n(1)} \in RB^1_{bad}.$$

   More than one copy of a rule in the same layer can be produced as a consequence of the generation process (Steps 1, 2 and 3). An equivalent Weighted Hierarchical Rule Base (WHRB) without repeated rules is obtained by maintaining a single instance for each rule and by assigning a weight w_i being the number of times each rule appears. Other equivalent WHRB can be found with $w_i \in [0, 1]$ by means of a normalization process over the weights.
4. *Genetic Weight Derivation and Rule Selection Process*. To improve rule cooperation in the final WHRB, rule simplification should be performed together with a learning of the associated weights. For this task, we consider an elitist generational GA with the Baker's stochastic universal sampling and the same fitness function $f'(RB)$ defined in Eq. (8). A double *coding scheme* ($c = c_1 + c_2$) for both, rule selection and weight derivation, is used. The c_1 part is a binary-coded string representing whether a rule is selected or not (alleles '1' or '0'). The c_2 part is a real-coded string representing the weights applied to each rule (values in $[0, 1]$). The initial pool is randomly generated, except for the first individual, setting to one all the genes in c_1 and including the weights obtained from Step 3 (summarization) in c_2. The

[3] The expansion factor α may be adapted in order to have more or less expanded rules.

same genetic operators described in Section 3.2 are used in this case (Alcalá et al., 2003) (except the mutation that flips the gene value in c_1).

3.4. TSK: local evolutionary learning of Takagi–Sugeno rules

In Alcalá, Alcalá-Fdez, Casillas, Cordón, & Herrera (2007), the use of local semantics-based Mamdani fuzzy rules as local fuzzy prototypes was proposed to obtain accurate local semantics-based TS rules (Takagi & Sugeno, 1985), considering the interaction between input and output variables and taking into account the fuzzy nature of these kind of rules. To do so, a two-stage genetic FRBS was designed following the MOGUL paradigm (Cordón, del Jesus, Herrera, & Lozano, 1999), a methodology to obtain genetic FRBSs under the Iterative Rule Learning (IRL) approach.

The local identification of prototypes induces competition among rules, considering only the quality of the approximation performed by each rule using an IRL-based approach. However, the global cooperation among rules should be considered in order to increase the generalization ability of the modeled system. Following the MOGUL approach, a post-processing stage is considered for this purpose. In this way, the learning method substantially reduces the search space size by dividing the genetic learning process in two stages. A complete description of this algorithm can be found in Alcalá et al. (2007). A brief description of this technique is presented in the next.

Table 2
Characteristics of the environments for the wall-following behavior.

Name	Dim. ($m \times m$)	Length (m)	#CC	#CX	#doors
wsc8a	15×15	70	4	7	1
gfs	14×10	43	10	6	0
rooms	19×12	86	12	6	4
autolab	28×20	154	21	11	10
office	26×15	146	23	10	8
mapunet	25×10	58	15	8	5
hospital	75×45	1046	98	69	43
real environment	11×12	28	7	3	0

Fig. 2. A *Pioneer II AT* robot equipped with a laser range scanner.

3.5. Local process for identifying prototypes

To obtain a set of local semantics-based Mamdani fuzzy rules (fuzzy prototypes) we have used a method described in Cordón & Herrera (2001) that is based on local covering measures to induce competition among rules, considering the *completeness* and *consistency* properties (Cordón et al., 1999). In our case, completeness is verified requiring that each example is covered to a degree $\epsilon \in \Re^+$. On the other hand, to verify the consistency, the *positive* and *negative example* concepts (Cordón et al., 1999) are considered. Thereby, the accuracy of a simple fuzzy rule, R_h, on the set of examples, E, is measured by using a multicriteria fitness function:

$$F(R_h) = \Psi_E(R_h) \cdot G_\omega(R_h) \cdot g_n(R_h^-), \tag{15}$$

designed to take into account three different criteria (Cordón et al., 1999): *high frequency value* ($\Psi_E(R_h)$), *high average covering degree over positive examples* ($G_\omega(R_h)$) and *small negative example set* ($g_n(R_h^-)$). This process is briefly summarized in the following steps:

(1) Perform a strong fuzzy partition for each variable (uniform triangular-shaped membership functions).
(2) Generate for each example e_l the global semantics-based fuzzy rule best covering it. Then, evaluate all the global fuzzy rules and select the rule with the highest value in the fitness function ($F(R_h)$).
(3) The most promising global fuzzy rule is locally tuned to identify the local fuzzy prototype best grouping the data located in the corresponding subspace. This process is computed by means of the $(1+1)$-Evolutionary Strategy ($(1+1)$-ES) described in Cordón & Herrera (2001) considering as fitness function $F'(R_h) = F(R_h) \cdot LNIR(R_h)$ where $LNIR(R_h)$ is a penalty function to avoid excessive proximity among prototypes (Cordón et al., 1999).
(4) Finally, the obtained prototype is added to the final set of fuzzy prototypes. Data covered to a certain degree by this set are removed and not considered for future iterations. The iterative process ends up when no more uncovered training data remains.

To obtain the TS consequents, once the set of local fuzzy prototypes is obtained and considering the same antecedents, the existing partial linear input–output relation is computed using the data located in each input subspace by means of the (μ, λ)-ES presented in Cordón & Herrera (1999) to minimize the objective function $f(RB)$ (Eq. (3)).

3.6. Post-processing stage

Two different processes are considered at this stage to improve $f(RB)$: the *genetic simplification process* and the *genetic tuning process*:

(1) *Genetic simplification process*: This process, described in Cordón & Herrera (2001), is based on a standard binary-coded GA and also considers the completeness property (τ). It has the aim of selecting the subset of rules best cooperating among the rules generated in the previous stage.
(2) *Genetic tuning process*: This method is an adaptation of Cordón & Herrera (1999) (tunes TS FRBSs based on global semantics) to local semantics. It is based on a hybrid GA-ES algorithm in which each individual represents a complete knowledge base. An $(1+1)$-ES is considered as a genetic operator to locally tune a percentage δ of the best individuals in each generation. In this method, the variation interval is independently estimated for each fuzzy set.

Table 3
Environment *wsc8a*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s, weights $= (0.89, 0.10, 0.01)$.

Method labels	#Examples	ϕ	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR 4-2-5-2|9-9	5070	0.0	72	55.36 ± 1.28	44.80 ± 0.80	2.92 ± 0.17	152.26 ± 2.22	6.34
	5070	0.1	71	47.15 ± 3.61	30.62 ± 3.89	1.97 ± 0.42	223.94 ± 21.32	5.50
	5070	0.2	67	48.64 ± 2.97	37.69 ± 1.50	2.70 ± 0.36	180.74 ± 9.05	3.46
	5070	0.3	64	50.62 ± 1.41	35.41 ± 1.17	2.53 ± 0.11	197.08 ± 8.16	**3.02**
	5070	0.4	63	47.71 ± 0.98	35.85 ± 1.48	2.16 ± 0.19	186.50 ± 9.12	4.48
	5070	0.5	63	52.62 ± 1.33	39.07 ± 0.88	2.84 ± 0.21	176.46 ± 3.83	4.45
	24,624	0.0	69	54.97 ± 4.08	45.89 ± 6.06	2.25 ± 1.37	172.22 ± 6.80	5.88
	24,624	0.1	67	47.12 ± 2.47	36.20 ± 1.53	4.45 ± 0.41	190.44 ± 11.96	4.97
	24,624	0.2	64	51.22 ± 1.79	39.10 ± 1.13	2.77 ± 0.09	177.58 ± 2.77	3.19
	24,624	0.3	59	49.66 ± 3.51	32.79 ± 3.00	5.11 ± 0.95	210.66 ± 17.04	3.03
	24,624	0.4	58	53.82 ± 4.50	43.51 ± 3.83	2.14 ± 0.87	168.60 ± 5.23	5.09
	24,624	0.5	52	53.38 ± 2.25	36.78 ± 2.76	4.06 ± 0.51	188.28 ± 12.81	5.36
WCOR 4-2-5-2|9-9	5070	0.0	79	49.14 ± 2.36	43.35 ± 1.59	2.81 ± 0.05	157.86 ± 5.35	2.44
	5070	0.1	69	58.24 ± 1.49	46.47 ± 0.82	2.74 ± 0.39	150.48 ± 4.41	8.77
	5070	0.2	63	49.60 ± 2.93	42.77 ± 1.46	5.41 ± 0.50	159.16 ± 6.50	**2.08**
	5070	0.3	60	53.69 ± 5.83	44.80 ± 2.42	2.86 ± 0.23	154.52 ± 7.99	4.84
	5070	0.4	55	46.26 ± 2.71	40.75 ± 1.63	4.13 ± 0.44	167.30 ± 7.84	5.29
	5070	0.5	52	58.03 ± 2.87	46.13 ± 1.80	3.89 ± 0.40	149.88 ± 4.42	8.61
	24,624	0.0	74	60.72 ± 2.06	45.96 ± 1.43	3.28 ± 0.31	151.70 ± 4.74	11.05
	24,624	0.1	59	60.40 ± 2.18	46.71 ± 0.79	3.38 ± 0.28	147.50 ± 2.60	10.69
	24,624	0.2	53	58.75 ± 2.12	45.61 ± 1.46	3.33 ± 0.09	151.88 ± 5.28	9.31
	24,624	0.3	49	61.45 ± 4.28	46.11 ± 2.01	3.49 ± 0.20	150.20 ± 6.14	11.69
	24,624	0.4	48	59.50 ± 3.31	46.24 ± 1.91	4.04 ± 0.26	149.06 ± 5.96	9.93
	24,624	0.5	45	51.71 ± 3.59	40.66 ± 2.00	3.63 ± 0.24	166.36 ± 9.19	3.47
HSWLR 4-2-3-2|9-9	5070	0.5	91	53.94 ± 0.67	30.23 ± 0.50	2.54 ± 0.16	224.40 ± 2.10	6.52
	5070	0.6	86	53.52 ± 0.12	30.99 ± 0.42	2.67 ± 0.09	219.58 ± 2.12	6.07
	5070	0.7	82	54.33 ± 1.02	30.11 ± 0.88	2.82 ± 0.09	227.00 ± 5.22	6.89
	5070	0.8	78	54.15 ± 0.14	30.08 ± 0.32	2.89 ± 0.06	226.46 ± 2.43	6.73
	5070	0.9	77	51.58 ± 0.47	31.83 ± 0.65	4.96 ± 0.17	213.64 ± 3.26	4.24
	5070	1.0	76	51.42 ± 0.78	30.46 ± 1.01	3.36 ± 0.13	222.34 ± 6.09	**4.23**
	24,624	0.5	82	54.64 ± 0.83	30.11 ± 0.54	2.55 ± 0.18	226.32 ± 5.20	7.17
	24,624	0.6	82	51.43 ± 1.92	26.10 ± 0.77	2.40 ± 0.27	260.68 ± 6.96	4.68
	24,624	0.7	79	52.62 ± 0.91	27.66 ± 0.60	2.53 ± 0.09	246.62 ± 4.51	5.59
	24,624	0.8	75	52.51 ± 1.77	28.45 ± 0.76	2.48 ± 0.09	238.52 ± 4.19	5.41
	24,624	0.9	71	52.35 ± 0.40	30.25 ± 0.15	2.57 ± 0.07	224.84 ± 2.64	5.09
	24,624	1.0	74	51.55 ± 1.62	30.78 ± 0.68	2.67 ± 0.19	219.68 ± 5.47	4.32
TSK 4-2-5-2	5070	0.5	95	57.71 ± 1.16	45.01 ± 0.78	3.16 ± 0.16	152.68 ± 2.62	8.44
	5070	0.6	95	62.25 ± 3.31	45.86 ± 1.62	2.90 ± 0.37	153.56 ± 4.77	12.44
	5070	0.7	94	59.93 ± 0.49	45.72 ± 0.83	3.26 ± 0.26	151.34 ± 4.00	10.37
	5070	0.8	96	58.91 ± 1.19	45.56 ± 1.21	3.20 ± 0.16	152.40 ± 4.51	9.46
	5070	0.9	93	57.25 ± 1.94	45.45 ± 1.43	3.20 ± 0.18	151.44 ± 4.80	7.98
	5070	1.0	95	57.54 ± 1.02	45.39 ± 1.19	3.29 ± 0.22	152.84 ± 4.42	8.25
	24,624	0.5	93	59.03 ± 19.00	39.04 ± 6.15	3.24 ± 0.16	161.84 ± 3.01	10.22
	24,624	0.6	89	64.14 ± 3.26	44.07 ± 1.62	3.25 ± 0.12	163.16 ± 7.15	14.32
	24,624	0.7	94	38.78 ± 1.23	28.85 ± 0.91	2.92 ± 0.16	231.24 ± 6.96	13.21
	24,624	0.8	96	50.38 ± 3.00	33.79 ± 1.24	5.06 ± 0.57	204.60 ± 6.77	2.96
	24,624	0.9	93	51.30 ± 2.12	31.29 ± 0.88	10.71 ± 0.77	221.90 ± 4.65	4.04
	24,624	1.0	98	49.21 ± 2.60	38.36 ± 1.43	3.15 ± 0.37	178.10 ± 6.61	**2.88**

4. Results

The study on the different methodologies to learn behaviors in mobile robotics requires a complete set of tests. Of course, tests on the real robot should be preferred. However, we want to analyze the influence of several parameters, and this means that a high number of knowledge bases must be considered. Moreover, tests should be done in different environments. For example, we have tried 344 different combinations for the wall-following behavior. Such a number of tests can not be done on the real robot. Thus, we have run the different knowledge bases using two simulators. These simulated environments represent situations completely different from the learning stage, as the training examples were obtained using the dataset generation methodology and, also, the simulation during the learning process was done with a set of equations (Section 2.5). To complete the study, a representative knowledge base of each of the data-driven learning algorithms (COR, WCOR, HSWLR and TSK) was selected and tested on the real robot.

In this section, we have analyzed the two behaviors described in Section 2: wall-following and moving object following. For the wall-following behavior, an exhaustive study is shown, comprising the performance of the data-driven learning algorithms, and the influence of several parameters (always in simulated environments). Finally we have tested the best controllers on the real robot. For the moving object following behavior, the study has analyzed the performance of the learning algorithms.

Table 4
Environment *wsc8a*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s, weights = (0.89, 0.10, 0.01).

