Learning Fuzzy Controllers in Mobile Robotics with Embedded Preprocessing

I. Rodríguez-Fdez*, M. Mucientes, A. Bugarín

Centro de Investigación en Tecnoloxías da Información (CITIUS), Universidade de Santiago de Compostela, SPAIN

Abstract

The automatic design of controllers for mobile robots usually requires two stages. In the first stage, sensorial data are preprocessed or transformed into high level and meaningful values of variables which are usually defined from expert knowledge. In the second stage, a machine learning technique is applied to obtain a controller that maps these high level variables to the control commands that are actually sent to the robot. This paper describes an algorithm that is able to embed the preprocessing stage into the learning stage in order to get controllers directly starting from sensorial raw data with no expert knowledge involved. Due to the high dimensionality of the sensorial data, this approach uses Quantified Fuzzy Rules (QFRs), that are able to transform low-level input variables into high-level input variables, reducing the dimensionality through summarization. The proposed learning algorithm, called Iterative Quantified Fuzzy Rule Learning (IQFRL), is based on genetic programming. IQFRL is able to learn rules with different structures, and can manage linguistic variables with multiple granularities. The algorithm has been tested with the implementation of the wall-following behavior both in several realistic simulated environments with different complexity and on a *Pioneer 3-AT* robot in two real environments. Results have been compared with several well-known learning algorithms combined with different data preprocessing techniques, showing that IQFRL exhibits a better and statistically significant performance. Moreover, three real world applications for which IQFRL plays a central role are also presented: path and object tracking with static and moving obstacles avoidance.

Keywords: mobile robotics, Quantified Fuzzy Rules, Iterative Rule Learning, Genetic Fuzzy System

1. Introduction

The control architecture of mobile robots usually includes a number of behaviors that are implemented as controllers, which are able to solve specific tasks such as motion planning, following a moving object, wall-following, avoiding collisions, etc. in real time. These behaviors are implemented as controllers whose outputs at each time point (control commands) depend on both the internal state of the robot and the environment in which it evolves. The robot sensors (e.g. laser range finders, sonars, cameras, etc.) are used in order to obtain the augmented state of the robot (internal state and environment). When the robot operates in real environments, both the data obtained by these sensors and the internal state of the robot present uncertainty or noise. Therefore, the use of mechanisms that manage them properly is necessary. The use of fuzzy rules is convenient to cope with this uncertainty, since it combines the interpretability and expressiveness of the rules with the ability of fuzzy logic for representing uncertainty.

The first step for designing controllers for mobile robots consists of the preprocessing of the raw sensor data: the low-level input variables obtained by the sensors are transformed into high-level variables that are significant for the behavior to be learned. Usually, expert knowledge is used for the definition of these high-level variables and the mapping from the sensorial data. After this preprocessing stage, machine learning algorithms can be used to automatically obtain the mapping from the high-level input variables to the robot control commands. This paper describes an algorithm that is able to perform the preprocessing stage embedded in the learning stage, thus avoiding the use of expert knowledge. Therefore, the mapping between low-level and high-level input variables is done automatically during the learning phase of the controller.

The data provided by the sensors is of high dimensionality. For example, a robot equipped with two laser range finders can generate over 720 low-level variables. However, in mobile robotics it is more common to work with sets or groupings of these variables, (e.g. "frontal sector") that are much more significant and relevant for the behavior. As a result, it is necessary to use a model that is capable of grouping low-level variables, thus reducing the dimensionality of the problem and providing meaningful descriptions. The model should provide propositions that are able to summarize the data with expressions like "part of the distances in the frontal sector are high". This kind of expressions can model the underlying knowledge in a better way than just using average, maximum or minimum values of sets of low level variables. Moreover, these expressions also include the definition of the set of low-level variables to be used. Since these propositions involve fuzzy quantifiers (e.g. "part"), they are called Quantified Fuzzy

*Corresponding author. Tel.: +34 881816392.
Email addresses: ismael.rodriguez@usc.es (I. Rodríguez-Fdez), manuel.mucientes@usc.es (M. Mucientes), alberto.bugarin.diz@usc.es (A. Bugarín)

Propositions (QFPs) [1]. QFP provide a formal model that is capable of modeling the knowledge involved in this grouping task.

Evolutionary algorithms have some characteristics that make them suitable for learning fuzzy rules. The well-known combination of evolutionary algorithms and fuzzy logic (genetic fuzzy systems) is one of the approaches that aims to manage the balance between accuracy and interpretability of the rules [2, 3]. As it was pointed out before, fuzzy rules can be composed of both conventional and QFPs (therefore, they will be referred to as QFRs). Furthermore, the transformation from low-level to high-level variables using QFPs produces a variable number of propositions in the antecedent of the rules. Therefore, genetic programming, where the structure of individuals is a tree of variable size derived from a context-free grammar, is here the most appropriate choice.

This paper describes an algorithm that is able to learn QFRs of variable structure for the design of controllers with embedded preprocessing in mobile robotics. This proposal, called Iterative Quantified Fuzzy Rule Learning (IQFRL), is based on the Iterative Rule Learning (IRL) approach and uses linguistic labels defined with unconstrained multiple granularity, i.e. without limiting the granularity levels. This proposal has been designed to solve control (regression) problems in mobile robotics having as input variables the internal state of the robot and the sensors data. Expert knowledge is only used to generate the training data for each of the situations of the task to be learned and, also, to define the context-free grammar that specifies the structure of the rules.

The main contributions of the paper are: (i) the proposed algorithm is able to learn using the state of the robot and the sensors data, with no preprocessing. Instead, the mapping between low-level variables and high-level variables is done embedded in the algorithm; (ii) the algorithm uses QFPs, a model able to summarize the low-level input data; (iii) moreover, IQFRL uses linguistic labels with unconstrained multiple granularity. With this approach, the interpretability of the membership functions used in the resulting rules is unaffected while the flexibility of representation remains. The proposal was validated in several simulated and real environments with the wall-following behavior. Results show a better and statistically significant performance of IQFRL over several combinations of well-known learning algorithms and preprocessing techniques. The approach was also tested in three real world behaviors that were built as a combination of controllers: path tracking with obstacles avoidance, object tracking with fixed obstacles avoidance, and object tracking with moving obstacle avoidance.

The paper is structured as follows: Section 2 summarizes recent work related with this proposal and Section 3 presents the QFRs model and its advantages in mobile robotics. Section 4 describes the IQFRL algorithm that has been used to learn the QFRs. Section 5 presents the obtained results, and Section 6 shows three real world applications of IQFRL in robotics. Finally, Section 7 points out the most relevant conclusions.

2. Related Work

The learning of controllers for autonomous robots has been dealt with by using different machine learning techniques. Among the most popular approaches can be found evolutionary algorithms [4, 5], neural networks [6] and reinforcement learning [7, 8]. Also hibridations of them, like evolutionary neural networks [9], reinforcement learning with evolutionary algorithms [10, 11], the widely used genetic fuzzy systems [12, 13, 14, 15, 16, 17, 18], or even more uncommon combinations like ant colony optimization with reinforcement learning [19] or differential evolution [20] or evolutionary group based particle swarm optimization [21] have been successfully applied. Furthermore, over the last few years, mobile robotic controllers have been getting some attention as a test case for the automatic design of type-2 fuzzy logic controllers [8, 5, 20].

An extensive use of expert knowledge is made in all of these approaches. In [12] 360 laser sensor beams are used as input data, and are heuristically combined into 8 sectors as inputs to the learning algorithm. On the other hand, in [9, 13, 14, 15, 16, 18, 19, 21] the input variables of the learning algorithm are defined by an expert. Moreover, in [13, 14, 16, 18, 20] the evaluation function of the evolutionary algorithm must be defined by an expert for each particular behavior. As in the latter case, the reinforcement learning approaches need the definition of an appropriate reward function using expert knowledge.

The approaches based on genetic fuzzy systems use different alternatives in the definition of the membership functions. In [10, 12, 16] the membership functions are defined heuristically. In [14, 15] labels have been uniformly distributed, but the granularity of each input variable is defined using expert knowledge. On the other hand, in [13, 17, 18, 19, 21] an approximative approach is used, i.e., different membership functions are learned for each rule, reducing the interpretability of the learned controller.

The main problem of learning behaviors using raw sensor input data is the curse of dimensionality. In [7], this issue has been managed from the reinforcement learning perspective, by using a probability density estimation of the joint space of states. Among all the approaches based on evolutionary algorithms, only in [4] no expert knowledge has been taken into account. In this work, the number of sensors and their position are learned from a reduced number of sensors.

In [22] a Genetic Cooperative-Competitive Learning (GCCL) approach was presented. The proposal learns knowledge bases without preprocessing raw data, but the rules involved approximative labels while the IQFRL proposal uses unconstrained multiple granularity. Moreover, in this approach it is difficult to adjust the balance between cooperation and competition, which is typical when learning rules in GCCL. As a result, the obtained rules where quite specific and the performance of the behavior was not comparable to other proposals based on expert knowledge.

3. Quantified Fuzzy Rules (QFRs)

3.1. QFRs for robotics

Machine learning techniques in mobile robotics are used to obtain the mapping from inputs to outputs (control commands). In general, two categories can be established for the input variables:

- High-level input variables: variables that provide, by themselves, information that is relevant and meaningful to the expert for modeling the system (e.g. the linear velocity of the robot, or the right-hand distance from the robot to a wall).
- Low-level input variables: variables that do not provide by themselves information for the expert to model the system (e.g. a single distance measure provided by a sensor). Relevance of these variables emerge when they are grouped into more significant sets of variables. For example, the control actions cannot be decided by simply analyzing the individual distance values provided by each beam of a laser range finder, since noisy measurements or gaps between objects (very frequent in cluttered environments) may occur. Instead, more significant variables and models involving complex groupings and structures are used.

Usually, high-level variables, or sectors, consisting of a set of laser beam measures instead of the beam measures themselves (e.g., right distance, frontal distance, etc.) are used in mobile robotics. The low-level input variables are transformed into high-level input variables in a preprocessing stage previous to the learning of the controller. Traditionally, this transformation and the resulting high-level input variables are defined using expert knowledge. Doing this preprocessing automatically during the learning phase demands a model that groups the low-level input variables in an expressive and meaningful way. Within this context Quantified Fuzzy Propositions (QFPs) such as *"part of the distances of the frontal sector are low"* are useful for representing relevant knowledge for the experts and therefore for performing intelligent control. Modeling with QFPs as in the previous example demands the definition of several elements:

- *part*: how many distances of the frontal sector must be low?
- *frontal sector*: which beams belong to the frontal sector?
- *low*: what is the actual semantics of low?

This example clearly sets out the need to use propositions that are different from the conventional ones. The use of QFPs in robotics eliminates the need of expert knowledge in two ways: i) the preprocessing of the low-level variables can be embedded in the learning stage; ii) the definition of the high-level variables obtained from low-level variables is done automatically, also during the learning stage. In this paper QFPs are used for representing knowledge about high-level variables that are defined as the grouping of low-level variables. Conventional fuzzy propositions are also used to represent conventional high-level variables, i.e., high-level variables not related to low-level ones (e.g. velocity).

IF *part of distances of FRONTAL SECTOR are LOW* and (1)
...
velocity is HIGH (2)
THEN *vlin is VERY LOW* and *vang is TURN LEFT*

Figure 1: An example of QFR to model the behavior of a mobile robot.

3.2. QFRs model

An example of a QFR is shown in Fig. 1, involving both QFPs (1) and conventional ones (2); the outputs of the rule are also fuzzy sets. In order to determine the degree to which the output of the rule will be applied, it is necessary to reason about the propositions (using, for example, the Mamdani's reasoning scheme).

The general expression for QFPs in this case is:

$$d\,(h)\ \textit{is}\ F_d^i\ \textit{in}\ Q^i\ \textit{of}\ F_b^i \tag{3}$$

where, for each i=1, ..., g_b^{max} (g_b^{max} being the maximum possible number of sectors of distances):

- $d\,(h)$ is the signal. In this example, it represents the distance measured by beam h.
- F_d^i is a linguistic value for variable $d\,(h)$ (e.g., *"low"*).
- Q^i is a (spatial, defined in the laser beam domain) fuzzy quantifier (e.g., *"part"*).
- F_b^i is a fuzzy set in the laser beam domain (e.g., the *"frontal sector"*).

Evaluation of the Degree of Fulfillment (*DOF*) for QFP (Eq. 3) is carried out using Zadeh's quantification model for proportional quantifiers (such as "most of", "part of", ...) [23]. This model allows to consider non-persistence, partial persistence and total persistence situations for the event "$d\,(h)$ is F_d^i" in the range of laser beams (spatial interval F_b^i). Therefore, for the considered example, it is possible to make a total or partial assessment on how many distances should be low, in order to decide the corresponding control action. This is a relevant feature of this model, since it allows to consider partial, single or total fulfillment of an event within the laser beams set.

The number of analyzed sectors of distances and their definition may vary for each of the rules. There can be very generic rules that only need to evaluate a single sector consisting of many laser beams, while other rules may need a finer granularity, with more specific laser sectors. Moreover, the rules may require a mix of QFPs and standard fuzzy propositions (for conventional high-level variables). Therefore, the automatic learning of QFRs demands an algorithm with the capability of managing rules with different structures.

4. Iterative Quantified Fuzzy Rule Learning of Controllers

4.1. Evolutionary learning of Knowledge Bases

Evolutionary learning methods follow two approaches in order to encode rules within a population of individuals [3, 24]:

- Pittsburgh approach: each individual represents the entire rule base.
- Michigan, IRL [25], and GCCL [26]: each individual codifies a rule. The learned rule base is the result of combining several individuals. The way in which the individuals interact during the learning process defines these three different approaches.

The discussion is focused on those approaches for which an individual represents a rule, discarding the Michigan approach as it is used in reinforcement learning problems in which the reward from the environment needs to be maximized [27]. Therefore, the IRL and GCCL approaches are analyzed.

In the IRL approach, the individuals compete among them but only a single rule is learned for each run (epoch) of the evolutionary algorithm. After each sequence of iterations, the best rule is selected and added to the final rule base. The selected rule must be penalized in order to induce niche formation in the search space. A common way to penalize the obtained rules is to delete the training examples that have been covered by the set of rules in the final rule base. The final step of the IRL approach is to check whether the obtained set of rules is a complete knowledge base. In the case it is not, the process is repeated. A weak point of this approach is that the cooperation among rules is not taken into account when a rule is evaluated. For example, a new rule could be added to the final rule base, deteriorating the behavior of the whole rule base over a set of examples that were already covered. The cooperation among rules can be improved with a posterior rules selection process.

In the GCCL approach the entire population codifies the rule base. That is, rules evolve together but competing among them to obtain the higher fitness. For this type of algorithm it is fundamental to include a mechanism to maintain the diversity of the population (niche induction). This mechanism must warrant that individuals of the same niche compete among themselves, but also has to avoid deleting those weak individuals that occupy a niche that remains uncovered. This is usually done using token competition [24].

Although GCCL works well for classification problems [1], the same does not occur for regression problems [22], mostly due to the difficulty of achieving in this realm an adequate balance between cooperation and competition. It is frequent in regression that an individual tries to capture examples seized by other individual, improving the performance on many of the examples, but decreasing the accuracy on a few ones. In subsequent iterations, new and more specific individuals replace the rule that was weakened. As a result, the individuals improve their individual fitness, but the performance of the knowledge base does not increase. In particular, for mobile robotics, the obtained knowledge bases over-fit the training data due to a *polarization* effect of the rule base: few very general rules and many very specific rules. Moreover, many times, the errors of the individual rules compensate each other, generating a good output of the rule base over the training data, but not on test data.

```
1: $KB_{cur} := \varnothing$
2: repeat
3:     $it := 0$
4:     $equal_{ind} := 0$
5:     Initialization
6:     Evaluation
7:     repeat
8:         Selection
9:         Crossover and Mutation
10:        Evaluation
11:        Replacement
12:        if $best_{ind}^{it-1} = best_{ind}^{it}$ then
13:            $equal_{ind} := equal_{ind} + 1$
14:        else
15:            $equal_{ind} := 0$
16:        end if
17:        $it := it + 1$
18:    until $(it \geq it_{min} \wedge equal_{ind} \geq it_{check}) \vee (it \geq it_{max})$
19:    $KB_{cur} := KB_{cur} \cup best_{ind}$
20:    $uncov_{ex} := uncov_{ex} - cov_{ex}$
21: until $uncov_{ex} = \varnothing$
```

Figure 2: IQFRL algorithm.

This proposal, called IQFRL (Iterative Quantified Fuzzy Rule Learning), is based on IRL. The learning process is divided into epochs (set of iterations), and at the end of each epoch a new QFR (Sec. 3.2) is obtained. The following sections describe each of the stages of the algorithm (Fig. 2).

4.2. Examples and Grammar

The learning process is based on a set of training examples. In mobile robotics, each example can be composed of several variables that define the state of the robot (position, orientation, linear and angular velocity, etc.), and the data measured by the sensors. If the robot is equipped with laser range finders, the sensors data are vectors of distances. A laser range finder provides the distances to the closest obstacle in each direction (Fig. 3) with a given angular resolution (number of degrees between two consecutive beams). In this paper, each example e^l is represented by a tuple:

$$e^l = (d(1), \ldots, d(N_b), velocity, vlin, vang) \tag{4}$$

where $d(h)$ is the distance measured by beam h, N_b is the number of beams (e.g. 722 for a robot equipped with two Sick LMS200 laser range scanners as in Fig. 3), *velocity* is the measured linear velocity of the robot, and *vlin* and *vang* are the

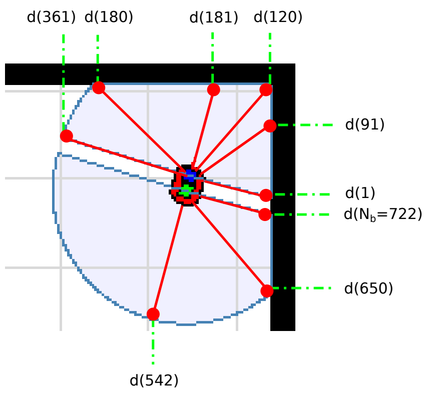


Figure 3: Some of the distances measured by a robot equipped with two laser range finders.

output variables (control commands for the linear and angular velocities respectively).

The individuals in the population include both conventional propositions and QFPs (Sec. 3.2). Also, the number of relevant inputs can be different. Therefore, genetic programming is the most appropriate approach, as each individual is a tree of variable size. In order to generate valid individuals of the population, and to produce right structures for the individuals after crossover and mutation, some constraints have to be added. With a context-free grammar all the valid structures of a tree (genotype) in the population can be defined in a compact form. A context-free grammar is a quadruple (V, Σ, P, S), where V is a finite set of variables, Σ is a finite set of terminal symbols, P is a finite set of rules or productions, and S the start symbol.

The basic grammar is described in Fig. 4. As usual, different productions for the same variable are separated by symbol "|". Fig. 5 represents a typical chromosome generated with this context-free grammar. Terminal symbols (leaves of the tree) are represented by ellipses, and variables as rectangles. There are two different types of antecedents:

- The sector antecedent. Consecutive beams are grouped into sectors in order to generate more general (high-level) variables (frontal distance, right distance, etc.). This type of antecedent is defined by the terminal symbols F_d, F_b and Q: i) the linguistic label F_d represents the measured distances (*HIGH* in Fig. 1, prop. 1); ii) F_b is the linguistic label that defines the sector, i.e., which beams belong to the sector ($FRONTAL\ SECTOR$ in Fig. 1, prop. 1); iii) Q is the quantifier (*part* in Fig. 1, prop. 1).
- The measured linear velocity of the antecedent is defined by the F_v linguistic label.

Finally, F_{lv} and F_{av} are the linguistic labels of the linear and angular velocity control commands respectively, which are the consequents of the rule.

The linguistic labels of the antecedent (F_v, F_d, F_b) are defined using a multiple granularity approach. The universe of discourse of a variable is divided into a different number of equally spaced labels for each granularity. Specifically, a granularity g^i_{var} divides the variable *var* in *i* uniformly spaced labels, i.e., $A^i_{var} = \{A^{i,1}_{var}, ..., A^{i,i}_{var}\}$. Fig. 6 shows a partitioning of up to granularity five. On the other hand, the linguistic labels of the consequents (F_{lv}, F_{av}) are defined using a single granularity approach[1].

- V = { rule, antecedent, consequent, sector }
- Σ = { F_{lv}, F_{av}, F_v, F_d, F_b, Q }
- S = rule
- P:
  1. rule $\longrightarrow$ antecedent consequent
  2. antecedent $\longrightarrow$ sector F_v | sector
  3. consequent $\longrightarrow$ F_{lv} F_{av}
  4. sector $\longrightarrow$ F_d Q F_b sector | F_d Q F_b

Figure 4: Basic context-free grammar for controllers in robotics.

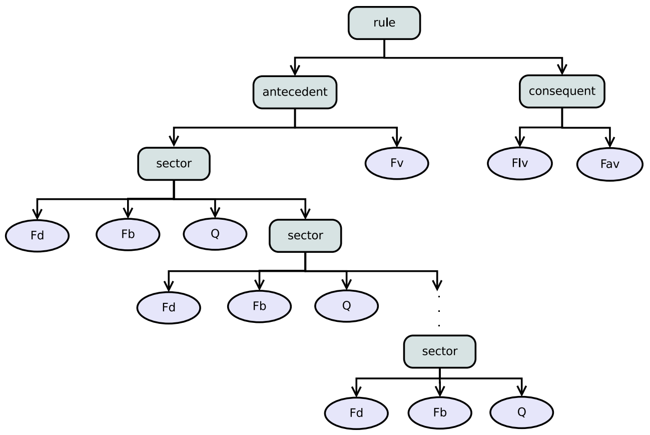


Figure 5: An individual representing a QFR that models the behavior of a robot.

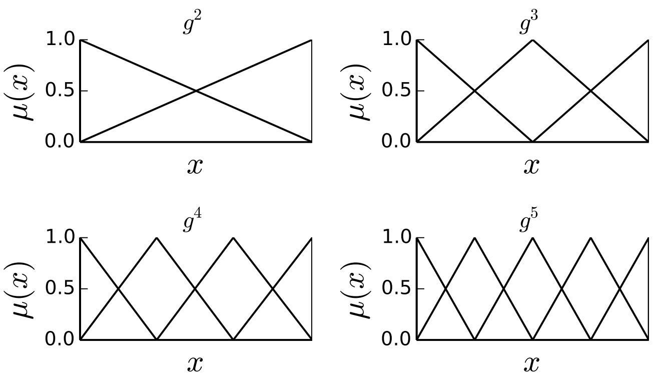
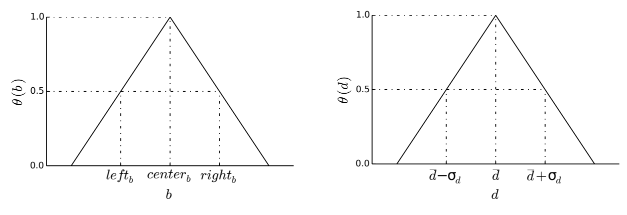


Figure 6: Multiple granularity approach from g^2_x to g^5_x.

[1]Multiple granularity makes no sense if the labels are defined as singletons, which is the usual choice for the output variables in control applications.

```
Require: $mask_{var}$
1:  $i := g^1_{var}$
2:  $result := \varnothing$
3:  loop
4:      for all $j \in [1, i]$ do
5:          if support($mask_{var}$) $\geq$ support($A^{i,\,j}_{var}$) then
6:              if similarity($mask_{var}$, $A^{i,\,j}_{var}$) >
                similarity($mask_{var}$, $result$) then
7:                  $result := A^{i,\,j}_{var}$
8:              end if
9:          else
10:             break loop
11:         end if
12:     end for
13:     $i := i + 1$
14: end loop
15: return $result$
```

Figure 7: Function that searches for the most similar label to $mask_{var}$.



(a) $mask_b$ (b) $mask_d$

Figure 8: $mask_{var}$ representations for beam (b) and distance (d) variables.

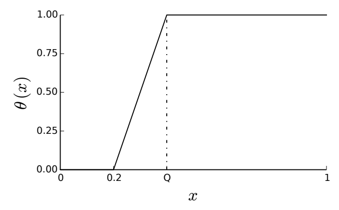


Figure 9: Example of a definition of the quantified label Q.

4.3. Initialization

An individual (Fig. 5) is generated for each example in the training set. The consequent part (F_{lv} and F_{av}) is initialized as $F_{var} = A^{g_{var},\,\beta}_{var}$ where $\beta = argmax_j\, \mu^{g_{var},\,j}_{var}\left(e^l\right)$, i.e., the label with the largest membership value for the example.

The initialization of the antecedent part of a rule requires obtaining the most similar linguistic label to a given fuzzy membership function (which is called mask label). As the maximum granularity of the linguistic labels in the antecedent part of a rule is not limited, the function `maskToLabel` (Fig. 7) is applied to obtain the most appropriate linguistic label. This function uses a similarity measure defined as [28]:

$$similarity(F_\phi,\; F_\psi) = 1 - \frac{\sum_{x \in X} |\mu_\phi(x) - \mu_\psi(x)|}{|X|} \tag{5}$$

where F_ϕ and F_ψ are the labels being compared and X is a finite set of points x uniformly distributed on the support of $\phi \cup \psi$.

The `maskToLabel` function (Fig. 7) receives a triangular membership function ($mask_{var}$) and searches for the label $A^{i,\,j}_{var}$ with the highest similarity (Eq. 5, line 6) with less or equal support (line 5), starting from g^1_{var} (line 1).

For the initialization of the quantified propositions (sectors), the distances measured in the example are divided into groups of consecutive laser beams whose deviation does not exceed a certain threshold (σ_{bd}). Each group represents a sector that is going to be included in the individual. Afterwards, for each of the previously obtained sectors, the components (F_b, F_d and Q) are calculated:

1. $F_b = maskToLabel(mask_b)$, with $mask_b = (left_b,\; center_b,\; right_b)$ where $left_b$ is the lower beam of the group, $right_b$ is the higher beam, $center_b$ is the middle beam and the following properties are satisfied: $\mu(left_b) = \mu(right_b) = 0.5$ and $\mu(center_b) = 1$ as shown in Fig. 8(a).
2. $F_d = maskToLabel(mask_d)$, with $mask_d = (\bar{d} - \sigma_d,\; \bar{d},\; \bar{d} + \sigma_d)$ where $\bar{d}$ is the mean of the distances measured by the beams of the group, σ_d is the standard deviation of these distances and the following properties are satisfied: $\mu(\bar{d} - \sigma_d) = \mu(\bar{d} + \sigma_d) = 0.5$ and $\mu(\bar{d}) = 1$ as shown in Fig. 8(b).
3. Q (Fig. 9) is calculated as the percentage of beams of the sector ($h \in F_b$) that fulfill F_d:

$$Q = \frac{\sum_{h \in F_b} min\left(\mu_{F_d}(d(h)),\; \mu_{F_b}(h)\right)}{\sum_{h \in F_b} \mu_{F_b}(h)} \tag{6}$$

Finally, the velocity antecedent F_v is initialized as $F_v = A^{g^i_v,\,\beta}_v$ where $\beta = argmax_j\; \mu^{g^i_v,\,j}_v\;(e^l)$ and g^i_v is the granularity that satisfies that two consecutive linguistic labels have a separation of σ_v, where σ_v is a threshold of the velocity deviation.