Method labels	#Examples	ϕ	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR 4-3-3-3|9-9	5070	0.0	95	52.50 ± 0.73	27.81 ± 0.65	10.57 ± 0.42	248.94 ± 5.78	5.47
	5070	0.1	91	58.26 ± 3.56	39.75 ± 0.84	4.34 ± 0.37	176.06 ± 4.76	9.46
	5070	0.2	88	63.16 ± 3.73	35.38 ± 0.61	2.87 ± 0.14	206.56 ± 6.02	14.31
	5070	0.3	88	53.20 ± 1.33	39.86 ± 0.71	5.67 ± 0.83	174.98 ± 1.79	4.89
	5070	0.4	82	44.27 ± 3.03	30.37 ± 2.66	7.97 ± 1.43	235.08 ± 20.17	8.12
	5070	0.5	81	55.65 ± 1.81	28.72 ± 0.50	12.61 ± 1.33	237.96 ± 2.64	8.21
	24,624	0.0	93	60.45 ± 2.49	35.26 ± 2.16	2.80 ± 0.13	196.88 ± 8.34	11.88
	24,624	0.1	93	63.53 ± 1.00	38.40 ± 2.52	3.69 ± 0.81	186.70 ± 11.19	14.34
	24,624	0.2	90	50.42 ± 2.83	32.26 ± 0.57	3.04 ± 0.27	212.68 ± 3.14	**3.15**
	24,624	0.3	82	66.32 ± 3.00	36.50 ± 2.29	4.35 ± 0.26	200.40 ± 9.00	17.04
	24,624	0.4	84	62.13 ± 0.87	37.85 ± 1.35	3.65 ± 0.22	185.78 ± 7.08	13.13
	24,624	0.5	73	73.56 ± 7.51	31.09 ± 3.32	4.11 ± 0.19	224.62 ± 21.24	24.10
WCOR 4-3-3-3|9-9	5070	0.0	106	56.42 ± 2.58	41.05 ± 1.70	2.95 ± 0.39	166.68 ± 8.00	7.67
	5070	0.1	95	54.29 ± 2.33	38.64 ± 1.76	3.12 ± 0.22	176.60 ± 7.09	6.00
	5070	0.2	86	52.45 ± 1.35	26.98 ± 1.33	14.32 ± 0.96	257.40 ± 9.12	5.51
	5070	0.3	81	52.79 ± 1.36	36.98 ± 1.85	7.37 ± 0.62	187.90 ± 9.78	4.81
	5070	0.4	74	50.34 ± 2.44	26.20 ± 1.01	15.19 ± 0.74	261.90 ± 11.30	3.69
	5070	0.5	71	53.66 ± 2.87	39.13 ± 2.08	6.65 ± 0.22	175.54 ± 10.75	5.38
	24,624	0.0	107	47.56 ± 2.21	37.39 ± 1.58	2.97 ± 0.12	182.82 ± 6.41	4.46
	24,624	0.1	96	51.32 ± 1.95	38.90 ± 1.49	3.46 ± 0.34	175.38 ± 5.34	**3.30**
	24,624	0.2	87	54.75 ± 0.94	40.06 ± 0.61	3.56 ± 0.06	170.88 ± 2.24	6.27
	24,624	0.3	74	52.93 ± 0.64	37.76 ± 0.53	9.44 ± 0.60	181.56 ± 3.40	4.86
	24,624	0.4	69	48.04 ± 2.79	27.27 ± 1.53	17.22 ± 0.60	246.40 ± 12.37	5.04
	24,624	0.5	66	53.50 ± 0.69	36.73 ± 0.97	9.80 ± 0.12	186.88 ± 4.02	5.48
TSK 4-3-3-3	5070	0.5	120	51.43 ± 1.36	37.54 ± 1.53	5.20 ± 0.50	182.54 ± 8.35	3.53
	5070	0.6	118	55.35 ± 1.38	39.67 ± 0.78	4.92 ± 0.39	174.22 ± 3.65	6.85
	5070	0.7	119	52.47 ± 0.42	34.63 ± 0.67	7.69 ± 0.19	197.08 ± 3.89	4.76
	5070	0.8	121	52.34 ± 0.48	39.04 ± 1.08	4.83 ± 0.23	175.90 ± 4.64	4.20
	5070	0.9	119	51.03 ± 0.74	35.32 ± 1.20	8.18 ± 0.88	194.72 ± 5.44	3.40
	5070	1.0	119	54.94 ± 0.95	36.08 ± 0.88	5.94 ± 0.37	190.34 ± 3.63	6.84
	24,624	0.5	150	53.40 ± 3.38	27.95 ± 0.99	2.56 ± 0.16	244.26 ± 8.97	6.27
	24,624	0.6	150	48.20 ± 1.63	36.56 ± 0.90	6.68 ± 0.58	186.08 ± 4.87	3.96
	24,624	0.7	153	51.29 ± 2.16	31.53 ± 2.03	8.15 ± 1.19	218.70 ± 16.12	4.01
	24,624	0.8	150	50.30 ± 1.85	37.80 ± 1.23	6.02 ± 0.34	180.56 ± 5.57	**2.49**
	24,624	0.9	150	48.89 ± 0.94	36.94 ± 0.61	6.03 ± 0.41	184.28 ± 2.63	3.31
	24,624	1.0	156	45.42 ± 1.50	29.55 ± 1.08	2.99 ± 0.06	229.34 ± 7.77	7.17

The following values have been considered for the parameters of each data-driven learning algorithm[4]:

- COR: 50 iterations, 30 ants, $\rho = 0.8, \alpha = 2, \beta = 2$, 0.3 as probability of mutation, 4 as mutation rate, 10 iterations of the local search, 30 as neighbor size in the local search, and 5 as number of iterations before restart.
- WCOR: 61 individuals, 1000 generations, 0.6 as crossover probability, 0.1 as mutation probability, and 0.35 for the a factor in the max-min-arithmetical crossover.
- HSWLR:
  - *Hierarchical generation*: 0.01 as δ cross level of significant intersection, 0.5 as τ used to calculate E_i, and 1.3 as α used to decide the expansion of rule.
  - *Genetic weight derivation and rule selection process*: 61 individuals, 2000 generations, 0.6 as crossover probability, 0.1 as mutation probability, and 0.35 for the factor a in the max-min-arithmetical crossover.
- TSK:
  - *Local process for identifying prototypes*: $\epsilon = 0.01, \omega = 0.01$ and $k = 0.1$ in the fitness function; $c = 0.9$ and five iterations for the $(1 + 1)$-ES; $\mu = 15, \lambda = 100, \gamma = 0.2 \cdot \mu = 3, \theta = 0.7$, $q = 5, \vec{r} = (r_{\vec{x}}, r_{\vec{\sigma}}, r_{\vec{\alpha}}) = (2, 0, 0), \vec{\zeta} = (\zeta_{\vec{x}}, \zeta_{\vec{\sigma}}, \zeta_{\vec{\alpha}}) = (\mu, \mu, 1), (n_\sigma, n_\alpha) = (0, 0)$ and 50 iterations for the (μ, λ)-ES.
  - *Genetic simplification process*: $\tau = 0.5$, 61 individuals, 1000 generations, 0.6 as crossover probability, and 0.1 as mutation probability.
  - *Genetic tuning process*: 61 individuals, 1000 generations, 0.6 as crossover probability, 0.1 as mutation probability, 0.35 for the factor a in the max-min-arithmetical crossover, $b = 5, d = 0.001$, 40 $(1 + 1)$-ES iterations, $\alpha = 0$ and $c = 0.9$ (the updating amount of the Rechenberg's 1/5-success rule in the $(1 + 1)$-ES (Bäck, 1996)).

4.1. Wall-following behavior

A total number of 344 tests have been done for this behavior. Such a high number of tests makes possible the study on the different learning algorithms and the analysis of the parameters. This number also includes the final tests on the real robot. Table 2 shows some of the characteristics of the environments[5] that have been used to analyze this behavior: the dimensions of the environment, the length of the path, the number of concave (#CC) and convex (#CX) corners, and the number of times that the robot has to cross a door (#doors). A concave corner is a corner in which the robot finds a wall in front of it. On the other hand, a convex corner is characterized by the existence of a gap in the wall (like an open

[4] With these values we have tried to facilitate the comparisons, selecting standard common parameters that work well in most cases instead of looking for very specific values for each approach.

[5] Environments *rooms*, *autolab*, *office* and *hospital* are part of the Player/Stage Project distribution (Gerkey, Vaughan, & Howard, 2003), while *mapunet* has been obtained from Howard & Roy (2003).

Table 5
Environment *gfs*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s, weights = (0.89, 0.10, 0.01).

Method labels	#Examples	ϕ	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR 4-2-5-2|9-9	5070	0.0	72	52.93 ± 1.26	32.63 ± 2.13	4.32 ± 0.39	134.44 ± 12.29	5.37
	5070	0.1	71	46.77 ± 1.54	23.39 ± 1.17	2.82 ± 0.24	195.58 ± 11.69	6.57
	5070	0.2	67	51.93 ± 0.87	30.13 ± 0.49	3.87 ± 0.18	146.40 ± 4.77	4.72
	5070	0.3	64	52.22 ± 1.41	28.01 ± 1.18	3.54 ± 0.22	158.82 ± 6.02	5.20
	5070	0.4	63	49.96 ± 2.48	31.12 ± 1.37	3.54 ± 0.33	140.98 ± 8.70	**2.92**
	5070	0.5	63	54.35 ± 1.62	29.23 ± 1.18	3.70 ± 0.25	152.40 ± 11.45	6.99
	24,624	0.0	69	52.14 ± 0.50	28.97 ± 0.83	3.57 ± 0.17	150.72 ± 4.51	5.03
	24,624	0.1	67	50.06 ± 2.38	29.96 ± 1.52	4.76 ± 0.40	147.40 ± 8.46	3.06
	24,624	0.2	64	49.76 ± 1.11	28.56 ± 1.07	3.74 ± 0.32	150.14 ± 11.64	3.36
	24,624	0.3	59	44.56 ± 1.10	21.83 ± 0.65	4.71 ± 0.20	199.96 ± 7.19	8.71
	24,624	0.4	58	51.71 ± 1.35	32.57 ± 0.43	4.21 ± 0.13	137.18 ± 2.38	4.28
	24,624	0.5	52	51.03 ± 5.75	25.05 ± 1.25	4.52 ± 0.37	178.04 ± 10.54	4.42
WCOR 4-2-5-2|9-9	5070	0.0	79	55.37 ± 4.98	35.02 ± 1.20	4.62 ± 0.40	128.40 ± 4.04	7.33
	5070	0.1	69	50.99 ± 1.09	34.32 ± 0.56	4.31 ± 0.18	130.80 ± 11.04	3.46
	5070	0.2	63	49.63 ± 1.74	34.05 ± 0.95	6.22 ± 0.38	127.84 ± 3.94	2.93
	5070	0.3	60	50.88 ± 1.64	35.21 ± 0.70	4.41 ± 0.20	125.66 ± 4.05	3.27
	5070	0.4	55	50.15 ± 0.65	34.11 ± 0.46	5.33 ± 0.39	126.24 ± 1.89	**2.72**
	5070	0.5	52	49.86 ± 2.85	33.83 ± 0.82	4.69 ± 0.12	130.36 ± 4.91	2.74
	24,624	0.0	74	54.50 ± 1.19	35.22 ± 0.54	4.77 ± 0.35	121.20 ± 1.94	6.53
	24,624	0.1	59	54.16 ± 1.86	34.93 ± 0.54	5.17 ± 0.34	123.42 ± 2.37	6.25
	24,624	0.2	53	53.48 ± 1.37	34.86 ± 0.85	4.84 ± 0.16	123.88 ± 3.40	5.65
	24,624	0.3	49	53.83 ± 2.56	34.30 ± 0.56	4.68 ± 0.22	128.66 ± 8.21	6.02
	24,624	0.4	48	54.34 ± 2.96	32.94 ± 1.61	4.98 ± 0.20	133.98 ± 10.40	6.61
	24,624	0.5	45	53.03 ± 1.09	31.43 ± 0.65	4.93 ± 0.18	136.44 ± 4.64	5.58
HSWLR 4-2-3-2|9-9	5070	0.5	91	51.24 ± 0.61	17.24 ± 0.31	2.48 ± 0.09	247.28 ± 5.12	5.39
	5070	0.6	86	51.53 ± 0.36	16.84 ± 0.47	2.41 ± 0.12	251.98 ± 7.01	5.69
	5070	0.7	82	50.72 ± 0.78	15.80 ± 0.18	2.41 ± 0.10	270.40 ± 4.89	5.07
	5070	0.8	78	50.91 ± 0.62	16.09 ± 0.19	2.44 ± 0.03	265.80 ± 3.71	5.21
	5070	0.9	77	50.40 ± 0.82	18.33 ± 0.40	3.02 ± 0.08	232.86 ± 2.80	**4.53**
	5070	1.0	76	51.49 ± 0.58	15.42 ± 0.38	2.56 ± 0.04	274.96 ± 7.11	5.80
	24,624	0.5	82	52.72 ± 0.61	15.56 ± 0.53	2.58 ± 0.16	271.76 ± 8.13	6.89
	24,624	0.6	82	48.90 ± 0.92	13.12 ± 0.31	2.09 ± 0.08	322.78 ± 6.58	5.68
	24,624	0.7	79	50.77 ± 0.63	15.03 ± 0.32	2.47 ± 0.14	288.26 ± 6.49	5.19
	24,624	0.8	75	51.09 ± 0.56	15.08 ± 0.36	2.34 ± 0.09	281.14 ± 5.13	5.47
	24,624	0.9	71	52.85 ± 1.57	16.53 ± 0.86	2.61 ± 0.13	261.40 ± 13.18	6.91
	24,624	1.0	74	52.28 ± 0.30	16.00 ± 0.39	2.51 ± 0.15	264.48 ± 2.73	6.45
TSK 4-2-5-2	5070	0.5	95	53.72 ± 0.69	32.80 ± 1.94	4.53 ± 0.40	131.34 ± 7.89	6.07
	5070	0.6	95	53.43 ± 1.49	32.05 ± 0.78	4.69 ± 0.22	134.80 ± 2.53	5.88
	5070	0.7	94	53.11 ± 1.33	33.27 ± 0.66	5.48 ± 0.13	129.26 ± 4.51	5.47
	5070	0.8	96	51.56 ± 1.02	32.41 ± 1.14	5.20 ± 0.32	132.40 ± 5.16	4.16
	5070	0.9	93	52.45 ± 1.11	33.27 ± 0.62	4.72 ± 0.16	128.88 ± 3.54	4.88
	5070	1.0	95	51.81 ± 1.34	32.75 ± 0.49	4.76 ± 0.22	131.12 ± 2.93	4.35
	24,624	0.5	93	47.67 ± 1.39	30.89 ± 0.50	3.90 ± 0.20	140.34 ± 2.76	5.01
	24,624	0.6	89	59.33 ± 2.06	30.42 ± 3.12	4.65 ± 0.51	138.87 ± 3.73	11.36
	24,624	0.7	94	51.97 ± 0.76	29.13 ± 1.41	3.68 ± 0.28	147.54 ± 8.80	4.86
	24,624	0.8	96	49.92 ± 3.25	27.64 ± 1.61	4.16 ± 0.21	153.54 ± 10.83	**3.31**
	24,624	0.9	93	51.31 ± 1.24	27.53 ± 1.38	5.38 ± 0.21	156.20 ± 6.88	4.43
	24,624	1.0	98	54.70 ± 1.80	31.43 ± 1.19	4.11 ± 0.34	141.34 ± 13.43	7.09

door) and is the most complex situation, as at the beginning of this type corner the sensors of the robot cannot detect the wall to follow. Finally, the action of crossing a door represents a high difficulty as the robot has to negotiate a convex corner with a very close wall in front of it.

For this behavior we have used the Player/Stage (Gerkey et al., 2003) robot software for the tests on the simulated environments and for the connection with the real robot. We have tested the controllers on a *Pioneer II AT* robot equipped with laser range scanners (Fig. 2). This robot has a skid-steer configuration: it is a four-wheel drive vehicle with the left-side drive wheels independent of the right-side drive wheels. With this characteristic each side is independent of the other, and the turn will be determined by the wheel speed and its direction of rotation. Thus, the robot can turn in its own track. For the simulations, the robot had two lasers (front and rear) covering the whole surrounding, while on the real robot only a front laser was used.

The first study that was performed comprised the following algorithms and parameters:

- Data-driven learning algorithms: COR, WCOR, HSWLR and TSK.
- Granularities of the membership functions: 4-2-5-2|9-9 ($RD-DQ-\theta_{wall}-LV|LA-AV$) and 4-3-3-3|9-9.
- Size of the training set: 5070 (p_i = 0.25, 0.5, 7.5, 0.2, 0.125, 0.05) and 24,624 (p_i = 0.2, 0.25, 5.0, 0.125, 0.125, 0.05).
- Six different values of ϕ (Eq. (8)): from 0.0 to 0.5 (COR and WCOR) and from 0.5 to 1.0 (HSWLR and TSK).
- Three environments: *wsc8a*, *gfs* and *rooms*.

Tables 3–8 contain the results of applying the proposed learning framework to the mentioned algorithms and parameters. Each row of the tables shows the results of a learned controller. In order to evaluate the quality of the controllers we have measured four different indicators: the right distance (Dist.), the linear velocity

Table 6
Environment *gfs*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s, weights = (0.89, 0.10, 0.01).