4.4. Evaluation

The fitness of an individual of the population is calculated as follows. Firstly, it is necessary to estimate the probability that an example e^l matches the output (C_j) associated to the j-th individual rule:

$$P\left(C_j \mid e^l\right) = \exp\left(-\frac{error^l_j}{ME}\right) \tag{7}$$

where ME is a parameter that defines the meaningful error and $error^l_j$ is the difference between output C_j and the output codified in the example:

$$error^l_j = \sum_k \left(\frac{y^l_k - c_{j,\,k}}{max_k - min_k}\right)^2 \tag{8}$$

where y^l_k is the value of the k-th output variable of example e^l, $c_{j,\,k}$ is the output of the k-th output variable associated to individual j, and max_k and min_k are the maximum and minimum values of output variable k. In regression problems, there can be several consequents that are different from the one codified in the example, but that produce small errors, i.e., that are very similar to the desired output. Thus, $P\left(C_j \mid e^l\right)$ can be interpreted as a normal distribution with covariance ME, and $error_j^l$ is the square of the difference between the mean (output codified in the example) and the output value proposed in the rule codified by the individual.

In an IRL approach, $C_j = C_{R_j}$, i.e., the output coded in individual j is the output associated to rule j. The fitness of an individual in the population is calculated as the combination of two values. On one hand, the accuracy with which the individual covers the examples, called confidence. On the other hand, the ability of generalization of the rule, called support. The confidence can be defined as:

$$confidence = \frac{\rho_u}{\sum_l DOF_j(e_u^l)} \tag{9}$$

where $DOF_j(e_u^l)$ is the degree of fulfillment of e_u^l for rule j, and $e_u^l \in uncov_{ex}$, where $uncov_{ex}$ is defined as:

$$uncov_{ex} = \{e^l \ : \ DOF_{KB_{cur}}(e^l) < DOF_{min}\} \tag{10}$$

i.e., the set of examples that are covered with a degree of fulfillment below DOF_{min} by the current final knowledge base (KB_{cur}) (line 19, Fig. 2), and ρ_u can be defined as:

$$\begin{aligned}\rho_u = \sum_l DOF_j(e_u^l) \ : \ P\left(C_j \mid e_u^l\right) > P_{min} \\ and\ DOF_j(e_u^l) > DOF_{min}\end{aligned} \tag{11}$$

where P_{min} is the minimum admissible accuracy. Therefore, the higher the accuracy over the examples covered by the rule (and not covered yet by the current knowledge base), the higher the confidence. Support is calculated as:

$$support = \frac{\rho_u}{\#uncov_{ex}} \tag{12}$$

Thus, support measures the percentage of examples that are covered with accuracy, related to the total number of uncovered examples. Finally, $fitness$ is defined as a linear combination of both values:

$$fitness = \alpha_f \cdot confidence + (1 - \alpha_f) \cdot support \tag{13}$$

which represents the strength of an individual over the set of examples in $uncov_{ex}$. $\alpha_f \in [0,\ 1]$ is a parameter that codifies the trade-off between accuracy and generalization of the rule.

4.5. Crossover

The matching of the pairs of individuals that are going to be crossed is implemented following a probability distribution defined as:

$$P_{close}(\alpha,\ \beta) = 1 \ - \ \frac{\sum_{k=1}^{N_c}\left(\frac{c_{\alpha,\ k} \ - \ c_{\beta,\ k}}{max_k \ - \ min_k}\right)^2}{N_c} \tag{14}$$

where $c_{\alpha,\ k}$ ($c_{\beta,\ k}$) is the value of the k-th output variable of individual α (β), and N_c is the number of consequents. With this probability distribution, the algorithm selects with higher probability mates that have similar consequents. The

```
Require: $ind_\alpha$, $ind_\beta$
 1: $a_\alpha = a_\beta = \varnothing$
 2: $N_a = g_b^{max} + 1$
 3: repeat
 4:    $m = random \in [1,\ N_a]$
 5:    if $m$ is a sector then
 6:       $a_\alpha = argmax_r\ similarity(F_{b,\ r},\ A_b^{g_b^{max},m}) \geq 0 : \forall r \in ind_\alpha$
 7:       $a_\beta = argmax_r\ similarity(F_{b,\ r},\ A_b^{g_b^{max},m}) \geq 0 : \forall r \in ind_\beta$
 8:    else
 9:       $a_\alpha = F_v \in ind_\alpha$
10:       $a_\beta = F_v \in ind_\beta$
11:    end if
12: until $(a_\alpha \neq \varnothing) \vee (a_\beta \neq \varnothing)$
```

Figure 10: Selection of antecedents for crossover.

objective is to extract information on which propositions of the antecedent part of the rules are important, and which are not. Crossover has been designed to generate more general individuals, as the initialization of the population produces very specific rules. The crossover operator generates two offsprings:

$$\begin{aligned} offspring_1 &= crossover(ind_i,\ ind_j) \\ offspring_2 &= crossover(ind_j,\ ind_i) \end{aligned} \tag{15}$$

This operator modifies a single proposition in antecedent part of the rule. As individuals have a variable number of antecedents, the total number of propositions can be different for two individuals. Moreover, the propositions can be defined using different granularities. Therefore, the first step is to select the propositions (one for each individual) to be crossed between both individuals (Fig. 10) as follows:

1. Get the most specific granularity of the sectors of the individuals to cross (g_b^{max}). Then, an antecedent $m \in [1,\ N_a]$ is selected, where N_a is g_b^{max} plus one, due to the velocity proposition.
2. Check the existence of this antecedent in both individuals, according to the following criteria:
   (a) If the antecedent m is a sector, then calculate for each proposition of each individual the similarity between the definition of the sector for the proposition and the linguistic label that defines sector m. Finally, select for each individual the proposition with the highest similarity.
   (b) If the antecedent m is the velocity, then the corresponding proposition is F_v (in case it exists).

Once the propositions to be crossed have been selected, an operation must be picked depending on the existence of the antecedent in both parents (table 1):

- If the proposition does not exist in the first individual but exists in the second one, then the proposition of the second individual is copied to the first one, as this proposition could be meaningful.

Table 1: Crossover operations

Individual 1	Individual 2	Action
no	yes	copy proposition from individual 2 to 1
yes	no	delete proposition in individual 1
yes	yes	combine propositions

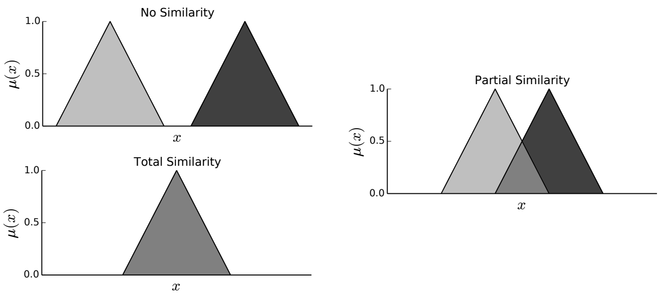


Figure 11: Different possibilities of similarity for the labels of equal proposition of two individuals used in the crossover operator.

- If the situation is the opposite to the previous one, then the proposition of the first individual is deleted, as it might be not important.
- If the proposition exists in both individuals, then both propositions are combined in order to obtain a proposition that generalizes both antecedents.

In this last case, the combination of propositions is done by taking into account the degree of similarity (Eq. 5) between them (Fig. 11). If the proposition is of type sector, the similarity takes into account both F_b and F_d labels. Only when both similarities are partial, the propositions are merged:

- If there is no similarity, then the propositions correspond to different situations. For example, *"the distance is high in part of the frontal sector"* and *"the distance is low in part of the frontal sector"*. This means that the proposition of the first individual might not contain meaningful information and it could be deleted to generalize the rule. For example, both individuals have the proposition *"the distance is high in part of the frontal sector"*.
- If the similarity is total, then, in order to obtain a new individual with different antecedents, the proposition is eliminated.
- Finally, if the similarity is partial, then the propositions are merged in order to obtain a new one that combines the information provided by the two original propositions. For example, *"the distance is high in part of the frontal sector"* and *"the distance is medium-high in part of the frontal sector"*. Therefore, the individual is generalized. The merge action is defined as the process of finding the label with the highest possible granularity that has some similarity with the labels of both original propositions. This is done for both F_b and F_d labels. Q is calculated as the minimum Q of both individuals.

4.6. Mutation

If crossover is not performed, both individuals are mutated. Mutation implements two different strategies (Fig. 12): generalize or specialize a rule. The higher the value of confidence (Eq. 9), the higher the probability to generalize the rule by mutation. This occurs with rules that cover their examples with high accuracy and that could be modified to cover other examples. On the contrary, when the confidence of the individual is low, this means that it is covering some of its examples with a low performance. In order to improve the rule some of the examples that are currently covered should be discarded in order to get a more specific rule.

For generalization, the following steps are performed:

1. Select an example $e^{sel} \in uncov^j_{ex}$, where $uncov^j_{ex} = \{e^l_u : DOF_j(e^l_u) < DOF_{min}\}$, i.e. the set of examples that belong to $uncov_{ex}$ and are not covered by individual j. The example is selected with a probability distribution given by $P\left(C_j \mid e^l_u\right)$ (Eq. 7). The higher the similarity between the output of the example and the consequent of rule j, the higher the probability of being selected.
2. The individual is modified in order to cover e^{sel}. Therefore, all the propositions that are not covering the example (those with $\mu_{prop}\left(e^{sel}\right) < DOF_{min}$) are selected for mutation.
    (a) For sector propositions (Eq. 1), there are three different ways in which the proposition can be modified: F_d, F_b, and Q. The modification is selected among the three possibilities, with a probability proportional to the $\mu_{prop}\left(e^{sel}\right)$ value after applying each one.
        i. F_d and F_b are generalized choosing the most similar label in the adjacent partition with lower granularity. The process is repeated until $\mu_{prop}\left(e^{sel}\right) \geq DOF_{min}$.
        ii. On the other hand, Q is decreased until $\mu_{prop}\left(e^{sel}\right) \geq DOF_{min}$.
    (b) For velocity propositions (Eq. 2), generalization is done choosing the most similar label in the adjacent partition with lower granularity until $\mu_{prop}\left(e^{sel}\right) > DOF_{min}$.

For specialization, the process is equivalent:

1. Select an example $e^{sel} \in cov^j_{ex}$, where $cov^j_{ex} = \{e^l_u : DOF_j(e^l_u) > DOF_{min}\}$, i.e. the set of examples that belong to $uncov_{ex}$ and are covered by individual j. The example is selected with a probability distribution that is inversely proportional to $P\left(C_j \mid e^l_u\right)$ (Eq. 7). The higher the similarity between the output of the example and the consequent of rule j, the lower the probability of being selected.
2. Only one proposition needs to be modified to specialize the individual. This proposition is selected randomly.
    (a) For sector propositions there are, again, three different ways in which the proposition can be modified: F_d, F_b, and Q. The modification

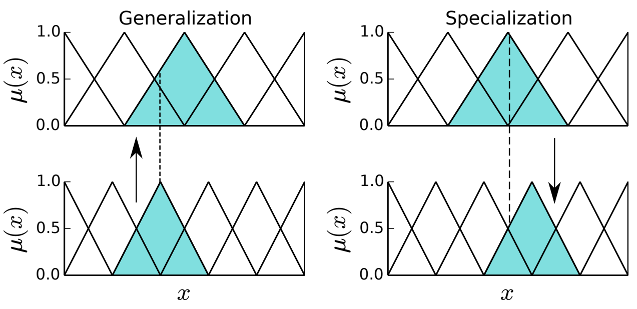


Figure 12: The strategies used for mutation for variables d, b and v.

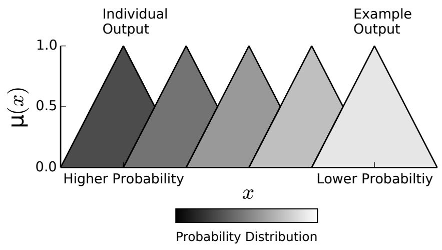


Figure 13: Probability distribution example for consequent mutation. Labels closest to the individual output have higher probability to be selected.

is selected among these three possibilities, with a probability that is inversely proportional to the $\mu_{prop}\left(e^{sel}\right)$ value after applying each one.

i. F_d and F_b are specialized, choosing the most similar label in the adjacent partition with higher granularity. The process is repeated until $\mu_{prop}\left(e^{sel}\right) < DOF_{min}$.

ii. On the other hand, Q is increased until $\mu_{prop}\left(e^{sel}\right) < DOF_{min}$.

(b) For velocity propositions, specialization is done by choosing the most similar label in the adjacent partition with higher granularity until $\mu_{prop}\left(e^{sel}\right) < DOF_{min}$.

Finally, once the antecedent is mutated, the consequent also mutates. Again, this mutation requires the selection of an example. If generalization was selected for the mutation of the antecedent, then the example will be e^{sel}. On the other hand, for specialization an example is randomly selected from those currently in cov_{ex}^{j}. For each variable in the consequent part of the rule, the label of the individual is modified selecting a label following a probability distribution (Fig. 13):

$$P\left(A_{var}^{g_{var},\gamma} \mid A_{var}^{g_{var},\alpha}, A_{var}^{g_{var},\beta}\right) = 1 - \frac{|\alpha - \gamma|}{|\alpha - \beta| + 1} \quad (16)$$

where $A_{var}^{g_{var},\alpha}$ is the label of each of the consequents of the individual, $A_{var}^{g_{var},\beta}$ is the label with the largest membership value for e^{sel} and $A_{var}^{g_{var},\gamma}$ is a label between them. Thus, the labels closer to the label of the individual have a higher probability to be selected, while the labels closer to the example label have a lower one.