Method labels	#Examples	ϕ	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR 4-3-3-3|9-9	5070	0.0	95	49.95 ± 1.56	22.00 ± 1.20	7.96 ± 0.42	198.00 ± 13.10	3.85
	5070	0.1	91	51.52 ± 3.12	26.53 ± 1.03	5.50 ± 0.26	163.56 ± 7.91	4.72
	5070	0.2	88	57.93 ± 3.31	23.99 ± 1.48	3.74 ± 0.31	184.32 ± 9.56	10.74
	5070	0.3	88	54.62 ± 5.98	29.85 ± 2.01	5.33 ± 0.42	149.90 ± 13.07	7.17
	5070	0.4	82	50.26 ± 1.19	26.09 ± 1.19	6.53 ± 0.24	170.18 ± 5.13	**3.63**
	5070	0.5	81	50.38 ± 1.70	22.55 ± 0.74	10.72 ± 0.35	193.90 ± 7.34	4.09
	24,624	0.0	93	56.99 ± 1.65	25.44 ± 0.62	3.29 ± 0.13	168.72 ± 4.41	9.75
	24,624	0.1	93	51.97 ± 2.88	22.21 ± 1.57	5.92 ± 0.61	182.54 ± 6.65	5.55
	24,624	0.2	90	54.58 ± 0.59	24.38 ± 0.42	3.47 ± 0.11	173.36 ± 5.27	7.68
	24,624	0.3	82	47.12 ± 1.54	27.09 ± 0.75	3.67 ± 0.51	164.50 ± 6.35	5.88
	24,624	0.4	84	56.74 ± 0.85	20.85 ± 0.92	3.73 ± 0.30	199.00 ± 8.57	9.98
	24,624	0.5	73	53.47 ± 4.18	21.95 ± 1.55	3.13 ± 0.22	194.50 ± 12.17	6.93
WCOR 4-3-3-3|9-9	5070	0.0	106	50.65 ± 0.58	29.27 ± 0.47	4.07 ± 0.07	147.88 ± 1.52	3.66
	5070	0.1	95	50.68 ± 0.99	28.04 ± 0.78	4.16 ± 0.23	154.14 ± 4.92	3.81
	5070	0.2	86	51.71 ± 1.52	24.38 ± 0.45	5.84 ± 0.58	174.90 ± 4.52	5.10
	5070	0.3	81	51.53 ± 1.79	28.91 ± 1.10	6.12 ± 0.34	152.86 ± 7.57	4.49
	5070	0.4	74	54.02 ± 1.98	23.12 ± 1.32	8.31 ± 0.24	193.50 ± 21.44	7.31
	5070	0.5	71	52.02 ± 0.91	29.35 ± 0.33	7.31 ± 0.33	149.94 ± 2.32	4.88
	24,624	0.0	107	48.73 ± 0.53	29.80 ± 0.39	4.29 ± 0.08	146.72 ± 1.99	4.16
	24,624	0.1	96	48.94 ± 0.72	29.37 ± 0.33	4.26 ± 0.17	147.96 ± 2.46	4.02
	24,624	0.2	87	50.62 ± 0.79	28.07 ± 0.45	4.05 ± 0.20	154.52 ± 2.46	3.75
	24,624	0.3	74	50.36 ± 0.62	29.15 ± 0.62	6.94 ± 0.71	149.72 ± 3.15	**3.41**
	24,624	0.4	69	48.42 ± 1.18	25.61 ± 0.27	9.32 ± 0.60	170.36 ± 3.25	4.86
	24,624	0.5	66	51.40 ± 0.79	28.37 ± 0.97	6.95 ± 0.18	153.10 ± 5.34	4.42
TSK 4-3-3-3	5070	0.5	120	50.39 ± 0.68	28.77 ± 0.70	4.81 ± 0.17	149.06 ± 3.15	3.47
	5070	0.6	118	52.09 ± 1.77	29.81 ± 1.13	5.91 ± 0.30	144.84 ± 5.99	4.90
	5070	0.7	119	51.27 ± 1.06	29.14 ± 0.79	5.68 ± 0.31	150.38 ± 6.18	4.23
	5070	0.8	121	51.62 ± 1.14	30.37 ± 1.09	4.79 ± 0.18	142.90 ± 5.88	4.42
	5070	0.9	119	50.22 ± 1.20	29.83 ± 0.56	5.83 ± 0.32	145.22 ± 5.02	**3.22**
	5070	1.0	119	53.67 ± 1.24	29.47 ± 0.88	5.52 ± 0.36	146.26 ± 5.46	6.36
	24,624	0.5	150	52.86 ± 1.08	28.40 ± 1.18	4.08 ± 0.24	152.38 ± 7.76	5.73
	24,624	0.6	150	49.25 ± 1.44	28.44 ± 0.36	6.10 ± 0.45	151.86 ± 3.95	3.83
	24,624	0.7	153	49.28 ± 0.88	27.68 ± 1.04	7.42 ± 0.28	155.42 ± 7.32	3.88
	24,624	0.8	150	49.85 ± 0.95	28.46 ± 0.49	5.91 ± 0.16	151.62 ± 2.05	3.29
	24,624	0.9	150	49.02 ± 1.06	27.99 ± 0.40	5.87 ± 0.18	154.82 ± 1.80	4.08
	24,624	1.0	156	48.89 ± 1.33	24.89 ± 0.91	3.49 ± 0.08	170.78 ± 5.74	4.51

(Vel.), the change in the linear velocity between two consecutive cycles (Vel.ch.)—which reflects the smoothness in the control—and the time. The average values of the indicators are calculated for each lap that the robot performs in the environment. Results presented in the tables are the average and standard deviation values over five laps of the average values of the indicators over one lap. Moreover, we have found worthy to evaluate the error (inverse of accuracy) of a controller with a numerical value. Of course, the error could be measured in many different ways. In particular, the analysis of the error of a behavior in mobile robotics is somehow subjective. Taking this into account, we have decided to measure the error for this behavior as:

$$error = 0.9 \cdot |Dist - d_{wall}| + 0.1 \cdot |Vel - v_{max}| \quad (16)$$

Thus, the lower its value, the better the accuracy of the controller. Tables 3–8 show in bold face the lower error (best accuracy) for each data-driven algorithm in the table. The objective of this first study is to analyze the influence of the number of labels, number of examples and ϕ values in the error of the controllers.

4.1.1. Number of labels

In order to study its influence, we have analyzed the error of each row in a table with its counterpart in another table. Thus, for example, we have compared the error of the first row of Tables 3 and 4, as the only difference between them is the number of labels. Finally, for each learning algorithm, we have counted the times that the error of one of the labels sets outperforms the other. Due to the characteristics of the HSWLR algorithm, only one set of labels has been used and, therefore, this algorithm has been excluded from this analysis. For the COR algorithm the set of labels 4-2-5-2|9-9 clearly outperforms (75% of the times) the set 4-3-3-3|9-9. On the contrary, the best labels set for WCOR (67% of the times) and TSK (81% of the times) is 4-3-3-3|9-9. Thus, the number of labels really influences the performance of the learned behavior: the higher the number of labels, the higher the error of the controller. Nevertheless, a higher number of labels also reduces the interpretability of the knowledge base.

These conclusions must be qualified for our labels sets selection, as COR obtains better results with a lower number of labels. The reason is that set 4-3-3-3|9-9 has a higher or equal number of labels for all the variables but θ_{wall} (orientation of the robot to the wall). The performance of this behavior requires a right distance close to d_{wall} (Eq. (16)), and this heavily depends on a precise classification of θ_{wall}. However, WCOR and TSK are algorithms that can obtain higher accuracies than COR, and can compensate the dependency in the number of labels of θ_{wall} with a higher global number of labels.

4.1.2. Number of examples

The analysis of the influence of the number of examples in the training set has followed the same structure as in the previous section. We have compared the accuracies of knowledge bases that have been learned with the same parameters but the number of examples. What we could expect is that if there are many exam-

Table 7
Environment *rooms*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s, weights = (0.89, 0.10, 0.01).

Method labels	#Examples	ϕ	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR 4-2-5-2|9-9	5070	0.0	72	50.66 ± 0.57	45.51 ± 0.25	2.61 ± 0.19	186.08 ± 1.16	2.04
	5070	0.1	71	47.99 ± 1.78	36.77 ± 2.10	1.96 ± 0.24	229.94 ± 11.07	4.13
	5070	0.2	67	47.97 ± 0.99	41.26 ± 0.19	2.13 ± 0.23	202.98 ± 1.44	3.70
	5070	0.3	64	51.22 ± 1.44	38.98 ± 1.53	1.95 ± 0.17	214.10 ± 6.41	3.20
	5070	0.4	63	50.12 ± 1.65	42.45 ± 1.42	2.12 ± 0.26	200.46 ± 7.49	**1.86**
	5070	0.5	63	52.07 ± 2.27	39.68 ± 2.14	2.17 ± 0.21	211.40 ± 13.92	3.90
	24,624	0.0	69	51.37 ± 1.95	40.63 ± 1.52	2.36 ± 0.30	197.10 ± 9.51	3.17
	24,624	0.1	67	46.34 ± 2.77	38.04 ± 1.60	5.39 ± 0.73	221.78 ± 9.57	5.49
	24,624	0.2	64	49.24 ± 1.26	40.69 ± 0.65	2.53 ± 0.11	206.78 ± 3.81	2.62
	24,624	0.3	59	48.73 ± 1.22	33.67 ± 1.24	6.65 ± 0.42	250.80 ± 12.26	3.78
	24,624	0.4	58	48.79 ± 1.66	44.14 ± 1.01	2.33 ± 0.18	192.44 ± 3.12	2.68
	24,624	0.5	52	44.41 ± 2.03	30.19 ± 2.27	4.98 ± 0.38	282.74 ± 19.52	8.01
WCOR 4-2-5-2|9-9	5070	0.0	79	49.33 ± 1.51	45.49 ± 0.53	2.50 ± 0.14	184.20 ± 1.62	**2.05**
	5070	0.1	69	54.08 ± 1.40	46.23 ± 0.31	2.44 ± 0.17	182.60 ± 1.77	5.05
	5070	0.2	63	46.79 ± 0.58	42.04 ± 0.76	8.15 ± 0.64	201.26 ± 3.30	4.69
	5070	0.3	60	51.54 ± 1.43	45.58 ± 0.49	2.54 ± 0.18	184.90 ± 1.26	2.83
	5070	0.4	55	47.59 ± 1.64	43.04 ± 1.32	6.22 ± 0.12	196.80 ± 5.31	3.87
	5070	0.5	52	56.60 ± 1.35	46.79 ± 0.35	3.25 ± 0.23	181.60 ± 1.74	7.26
	24,624	0.0	74	56.29 ± 2.06	46.27 ± 0.70	2.70 ± 0.16	181.36 ± 2.33	7.03
	24,624	0.1	59	54.97 ± 1.46	46.18 ± 0.71	2.84 ± 0.24	182.10 ± 1.69	5.86
	24,624	0.2	53	57.70 ± 0.81	46.86 ± 0.40	3.04 ± 0.04	180.02 ± 1.36	8.24
	24,624	0.3	49	59.32 ± 0.47	46.37 ± 0.40	2.80 ± 0.22	182.76 ± 1.82	9.75
	24,624	0.4	48	55.16 ± 1.55	45.39 ± 0.45	3.07 ± 0.21	184.56 ± 2.29	6.11
	24,624	0.5	45	50.13 ± 0.57	40.26 ± 0.33	2.88 ± 0.05	204.90 ± 0.82	2.09
HSWLR 4-2-3-2|9-9	5070	0.5	91	51.87 ± 0.30	28.35 ± 0.47	1.83 ± 0.18	292.16 ± 3.69	4.85
	5070	0.6	86	51.36 ± 1.31	29.51 ± 0.47	1.95 ± 0.10	281.96 ± 3.38	4.27
	5070	0.7	82	51.14 ± 0.87	27.19 ± 0.23	1.81 ± 0.07	305.54 ± 1.89	4.31
	5070	0.8	78	52.57 ± 0.73	27.33 ± 0.63	1.93 ± 0.08	304.50 ± 4.84	5.58
	5070	0.9	77	48.78 ± 0.71	30.50 ± 0.13	4.52 ± 0.16	274.84 ± 1.19	4.05
	5070	1.0	76	50.09 ± 0.88	28.71 ± 0.29	3.04 ± 0.20	290.70 ± 3.66	**3.21**
	24,624	0.5	82	50.40 ± 1.53	27.25 ± 0.78	1.71 ± 0.16	303.38 ± 8.93	3.64
	24,624	0.6	82	48.34 ± 1.52	25.02 ± 0.45	1.41 ± 0.18	331.12 ± 6.17	4.99
	24,624	0.7	79	49.33 ± 1.67	25.33 ± 0.47	1.64 ± 0.12	328.70 ± 5.72	4.07
	24,624	0.8	75	48.18 ± 1.25	27.43 ± 0.48	1.51 ± 0.11	301.50 ± 4.69	4.90
	24,624	0.9	71	49.13 ± 0.95	29.09 ± 0.47	1.75 ± 0.11	284.78 ± 4.45	3.87
	24,624	1.0	74	50.31 ± 1.46	29.78 ± 0.54	1.71 ± 0.16	277.56 ± 4.04	3.30
TSK 4-2-5-2	5070	0.5	95	52.69 ± 2.22	44.81 ± 1.14	2.80 ± 0.13	187.36 ± 3.15	3.94
	5070	0.6	95	58.01 ± 3.26	45.21 ± 1.20	2.77 ± 0.18	185.96 ± 2.92	8.69
	5070	0.7	94	52.84 ± 2.02	44.17 ± 0.82	3.38 ± 0.20	191.00 ± 3.08	4.14
	5070	0.8	96	52.25 ± 1.13	43.74 ± 0.42	3.22 ± 0.13	192.20 ± 1.47	3.65
	5070	0.9	93	53.43 ± 0.95	44.33 ± 0.79	3.10 ± 0.06	189.70 ± 2.66	4.65
	5070	1.0	95	50.99 ± 0.98	44.10 ± 0.77	3.24 ± 0.07	190.44 ± 2.65	**2.48**
	24,624	0.5	93	43.69 ± 2.48	40.21 ± 1.93	2.28 ± 0.24	209.98 ± 8.64	7.66
	24,624	0.6	89	65.82 ± 1.71	42.92 ± 3.57	2.69 ± 0.17	194.18 ± 5.53	15.95
	24,624	0.7	94	40.88 ± 1.05	32.82 ± 0.66	2.71 ± 0.05	254.26 ± 4.61	10.93
	24,624	0.8	96	49.40 ± 1.26	32.24 ± 0.43	5.86 ± 0.11	263.02 ± 4.03	3.32
	24,624	0.9	93	48.24 ± 2.46	30.61 ± 1.01	9.80 ± 0.56	280.56 ± 6.67	4.52
	24,624	1.0	98	45.37 ± 1.46	37.77 ± 0.20	2.60 ± 0.12	221.10 ± 2.25	6.39

ples in the training set that are useless for learning (there are many examples that represent similar situations), learning will go worse. This happens as areas of the input space that have a lower concentration of examples will contribute less to the fitness function (Eq. (3)), while areas with a higher number of examples will be considered more important. In summary, the training set will be unbalanced, which tends to go in detriment of the learning process.

On the one hand, results show a slightly better error of the reduced training set (5070 examples) for WCOR (67% of the best tests correspond to this dataset) and COR (61%), while for the other methodologies results are similar for both training sets: HSWLR (56%) and TSK (47%). In general, we can conclude that the performance of the controllers is quite independent on the size of the training set, although a lower number of examples seems to be slightly better for accuracy. On the other hand, the number of rules of the knowledge bases is, in most of the comparisons, slightly higher for the smaller dataset. So, we can conclude that more examples generate more interpretable knowledge bases.

4.1.3. ϕ value

ϕ is a parameter that reflects the importance that the fitness function gives to the reduction in the number of rules (Eq. (8)). Therefore, high values of ϕ generate knowledge bases with a lower number of rules, increasing the interpretability but also decreasing accuracy[6]. Results show that there is not a ϕ value that systematically outperforms the others. Therefore, we can conclude that if ϕ takes values in a reasonable range, accuracies are not affected in any of the learning algorithms.

[6] For the TSK algorithm, there is not a reduction in the number of rules with ϕ because the genetic simplification process also considers the completeness property.

Table 8
Environment *rooms*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s, weights = (0.89, 0.10, 0.01).

Method labels	#Examples	ϕ	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR 4-3-3-3|9-9	5070	0.0	95	49.99 ± 1.11	26.68 ± 1.13	11.38 ± 0.26	317.14 ± 10.43	3.34
	5070	0.1	91	51.87 ± 0.94	38.11 ± 0.92	5.71 ± 0.30	225.88 ± 3.74	3.87
	5070	0.2	88	60.75 ± 9.21	34.16 ± 2.73	2.59 ± 0.09	253.64 ± 18.55	12.26
	5070	0.3	88	46.80 ± 0.59	37.82 ± 0.41	6.76 ± 0.31	227.16 ± 1.03	5.10
	5070	0.4	82	43.81 ± 0.65	32.56 ± 0.70	6.36 ± 0.54	258.68 ± 4.78	8.32
	5070	0.5	81	51.57 ± 0.99	25.57 ± 0.86	16.72 ± 0.28	337.94 ± 9.21	4.86
	24,624	0.0	93	54.56 ± 1.32	33.71 ± 1.51	2.13 ± 0.16	250.30 ± 12.68	6.73
	24,624	0.1	93	56.85 ± 0.92	36.15 ± 0.51	2.60 ± 0.17	231.92 ± 2.43	8.55
	24,624	0.2	90	47.67 ± 0.55	33.14 ± 0.30	2.44 ± 0.07	255.84 ± 3.01	4.78
	24,624	0.3	82	49.23 ± 1.44	35.65 ± 1.01	6.24 ± 0.16	241.80 ± 6.41	**3.13**
	24,624	0.4	84	58.47 ± 0.49	36.36 ± 0.29	3.41 ± 0.16	235.28 ± 3.48	9.99
	24,624	0.5	73	55.90 ± 1.36	33.85 ± 1.24	5.77 ± 0.29	259.00 ± 10.43	7.93
WCOR 4-3-3-3|9-9	5070	0.0	106	50.08 ± 1.42	39.78 ± 0.83	2.44 ± 0.20	208.56 ± 3.55	**2.09**
	5070	0.1	95	50.64 ± 1.24	39.00 ± 0.76	2.58 ± 0.11	212.76 ± 3.19	2.68
	5070	0.2	86	51.16 ± 0.83	26.63 ± 0.57	12.06 ± 0.79	330.34 ± 5.77	4.38
	5070	0.3	81	51.17 ± 0.77	37.19 ± 0.27	9.15 ± 0.24	234.04 ± 2.70	3.33
	5070	0.4	74	48.97 ± 1.99	24.82 ± 0.53	13.95 ± 0.22	353.00 ± 2.89	4.45
	5070	0.5	71	50.74 ± 0.33	37.76 ± 0.72	8.97 ± 0.20	227.68 ± 3.49	2.89
	24,624	0.0	107	47.87 ± 1.07	40.16 ± 0.73	2.98 ± 0.15	208.18 ± 2.58	3.90
	24,624	0.1	96	48.63 ± 1.65	40.79 ± 0.95	2.76 ± 0.29	205.68 ± 5.18	3.15
	24,624	0.2	87	50.05 ± 0.39	39.13 ± 0.20	2.72 ± 0.07	213.74 ± 1.15	2.13
	24,624	0.3	74	51.00 ± 0.24	37.83 ± 0.66	9.80 ± 0.18	226.14 ± 1.35	3.12
	24,624	0.4	69	47.61 ± 2.31	27.72 ± 1.27	15.08 ± 0.30	298.00 ± 8.33	5.38
	24,624	0.5	66	50.49 ± 0.63	36.35 ± 0.74	11.09 ± 0.22	244.00 ± 4.64	2.81
TSK 4-3-3-3	5070	0.5	120	49.07 ± 1.08	37.05 ± 0.82	4.96 ± 0.21	227.58 ± 4.46	3.13
	5070	0.6	118	52.90 ± 1.03	39.15 ± 0.48	4.13 ± 0.37	216.00 ± 2.79	4.70
	5070	0.7	119	50.21 ± 0.82	35.49 ± 0.29	5.67 ± 0.48	237.96 ± 2.54	**2.64**
	5070	0.8	121	50.79 ± 0.95	39.26 ± 0.85	3.93 ± 0.14	215.12 ± 4.17	2.79
	5070	0.9	119	47.52 ± 1.25	40.04 ± 0.48	4.57 ± 0.73	211.60 ± 1.83	4.23
	5070	1.0	119	53.41 ± 0.79	34.97 ± 0.37	5.73 ± 0.12	243.70 ± 3.90	5.57
	24,624	0.5	150	48.74 ± 1.09	36.34 ± 1.01	2.20 ± 0.13	230.76 ± 8.58	3.50
	24,624	0.6	150	48.46 ± 0.62	38.67 ± 0.56	4.92 ± 0.15	218.58 ± 2.58	3.52
	24,624	0.7	153	49.73 ± 0.90	34.53 ± 0.17	9.01 ± 0.57	245.20 ± 2.26	2.79
	24,624	0.8	150	48.43 ± 1.19	38.27 ± 0.88	4.53 ± 0.18	219.96 ± 4.75	3.59
	24,624	0.9	150	49.38 ± 1.26	38.96 ± 0.58	4.40 ± 0.08	215.98 ± 3.47	2.66
	24,624	1.0	156	43.66 ± 1.01	27.76 ± 0.33	2.62 ± 0.07	300.84 ± 3.31	8.93

Table 9
Environment *wsc8a*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s.

Method and configuration	Weights	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR	(0.60, 0.35, 0.05)	74	73.50 ± 4.41	34.91 ± 3.67	3.61 ± 0.68	203.34 ± 23.45	23.66
#Labels: 4-3-3-3|9-9	(0.70, 0.20, 0.10)	74	41.05 ± 1.13	36.15 ± 0.42	7.89 ± 0.63	189.86 ± 3.04	10.44
#Examples = 5070	(0.89, 0.10, 0.01)	88	53.20 ± 1.33	39.86 ± 0.71	5.67 ± 0.83	174.98 ± 1.79	**4.89**
$\phi = 0.3$	(0.95, 0.04, 0.01)	86	55.15 ± 0.73	29.69 ± 1.30	11.80 ± 1.06	231.56 ± 9.33	7.67
WCOR	(0.60, 0.35, 0.05)	76	70.97 ± 3.23	41.29 ± 2.34	2.59 ± 0.20	204.36 ± 18.09	20.74
#Labels: 4-3-3-3|9-9	(0.70, 0.20, 0.10)	79	45.06 ± 0.32	41.58 ± 0.33	2.76 ± 0.13	163.06 ± 1.82	6.29
#Examples = 5070	(0.89, 0.10, 0.01)	81	52.79 ± 1.36	36.98 ± 1.85	7.37 ± 0.62	187.90 ± 9.78	4.81
$\phi = 0.3$	(0.95, 0.04, 0.01)	76	49.49 ± 0.58	16.97 ± 0.16	13.17 ± 0.64	406.80 ± 9.97	**4.76**
HSWLR	(0.60, 0.35, 0.05)	56	49.59 ± 1.34	34.60 ± 0.69	2.50 ± 0.07	199.90 ± 2.55	2.91
#Labels: 4-2-3-2|9-9	(0.70, 0.20, 0.10)	67	49.99 ± 0.80	33.77 ± 0.20	2.67 ± 0.06	203.68 ± 2.73	**2.63**
#Examples = 5070	(0.89, 0.10, 0.01)	76	51.42 ± 0.78	30.46 ± 1.01	3.36 ± 0.13	222.34 ± 6.09	4.23
$\phi = 1$	(0.95, 0.04, 0.01)	76	53.87 ± 0.51	25.68 ± 0.44	6.17 ± 0.19	264.04 ± 3.19	6.92
TSK	(0.60, 0.35, 0.05)	103	77.47 ± 3.44	44.41 ± 2.20	2.36 ± 0.26	226.90 ± 9.20	26.28
#Labels: 4-3-3-3	(0.70, 0.20, 0.10)	106	42.90 ± 1.89	36.41 ± 2.82	2.68 ± 0.21	181.64 ± 4.31	8.75
#Examples = 5070	(0.89, 0.10, 0.01)	120	51.43 ± 1.36	37.54 ± 1.53	5.20 ± 0.50	182.54 ± 8.35	**3.53**
$\phi = 0.5$	(0.95, 0.04, 0.01)	114	48.77 ± 2.92	27.08 ± 1.24	8.27 ± 0.32	249.98 ± 9.43	4.40

In order to implement other studies, in the remaining of this section we have selected a combination of the analyzed parameters for each of the data-driven learning algorithms. The selection of the values of the parameters has been done not for picking the best representative of each methodology, but for making comparative studies. With that objective we have selected only one set of examples and one set of labels (in the HSWLR approach this set is different due to its special characteristics). The combinations are the following:

- COR: #Labels: 4-3-3-3|9-9, #Examples = 5070, $\phi = 0.3$.
- WCOR: #Labels: 4-3-3-3|9-9, #Examples = 5070, $\phi = 0.3$.
- HSWLR: #Labels: 4-2-3-2|9-9, #Examples = 5070, $\phi = 1.0$.
- TSK: #Labels: 4-3-3-3, #Examples = 5070, $\phi = 0.5$.

4.1.4. Weights in the scoring function

It would be interesting to know how the weights (w_i) included in the definition of α_i (Eq. (2)) influence the error of the controllers. Tables 9–11 show this analysis for four different combinations of weights. For the analysis, it must be taken into account that for this behavior the first objective is to place the robot at the reference distance, then select the highest possible speed, and finally choose the correct orientation. We have ranked the errors for each algo-

Table 10
Environment *gfs*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s.

Method and configuration	Weights	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR	(0.60, 0.35, 0.05)	74	48.14 ± 3.07	26.33 ± 1.64	4.68 ± 0.41	161.14 ± 21.60	**5.04**
#Labels: 4-3-3-3|9-9	(0.70, 0.20, 0.10)	74	41.01 ± 1.08	28.50 ± 0.37	5.69 ± 0.19	160.58 ± 3.76	11.24
#Examples = 5070	(0.89, 0.10, 0.01)	88	54.62 ± 5.98	29.85 ± 2.01	5.33 ± 0.42	149.90 ± 13.07	7.17
$\phi = 0.3$	(0.95, 0.04, 0.01)	86	56.51 ± 0.28	22.84 ± 0.93	7.35 ± 0.37	187.40 ± 3.42	9.58
WCOR	(0.60, 0.35, 0.05)	76	41.42 ± 2.94	28.32 ± 2.50	2.60 ± 0.46	138.24 ± 8.39	10.89
#Labels: 4-3-3-3|9-9	(0.70, 0.20, 0.10)	79	44.66 ± 1.34	33.52 ± 0.53	4.19 ± 0.23	136.50 ± 3.18	7.45
#Examples = 5070	(0.89, 0.10, 0.01)	81	51.53 ± 1.79	28.91 ± 1.10	6.12 ± 0.34	152.86 ± 7.57	**4.49**
$\phi = 0.3$	(0.95, 0.04, 0.01)	76	53.02 ± 4.52	11.16 ± 0.61	5.05 ± 0.13	396.06 ± 18.53	7.60
HSWLR	(0.60, 0.35, 0.05)	56	49.06 ± 0.73	23.63 ± 0.64	3.02 ± 0.11	186.04 ± 3.42	4.48
#Labels: 4-2-3-2|9-9	(0.70, 0.20, 0.10)	67	49.85 ± 0.45	22.97 ± 0.25	3.22 ± 0.07	189.76 ± 3.03	**3.84**
#Examples = 5070	(0.89, 0.10, 0.01)	76	51.49 ± 0.58	15.42 ± 0.38	2.56 ± 0.04	274.96 ± 7.11	5.80
$\phi = 1$	(0.95, 0.04, 0.01)	76	52.12 ± 0.61	12.88 ± 0.15	3.11 ± 0.06	321.16 ± 7.05	6.62
TSK	(0.60, 0.35, 0.05)	103	54.49 ± 0.49	30.50 ± 1.03	2.58 ± 0.36	155.88 ± 10.19	6.99
#Labels: 4-3-3-3	(0.70, 0.20, 0.10)	106	38.54 ± 2.00	27.80 ± 1.61	2.92 ± 0.20	161.00 ± 10.00	13.53
#Examples = 5070	(0.89, 0.10, 0.01)	120	50.39 ± 0.68	28.77 ± 0.70	4.81 ± 0.17	149.06 ± 3.15	**3.47**
$\phi = 0.5$	(0.95, 0.04, 0.01)	114	51.59 ± 2.19	22.14 ± 0.74	5.73 ± 0.65	187.18 ± 9.55	5.22

Table 11
Environment *rooms*, $d_{wall} = 50$ cm, $v_{max} = 60$ cm/s.

Method and configuration	Weights	#Rules	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR	(0.60, 0.35, 0.05)	74	55.18 ± 5.18	31.95 ± 1.33	4.29 ± 1.27	269.56 ± 22.10	7.47
#Labels: 4-3-3-3|9-9	(0.70, 0.20, 0.10)	74	36.11 ± 0.60	37.27 ± 0.44	6.14 ± 0.25	234.60 ± 3.59	14.77
#Examples = 5070	(0.89, 0.10, 0.01)	88	46.80 ± 0.59	37.82 ± 0.41	6.76 ± 0.31	227.16 ± 1.03	**5.10**
$\phi = 0.3$	(0.95, 0.04, 0.01)	86	54.09 ± 1.49	25.72 ± 1.25	12.80 ± 0.59	321.16 ± 8.17	7.11
WCOR	(0.60, 0.35, 0.05)	76	44.72 ± 4.78	36.32 ± 3.39	1.57 ± 0.16	248.94 ± 24.87	7.12
#Labels: 4-3-3-3|9-9	(0.70, 0.20, 0.10)	79	42.34 ± 0.45	43.76 ± 0.23	2.47 ± 0.08	194.54 ± 0.96	8.52
#Examples = 5070	(0.89, 0.10, 0.01)	81	51.17 ± 0.77	37.19 ± 0.27	9.15 ± 0.24	234.04 ± 2.70	**3.33**
$\phi = 0.3$	(0.95, 0.04, 0.01)	76	50.55 ± 0.93	16.87 ± 0.26	10.41 ± 0.89	499.88 ± 12.38	4.81
HSWLR	(0.60, 0.35, 0.05)	56	46.36 ± 0.78	33.84 ± 0.72	2.24 ± 0.10	254.10 ± 6.80	5.89
#Labels: 4-2-3-2|9-9	(0.70, 0.20, 0.10)	67	49.41 ± 0.81	34.07 ± 0.26	2.87 ± 0.08	245.74 ± 1.40	**3.12**
#Examples = 5070	(0.89, 0.10, 0.01)	76	50.09 ± 0.88	28.71 ± 0.29	3.04 ± 0.20	290.70 ± 3.66	3.21
$\phi = 1$	(0.95, 0.04, 0.01)	76	51.43 ± 0.17	22.53 ± 0.36	4.42 ± 0.19	367.42 ± 5.24	5.03
TSK	(0.60, 0.35, 0.05)	103	51.70 ± 3.69	32.68 ± 2.68	1.76 ± 0.24	291.34 ± 31.31	4.26
#Labels: 4-3-3-3	(0.70, 0.20, 0.10)	106	30.05 ± 0.31	27.72 ± 0.37	1.06 ± 0.10	311.90 ± 2.13	21.18
#Examples = 5070	(0.89, 0.10, 0.01)	120	49.07 ± 1.08	37.05 ± 0.82	4.96 ± 0.21	227.58 ± 4.46	**3.13**
$\phi = 0.5$	(0.95, 0.04, 0.01)	114	48.81 ± 3.22	26.61 ± 1.40	7.10 ± 0.68	305.48 ± 3.81	4.41

Table 12
Environment *wsc8a*, weights = (0.89, 0.10, 0.01).