4.7. Selection and replacement

Selection has been implemented following the binary tournament strategy. Replacement follows an steady-state approach. The new individuals and those of the previous population are joined, and the best pop_{max} individuals are selected for the next population.

4.8. Epoch loop

An epoch is a set of iterations at the end of which a new rule is added to KB_{cur}. The stopping criterion of each epoch (inner loop in Fig. 2) is the number of iterations, but this limit varies according to the following criteria: once the number of iterations (it) reaches it_{min}, the algorithm stops if there are it_{check} consecutive iterations (counted by $equal_{ind}$) with no change in the best individual ($best_{ind}$). If the number of iterations reaches the maximum (it_{max}), then the algorithm stops regardless of the previous condition.

When the epoch ends, the rule defined in $best_{ind}$ is added to KB_{cur}. Moreover, the examples that are covered with accuracy (according to the criterion in Eq. 11) are marked as covered by the algorithm (line 20, Fig. 2). Finally, the algorithm stops when there are no uncovered examples.

4.9. Rule subset selection

After the end of the iterative part of the algorithm, the performance of the obtained rule base can be improved selecting a subset of rules with better cooperation among them. The rule selection algorithm described in [1] has been used. The rule selection process has the following steps:

1. Generate $\#R_{gp}$ rule bases, where $\#R_{gp}$ is the number of rules of the population obtained by the IQFRL algorithm (RB_{gp}) Each rule base is coded as: $RB_i = r_1^i \cdots r_{\#R_{gp}}^i$, with:

$$r_j^i = \begin{cases} 0, & if\ j > i \\ 1, & if\ j \leq i \end{cases} \quad (17)$$

   where r_j^i indicates if the j-th rule of RB_{gp} is included ($r_j^i = 1$) or not ($r_j^i = 0$) in RB_i. With this codification, RB_i will contain the best i rules of RB_{gp}, as these rules have been ranked in decreasing order of their individual fitness. Notice that $RB_{\#R_{gp}}$ is RB_{gp}
2. Evaluate all the rule bases, and select the best one, RB_{sel}.
3. Execute a local search on RB_{sel} to obtain the best rule set, RB_{best}.

The last step was implemented with the iterated local search (ILS) algorithm [29].
threshold (maxRestarts).

5. Results

5.1. Experimental setup

The proposed algorithm has been validated with the well-known in mobile robotics wall-following behavior. The main objectives of a controller for this behavior are: to keep

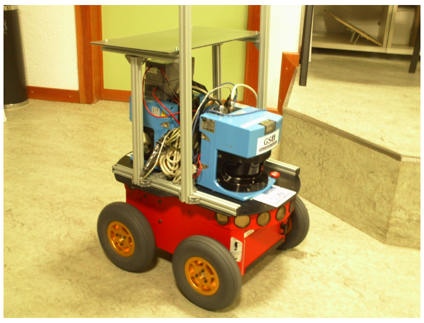


Figure 14: *Pioneer 3-AT* robot equipped with two laser range scanners.

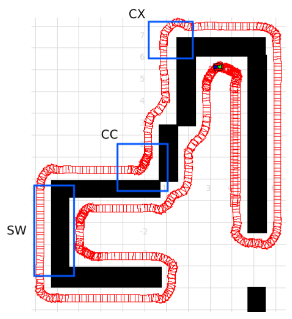


Figure 15: The three different situations for the wall-following behavior.

a suitable distance between the robot and the wall, to move at the highest possible velocity, and to implement smooth control actions. The Player/Stage robot software [30] has been used for the tests on the simulated environments and also for the connection with the real robot *Pioneer 3-AT* (Fig. 14). This real robot was equipped with two laser range scanners with an amplitude of 180° and a precision of 0.5° (i.e. 361 measurements for each laser scan). Without loss of generality, all the examples and tests here described were made with the robot following the wall at its right.

The examples that have been used for learning were generated for three different situations (Fig. 15) that have been identified by an expert:

1. Convex corner: it is characterized by the existence of a gap in the wall (like an open door) (labeled CX in Fig. 15).
2. Concave corner: it is a situation in which the robot finds a wall in front of it (labeled CC in Fig. 15).
3. Straight wall: any other situation (labeled SW in Fig. 15).

For each of the above situations, the robot was placed in different positions and the associated control order was the one that minimized the error. Therefore, each example consists of 722 distances (one for each laser beam), the current linear velocity of the robot, and the control commands (linear and angular velocity). The expert always tried to follow the wall at, approximately, 50 cm and the maximum values for the linear and angular velocities were 50 cm/s and $45^o s^{-1}$ respectively. 572 training examples were generated for the straight wall situation, 540 for the convex corner and 594 for the concave corner.

The IQFRL algorithm was used to learn a different controller for each of the three situations. In order to decide which

Table 2: Characteristics of the test environments

Environment	Dim. ($m \times m$)	Length (m)	#CC	#CX	#doors
home	8×10	20	8	3	1
gfs_b	14×10	43	10	6	0
dec	19×12	53	8	4	0
domus	26×16	60	9	6	3
citius	16×10	63	12	6	2
raid_a	16×16	66	16	12	0
wsc8a	15×15	70	4	7	1
home_b	18×11	76	17	6	2
raid_b	20×10	86	12	10	2
rooms	19×19	86	12	6	4
flower	22×20	98	9	6	1
office	26×26	146	23	10	8
autolab	26×28	154	21	11	10
maze	18×18	205	13	9	0
hospital	74×45	1046	98	69	43
real env 1	9×8	20	7	3	0
real env 2	10×5	26	7	3	0

knowledge base should be used at each time instant, the classification version of IQFRL (IQFRL-C, see Appendix A) was used. In this way, IQFRL learning could be tested with three completely different controllers.

In order to analyze the performance of the proposed learning algorithm, several tests were done in 15 simulated environments and two real ones. Table 2 shows some of the characteristics of the environments: the dimensions of the environment, the path length, the number of concave (#CC) and convex (#CX) corners, and the number of times that the robot has to cross a door (#doors). 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.

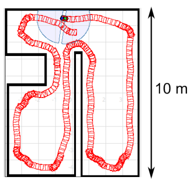
The simulated environments are shown in Figs. 16 and 17. The trace of the robot is represented by marks, and the higher the concentration of marks, the lower the velocity of the robot. Furthermore, Fig. 18 shows the real environments. Each of them represents an occupancy grid map of the environment, together with the trajectory of the robot.

5.2. *Algorithms and parameters*

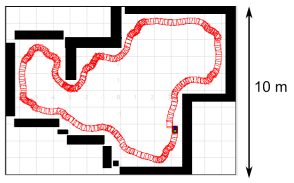
The following values were used for the parameters of the evolutionary algorithm: $ME = 0.02$, $DOF_{min} = 0.001$, $\alpha_f = 0.99$, $P_{cross} = 0.8$, $pop_{max} = 70$, $it_{min} = 50$, $it_{check} = 10$, $it_{max} = 100$, $\sigma_{bd} = 0.01$, $\sigma_v = 0.1$ and $P_{min} = 0.17$. P_{min} is a parameter that has a high influence in the performance of the system. A single value of P_{min} was used in testing, obtained from Eqs. 7 and 8 for the case the error for each consequent is one label (Eq. 8). The granularities and the universe of discourse of each output of a rule are shown in table 3. For the rule subset selection algorithm, the parameters have values of $radius_{nbhood} = 1$ and $maxRestarts = 2$.

The fuzzy inference system used for the learned fuzzy rule sets uses the minimum t-norm for both the implication and conjunction operators, and the weighted average method as defuzzification operator.

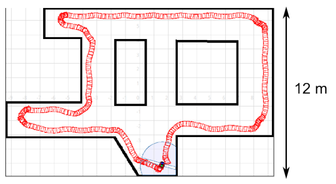
The IQFRL approach was compared with three different algorithms:



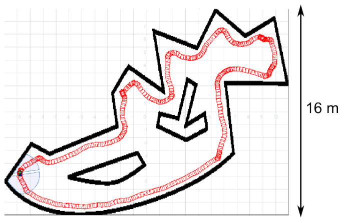
(a) *home*



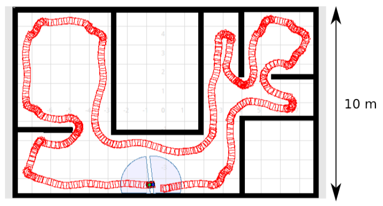
(b) *gfs_b*



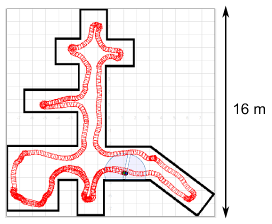
(c) *dec*



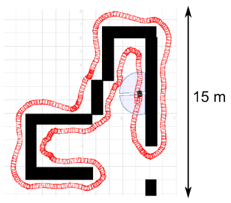
(d) *domus*



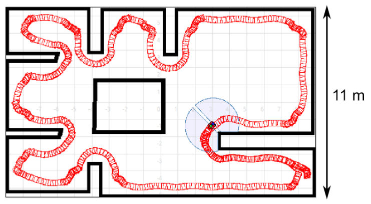
(e) *citius*



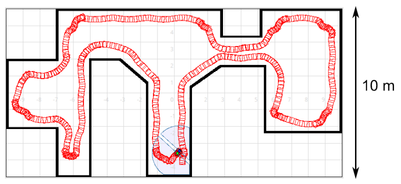
(f) *raid_a*



(g) *wsc8a*



(h) *home_b*



(i) *raid_b*

Figure 16: Path of the robot along the simulated environments (I).

Table 3: Universe of discourse and granularities

Variable	Min	Max	Granularities
Distance	0	1.5	All
Beam	0	721	All
Quantifier	10	100	–
Velocity	0	0.5	All
Lineal velocity	0	0.5	{9}
Angular velocity	$-\pi/4$	$\pi/4$	{19}

- Methodology to Obtain Genetic fuzzy rule-based systems Under the iterative Learning approach (MOGUL): a three-stage genetic algorithm [31]:
  1. An evolutionary process for learning fuzzy rules, with two components: a fuzzy-rule generating method based on IRL, and an iterative covering method.
  2. A genetic simplification process for selecting rules.
  3. A genetic tuning process, that tunes the membership functions for each fuzzy rule or for the complete rule base.

  The soft-constrained MOGUL was used, as it has better performance in very hard problems [25][2].

- Multilayer Perceptron Neural Network (MPNN): a single-hidden-layer neural network trained with the BFGS method [33] with the following parameters: *abstol* = 0.01, *reltol* = 0.0001 and *maxit* = 500. The number of neurons in the hidden layer varies from n to $2{\cdot}n$, being n the number of inputs[3].

- ν-Support Vector Regression (ν-SVR)[4]: a ν-SVM [36] version for regression with a Gaussian RBF kernel. The parameter sigma is estimated based upon the 0.1 and 0.9 quantile of $||x - x'||^2$.

As mentioned before, in the IQFRL proposal the preprocessing of raw sensor data is embedded in the learning algorithm. Since the algorithms for the comparison need to preprocess the data before the learning phase, three different approaches were used for the transformation of the sensor data:

- *Min*: the beams of the laser range finder are grouped in n equal sized sectors. For each sector, the minimum distance value is selected as input.

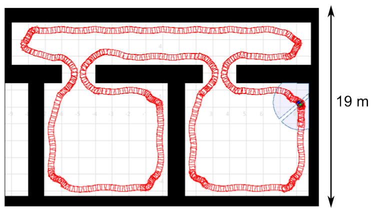
- *Sample*: n equidistant beams are selected as the input data.

- PCA: Principal Component Analysis computes the most meaningful basis to re-express the data. It is a simple, non-parametric method for extracting relevant information. The variances associated with the principal components can be examined in order to select only those that cover a percentage of the total variance.

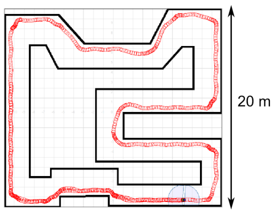
[2]The implementation in *Keel* [32], an open source (GPLv3) Java software tool to assess evolutionary algorithms for Data Mining problems, was used.

[3]The package *nnet* [34] of the statistical software R was used.

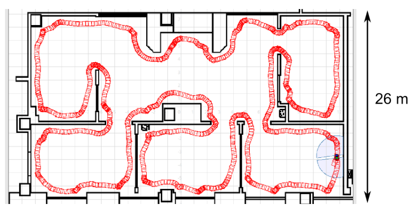
[4]The package *kernlab* [35] of the statistical software R was used.