Method and configuration	d_{wall}	v_{max}	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR	75	60	88.65 ± 1.09	44.31 ± 0.48	4.52 ± 0.44	158.14 ± 0.64	13.85
#Labels: 4-3-3-3|9-9	50	60	53.20 ± 1.33	39.86 ± 0.71	5.67 ± 0.83	174.98 ± 1.79	4.89
#Examples = 5070	40	60	43.59 ± 0.97	39.75 ± 0.58	5.60 ± 0.97	175.62 ± 2.76	5.26
$\phi = 0.3$, #R = 88	30	50	30.53 ± 2.69	28.90 ± 3.97	7.15 ± 1.14	210.04 ± 4.66	**2.59**
WCOR	75	60	85.51 ± 0.47	42.59 ± 0.39	4.84 ± 0.51	161.54 ± 1.19	11.20
#Labels: 4-3-3-3|9-9	50	60	52.79 ± 1.36	36.98 ± 1.85	7.37 ± 0.62	187.90 ± 9.78	4.81
#Examples = 5070	40	60	41.36 ± 0.87	35.06 ± 0.66	7.89 ± 0.55	196.22 ± 3.39	**3.72**
$\phi = 0.3$, #R = 81	30	50	32.46 ± 0.46	31.21 ± 0.27	7.93 ± 0.28	219.72 ± 2.15	4.09
HSWLR	75	60	79.77 ± 0.48	30.13 ± 0.49	2.79 ± 0.18	228.58 ± 1.87	7.28
#Labels: 4-2-3-2|9-9	50	60	51.42 ± 0.78	30.46 ± 1.01	3.36 ± 0.13	222.34 ± 6.09	4.23
#Examples = 5070	40	60	40.22 ± 0.22	30.47 ± 0.56	3.77 ± 0.09	222.46 ± 4.05	3.15
$\phi = 1$, #R = 76	30	50	30.28 ± 0.11	27.83 ± 0.23	4.07 ± 0.09	244.14 ± 1.84	**2.47**
TSK	75	60	81.66 ± 0.92	39.87 ± 0.33	5.39 ± 0.39	170.90 ± 0.73	8.01
#Labels: 4-3-3-3	50	60	51.43 ± 1.36	37.54 ± 1.53	5.20 ± 0.50	182.54 ± 8.35	3.53
#Examples = 5070	40	60	39.31 ± 1.56	35.50 ± 1.68	5.09 ± 0.32	193.80 ± 8.90	3.07
$\phi = 0.5$, #R = 120	30	50	30.16 ± 0.60	32.13 ± 2.63	5.42 ± 0.50	223.56 ± 3.33	**1.93**

Table 13
Environment *gfs*, weights = (0.89, 0.10, 0.01).

Method and configuration	d_{wall}	v_{max}	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR	75	60	70.37 ± 14.18	25.03 ± 4.23	5.92 ± 0.79	137.80 ± 8.15	7.66
#Labels: 4-3-3-3|9-9	50	60	54.62 ± 5.98	29.85 ± 2.01	5.33 ± 0.42	149.90 ± 13.07	7.17
#Examples = 5070	40	60	41.48 ± 3.16	30.37 ± 1.38	6.25 ± 0.56	146.34 ± 15.29	4.30
$\phi = 0.3$, #R = 88	30	50	31.31 ± 1.23	25.44 ± 3.43	7.77 ± 1.27	173.94 ± 4.84	**3.64**
WCOR	75	60	73.52 ± 6.38	26.10 ± 2.51	5.47 ± 0.61	143.70 ± 5.44	4.72
#Labels: 4-3-3-3|9-9	50	60	51.53 ± 1.79	28.91 ± 1.10	6.12 ± 0.34	152.86 ± 7.57	**4.49**
#Examples = 5070	40	60	43.14 ± 0.35	28.54 ± 0.68	6.27 ± 0.18	158.32 ± 4.90	5.97
$\phi = 0.3$, #R = 81	30	50	33.11 ± 0.51	26.00 ± 0.34	6.72 ± 0.18	181.46 ± 2.48	5.20
HSWLR	75	60	74.03 ± 1.22	13.34 ± 0.24	2.55 ± 0.15	293.90 ± 6.93	5.54
#Labels: 4-2-3-2|9-9	50	60	51.49 ± 0.58	15.42 ± 0.38	2.56 ± 0.04	274.96 ± 7.11	5.80
#Examples = 5070	40	60	42.03 ± 0.72	19.67 ± 0.49	3.47 ± 0.24	226.80 ± 4.19	5.86
$\phi = 1$, #R = 76	30	50	31.87 ± 0.25	19.89 ± 0.41	3.84 ± 0.05	232.24 ± 2.52	**4.69**
TSK	75	60	77.53 ± 2.78	26.19 ± 1.15	5.33 ± 0.38	150.48 ± 6.23	5.66
#Labels: 4-3-3-3	50	60	50.39 ± 0.68	28.77 ± 0.70	4.81 ± 0.17	149.06 ± 3.15	**3.47**
#Examples = 5070	40	60	39.57 ± 1.67	28.96 ± 0.66	5.57 ± 0.21	156.44 ± 3.10	3.49
$\phi = 0.5$, #R = 120	30	50	32.89 ± 2.87	27.26 ± 0.55	4.98 ± 0.16	177.26 ± 11.04	4.88

Table 14
Environment *rooms*, weights = (0.89, 0.10, 0.01).

Method and Configuration	d_{wall}	v_{max}	Dist.(cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR	75	60	78.23 ± 0.74	38.14 ± 0.58	8.48 ± 0.62	212.04 ± 2.05	5.09
#Labels: 4-3-3-3|9-9	50	60	46.80 ± 0.59	37.82 ± 0.41	6.76 ± 0.31	227.16 ± 1.03	5.10
#Examples = 5070	40	60	41.70 ± 3.63	36.42 ± 1.39	7.42 ± 0.32	250.30 ± 19.71	3.89
$\phi = 0.3$, #R = 88	30	50	31.08 ± 0.99	29.89 ± 2.17	10.23 ± 1.02	289.2 ± 6.37	**2.98**
WCOR	75	60	83.28 ± 0.67	37.50 ± 0.22	7.02 ± 0.59	213.48 ± 2.66	9.70
#Labels: 4-3-3-3|9-9	50	60	51.17 ± 0.77	37.19 ± 0.27	9.15 ± 0.24	234.04 ± 2.70	3.33
#Examples = 5070	40	60	42.01 ± 0.54	35.82 ± 0.46	7.91 ± 0.19	240.38 ± 3.53	4.23
$\phi = 0.3$, #R = 81	30	50	31.37 ± 0.17	31.41 ± 0.21	8.96 ± 0.34	278.54 ± 2.49	**3.09**
HSWLR	75	60	78.76 ± 0.30	24.56 ± 0.23	2.42 ± 0.09	323.70 ± 2.38	6.93
#Labels: 4-2-3-2|9-9	50	60	50.09 ± 0.88	28.71 ± 0.29	3.04 ± 0.20	290.70 ± 3.66	3.21
#Examples = 5070	40	60	39.58 ± 0.28	30.76 ± 0.21	3.54 ± 0.03	276.68 ± 2.44	3.30
$\phi = 1$, #R = 76	30	50	29.29 ± 0.25	28.75 ± 0.18	3.95 ± 0.16	302.34 ± 1.69	**2.76**
TSK	75	60	79.59 ± 0.36	36.12 ± 0.44	4.01 ± 0.09	220.14 ± 2.92	6.52
#Labels: 4-3-3-3	50	60	49.07 ± 1.08	37.05 ± 0.82	4.96 ± 0.21	227.58 ± 4.46	3.13
#Examples = 5070	40	60	38.01 ± 0.76	36.38 ± 0.60	4.37 ± 0.29	237.74 ± 2.83	4.15
$\phi = 0.5$, #R = 120	30	50	28.64 ± 0.24	31.46 ± 0.18	4.83 ± 0.12	281.26 ± 1.15	**3.08**

Table 15
$d_{wall} = 50$ cm; $v_{max} = 60$ cm/s; weights = (0.89, 0.10, 0.01); #Labels: 4-3-3-3|9-9 (COR and WCOR), 4-2-3-2|9-9 (HSWLR), and 4-3-3-3 (TSK); #rules: 88 (COR), 81 (WCOR), 76 (HSWLR), and 120 (TSK).

Method	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
Environment: autolab					
COR	56.88 ± 0.91	25.69 ± 0.79	10.79 ± 0.21	587.96 ± 39.72	9.62
WCOR	52.97 ± 1.10	33.47 ± 0.89	7.12 ± 0.52	455.98 ± 41.60	5.33
HSWLR	51.50 ± 0.34	23.50 ± 0.97	3.05 ± 0.14	618.40 ± 20.98	5.00
TSK	51.87 ± 2.99	33.05 ± 1.33	4.61 ± 0.11	465.56 ± 15.33	**4.38**
Environment: office					
COR	55.97 ± 1.65	32.48 ± 0.90	4.06 ± 0.28	457.58 ± 15.00	8.13
WCOR	54.59 ± 1.10	33.13 ± 0.97	6.76 ± 0.53	448.16 ± 10.36	6.82
HSWLR	53.43 ± 1.22	24.69 ± 0.66	3.73 ± 0.11	594.74 ± 13.16	6.62
TSK	53.75 ± 0.97	34.26 ± 0.65	5.24 ± 0.22	432.38 ± 10.48	**5.95**
Environment: mapunet					
COR	49.11 ± 1.98	19.75 ± 1.72	7.64 ± 0.50	318.28 ± 15.21	**4.83**
WCOR	47.90 ± 2.17	24.36 ± 1.39	7.51 ± 0.46	258.72 ± 16.76	5.45
HSWLR	42.43 ± 3.30	14.99 ± 1.85	4.45 ± 0.42	410.62 ± 18.81	11.31
TSK	46.80 ± 2.58	22.14 ± 2.71	6.67 ± 0.88	260.50 ± 7.79	6.67
Environment: hospital					
COR	54.12 ± 0.92	35.63 ± 0.77	6.95 ± 0.28	2864.92 ± 45.27	**6.15**
WCOR	55.26 ± 1.01	33.71 ± 0.14	6.52 ± 0.12	3073.98 ± 23.63	7.36
HSWLR	54.60 ± 1.65	25.07 ± 0.49	3.89 ± 0.06	4209.68 ± 166.14	7.63
TSK	54.50 ± 1.49	34.31 ± 0.32	5.01 ± 0.11	3053.74 ± 123.72	6.62

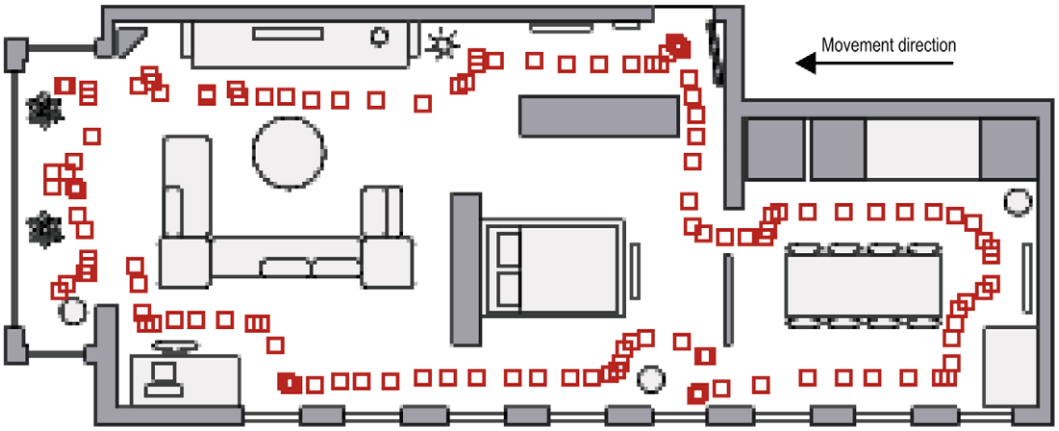


Fig. 3. Path of the robot in environment *mapunet* for the controller WCOR (#Labels: 4-3-3-3|9-9, #Examples = 5070, $\phi = 0.3$, #R = 81).

rithm and each environment. Then, we have added the ranks over all the environments. For COR, WCOR and TSK, it can be observed that the worst error corresponds to the combination of weights that gives a higher value to the less important objective (orientation). For the same reason, the first combination of weights obtains the second worst error. The other two combinations give a very low weight to the orientation. Also both of them have a high weight for the distance. Thus, both combinations get good values of the distance. However, due to the low weight in velocity, the last combination has obtained very low average speeds. The pattern followed by HSWLR is completely different from the other methods. That, shows again the difficulties in obtaining a good controller under this paradigm.

It is important to notice that the higher the weight of the distance, the higher the number of rules. The reason is that it is much more difficult to control a correct distance to the wall than a high speed. So, controllers that have been learned with a low weight in distance are simpler and their performance is lower. Nevertheless, this effect has a limit if the weight of the velocity is low. On the contrary, we should hope that a higher weight in velocity (lower weight in distance) would obtain higher average velocities. This is partially true, as in some cases due to the simplicity in the control of the distance the controller cannot take advantage of a better control in velocity.

4.1.5. d_{wall} and v_{max}

All the knowledge bases have been learned with $d_{wall} = 50$ and $v_{max} = 60$. Nevertheless, a final user could change these parameters for the execution of a controller on the robot. So, if the controller is learned with the mentioned values, how is accuracy affected executing the controller with different values for d_{wall} and v_{max}? The analysis is shown in Tables 12–14. In most of the tests, performance is not affected, reflecting the high robustness of the learning framework. There are a couple of exceptions in environment *wsc8a* for $d_{wall} = 75$, where distance takes higher values than the reference, but also with very high average values in speed.

Finally, we have chosen another four environments to test more in deep the selected controller for each of the learning algorithms. These environments have paths with a higher length, and with a structure like floor plans of buildings. They are *autolab*, *office*, *mapunet* (this environment is furnished) and *hospital*. Results are shown in Table 15. As we are interested in the performance of the controllers in all the environments, we have added the errors of each controller in the four environments, obtaining the following results: TSK (23.62), WCOR (24.96), COR (28.73) and HSWLR (30.56). Nevertheless, the error is a parameter that gives an approximation of the performance of a controller, but sometimes that performance must be analyzed in more detail. The only way to really evaluate the performance is by visual inspection of the movement of the robot.

Figs. 3 and 4 show the path of the robot along environments *mapunet* and *hospital* for the WCOR controller. Each red square indicates the position of the robot each six control cycles. The higher the concentration of marks, the lower the linear velocity of the robot. Environment *mapunet* (Fig. 3)[7] is the floor plan of a furnished apartment. The path that the robot has to follow is really complex, as there are several deadlocks in which the robot has to turn on its own, very narrow passageways, and obstacles with irregular shapes. One of the particularities of this environment is that in some narrow areas the robot cannot follow the wall at the reference distance, because this would get it into a collision with the left wall. Thus, all the average distances are under d_{wall}.

Environment *hospital* (Fig. 4) represents the floor plan of a hospital. The robot has to follow a path of more than one kilometer long, with more than 150 corners, and has to cross more than 40 doors (Table 2). The performance of COR, WCOR and TSK is very similar, while the HSWLR approach obtains a much lower average speed.

4.1.6. Final comments on accuracy and interpretability

In summary, from the accuracy point of view, we can conclude that the performance of the learned controllers is, in general, good for the different algorithms and combinations of values in the parameters. The study about the dependencies of accuracy with the values of the design parameters reflects that the learning framework is quite sensible to the weights in SF (Eq. (1)) as with these weights we are defining the behavior to be learned. Also, the granularity of the most important input variables influences the accuracy of the controllers, although in a lower degree. On the other hand, the number of examples and the value of ϕ do not affect the performance if they take values in a reasonable range.

The interpretability of each of the learning algorithms has been described in Section 3. In this section, we have analyzed the number of rules, as a higher number reduces interpretability. Of course, the number of labels is the parameter that influences more the number of rules. Moreover, the value of ϕ allows to reduce the number of rules, increasing interpretability. Also, a higher number of examples slightly improves interpretability. Finally, the weights of SF also modify the number of rules: the higher the weights of the most complex (difficult to control) variables, the higher the num-

[7] The associated video can be downloaded from http://www.gsi.dec.usc.es/mucientes/videos/wallfollowing_mapunet_eswa09.avi.

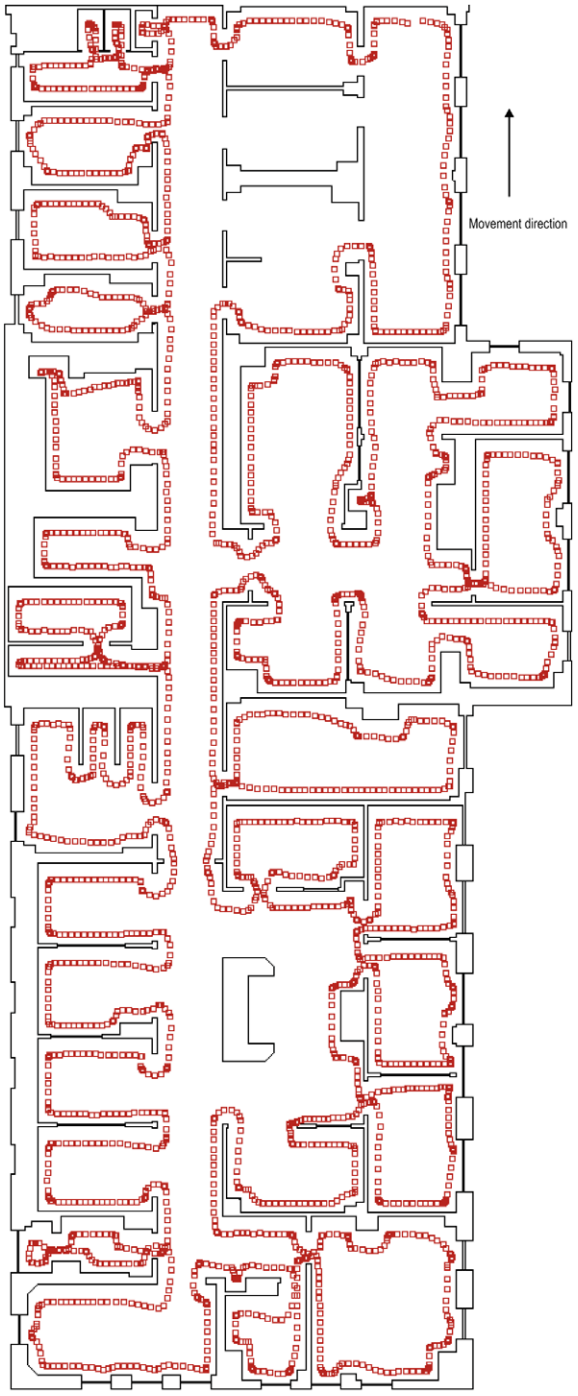


Fig. 4. Path of the robot in environment *hospital* for the controller WCOR (#Labels: 4-3-3-3|9-9, #Examples = 5070, $\phi = 0.3$, #R = 81).

ber of rules, although the interaction among variables must also be taken into account.

Fig. 5 shows the trade-off between the error and the number of rules for each of the data-driven learning algorithms. For each combination of parameters and each environment, the best controller of each methodology was selected. Then, the average value of the error was calculated and represented against the number of rules. The graphic proves that, although TSK is the most accurate method, the number of rules is really high. Nevertheless, WCOR obtains a similar accuracy, but with a much more reduced rule base. Therefore, we can conclude that WCOR is the algorithm with the best accuracy/interpretability trade-off for the learning framework.

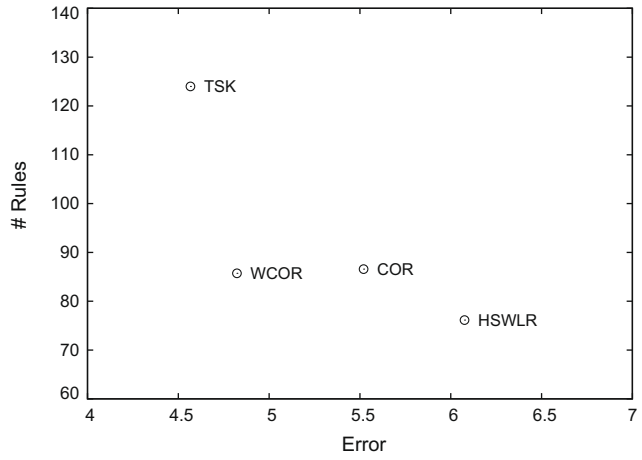


Fig. 5. Average values of the most accurate results for each environment and learning method for the wall-following behavior.

Table 16
Real robot. $d_{wall} = 50$ cm; $v_{max} = 60$ cm/s; weights = (0.89, 0.10, 0.01); #Labels: 4-3-3-3|9-9 (COR and WCOR), 4-2-3-2|9-9 (HSWLR), and 4-3-3-3 (TSK); #rules: 88 (COR), 81 (WCOR), 76 (HSWLR), and 120 (TSK).

Method	Dist. (cm)	Vel. (cm/s)	Vel.ch. (cm/s)	Time (s)	Error
COR	44.64 ± 3.76	25.04 ± 0.33	5.26 ± 0.25	124.13 ± 13.07	8.32
WCOR	50.04 ± 2.29	26.39 ± 0.99	6.78 ± 0.05	114.23 ± 7.04	**3.40**
HSWLR	46.79 ± 0.76	18.03 ± 0.42	4.05 ± 0.30	153.70 ± 7.50	7.09
TSK	46.74 ± 0.37	27.00 ± 0.61	7.28 ± 0.04	118.91 ± 7.92	6.23

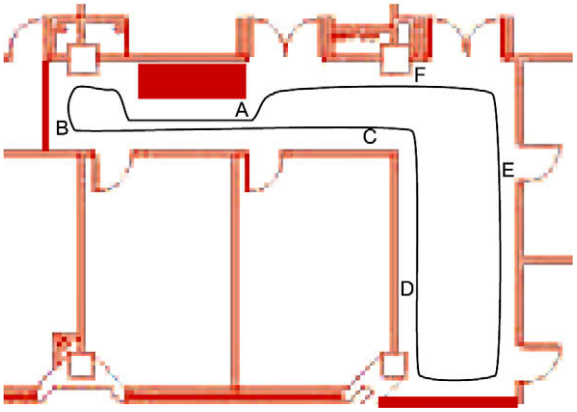
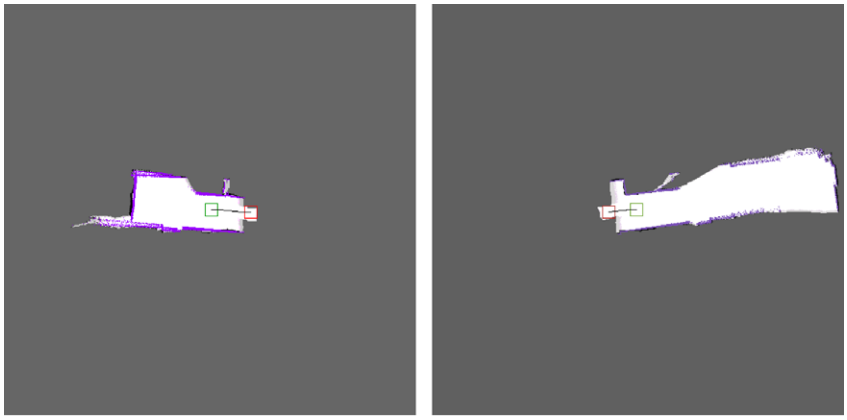


Fig. 6. Approximate path followed by a *Pioneer II AT* robot in a real environment.

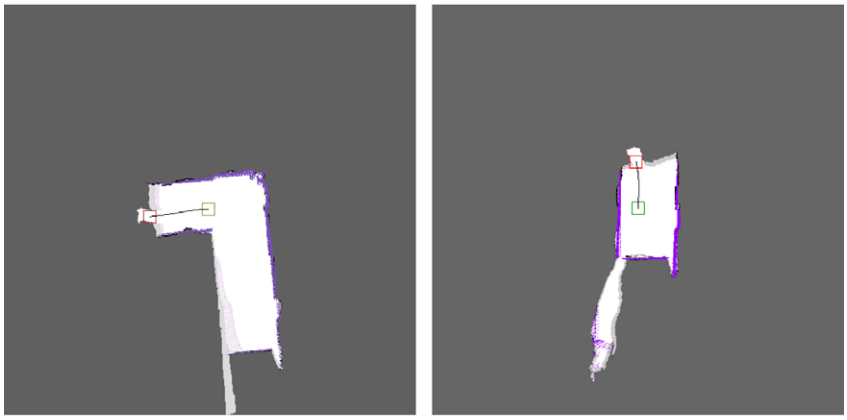
4.1.7. Real robot

As has been mentioned, on the real robot the sensory information is provided by a laser range finder heading to the front of the robot. This configuration covers 180° and not the whole surrounding. The practical implication of this configuration is that in the convex corners the robot fails to detect the wall that was following earlier, and has to look for the next wall "blindly" during a higher number of control cycles.

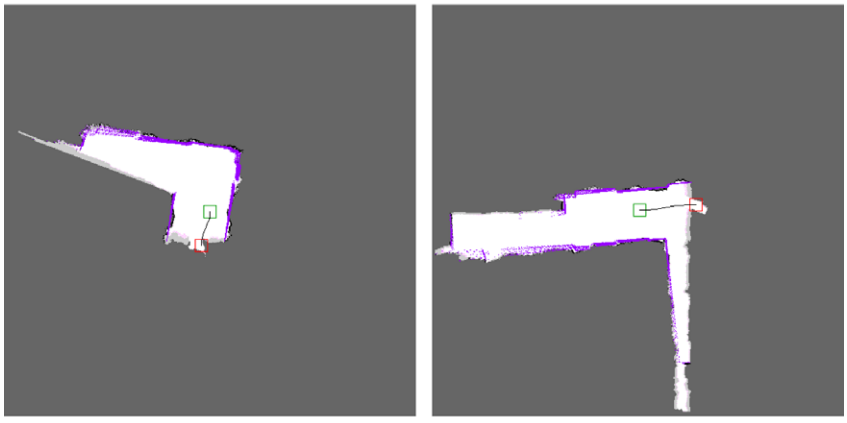
Table 16 shows the results of the tests on the real robot. The average values and standard deviations have been calculated along three laps. WCOR clearly outperforms the other methodologies, obtaining the best distance, and the second fastest speed (only TSK improves that velocity). Although TSK is faster, the time spent per lap is higher than WCOR. The reason is that the trajectory implemented by the robot is not the same, as the average distance is different and, also, the path of the robot along a straight wall is not always rectilinear, but sinuous (depending on the accuracy of



(a) Position A in Fig. 6. (b) Position B in Fig. 6.



(c) Position C in Fig. 6. (d) Position D in Fig. 6.



(e) Position E in Fig. 6. (f) Position F in Fig. 6.

Fig. 7. Reconstruction of the path of a *Pioneer II AT* robot in a real environment.

the controller) due to the existence of closed doors that slightly modify the distance to the wall in a few centimeters.

Results on the real robot clearly fulfill one of the requisites that the learning framework should have: the controllers, that have been learned off-line, can run on the real robot with a good performance and without any tuning stage. Moreover, the four selected learning methodologies obtain good accuracies.

Fig. 6 shows the approximate path followed by the robot in the real environment[8]. During the test, all the doors of the floor plan were closed. Due to the existence of odometry errors, the real trace of the robot in real environments cannot be reproduced as a whole. Nevertheless, in order to show in a realistic way the performance of the robot, Fig. 7 shows some snapshots along the path[9]. Each of them represents an occupancy grid map of the environment, together with the trajectory of the robot along 50 control cycles (initial position in red and final position in green) and the laser range scanner measurements (displayed in violet). The snapshots have been obtained in the points of the path labeled in Fig. 6. The grid map has different gray levels, reflecting the probability of being occupied by an obstacle or free (walls appear in black).

[8] The associated video can be downloaded from http://www.gsi.dec.usc.es/mucientes/videos/wallfollowing_realrobot_eswa09.wmv.

[9] For small displacements of the robot, odometry errors are not significant.

Table 17
Trajectory: *zigzag*, weights = (1000, 60, 4), #Examples = 6435.

Method labels	ϕ	#Rules	δd (cm)	δdev (°)	*Rvc*	Error
COR 3-5-3-3|9-9	0.0	135	33.98 ± 1.70	20.79 ± 0.76	5.45 ± 0.74	32.66
	0.1	118	31.36 ± 0.47	24.42 ± 0.38	4.85 ± 0.27	30.67
	0.2	108	29.95 ± 0.65	21.52 ± 0.44	6.14 ± 0.44	**29.11**
	0.3	121	34.14 ± 0.51	22.20 ± 0.55	5.13 ± 0.54	32.95
	0.4	95	43.57 ± 2.46	23.25 ± 1.44	6.10 ± 0.13	41.54
	0.5	104	35.46 ± 0.68	21.14 ± 0.67	6.51 ± 0.43	34.03
WCOR 3-5-3-3|9-9	0.0	134	31.09 ± 0.19	20.65 ± 0.55	6.53 ± 0.12	30.05
	0.1	127	31.81 ± 0.15	20.93 ± 0.15	5.70 ± 0.17	30.72
	0.2	120	29.26 ± 0.16	21.07 ± 0.18	5.42 ± 0.19	28.44
	0.3	104	27.22 ± 0.39	20.19 ± 0.22	6.10 ± 0.29	**26.52**
	0.4	95	30.25 ± 0.54	20.84 ± 0.81	6.07 ± 0.51	29.31
	0.5	94	29.55 ± 0.14	19.65 ± 0.40	5.83 ± 0.33	28.56
HSWLR 3-5-3-3|9-9	0.5	136	33.11 ± 1.44	21.13 ± 0.48	5.67 ± 0.49	31.91
	0.6	135	32.21 ± 0.40	21.12 ± 0.64	5.86 ± 0.24	31.10
	0.7	136	31.80 ± 0.25	21.09 ± 0.32	6.08 ± 0.17	**30.73**
	0.8	136	32.11 ± 0.20	21.16 ± 0.56	6.14 ± 0.47	31.02
	0.9	135	33.33 ± 0.59	21.00 ± 0.72	5.72 ± 0.27	32.10
	1.0	136	31.98 ± 0.47	20.91 ± 0.56	5.90 ± 0.36	30.87
TSK 3-5-3-3	0.5	183	23.21 ± 1.39	17.51 ± 0.64	17.12 ± 1.48	**22.64**
	0.6	185	30.96 ± 5.38	18.66 ± 1.73	7.95 ± 1.03	29.73
	0.7	164	33.57 ± 1.75	19.71 ± 0.88	9.48 ± 1.02	32.18
	0.8	176	34.90 ± 1.27	18.14 ± 0.68	10.38 ± 0.95	33.22
	0.9	175	23.19 ± 1.59	18.76 ± 1.29	17.21 ± 0.84	22.75
	1.0	174	37.86 ± 2.52	19.78 ± 1.24	11.93 ± 1.31	36.05

Table 18
Trajectory: *chicane*, weights = (1000, 60, 4), #Examples = 6435.