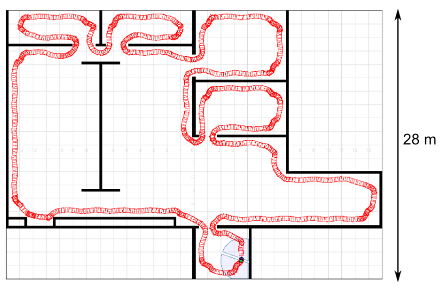
(a) *rooms*



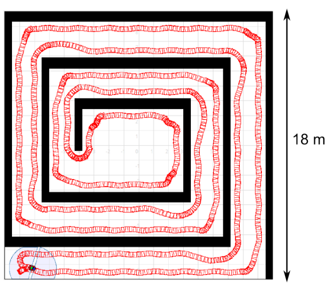
(b) *flower*



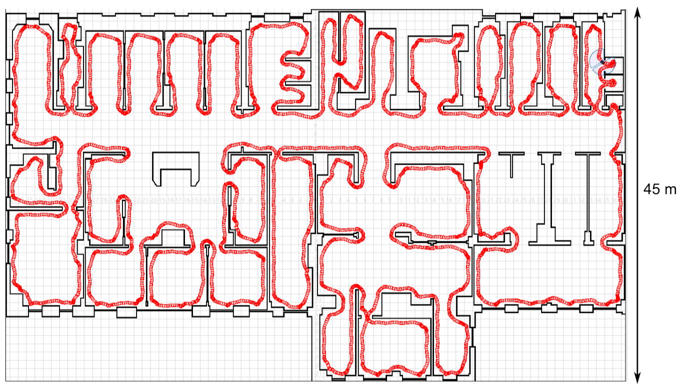
(c) *office*



(d) *autolab*

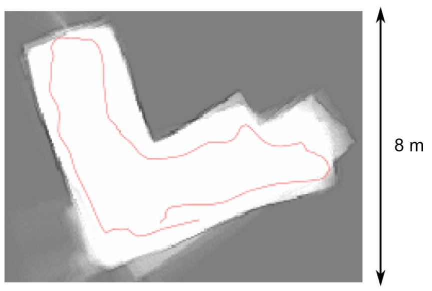


(e) *maze*

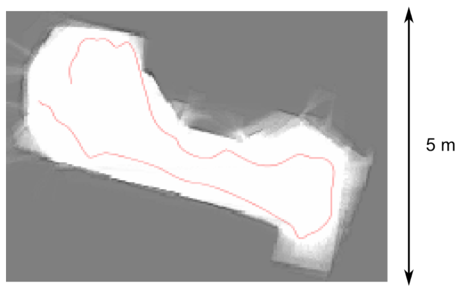


(f) *hospital*

Figure 17: Path of the robot along the simulated environments (II).



(a) *real environment 1*



(b) *real environment 2*

Figure 18: Path of the robot along the real environments.

Different parameters have been used for the preprocessing approaches. For *Min* and *Sample* methods, the number of obtained inputs (n) was changed. For *PCA*, the percentage of variance (σ_{PCA}) indicates the principal components selected as input data. Table 4 shows the parameters used for the preprocessing methods. Moreover, table 5 shows the number of inputs obtained with PCA for the three datasets with each configuration.

Table 4: Different configurations for the preprocessing methods.

Preprocessing	Configuration
Min (n)	$\{4, 8, 16, 32, 64\}$
Sample (n)	$\{4, 8, 16, 32, 64\}$
PCA (σ_{PCA})	$\{0.90, 0.95, 0.975, 0.99, 0.999\}$

5.3. *Comparison and statistical significance*

Table 6 shows the training and test errors over a 5-fold cross-validation. For each algorithm and dataset the mean and

Table 5: Number of inputs obtained with PCA.

σ_{PCA}	Straight	Convex	Concave
0.90	35	15	27
0.95	51	24	40
0.975	66	35	53
0.99	85	57	68
0.999	127	99	109

Table 6: Training and test errors

Alg.	Preproc.	Dataset	Training	Test
IQFRL	–	Straight	0.11 ± 0.03	0.14 ± 0.03
		Convex	0.10 ± 0.01	0.12 ± 0.02
		Concave	0.04 ± 0.01	0.05 ± 0.01
MOGUL	min 16	other	0.01 ± 0.00	0.10 ± 0.01
		convex	0.01 ± 0.00	0.05 ± 0.01
		concave	0.00 ± 0.00	0.05 ± 0.01
	sample 16	other	0.01 ± 0.00	0.09 ± 0.02
		convex	0.02 ± 0.00	0.05 ± 0.01
		concave	0.00 ± 0.00	0.04 ± 0.01
MPNN	min 8	other	0.01 ± 0.00	0.06 ± 0.10
		convex	0.02 ± 0.01	0.03 ± 0.05
		concave	0.00 ± 0.00	0.04 ± 0.06
	sample 8	other	0.02 ± 0.00	0.18 ± 0.25
		convex	0.03 ± 0.01	0.02 ± 0.02
		concave	0.01 ± 0.00	0.17 ± 0.34
ν-SVR	min 16	other	0.01 ± 0.00	0.02 ± 0.02
		convex	0.03 ± 0.01	0.02 ± 0.01
		concave	0.01 ± 0.00	0.00 ± 0.00
	sample 16	other	0.02 ± 0.01	0.02 ± 0.02
		convex	0.04 ± 0.02	0.02 ± 0.01
		concave	0.01 ± 0.00	0.01 ± 0.00

standard deviation of the error (Eq. 8) were calculated.

For each preprocessing technique, a 5-fold cross-validation was performed for each combination of the parameters of the algorithms. For example, for the *Min* preprocessing with 16 equal size sectors, a 5-fold cross-validation was run for each number of neurons between 17 and 34 for the MPNN approach. Only the configuration of the algorithm with lowest test error for each configuration of the preprocessing methods was used for comparison purposes. Moreover, only those configurations of preprocessing techniques with the best results are shown in the tables of this section. Results for *PCA* preprocessing have not been included, as the learning algorithms were not able to obtain adequate controllers.

Although, the MSE (Mean Squared Error) is the usual measure of the performance of the algorithms, this is not a sufficient criterion in mobile robotics. A good controller must be robust and able to provide a good and smooth output in any situation. The only way to validate the controller is to test it on environments (simulated and real) with different difficulties and assessing on these tests a number of quality parameters such as mean distance to the wall, mean velocity along the paths, . . .

Table 8 contains the results of the execution of each of the algorithms for the different simulated environments (Figs. 16 and 17). Furthermore, table 9 shows the average results for the following five different indicators: the distance to the wall at its right (Dist.), the linear velocity (Vel.), the change in the linear velocity between two consecutive cycles (Vel.ch.) —which reflects the smoothness in the control—, the time per lap, and the number of blockades of the robot along the path and cannot recover.

The robot is blocked if it hits a wall or if it does not move for 5 s. In this situation the robot is placed parallel to the wall at a distance of 0.5 m. The average values of the five indicators are calculated for each lap that the robot performs in the environment. Results presented in the table are the average and standard deviation values over five laps of the average values of the indicators over one lap. The dash symbol in the results table indicates that the controller could not complete the path. This usually occurs when the number of blockades per meter is high (greater than 5 blockades in a short period of time) or when the robot completely deviates from the path.

Moreover, in order to evaluate the performance of a controller with a numerical value a general quality measure was defined. It is based on the error measure defined in [15], but including the number of blockades:

$$quality = \frac{1}{1 + (1 + \#Blockades) \cdot (0.9 \cdot |Dist - d_{wall}| + 0.1 \cdot |Vel - v_{max}|)} \qquad (18)$$

where d_{wall} is the reference distance to the wall (50 cm) and v_{max} is the maximum value of the velocity (50 cm/s). The higher the quality, the better the controller. This measure takes the number of blockades into account in a linear form for comparison purposes. However, it should be noted that controllers with just a single blockade are not reliable and should not be implemented on a real robot.

In general, all the algorithms except MPNN with *Sample 16* preprocessing, produced a distance that is very close to the reference (between 40 cm and 60 cm to the wall at its right). Note that in cases where the best distance is very different from that obtained by IQFRL, this is because several blockades happened. Therefore, those controllers have the advantage of being continually repositioned into the perfect situation. The best results in speed are those obtained by ν-SVR and MOGUL but, in general, due to a worsening in the distance to the wall or an increase in the number of blockades. The same applies to the speed change. In those cases where it is too low, like in some cases for MOGUL or MPNN, the robot is not able to trace some curves safely. IQFRL is the approach that gets the best

Table 7: Non-parametric test for *quality* of table 8.

Alg.	Preprocessing	Ranking	Holm p-value
IQFRL	–	1.53	–
MOGUL	min 16	4.9	0.012
	sample 16	3.47	0.025
MLPNN	min 8	5.57	0.010
	sample 8	6	0.008
ν-SVR	min 16	3.83	0.017
	sample 16	2.7	0.05
Friedman p-value = 0.00			
Holm's rejects hypothesis with p-value <= 0.05			