Method labels	ϕ	#Rules	δd (cm)	δdev (°)	*Rvc*	Error
COR 3-5-3-3|9-9	0.0	135	10.94 ± 0.40	4.60 ± 0.18	7.61 ± 0.35	10.31
	0.1	118	11.46 ± 0.29	5.00 ± 0.11	5.59 ± 0.41	10.81
	0.2	108	10.53 ± 0.14	3.88 ± 0.07	8.92 ± 0.24	9.87
	0.3	121	7.19 ± 0.34	5.56 ± 0.09	6.37 ± 0.14	**7.03**
	0.4	95	12.14 ± 0.76	4.48 ± 0.09	5.37 ± 0.30	11.37
	0.5	104	9.27 ± 0.44	4.46 ± 0.08	5.40 ± 0.25	8.79
WCOR 3-5-3-3|9-9	0.0	134	10.57 ± 0.78	4.68 ± 0.06	7.45 ± 0.69	9.98
	0.1	127	10.40 ± 0.27	4.94 ± 0.06	7.06 ± 0.47	9.85
	0.2	120	10.04 ± 0.28	4.41 ± 0.12	6.65 ± 0.21	9.48
	0.3	104	10.52 ± 0.22	4.55 ± 0.04	6.96 ± 0.56	9.92
	0.4	95	9.94 ± 0.37	4.52 ± 0.08	5.94 ± 0.32	**9.40**
	0.5	94	10.46 ± 0.40	4.56 ± 0.04	5.72 ± 0.28	9.87
HSWLR 3-5-3-3|9-9	0.5	136	10.99 ± 0.18	4.01 ± 0.08	6.89 ± 0.32	**10.29**
	0.6	135	11.06 ± 0.33	4.11 ± 0.16	6.24 ± 0.55	10.37
	0.7	136	11.41 ± 0.34	3.96 ± 0.07	6.11 ± 0.41	10.67
	0.8	136	11.23 ± 0.47	3.99 ± 0.32	6.37 ± 0.27	10.51
	0.9	135	11.23 ± 0.49	4.10 ± 0.11	6.25 ± 0.52	10.52
	1.0	136	11.23 ± 0.34	4.04 ± 0.16	6.38 ± 0.17	10.51
TSK 3-5-3-3	0.5	183	10.49 ± 0.91	4.11 ± 0.36	9.74 ± 0.84	**9.85**
	0.6	185	18.06 ± 2.56	4.20 ± 0.28	12.76 ± 2.13	16.67
	0.7	164	14.02 ± 3.79	4.75 ± 0.26	15.23 ± 2.41	13.09
	0.8	176	11.33 ± 1.79	5.06 ± 1.03	9.09 ± 2.25	10.70
	0.9	175	10.94 ± 0.51	5.49 ± 0.42	12.97 ± 1.61	10.40
	1.0	174	13.28 ± 2.63	4.99 ± 0.08	9.29 ± 1.95	12.45

4.2. Moving object following behavior

The objective of this section is to prove that the framework to learn behaviors is also valid for other behaviors, and other robots with different drive configurations and sensors. For this reason we will only analyze the performance of the controllers in five different situations, but without modifying the number of labels, the number of examples and the weights. One hundred and twenty tests have been done using the *Nomad 200* robot simulation software. The *Nomad 200* robot is equipped with a ring of 16 ultrasound sensors uniformly distributed. The robot has synchronous drive, thus the three wheels point always in the same direction. With this configuration the robot is able to turn on its track.

Tables 17–21 show the results obtained for five different types of trajectories of the moving object: *zigzag*, *chicane*, *loop*, *spiral* and *zigzag–chicane*. In all the experiments the robot has to follow the moving object at a distance of $d_{ref} = 150$ cm and with an angle of $dev_{ref} = 0$. Each row of the tables shows the results of a learned controller. In order to evaluate the quality of the controllers we have measured three different indicators:

- $\delta d = |d - d_{ref}|$: error in the distance.
- $\delta dev = |dev - dev_{ref}|$: error in the angle.

Table 19
Trajectory: *loop*, weights = (1000, 60, 4), #Examples = 6435.

Method labels	ϕ	#Rules	δd (cm)	δdev (°)	Rvc	Error
COR 3-5-3-3|9-9	0.0	135	28.98 ± 0.43	9.68 ± 0.10	5.68 ± 1.02	27.05
	0.1	118	26.36 ± 0.29	10.08 ± 0.13	6.99 ± 0.31	**24.73**
	0.2	108	28.42 ± 0.30	9.19 ± 0.16	8.75 ± 0.79	26.50
	0.3	121	33.21 ± 0.20	10.80 ± 0.20	4.29 ± 0.36	30.97
	0.4	95	43.22 ± 1.16	10.88 ± 0.17	5.07 ± 0.30	39.99
	0.5	104	33.82 ± 0.57	10.25 ± 0.20	4.58 ± 0.50	31.46
WCOR 3-5-3-3|9-9	0.0	134	33.93 ± 0.86	9.89 ± 0.16	5.03 ± 0.95	31.53
	0.1	127	25.29 ± 0.68	9.27 ± 0.12	5.87 ± 0.45	**23.69**
	0.2	120	27.56 ± 0.32	9.47 ± 0.08	6.23 ± 0.75	25.75
	0.3	104	26.49 ± 0.25	9.54 ± 0.22	7.53 ± 0.63	24.80
	0.4	95	28.24 ± 0.60	9.76 ± 0.14	5.17 ± 0.32	26.39
	0.5	94	27.01 ± 0.38	9.50 ± 0.09	5.43 ± 0.65	25.26
HSWLR 3-5-3-3|9-9	0.5	136	27.73 ± 0.33	9.53 ± 0.11	6.42 ± 0.67	25.91
	0.6	135	27.33 ± 0.27	9.60 ± 0.11	5.95 ± 0.20	25.56
	0.7	136	27.32 ± 0.29	9.52 ± 0.04	5.94 ± 0.63	25.54
	0.8	136	27.68 ± 0.40	9.54 ± 0.09	5.88 ± 0.29	25.87
	0.9	135	29.18 ± 0.20	9.64 ± 0.11	6.03 ± 0.27	27.23
	1.0	136	27.26 ± 0.33	9.54 ± 0.12	5.73 ± 0.28	**25.49**
TSK 3-5-3-3	0.5	183	32.64 ± 5.18	12.52 ± 0.60	13.81 ± 1.14	30.63
	0.6	185	17.39 ± 1.55	11.29 ± 0.17	6.84 ± 0.98	**16.78**
	0.7	164	20.91 ± 1.09	14.17 ± 0.89	15.16 ± 1.77	20.24
	0.8	176	25.36 ± 6.01	11.06 ± 1.12	7.69 ± 1.09	23.93
	0.9	175	18.01 ± 6.39	9.68 ± 3.11	14.53 ± 2.82	17.18
	1.0	174	20.83 ± 2.75	10.61 ± 1.37	14.38 ± 0.92	19.81

Table 20
Trajectory: *spiral*, weights = (1000, 60, 4), #Examples = 6435.

Method labels	ϕ	#Rules	δd (cm)	δdev (°)	Rvc	Error
COR 3-5-3-3|9-9	0.0	135	26.40 ± 4.92	8.58 ± 4.00	7.41 ± 2.62	24.62
	0.1	118	22.17 ± 0.31	7.09 ± 0.08	7.79 ± 0.82	**20.66**
	0.2	108	23.37 ± 0.48	6.38 ± 0.09	12.71 ± 1.41	21.67
	0.3	121	24.28 ± 0.30	7.60 ± 0.10	4.63 ± 0.29	22.61
	0.4	95	33.58 ± 0.66	7.72 ± 0.17	6.10 ± 0.13	30.99
	0.5	104	26.57 ± 0.88	7.25 ± 0.11	5.33 ± 0.45	24.64
WCOR 3-5-3-3|9-9	0.0	134	28.01 ± 0.29	6.88 ± 0.10	7.82 ± 0.16	25.90
	0.1	127	21.33 ± 0.43	6.44 ± 0.06	7.11 ± 0.42	**19.84**
	0.2	120	22.48 ± 0.20	6.60 ± 0.17	8.80 ± 0.36	20.89
	0.3	104	22.11 ± 0.16	6.52 ± 0.08	9.76 ± 0.42	20.55
	0.4	95	23.05 ± 0.21	6.75 ± 0.05	6.95 ± 0.57	21.42
	0.5	94	22.06 ± 0.35	6.55 ± 0.11	7.28 ± 0.38	20.51
HSWLR 3-5-3-3|9-9	0.5	136	23.29 ± 0.09	6.60 ± 0.11	7.74 ± 0.94	21.62
	0.6	135	22.88 ± 0.28	6.63 ± 0.08	6.78 ± 0.61	21.26
	0.7	136	22.92 ± 0.19	6.64 ± 0.06	7.44 ± 0.51	21.29
	0.8	136	22.97 ± 0.47	6.66 ± 0.06	7.53 ± 0.64	21.34
	0.9	135	24.45 ± 0.32	6.81 ± 0.14	7.15 ± 1.13	22.69
	1.0	136	22.72 ± 0.26	6.62 ± 0.08	6.95 ± 0.52	**21.11**
TSK 3-5-3-3	0.5	183	26.09 ± 2.50	9.39 ± 0.52	18.38 ± 1.52	24.42
	0.6	185	13.81 ± 0.97	7.87 ± 0.28	4.69 ± 0.90	13.22
	0.7	164	18.56 ± 3.77	10.50 ± 1.63	17.75 ± 2.90	17.75
	0.8	176	22.02 ± 1.93	8.68 ± 0.57	9.51 ± 0.63	20.69
	0.9	175	13.11 ± 4.21	7.27 ± 0.39	16.03 ± 0.49	**12.53**
	1.0	174	15.82 ± 0.64	6.77 ± 0.75	19.32 ± 0.94	14.92

- $Rvc = \frac{vc_r+1}{vc_m+1}$: relative velocity change, where vc_r and vc_m are the average velocity changes for the robot and the moving object respectively.

The results presented in the tables are the average and standard deviation values over five executions of the average values of the indicators over one execution. Again, we have added a new column to represent with a numerical value a possible definition of the error of a controller:

$$error = 0.9 \cdot \delta d + 0.1 \cdot \delta dev \quad (17)$$

For each data-driven algorithm in a table (Tables 17–21) the lowest error (best accuracy) has been marked in bold face. Moreover, in Figs. 8 and 9 the paths of the robot and the moving object for method TSK and $\phi = 0.5$ are shown for traces *spiral* (Table 20) and *zig-zag–chicane* (Table 21). The trajectories are represented by circular marks, and the higher the concentration of marks, the lower the speed. In order to visualize adequately both trajectories, the path of the robot has been shifted along the *x*-axis (Fig. 8) and the *y*-axis (Fig. 9). For example, in Fig. 8, the starting point of the robot (A_r) and the moving object (A_m) have the same *y* coordinate (as can be seen in the figure), but the *x* coordinate of the robot should be 1.5 m (d_{ref}) under the moving object *x* coordinate. The circular

Table 21
Trajectory: *zigzag–chicane*, weights = (1000, 60, 4), #Examples = 6435

Method labels	ϕ	#Rules	δd (cm)	δdev (°)	*Rvc*	Error
COR 3-5-3-3|9-9	0.0	135	28.82 ± 0.15	18.27 ± 0.14	5.49 ± 0.14	27.77
	0.1	118	27.70 ± 0.24	21.09 ± 0.44	5.04 ± 0.14	27.04
	0.2	108	25.51 ± 0.87	18.86 ± 0.31	5.98 ± 0.21	**24.85**
	0.3	121	28.07 ± 0.32	19.18 ± 0.40	4.72 ± 0.47	27.18
	0.4	95	30.72 ± 2.13	18.48 ± 0.42	5.73 ± 0.37	29.50
	0.5	104	27.21 ± 0.37	18.42 ± 0.14	5.99 ± 0.20	26.33
WCOR 3-5-3-3|9-9	0.0	134	26.33 ± 0.32	18.28 ± 0.32	6.10 ± 0.24	25.53
	0.1	127	27.41 ± 0.46	18.38 ± 0.45	5.69 ± 0.35	26.51
	0.2	120	25.50 ± 0.29	18.02 ± 0.18	5.88 ± 0.38	24.75
	0.3	104	23.62 ± 0.45	17.84 ± 0.11	5.65 ± 0.39	**23.04**
	0.4	95	25.80 ± 0.34	17.88 ± 0.20	6.02 ± 0.25	25.01
	0.5	94	25.13 ± 0.39	17.83 ± 0.27	5.68 ± 0.24	24.40
HSWLR 3-5-3-3|9-9	0.5	136	27.83 ± 0.51	18.27 ± 0.48	6.46 ± 0.20	26.87
	0.6	135	28.01 ± 0.59	18.29 ± 0.40	6.33 ± 0.18	27.04
	0.7	136	27.61 ± 0.48	18.47 ± 0.56	6.48 ± 0.37	**26.70**
	0.8	136	27.75 ± 0.65	18.37 ± 0.41	6.36 ± 0.37	26.81
	0.9	135	27.91 ± 0.57	18.00 ± 0.12	6.08 ± 0.29	26.92
	1.0	136	28.11 ± 0.48	18.05 ± 0.42	6.31 ± 0.27	27.10
TSK 3-5-3-3	0.5	183	20.43 ± 1.88	15.38 ± 0.74	13.49 ± 0.19	**19.93**
	0.6	185	26.96 ± 5.35	14.98 ± 1.04	6.92 ± 0.58	25.76
	0.7	164	33.55 ± 1.07	17.65 ± 0.67	9.96 ± 1.42	31.96
	0.8	176	33.76 ± 0.30	15.65 ± 0.44	8.37 ± 0.59	31.95
	0.9	175	22.22 ± 0.94	17.81 ± 0.51	13.47 ± 0.68	21.78
	1.0	174	35.47 ± 2.27	16.73 ± 0.66	9.26 ± 0.32	33.60

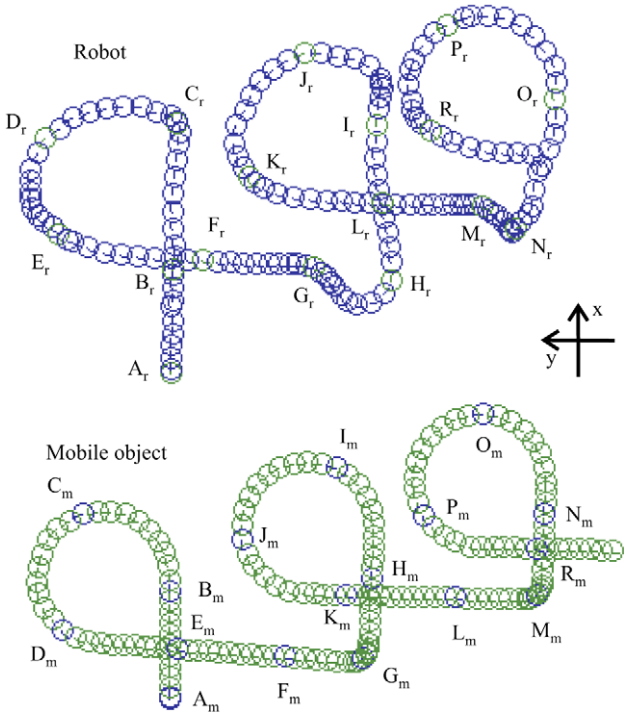


Fig. 8. Robot following a moving object with spiral-like trajectory.

marks that represent the trajectories have a frequency of two control cycles between them. Also, in Fig. 9 the *x* coordinate of the robot at the starting point (A_r) is the real one, but the *y* coordinate should be the same as the *y* coordinate of the starting point of the moving object (A_m).

To analyze the accuracies of the controllers for this behavior, it should be taken into account that the difficulty in placing the robot in the objective position is very high, as this objective position changes with the movement of the object. Moreover, the moving object frequently changes its linear velocity and heading direction, making impossible to obtain low values of δd and δdev. Both the robot and the moving object have the same kinematic characteristics. This means that if the moving object accelerates suddenly and reaches the highest speed, the robot will be unable to catch it. The practical implication is that in a situation like this, the errors in distance have to be always clearly over zero. The characteristics of the traces followed by the moving object are shown in Table 22.

Errors (Tables 17–21) show that if we choose the best representative of each learning methodology, and we compare that representative with the other representatives in the table, the best performance is always obtained by a TSK controller. The ranking in accuracy continues with WCOR, COR, and finally HSWLR. This occurs in all the traces but *chicane*, which is the easiest one. In that trace the values of the error are, in general, very low and the differences among methodologies are not meaningful. Also, the analysis of the influence of parameter ϕ over the error showed again that there was not a pattern that indicated a better ϕ value. Thus, if ϕ takes values in that interval, performance is not affected significantly.

Fig. 10 shows the trade-off between the average values of the error and the number of rules of each of the data-driven learning algorithms. Again, the graphic proves that, although TSK is the most accurate method, the number of rules is really high. For this behavior, the differences in the performance of TSK and WCOR are higher, but the same happens with the number of rules. We can conclude that WCOR is the most complete algorithm, with good accuracy and the lowest number of rules.