Table 8: Average results ($x \pm \sigma$) for each simulated environment

Alg.	Prepr.	Env.	Dist.(cm)	Vel.(cm/s)	Vel.ch.(cm/s)	Time(s)	# Blockades	*quality*
IQFRL	–	home	55.70 ± 0.25	27.00 ± 0.66	5.59 ± 0.14	164.63 ± 5.26	0.00 ± 0.00	0.12
		gfs_b	55.98 ± 1.70	22.37 ± 0.92	7.01 ± 0.74	163.90 ± 9.72	0.00 ± 0.00	0.11
		dec	57.33 ± 1.02	32.47 ± 0.67	5.83 ± 0.18	168.63 ± 1.76	0.00 ± 0.00	0.11
		domus	54.84 ± 0.53	29.97 ± 0.19	5.97 ± 0.49	198.80 ± 1.39	0.00 ± 0.00	0.14
		citius	54.80 ± 0.87	26.11 ± 0.78	6.28 ± 0.64	249.50 ± 8.25	0.00 ± 0.00	0.13
		raid_a	59.56 ± 0.72	25.35 ± 0.14	6.87 ± 0.55	262.00 ± 6.15	0.00 ± 0.00	0.08
		wsc8a	56.96 ± 1.00	27.45 ± 0.84	7.70 ± 0.33	233.10 ± 5.28	0.00 ± 0.00	0.11
		home_b	58.41 ± 0.72	25.53 ± 0.46	7.06 ± 0.36	300.07 ± 8.15	0.00 ± 0.00	0.09
		raid_b	58.22 ± 0.44	28.57 ± 0.40	6.60 ± 0.47	242.23 ± 3.82	0.00 ± 0.00	0.09
		rooms	57.38 ± 0.34	30.97 ± 0.34	6.38 ± 0.43	261.93 ± 4.60	0.00 ± 0.00	0.10
		flower	53.46 ± 0.25	33.85 ± 0.40	4.13 ± 0.34	290.77 ± 4.13	0.00 ± 0.00	0.17
		office	51.37 ± 0.57	24.20 ± 0.18	6.65 ± 0.25	578.27 ± 2.92	0.00 ± 0.00	0.21
		autolab	52.91 ± 0.20	28.75 ± 0.31	5.57 ± 0.48	499.33 ± 9.74	0.00 ± 0.00	0.17
		maze	52.43 ± 0.22	35.88 ± 0.40	3.64 ± 0.28	567.73 ± 5.29	0.00 ± 0.00	0.22
		hospital	51.09 ± 0.19	26.68 ± 0.10	6.18 ± 0.35	3608.07 ± 21.72	0.00 ± 0.00	0.23
MOGUL	min 16	home	55.12 ± 0.69	30.43 ± 1.30	5.40 ± 0.45	181.10 ± 12.70	7.33 ± 2.05	0.05
		gfs_b	54.75 ± 0.96	24.44 ± 1.02	6.81 ± 0.39	208.87 ± 12.86	14.00 ± 2.16	0.04
		dec	55.46 ± 1.14	36.13 ± 0.43	5.17 ± 0.15	190.50 ± 6.29	8.67 ± 1.25	0.07
		domus	55.75 ± 0.64	31.44 ± 1.80	5.47 ± 0.33	224.60 ± 10.25	8.33 ± 0.47	0.09
		citius	53.47 ± 1.27	29.48 ± 0.13	5.60 ± 0.53	302.57 ± 13.58	18.33 ± 2.62	0.05
		raid_a	57.53 ± 0.32	26.53 ± 0.28	6.41 ± 0.09	363.87 ± 10.21	27.33 ± 3.40	0.02
		wsc8a	54.80 ± 0.35	27.57 ± 0.63	6.26 ± 0.59	346.90 ± 29.40	26.67 ± 5.44	0.02
		home_b	56.75 ± 0.49	27.49 ± 0.62	6.58 ± 0.37	379.57 ± 2.05	22.00 ± 1.41	0.05
		raid_b	57.48 ± 0.70	32.38 ± 0.17	5.84 ± 0.38	280.17 ± 14.29	14.67 ± 3.40	0.03
		rooms	54.79 ± 0.38	30.57 ± 1.04	5.14 ± 0.37	350.33 ± 28.04	20.00 ± 5.89	0.02
		flower	53.33 ± 0.39	38.05 ± 1.00	3.70 ± 0.71	310.27 ± 13.41	11.67 ± 4.03	0.05
		office	51.48 ± 0.29	25.09 ± 0.49	6.77 ± 0.20	762.20 ± 5.28	49.67 ± 1.25	0.10
		autolab	51.95 ± 0.71	30.54 ± 1.24	5.23 ± 0.27	612.00 ± 23.46	31.00 ± 2.45	0.07
		maze	52.25 ± 0.55	37.55 ± 1.53	2.87 ± 0.25	690.93 ± 53.92	32.00 ± 6.38	0.04
		hospital	51.33 ± 0.06	26.86 ± 0.08	5.89 ± 0.29	4908.07 ± 56.12	313.33 ± 10.34	0.02
	sample 16	home	56.76 ± 0.20	29.57 ± 0.35	4.73 ± 0.15	161.97 ± 0.69	1.67 ± 0.47	0.08
		gfs_b	56.16 ± 1.62	23.66 ± 0.88	8.16 ± 0.51	160.80 ± 9.65	1.67 ± 1.25	0.05
		dec	57.69 ± 0.66	37.95 ± 0.96	6.03 ± 0.25	148.87 ± 3.94	0.67 ± 0.47	0.08
		domus	56.04 ± 0.20	36.61 ± 1.14	6.35 ± 0.57	165.63 ± 8.93	1.00 ± 0.82	0.08
		citius	51.38 ± 1.72	27.40 ± 0.21	6.13 ± 0.52	241.53 ± 3.38	1.00 ± 0.82	0.14
		raid_a	57.44 ± 0.51	26.18 ± 1.39	6.61 ± 0.44	275.63 ± 18.82	3.67 ± 2.05	0.03
		wsc8a	54.67 ± 0.24	30.18 ± 0.65	9.03 ± 0.42	220.87 ± 2.26	1.67 ± 0.94	0.08
		home_b	57.17 ± 0.36	26.80 ± 0.91	6.73 ± 0.72	303.77 ± 9.52	3.00 ± 0.82	0.06
		raid_b	60.38 ± 0.38	34.73 ± 1.26	6.15 ± 0.41	206.20 ± 10.12	0.33 ± 0.47	0.06
		rooms	56.05 ± 0.09	32.11 ± 1.49	6.39 ± 0.64	254.50 ± 18.03	0.67 ± 0.94	0.07
		flower	55.24 ± 0.58	41.58 ± 0.32	3.92 ± 0.27	244.67 ± 6.56	2.00 ± 1.41	0.07
		office	50.33 ± 0.10	22.76 ± 0.56	6.22 ± 0.46	655.40 ± 21.32	11.33 ± 2.36	0.09
		autolab	50.57 ± 0.27	29.62 ± 0.69	5.29 ± 0.32	498.67 ± 8.12	2.33 ± 1.25	0.15
		maze	55.05 ± 0.34	40.22 ± 0.78	3.25 ± 0.18	512.33 ± 8.86	0.67 ± 0.47	0.11
		hospital	51.54 ± 0.25	25.97 ± 0.14	6.11 ± 0.34	3964.70 ± 15.81	64.67 ± 8.18	0.03
MPNN	min 8	home	58.34 ± 1.07	29.34 ± 2.66	4.12 ± 0.11	122.73 ± 6.36	6.33 ± 0.47	0.07
		gfs_b	58.56 ± 1.41	28.14 ± 0.71	7.66 ± 0.28	149.20 ± 11.03	4.33 ± 1.70	0.04
		dec	56.10 ± 0.32	35.13 ± 0.38	4.28 ± 0.19	173.37 ± 5.58	3.33 ± 0.94	0.07
		domus	–	–	–	–	–	0.00
		citius	55.73 ± 0.90	27.22 ± 0.68	4.97 ± 0.28	324.37 ± 31.51	15.00 ± 3.56	0.03
		raid_a	57.90 ± 0.61	23.05 ± 1.01	6.85 ± 0.26	403.60 ± 13.17	27.33 ± 1.70	0.04
		wsc8a	56.24 ± 0.81	30.96 ± 1.03	7.97 ± 0.11	238.77 ± 5.45	7.33 ± 0.47	0.08
		home_b	58.02 ± 0.54	24.19 ± 1.81	7.13 ± 0.45	922.87 ± 568.90	66.33 ± 41.02	0.00
		raid_b	59.99 ± 1.57	27.98 ± 2.64	5.07 ± 0.63	1137.37 ± 842.79	38.33 ± 20.53	0.00
		rooms	–	–	–	–	–	0.00
		flower	–	–	–	–	–	0.00
		office	55.84 ± 0.48	28.48 ± 0.19	8.18 ± 0.32	626.33 ± 5.59	32.00 ± 1.41	0.05
		autolab	–	–	–	–	–	0.00
		maze	52.75 ± 0.32	42.53 ± 1.09	2.81 ± 0.38	621.07 ± 45.13	28.00 ± 3.56	0.06
		hospital	55.94 ± 0.02	28.45 ± 0.33	7.47 ± 0.17	3730.00 ± 166.69	205.00 ± 13.93	0.01
	sample 8	home	–	–	–	–	–	0.00
		gfs_b	62.11 ± 0.47	21.96 ± 0.35	7.22 ± 0.14	172.00 ± 0.78	1.00 ± 0.00	0.07
		dec	64.67 ± 2.80	30.23 ± 4.21	6.03 ± 0.49	285.70 ± 118.55	1.00 ± 0.82	0.04
		domus	–	–	–	–	–	0.00
		citius	68.36 ± 5.48	20.98 ± 2.23	6.69 ± 0.18	603.33 ± 243.12	16.33 ± 11.81	0.00
		raid_a	74.58 ± 8.62	19.36 ± 3.61	6.41 ± 1.03	450.60 ± 259.59	5.00 ± 3.56	0.01
		wsc8a	61.04 ± 0.62	23.70 ± 0.41	7.90 ± 0.52	279.37 ± 6.70	1.00 ± 0.82	0.04
		home_b	85.20 ± 8.36	16.95 ± 3.09	6.40 ± 0.62	2477.03 ± 1471.27	26.33 ± 15.15	0.00
		raid_b	70.10 ± 4.50	21.41 ± 1.56	7.18 ± 0.48	1780.97 ± 1445.69	49.00 ± 43.69	0.00
		rooms	60.75 ± 0.47	34.39 ± 0.65	6.81 ± 0.22	237.53 ± 5.88	0.00 ± 0.00	0.08
		flower	61.77 ± 2.11	28.82 ± 0.35	7.54 ± 0.21	912.67 ± 231.62	51.33 ± 13.02	0.01
		office	57.22 ± 1.31	21.01 ± 0.31	5.58 ± 0.15	783.03 ± 7.34	28.33 ± 0.47	0.07
		autolab	–	–	–	–	–	0.00
		maze	–	–	–	–	–	0.00
		hospital	74.51 ± 11.21	21.74 ± 3.14	5.33 ± 1.18	555.43 ± 721.71	16.33 ± 23.10	0.00
ν-SVR	min 16	home	–	–	–	–	–	0.00
		gfs_b	57.82 ± 0.58	26.35 ± 0.81	8.67 ± 0.52	140.20 ± 3.94	0.00 ± 0.00	0.10
		dec	59.14 ± 0.05	39.01 ± 0.98	6.59 ± 0.48	143.03 ± 3.39	0.00 ± 0.00	0.10
		domus	–	–	–	–	–	0.00
		citius	55.14 ± 0.78	25.65 ± 0.29	5.90 ± 0.28	258.87 ± 2.23	0.00 ± 0.00	0.12
		raid_a	58.52 ± 0.29	30.31 ± 0.67	9.36 ± 0.28	208.97 ± 4.84	0.00 ± 0.00	0.09
		wsc8a	58.33 ± 0.24	30.09 ± 0.27	10.70 ± 0.38	218.33 ± 1.56	0.00 ± 0.00	0.10
		home_b	61.26 ± 0.46	27.87 ± 0.50	8.05 ± 0.10	289.40 ± 2.94	1.00 ± 0.00	0.07
		raid_b	–	–	–	–	–	0.00
		rooms	–	–	–	–	–	0.00
		flower	58.66 ± 0.11	38.42 ± 0.64	4.76 ± 0.22	257.10 ± 4.64	0.00 ± 0.00	0.10
		office	51.37 ± 0.47	23.92 ± 0.40	7.67 ± 0.10	582.23 ± 5.47	0.00 ± 0.00	0.21
		autolab	54.18 ± 0.25	28.04 ± 0.22	6.38 ± 0.19	522.07 ± 3.83	0.67 ± 0.47	0.10
		maze	61.07 ± 0.70	32.60 ± 1.02	3.06 ± 0.17	675.67 ± 63.68	1.00 ± 0.82	0.04
		hospital	53.42 ± 0.36	25.42 ± 0.08	6.58 ± 0.21	3833.90 ± 13.48	6.67 ± 1.70	0.06
	sample 16	home	–	–	–	–	–	0.00
		gfs_b	57.63 ± 0.24	27.77 ± 0.64	8.21 ± 0.62	132.20 ± 3.00	0.00 ± 0.00	0.10
		dec	57.31 ± 0.03	39.00 ± 0.29	5.72 ± 0.10	142.47 ± 0.66	0.00 ± 0.00	0.12
		domus	–	–	–	–	–	0.00
		citius	55.31 ± 0.82	29.69 ± 0.58	6.28 ± 0.24	221.00 ± 3.99	0.00 ± 0.00	0.13
		raid_a	56.89 ± 0.09	31.38 ± 0.20	8.37 ± 0.35	201.27 ± 0.59	0.00 ± 0.00	0.11
		wsc8a	56.27 ± 0.03	32.09 ± 0.21	9.37 ± 0.31	203.50 ± 1.84	0.00 ± 0.00	0.12
		home_b	60.62 ± 0.70	29.24 ± 0.06	8.27 ± 0.09	275.70 ± 6.91	0.33 ± 0.47	0.06
		raid_b	57.54 ± 1.24	36.64 ± 0.70	6.25 ± 0.44	273.67 ± 9.39	12.33 ± 1.25	0.05
		rooms	57.95 ± 0.49	34.88 ± 0.26	6.37 ± 0.26	233.53 ± 0.12	0.00 ± 0.00	0.10
		flower	56.64 ± 0.14	40.23 ± 0.13	5.16 ± 0.06	244.13 ± 1.13	0.00 ± 0.00	0.13
		office	51.34 ± 0.13	26.54 ± 0.11	7.65 ± 0.16	522.00 ± 3.41	0.00 ± 0.00	0.22
		autolab	53.26 ± 0.17	31.50 ± 0.38	6.19 ± 0.04	462.23 ± 3.94	0.00 ± 0.00	0.17
		maze	–	–	–	–	–	0.00
		hospital	52.54 ± 0.13	28.57 ± 0.15	6.71 ± 0.19	3359.17 ± 13.58	0.00 ± 0.00	0.18

Table 9: Average results ($x \pm \sigma$) for all simulated environments

Alg.	Prepr.	Dist.(cm)	Vel.(cm/s)	Vel.ch.(cm/s)	# Blockades	*quality*
IQFRL	–	55.36 ± 2.57	28.34 ± 3.60	6.10 ± 1.04	0.00 ± 0.00	0.14 ± 0.05
MOGUL	min 16	54.42 ± 1.99	30.30 ± 4.13	5.54 ± 1.05	40.33 ± 73.78	0.05 ± 0.02
	sample 16	55.10 ± 2.83	31.02 ± 5.76	6.07 ± 1.39	6.42 ± 15.78	0.08 ± 0.03
MPNN	min 16	56.86 ± 1.87	29.59 ± 5.09	6.05 ± 1.76	39.39 ± 55.37	0.03 ± 0.03
	sample 16	67.30 ± 7.88	23.69 ± 4.99	6.64 ± 0.76	17.79 ± 18.12	0.02 ± 0.03
ν-SVR	min 16	57.17 ± 3.05	29.79 ± 4.83	7.07 ± 2.05	0.85 ± 1.88	0.07 ± 0.06
	sample 16	56.11 ± 2.49	32.29 ± 4.27	7.05 ± 1.23	1.05 ± 3.40	0.10 ± 0.06

Table 10: Average results ($x \pm \sigma$) of IQFRL for the real environments

Env.	Dist.(cm)	Vel.(cm/s)	Vel.ch.(cm/s)	Time(s)	# Blockades	*quality*
real env 1	54.13 ± 2.59	19.86 ± 1.52	1.36 ± 0.21	100.70	0.00 ± 0.00	0.13
real env 2	59.29 ± 2.74	21.94 ± 1.43	1.72 ± 2.50	118.50	0.00 ± 0.00	0.08

quality values, reflecting not only the adequate values for the distance, velocity and smoothness in all the environments but, also, its robustness: it is the unique approach that never blocked or failed to complete the laps in any of the environments.

In order to compare the experimental results, non-parametric tests of multiple comparisons have been used. Their use is recommended in those cases in which the objective is to compare the results of a new algorithm against various methods simultaneously. The Friedman test with Holm post-hoc test was selected as the method for detecting significant differences among the results. The test is performed for the *quality* indicator in table 8.

The statistical test (table 7) shows that the difference of the quality of the IQFRL approach is statistically significant. Only ν-SVR and MOGUL with sample 16 preprocessing are comparable to IQFRL, as the number of blockades is very low or null in some environments.

Additionally, table 10 shows the results obtained by IQFRL in two real environments. As in the previous tables, the results are the average and standard deviation over 5 laps. The distance to the wall is lower than 60 cm, showing a good behavior, although the velocity seems to be low, this is because corners are very close to each other and the robot does not have time to accelerate. Also, the velocity change reflects a very smooth movement as changes in velocity take more time in the real robot.

Finally, the IQFRL proposal was compared with the proposals presented in [15] for learning rules for the wall-following behavior. The purpose of this comparison is to check if IQFRL is competitive against other methods which use expert knowledge for sensor data preprocessing. Four different approaches were used: the COR methodology, the weighted COR methodology (WCOR), Hierarchical Systems of Weighted Linguistic Rules (HSWLR) and a local evolutionary learning of Takagi-Sugeno rules (TSK). For these approaches, four input variables were defined by an expert: right distance, left distance, velocity, and the orientation (alignment) of the robot to the wall at its right. Moreover, the granularities of each variable were also defined by the expert. Table 12 presents the comparison between these approaches and the IQFRL proposal on those environments which are common.

The IQFRL approach exhibited the highest quality in the two most complex environments (office and hospital). Moreover, table 11 shows the non-parametric tests performed over *quality*. The Friedman p-value is higher than in table 9, due to the low number of environments available for comparisons. As can be seen, there is no statistically significant difference regarding the *quality*. That is, the controllers learned with embedded preprocessing has similar performance to the methods that use expert knowledge to preprocess the data.

Table 11: Non-parametric test for *quality* of table 12.

Alg.	Ranking	Holm p-value
IQFRL	2.9	–
COR	3.9	0.01
WCOR	3.7	0.017
HSWLR	2.8	0.05
TSK	1.7	0.006
Friedman p-value = 0.19		
Holm's rejects hypothesis with p-value <= 0.005		

5.4. *Complexity of the Rules*

An example of a rule learned by IQFRL is presented in Fig. 19. The antecedent part is composed of a single QFP. The linguistic value $A_d^{5,1}$ indicates a low distance, while $A_b^{4,1}$ denotes that the beams sector of the proposition is formed by the frontal and right parts of the robot. Therefore, the rule describes a situation where the robot is too close to the wall and, if it continues, it will collide. Because of that, the consequent indicates a zero linear velocity and a turn of the robot to the left, in order to get away from the wall without getting the robot into risk.

Table 13 shows the number of rules learned for the different situations by each of the methods based on rules. MOGUL is implemented as a multiple-input single-output (MISO) algorithm, therefore for each output, different rule bases were learned. Moreover, table 14 shows the complexity of the learned rules in terms of mean and standard deviation of

Table 12: Average results ($x \pm \sigma$) of IQFRL and several approaches with preprocessing based on expert knowledge [15]

Alg.	Env.	Dist.(cm)	Vel.(cm/s)	Vel.ch.(cm/s)	Time(s)	# Blockades	*quality*
IQFRL	wsc8a	56.96 ± 1.00	27.45 ± 0.84	7.70 ± 0.33	233.10 ± 5.28	0.00 ± 0.00	0.11
	rooms	57.38 ± 0.34	30.97 ± 0.34	6.38 ± 0.43	261.93 ± 4.60	0.00 ± 0.00	0.10
	autolab	52.91 ± 0.20	28.75 ± 0.31	5.57 ± 0.48	499.33 ± 9.74	0.00 ± 0.00	0.17
	office	51.37 ± 0.57	24.20 ± 0.18	6.65 ± 0.25	578.27 ± 2.92	0.00 ± 0.00	0.21
	hospital	51.09 ± 0.19	26.68 ± 0.10	6.18 ± 0.35	3608.07 ± 21.72	0.00 ± 0.00	0.23
COR	wsc8a	53.20 ± 1.33	39.86 ± 0.71	5.67 ± 0.83	174.98 ± 1.79	0.00 ± 0.00	0.17
	rooms	46.80 ± 0.59	37.82 ± 0.41	6.76 ± 0.31	227.16 ± 1.03	0.00 ± 0.00	0.16
	autolab	56.88 ± 0.91	25.69 ± 0.79	10.79 ± 0.21	587.96 ± 39.72	0.00 ± 0.00	0.09
	office	55.97 ± 1.65	32.48 ± 0.90	4.06 ± 0.28	457.58 ± 15.00	0.00 ± 0.00	0.11
	hospital	54.12 ± 0.92	35.63 ± 0.77	6.95 ± 0.28	2864.92 ± 45.27	0.00 ± 0.00	0.14
WCOR	wsc8a	52.79 ± 1.36	36.98 ± 1.85	7.37 ± 0.62	187.90 ± 9.78	0.00 ± 0.00	0.17
	rooms	51.17 ± 0.77	37.19 ± 0.27	9.15 ± 0.24	234.04 ± 2.70	0.00 ± 0.00	0.23
	autolab	52.97 ± 1.10	33.47 ± 0.89	7.12 ± 0.52	455.98 ± 41.60	0.00 ± 0.00	0.16
	office	54.59 ± 1.10	33.13 ± 0.97	6.76 ± 0.53	448.16 ± 10.36	0.00 ± 0.00	0.13
	hospital	55.26 ± 1.01	33.71 ± 0.14	6.52 ± 0.12	3073.98 ± 23.63	0.00 ± 0.00	0.12
HSWLR	wsc8a	51.42 ± 0.78	30.46 ± 1.01	3.36 ± 0.13	222.34 ± 6.09	0.00 ± 0.00	0.19
	rooms	50.09 ± 0.88	28.71 ± 0.29	3.04 ± 0.20	290.70 ± 3.66	0.00 ± 0.00	0.24
	autolab	51.50 ± 0.34	23.50 ± 0.97	3.05 ± 0.14	618.40 ± 20.98	0.00 ± 0.00	0.17
	office	53.43 ± 1.22	24.69 ± 0.66	3.73 ± 0.11	594.74 ± 13.16	0.00 ± 0.00	0.13
	hospital	54.60 ± 1.65	25.07 ± 0.49	3.89 ± 0.06	4209.68 ± 166.14	0.00 ± 0.00	0.12
TSK	wsc8a	51.43 ± 1.36	37.54 ± 1.53	5.20 ± 0.50	182.54 ± 8.35	0.00 ± 0.00	0.22
	rooms	49.07 ± 1.08	37.05 ± 0.82	4.96 ± 0.21	227.58 ± 4.46	0.00 ± 0.00	0.24
	autolab	51.87 ± 2.99	33.05 ± 1.33	4.61 ± 0.11	465.56 ± 15.33	0.00 ± 0.00	0.19
	office	53.75 ± 0.97	34.26 ± 0.65	5.24 ± 0.22	432.38 ± 10.48	0.00 ± 0.00	0.14
	hospital	54.50 ± 1.49	34.31 ± 0.32	5.01 ± 0.11	3053.74 ± 123.72	0.00 ± 0.00	0.13

Table 13: Number of rules learned

Alg.	Preproc.	Output	$\#R_{straight}$	$\#R_{convex}$	$\#R_{concave}$
IQFRL	–	Both	108.00 ± 18.88	47.80 ± 16.09	40.40 ± 10.65
MOGUL	min 16	*vlin*	548.60 ± 25.60	308.20 ± 12.12	680.20 ± 24.43
		vang	547.00 ± 16.37	302.80 ± 21.57	712.40 ± 23.79
	sample 16	*vlin*	507.80 ± 29.88	268.20 ± 12.66	664.80 ± 8.52
		vang	530.20 ± 26.48	252.80 ± 8.28	709.80 ± 34.19

Table 14: Complexity of the rules

Alg.	Preproc.	Dataset	Output	Propositions	g_d	g_b	g_v
IQFRL	–	Straight	Both	2.74 ± 0.94	7.02 ± 10.52	5.98 ± 5.62	6.21 ± 1.53
		Convex		2.68 ± 0.69	15.37 ± 23.59	11.22 ± 8.50	6.55 ± 1.03
		Concave		2.78 ± 1.18	3.80 ± 1.79	7.07 ± 6.86	6.16 ± 1.42
MOGUL	min 16	Straight	*vlin*	17.00 ± 0.00	24.35 ± 109.80	16.00 ± 0.00	39.44 ± 137.45
			vang	17.00 ± 0.00	24.49 ± 107.66	16.00 ± 0.00	35.19 ± 117.75
		Convex	*vlin*	17.00 ± 0.00	32.34 ± 125.75	16.00 ± 0.00	51.68 ± 172.27
			vang	17.00 ± 0.00	38.99 ± 144.86	16.00 ± 0.00	45.07 ± 146.38
		Concave	*vlin*	17.00 ± 0.00	22.93 ± 100.76	16.00 ± 0.00	32.79 ± 106.77
			vang	17.00 ± 0.00	23.35 ± 103.39	16.00 ± 0.00	37.56 ± 122.75
	sample 16	Straight	*vlin*	17.00 ± 0.00	26.23 ± 117.41	16.00 ± 0.00	33.98 ± 108.16
			vang	17.00 ± 0.00	26.25 ± 116.18	16.00 ± 0.00	37.60 ± 126.86
		Convex	*vlin*	17.00 ± 0.00	25.68 ± 103.29	16.00 ± 0.00	49.61 ± 160.18
			vang	17.00 ± 0.00	31.06 ± 119.50	16.00 ± 0.00	46.56 ± 151.27
		Concave	*vlin*	17.00 ± 0.00	23.50 ± 105.79	16.00 ± 0.00	33.62 ± 112.09
			vang	17.00 ± 0.00	23.95 ± 106.27	16.00 ± 0.00	34.63 ± 121.52

the number of propositions and granularities for each input variable.

The IQFRL approach is able to learn knowledge bases with a much lower number of rules than MOGUL, even though it is learning both outputs at the same time. The learning of QFRs results in a low number of propositions per rule, thus

IF
$d(h)$ is $A_d^{5,\,1}$ in 50 percent of $A_b^{4,\,1}$
THEN
$vlin$ is A^1_{vlin} and
$vang$ is A^{19}_{vang}

Figure 19: A typical rule learned by IQFRL. $A_d^{5,\,1}$ indicates a low distance and $A_b^{4,\,1}$ indicates the frontal and right sectors.

demonstrating its generalization ability, in spite of the huge input space dimensionality. Moreover, the granularities of each of the input variables are, in general, also low. Therefore, the learned knowledge bases show a low complexity without losing accuracy.

6. Real World Applications

Two of the most used behaviors in mobile robotics are path and object tracking. In recent years several real applications of these behaviors have been described in the literature in different realms. For instance, in [37], a tour-guide robot that can either follow a predefined route or a tour-guide person was shown. With a similar goal, an intelligent hospital service robot was presented in [38]. In this case, the robot can improve the services provided in the hospital through autonomous navigation based on following a path. More recently, in [39] a team of robots that cooperate in a building developing maintenance and surveillance tasks was presented.

More dynamic environments were described in [40, 41], where the robot had to operate in buildings and populated urban areas. These environments introduce numerous challenges to autonomous mobile robots as they are highly complex. Finally, in [42] the authors presented a motion planner that was able to generate paths taking into account the uncertainty due to controls and measurements.

In these and other real applications, the robot has to deal with static and moving objects, including the presence of people surrounding the robot, etc. All these difficulties make necessary the combination of behaviors to perform tasks like path or people tracking in real environments. In order to implement these tasks in a safe way, the robot must be endowed with the ability to avoid collisions with all the objects in the environment while implementing the tasks. These behaviors are challenging tasks that allow us to show the performance of the IQFRL-based approach in realistic conditions. The following behaviors are considered in this section, in order of increasing complexity:

1. *Path tracking with obstacles avoidance*. In this behavior, the mobile robot must follow a path with obstacles in it. A typical application of this behavior is a tour-guide robot that has to follow a predefined tour in a museum. Although in the initial path there were no obstacles in the trajectory, the modification of the environment with new exhibitors and the presence of people make it necessary that the robot modify the predefined route, avoiding the collision with the obstacles and returning to the predefined path as quickly as possible.
2. *Object tracking with fixed obstacles avoidance*. In this case, the robot has to follow the path of a moving object while being at a reference distance to the object. For instance, a tour-guide person being followed by a robot with extended information on a screen. If the followed object comes too close to an obstacle, the robot must avoid the collision while maintaining the tracking behavior.
3. *Object tracking with moving obstacle avoidance*. This behavior is a modification of the previous one, and presents a more difficult problem. In addition to the fixed obstacles avoidance, the robot has to track an object while preventing collisions with moving obstacles that are crossing between the robot and the tracked object. These moving obstacles can be persons walking around or even other mobile robots doing their own behaviors.

In order to perform these behaviors, a fusion of two different controllers has been developed. On one hand, a tracking controller [43] was used in order to follow the path or the moving object. On the other hand, the wall-following controller learned with the IQFRL algorithm was used as the collision avoidance behavior. Section 5.3 showed that this controller is robust and operates safely while performing the task. There were no blockades during the behavior in all the tests, neither from collisions nor from other reasons. The way in which the wall-following behavior is used in order to avoid collisions is: given an obstacle that is too close to the robot, it can be surrounded following the border of this obstacle in order to avoid a collision with it. The controller described in this paper follows the wall on its right, while for this task, the obstacle can be on both sides. This can easily be solved by a simple permutation of the laser beams depending on which side the obstacle is detected.

The wall-following behavior is only executed when the robot is too close to an object —a value of 0.4 m has been used as threshold. The objective of the controller is to drive the robot to a state in which there is no danger of collision —a value of 0.5 m has been established as a safe distance. As long as the robot is in a safe state the tracking behavior is resumed. This behavior controls the linear and angular velocities of the robot in order to place it at an objective point in every control cycle. This point is defined using the desired distance between the robot and the moving object. The tracking controller uses four different input variables:

- The distance between the robot and the objective point:
$$d = \frac{\sqrt{(x_r - x_{obj})^2 + (y_r - y_{obj})^2}}{d_{ref}} \tag{19}$$
where (x_r, y_r) are the coordinates of the robot, (x_{obj}, y_{obj}) are the coordinates of the objective point and d_{ref} is the reference distance between the robot and the objective point.
- The deviation of the robot with respect to the objective point:
$$dev = \arctan\left(\frac{y_{obj} - y_r}{x_{obj} - x_r}\right) - \theta_r \tag{20}$$

where θ_r is the angle of the robot. A negative value of the deviation indicates that the robot is moving in a direction to the left of the objective point, while a positive value means that it is moving to the right.

- The difference of velocity between the robot and the objective point:

$$\Delta v = \frac{v_r - v_m}{v_{max}} \tag{21}$$

where v_r, v_m and v_{max} are the linear velocities of the robot, the moving object, and the maximum velocity attainable by the robot.

- The difference in angle between the object and the robot:

$$\Delta\theta = \theta_m - \theta_r \tag{22}$$

where θ_m is the angle of the moving object.

The reference distance (d_{ref}) is different depending on the type of behavior. For the path tracking behavior, there is no moving object tracking and, therefore, the robot follows the path with $d_{ref} = 0$ in order to do a perfect path tracking. In the other two behaviors the robot follows a moving object, so it is necessary to keep a safe distance —a value of $d_{ref} = 0.5$ m was used in the experiments shown in this section.

The three behaviors have been validated in two different environments (*M1* and *Domus*) which try to reproduce the plant of a museum (Fig. 20). Figs. 20(a) and 20(b) show the path tracking with obstacles avoidance behavior. The orange (medium grey) path represents the trajectory that has to be followed by the robot. This path also includes information of the velocity that the robot should have at each point. The higher the concentration of marks, the lower the linear velocity in that point of the path. Moreover, the path was generated without obstacles and, once the obstacles were added to the environment, the robot was placed at the beginning of the path in order to track it. The cyan (light grey) path indicates the trajectory implemented by the robot using the proposed combination of controllers (wall-following and tracking). It can be seen that the robot avoids successfully all the obstacles in its path, i.e., the wall following behavior deviates the robot from the predefined path when an obstacle generates a possibility of collision. When the robot overcomes the obstacle, it returns to the predefined path as quickly as possible.