4.3. General rules for the design of behaviors

Once the study on the data-driven algorithms and the influence in the values of the parameters has been analyzed, it would be interesting to extract some general rules for the design of behaviors under the learning framework. We can point out the following:

- Three of the data-driven algorithms (COR, WCOR and TSK) obtain consistently a very good performance. If the designer requires high interpretability, COR should be selected. On the

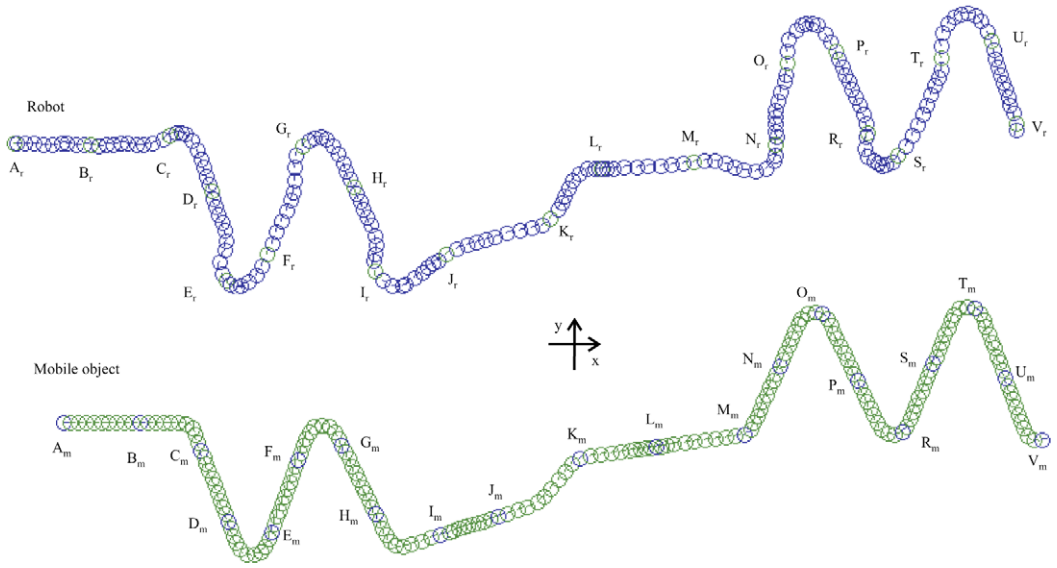


Fig. 9. Robot following a moving object with zigzag–chicane-like trajectory.

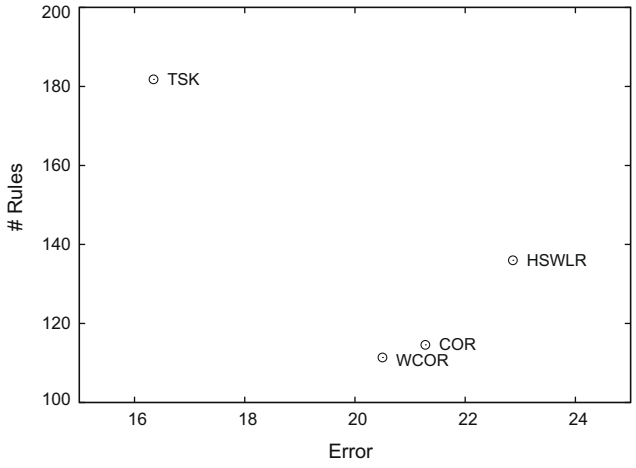


Fig. 10. Average values of the most accurate results for each environment and learning method for the moving object following behavior.

Table 22
Characteristics of the traces of the moving object.

Trace	Max. speed	Max. speed changes	Max. heading changes	Difficulty
zigzag	Medium	None	Very strong	Very high
chicane	High	Strong	Strong	Medium
loop	High	Medium	Medium	High
spiral	High	Strong	Medium	High
zigzag–chicane	High	Strong	Very strong	Very high

other hand, if accuracy is the main goal, TSK is the best candidate. Nevertheless, WCOR is the algorithm with the better trade-off between accuracy and interpretability.

- The granularities of the universes of discourse of the input variables are of great importance not only from the interpretability point of view (the higher the granularities, the higher the number of rules), but also for accuracy. In particular, accuracy is very sensible to the number of labels of the most significant input variables.
- The number of training examples, which have been generated with the dataset generation module, do not have a great influence on accuracy. Nevertheless a higher number of examples slightly reduces the number of rules. However, it also increases the learning time. From our point of view it is not useful to reduce the number of rules increasing the number of examples, as we have most adequate parameters to do that. In summary, a good choice is to try to generate a reduced number of examples.
- The best option to reduce the number of rules is to increase the value of ϕ, as it does not reduce the accuracy of the controller if the value is selected in the proposed intervals.
- The performance of the controllers is quite sensible to the values of the weights in SF (Eq. (1)). A good and very simple solution for the selection of the values of the weights consists, first, in making a ranking of the importance of the input variables. Then, a weight of one is assigned to the most important variable, and the next variables take weights ten times lower than the previous variable. Modifications around these values do not affect accuracy too much.

5. Conclusions

In this paper we have performed an exhaustive study on a set of data-driven evolutionary-based algorithms that cover a wide range of the accuracy/interpretability trade-off. Also, we have analyzed the influence of some design parameters both in accuracy and in the number of rules. The learning framework has been applied to two different behaviors to prove the applicability of the methodology for the learning of controllers in mobile robotics. The number of tests that have been implemented (344 for wall-following and 120 for moving object following behaviors) both in simulations and on a *Pioneer II AT* robot show, in general, very good accuracies for most of the combinations of algorithms and values of parameters. Taking into account all the analyzed data, some general rules for the design of behaviors under the learning framework have been extracted. The reliability of this design rules is high due to the number of tests, the number of combinations in the values of the parameters, and the results obtained on the real robot. One of the most interesting aspects of the methodology is the ability to directly run the learned controller on the real robot. This, together with the fact that the steps for

the design of behaviors have been clearly established, will facilitate the implementation of controllers in the growing field of service robots.

References

Alcalá, R., Alcalá-Fdez, J., Casillas, J., Cordón, O., & Herrera, F. (2006). Hybrid learning models to get the interpretability-accuracy trade-off in fuzzy modelling. *Soft Computing, 10*(9), 717–734.

Alcalá, R., Alcalá-Fdez, J., Casillas, J., Cordón, O., & Herrera, F. (2007). Local identification of prototypes for genetic learning of accurate TSK fuzzy rule-based systems. *International Journal of Intelligent Systems, 22*(9), 909–941.

Alcalá, R., Cano, J., Cordón, O., Herrera, F., Villar, P., & Zwir, I. (2003). Linguistic modeling with hierarchical systems of weighted linguistic rules. *International Journal of Approximate Reasoning, 32*(2–3), 187–215.

Alcalá, R., Casillas, J., Cordón, O., & Herrera, F. (2002). Improving simple linguistic fuzzy models by means of the weighted COR methodology. In *Proceedings of the 8th Ibero-American conference (AI) on advances in artificial intelligence – IBERAMIA 2002. LNAI* (Vol. 2527, pp. 294–302). Sevilla, Spain: Springer-Verlag.

Angle, C. (2008). Home robot explosion. PC Magazine. <http://blogs.spectrum.ieee.org/automaton/2008/01/14/irobot_founder_on_the_next_25.html> (January 2008).

Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. Oxford University Press.

Beom, H. R., & Cho, H. S. (1995). A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, 25*(3), 464–477.

Berlanga, A., Sanchis, A., Isasi, P., & Molina, J. M. (2000). A general learning co-evolution method to generalize autonomous robot navigation behavior. In *Proceedings of the congress on evolutionary computation, La Jolla, CA (USA)* (pp. 769–776).

Bonarini, A. (1997). Anytime learning and adaptation of structured fuzzy behaviors. *Adaptive Behavior, 5*, 281–315.

Bonissone, P. P., Subbu, R., Eklund, N., & Kiehl, T. R. (2006). Evolutionary algorithms + domain knowledge = real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation, 10*(3), 256–280.

Casillas, J., Cordón, O., de Viana, I. F., & Herrera, F. (2005). Learning cooperative linguistic fuzzy rules using the best–worst ant system algorithm. *International Journal of Intelligent Systems, 20*, 433–452.

Casillas, J., Cordón, O., & Herrera, F. (2002). COR: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 32*(4), 526–537.

Casillas, J., Cordón, O., & Herrera, F. (2002). Techniques for constructing intelligent systems. *Different approaches to induce cooperation in fuzzy linguistic models under the COR methodology*. Springer-Verlag.

Casillas, J., Cordón, O., Herrera, F., & Magdalena, L. (Eds.). (2003a). *Accuracy improvements in linguistic fuzzy modeling*. Heidelberg, Germany: Springer.

Casillas, J., Cordón, O., Herrera, F., & Magdalena, L. (Eds.). (2003b). *Interpretability issues in fuzzy modeling*. Heidelberg, Germany: Springer.

Chen, L.-H., & Chiang, C.-H. (2004). An intelligent control systems with a multi-objective self-exploration process. *Fuzzy Sets and Systems, 143*, 275–294.

Cho, J. S., & Park, D. J. (2000). Novel fuzzy logic control based on weighting of partially inconsistent rules using neural network. *Journal of Intelligent Fuzzy Systems, 8*, 99–110.

Cordón, O., del Jesus, M., Herrera, F., & Lozano, M. (1999). Mogul: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. *International Journal of Intelligent Systems, 14*(9), 1123–1153.

Cordón, O., & Herrera, F. (1999). A two-stage evolutionary process for designing tsk fuzzy rule-based systems. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, 29*(6), 703–715.

Cordón, O., & Herrera, F. (2001). Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems. *Fuzzy Sets and Systems, 118*(2), 235–255.

Cordón, O., Herrera, F., Hoffmann, F., & Magdalena, L. (2001). *Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases*. Singapore: World Scientific.

Dahl, T., & Giraud-Carrier, C. (2004). Evolution-inspired incremental development of complex autonomous intelligence. In *Proceeding of the 8th international conference on intelligent autonomous systems (IAS)* (pp. 395–402). The Netherlands: Amsterdam.

Floreano, D., & Mondada, F. (1998). Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks, 11*, 1461–1478.

Fuller, R. (2000). *Introduction to neuro-fuzzy systems*. Heidelberg, Germany: Springer-Verlag.

Gerkey, B., Vaughan, R., & Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics (ICAR), Coimbra (Portugal)* (pp. 317–323).

Gu, D., Hu, H., Reynolds, J., & Tsang, E. (2003). Ga-based learning in behaviour based robotics. In *Proceedings of the IEEE international symposium on computational intelligence in robotics and automation, Kobe (Japan)* (pp. 1521–1526).

Gu, D., Hu, H., & Spacek, L. (2003). Learning fuzzy logic controller for reactive robot behaviours. In *Proceedings of the IEEE/ASME international conference on advanced intelligent mechatronics (AIM 2003)* (pp. 46–51).

Hagras, H., Callaghan, V., & Collin, M. (2004). Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online fuzzy-genetic system. *Fuzzy Sets and Systems, 141*, 107–160.

Hoffmann, F. (2003). An overview on soft computing in behavior based robotics. In *Proceedings of the IFSA 2003* (pp. 544–551).

Howard, A., & Roy, N. (2003). The robotics data set repository (radish), <http://www.radish.sourceforge.net>.

Hui, N., Mahendar, V., & Pratihar, D. (2006). Time-optimal collision-free navigation of a car-like mobile robot using neuro-fuzzy approaches. *Fuzzy Sets and Systems, 157*, 2171–2204.

Izumi, K., Watanabe, K., & Jin, S.-H. (1999). Obstacle avoidance of mobile robot using fuzzy behavior-based control with module learning. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 454–459).

Kalmár, Z., Szepesvári, C., & Lörincz, A. (1998). Module-based reinforcement learning: Experiments with a real robot. *Machine Learning, 31*, 55–85.

Katagami, D., & Yamada, S. (2000). Interactive classifier system for real robot learning. In *IEEE international workshop on robot–human interaction (ROMAN-2000), Osaka (Japan)* (pp. 258–263).

Lee, K. J., & Zhang, B. T. (2000). Learning robot behaviors by evolving genetic programs. In *Proceedings of the 26th international conference on industrial electronics, control and instrumentation (IECON-2000)* (Vol. 4, pp. 2867–2872).

Lin, C. K. (2003). A reinforcement learning adaptative fuzzy controller for robots. *Fuzzy Sets and Systems, 137*, 339–352.

Miglino, O., Lund, H. H., & Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life, 2*(4), 417–434.

Mucientes, M., & Casillas, J. (2007). Quick design of fuzzy controllers with good interpretability in mobile robotics. *IEEE Transactions on Fuzzy Systems, 15*(4), 636–651.

Mucientes, M., Iglesias, R., Regueiro, C. V., Bugarín, A., & Barro, S. (2003). A fuzzy temporal rule-based velocity controller for mobile robotics. *Fuzzy Sets and Systems, 134*, 83–99.

Mucientes, M., Iglesias, R., Regueiro, C. V., Bugarín, A., Cariñena, P., & Barro, S. (2001). Fuzzy temporal rules for mobile robot guidance in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, 31*(3), 391–398.

Mucientes, M., Moreno, D. L., Bugarín, A., & Barro, S. (2006). Evolutionary learning of a fuzzy controller for wall-following behavior in mobile robotics. *Soft Computing, 10*(10), 881–889.

Mucientes, M., Moreno, D. L., Bugarín, A., & Barro, S. (2007). Design of a fuzzy controller in mobile robotics using genetic algorithms. *Applied Soft Computing, 7*, 540–546.

Nauck, D., Klawonn, F., & Kruse, R. (1997). *Fundations of neuro-fuzzy systems*. New York, NY, USA: John Wiley & Sons.

Nelson, A. L., Grant, E., Barlow, G., & White, M. (2003). Evolution of complex autonomous robot behaviors using competitive fitness. In *Proceedings of the IEEE international conference on integration of knowledge intensive multi-agent systems, Boston (USA)* (pp. 145–150).

Nozaki, K., Ishibuchi, H., & Tanaka, H. (1997). A simple but powerful heuristic method for generating fuzzy rules from numerical data. *Fuzzy Sets and Systems, 86*(3), 251–270.

Oh, C. K., & Barlow, G. J. (2004). Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming. In *Proceedings of the congress on evolutionary computation, Portland (USA)* (pp. 1538–1545).

Pal, N. R., & Pal, K. (1999). Handling of inconsistent rules with an extended model of fuzzy reasoning. *Journal of Intelligent and Fuzzy Systems, 7*, 55–73.

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

Shiah, S.-J., & Young, K.-Y. (2004). Robot motion classification from standpoint of learning control. *Fuzzy Sets and Systems, 144*, 285–296.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics, 15*(1), 116–132.

Takahashi, Y., & Asada, M. (2003). Multi-layered learning systems for vision-based behavior acquisition of a real mobile robot. In *Proceedings of SICE annual conference 2003* (pp. 2937–2942).

Thongchai, S. (2002). Behavior-based learning fuzzy rules for mobile robots. In *Proceedings of the American control conference, Anchorage, AK (USA)* (pp. 995–1000).

Tuci, E., Quinn, M., & Harvey, I. (2003). An evolutionary ecological approach to the study of learning behaviour using a robot based model. *Adaptive Behavior, 10*(3/4), 201–221.

Wang, Y., Huber, M., Papudesi, V., & Cook, D. (2003). User-guided reinforcement learning of robot assistive tasks for an intelligent environment. In *Proceedings of the IEEE/RJS international conference on intelligent robots and systems, Las Vegas (USA)* (pp. 424– 429).

Wang, L.-X., & Mendel, J. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics, 22*(6), 1414–1427.

Yamada, S. (2005). Evolutionary behavior learning for action-based environment modeling by a mobile robot. *Applied Soft Computing*, 245–257.

Zhou, C. (2002). Robot learning with ga-based fuzzy reinforcement learning agents. *Information Sciences, 145*, 45–68.