In the case of the moving object tracking with fixed obstacles avoidance behavior (Figs. 20(c) and 20(d)), the cyan (light grey) line represents the path of the robot due to the combination of the controllers. Also, the orange (medium grey) path shows the trajectory of the moving object tracked by the robot. In this behavior, the moving object goes too close to some obstacles in several situations, forcing the controller to execute the wall following behavior in order to avoid collisions. Moreover, the wall-following controller is also executed when the moving object turns the corners very close to the obstacles, at a distance that is unsafe for the robot.

The last and most complex behavior is moving object tracking with moving obstacle avoidance (Figs. 20(e) and 20(f)). The cyan (light grey) path shows, once again, the path followed by the robot when it tracks the moving object (orange / medium grey path) while avoiding static and moving obstacles. Also, the path followed by the moving obstacle that should be avoided by the robot is shown in blue (dark grey). The arrows along the path indicate the places in which the obstacle interferes with the robot. This behavior shows the ability of the controller learned with the IQFRL algorithm to avoid collisions, even when the moving obstacle tries to force the robot to fail: the controller can detect the situation and perform the task safely, avoiding collisions.

7. Conclusions

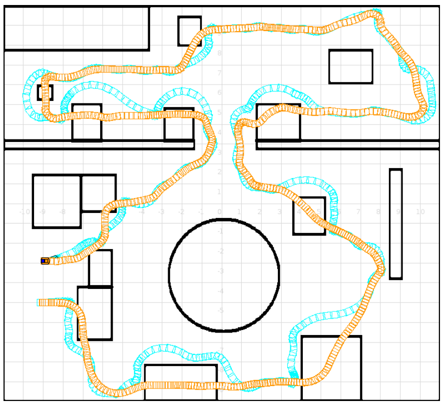
This paper describes a new algorithm which is able to learn controllers with embedded preprocessing for mobile robotics. The transformation of the low-level variables into high-level variables is done through the use of Quantified Fuzzy Propositions and Rules. Furthermore, the algorithm involves linguistic labels defined by multiple granularity without limiting the granularity levels. The algorithm was extensively tested with the wall-following behavior both in several simulated environments and on a *Pioneer 3-AT* robot in two real environments. The results were compared with some of the most well-known algorithms for learning controllers in mobile robotics. Non-parametric significance tests have been performed, showing a very good and a statistically significant performance of the IQFRL approach.

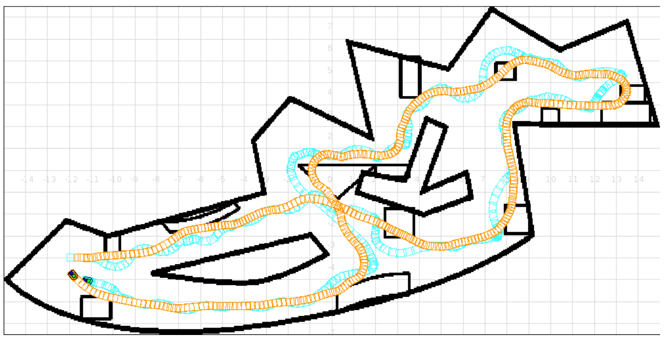
8. Acknowledgements

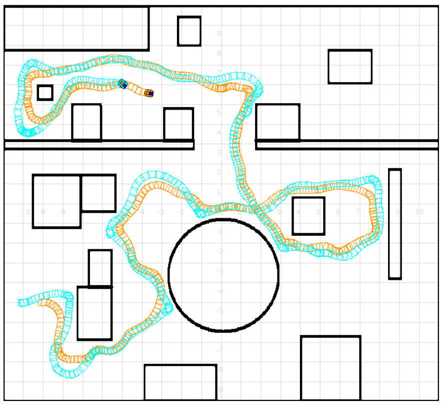
This work was supported by the Spanish Ministry of Economy and Competitiveness under grants TIN2011-22935 and TIN2011-29827-C02-02. I. Rodriguez-Fdez is supported by the Spanish Ministry of Education, under the FPU national plan (AP2010-0627). M. Mucientes is supported by the Ramón y Cajal program of the Spanish Ministry of Economy and Competitiveness. This work was supported in part by the European Regional Development Fund (ERDF/FEDER) under the projects CN2012/151 and CN2011/058 of the Galician Ministry of Education.

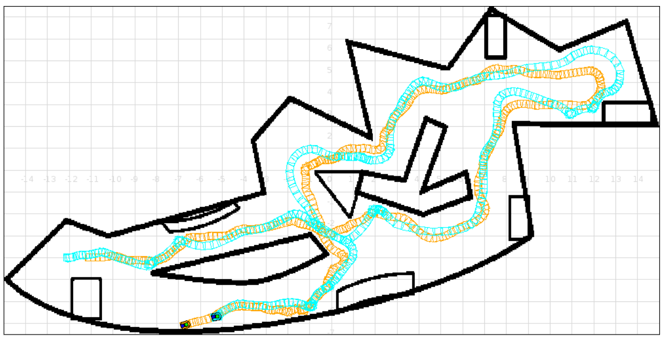
Appendix A. IQFRL for Classification (IQFRL-C)

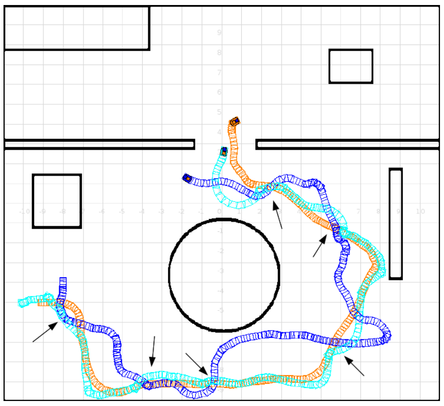
This section describes the modifications that are necessary to accomplish for adapting the IQFRL algorithm for classification problems.


(a) *Path tracking with obstacles avoidance in M1.*


(b) *Path tracking with obstacles avoidance in Domus.*


(c) *Object tracking with fixed obstacles avoidance in M1.*


(d) *Object tracking with fixed obstacles avoidance in Domus.*


(e) *Object tracking with moving obstacle avoidance in M1.*

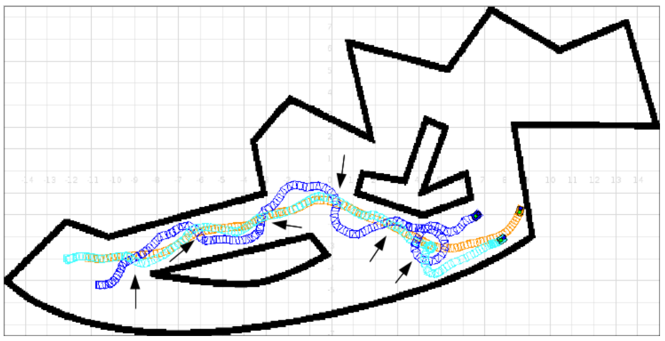

(f) *Object tracking with moving obstacle avoidance in Domus.*

Figure 20: Experiments on real applications. Colors code: 1) Original path to be tracked in orange (medium grey); 2) Robot path in cyan (light grey); 3) Moving obstacle path in blue (dark grey). The arrows along the path in Figs. 20(e) and 20(f) indicate the places in which the moving obstacle interferes with the robot.

Appendix A.1. Examples and Grammar

The structure of the examples used for classification is very similar to the one described in expression 4:

$$e^l = (d(1), \ldots, d(N_b), velocity, class) \qquad (A.1)$$

where *class* represents the class of the example.

Furthermore, the consequent production (production 3) of the grammar (Fig. 4) must be modified to:

3. consequent $\longrightarrow F_c$

where F_c is the linguistic label of the class. The output variable (*class*) has a granularity $g_c^{\#class}$.

Appendix A.2. Initialization

The consequent of the rules is initialized as $F_c = A_c^{\gamma}$ where γ is the class that represents the example. Only those examples whose class is different from the default class (A_c^f) are used in the initialization of a new individual.

Appendix A.3. Evaluation

For each individual (rule) of the population, the following values are calculated:

- True positives (*tp*):
  - $\#tp = \left|\left\{e^l : C_l = C_j \wedge DOF_j\left(e^l\right) > 0\right\}\right|$, where C_l is the class of example e^l, C_j is the class in the consequent of the j-th rule, and $DOF_j\left(e^l\right)$ is the *DOF* of the j-th rule for the example e^l. *#tp* represents the number of examples that have been correctly classified by the rule.
  - $tpd = \sum_l DOF_j\left(e^l\right) : C_l = C_j$, i.e., the sum of the *DOF*s of the examples contributing to *#tp*.
  - $tp = \#tp + tpd/\#tp$
- False positives (*fp*):
  - $\#fp = \left|\left\{e^l : C_l \neq C_j \wedge DOF_j\left(e^l\right) > 0\right\}\right|$: number of patterns that have been classified by the rule but belong to a different class.
  - $fpd = \sum_l DOF_j\left(e^l\right) : C_l \neq C_j$, i.e., the sum of the *DOF*s of the patterns that contribute to *#fp*.
  - $fp = \#fp + fpd/\#fp$
- False negatives (*fn*):
  - $\#fn = n_{ex}^{C_j} - \#tp$, where $n_{ex}^{C_j} = \left|\left\{e^l : C_l = C_j\right\}\right|$. *#fn* is the number of examples that have not been classified by the rule but belong to the class in the consequent of the rule.

The values of *tp* and *fp* take into account not only the number of examples that are correctly/incorrectly classified, but also the degree of fulfillment of the rule for each of the examples. In case that $tpd \approx 0$, then $tp \approx \#tp$, while if it is high ($tpd \approx \#tp$) then $tp \approx \#tp + 1$. Taking into account these definitions, the accuracy of an individual of the population can be described as:

$$confidence = \frac{1}{10^{fp}} \qquad (A.2)$$

while the ability of generalization of a rule is calculated as:

$$support = \frac{tp}{tp + \#fn} \qquad (A.3)$$

Finally, *fitness* is defined as the combination of both values:

$$fitness = confidence \cdot support \qquad (A.4)$$

which represents the strength of an individual.

Table A.15: Number of rules learned for dataset by IQFRL-C

$\#R_{straight}$	$\#R_{convex}$	$\#R_{concave}$
–	21.20 ± 4.35	10.00 ± 1.41

Table A.16: Complexity of the rules learned by IQFRL-C

Propositions	g_d	g_b	g_v
2.67 ± 0.90	7.58 ± 12.65	8.41 ± 8.49	5.83 ± 1.65

Appendix A.4. Mutation

For classification, the probability that an example matches the output associated to a rule (Eq. 7) is binary. Therefore, in order to select the example (e^{sel}) that is going to be used for mutation, the following criteria is used:

- For generalization, the probability for an example e^l to be selected is:

$$P(e^l = e^{sel}) = 1 - \frac{\sum_j DOF_j\left(e^l\right) \cdot confidence_j}{\sum_j DOF_j\left(e^l\right)} \qquad (A.5)$$

  where $confidence_j$ is the *confidence* (Eq. A.2) of the j-th individual. This probability measures the accuracy with which the individuals of the population cover the example e^l.

- For specialization, the mutated individual uncovers the example e^{sel}. The probability to select e^l for specialization is calculated as follow:

$$P(e^l = e^{sel}) = 1 - DOF_j\left(e^l\right) \qquad (A.6)$$

Finally, the consequent is mutated considering the class of the examples covered by the individual. Thus, the probability that the consequent of the individual j change to the class C_γ is defined as:

$$P\left(j \mid C_\gamma\right) = \frac{\sum_l DOF_j\left(e^l\right) : C_l = C_\gamma}{\sum_l DOF_j\left(e^l\right)} \qquad (A.7)$$

Appendix A.5. Performance

The parameters used for IQFRL-C are the same as for regression (Sec. 5.2). Moreover, the default class is straight wall. Tables A.15 and A.16 show the number of rules learned by the classification method IQFRL-C and the complexity of the rules learned in terms of mean and standard deviation of the number of propositions and granularities for each input variable. The number of rules for each situation is very low, resulting in very interpretable knowledge bases. Furthermore, the complexity of the rules is also low, as the number of propositions and granularities learned show that the rules are very general.

Table A.17 shows the confusion matrix for the learned classifier. The matrix was obtained as the average of a 5-fold cross-validation over the sets. Moreover, the performance of the classifier was analyzed with the accuracy and the Cohen's κ [44]. Both measures are very close to 1, showing the high performance of the classifier obtained with IQFRL-C.

Table A.17: Confusion matrix for the classifier

Actual/Predicted	Straight	Convex	Concave
Straight	30.85	2.40	0.23
Convex	0.70	30.97	0.00
Concave	0.23	0.06	34.55

Accuracy = 0.96
Cohen's κ = 0.94

References

[1] M. Mucientes, A. Bugarín, People detection through quantified fuzzy temporal rules, Pattern Recognition 43 (2010) 1441–1453.

[2] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, L. Magdalena, Ten years of genetic fuzzy systems: current framework and new trends, Fuzzy sets and systems 141 (1) (2004) 5–31.

[3] F. Herrera, Genetic fuzzy systems: taxonomy, current research trends and prospects, Evolutionary Intelligence 1 (1) (2008) 27–46.

[4] B. Bonte, B. Wyns, Automatically designing robot controllers and sensor morphology with genetic programming, in: Proceedings of the 6th IFIP Artificial Intelligence Applications and Innovations (AIAI), 2010, pp. 86–93.

[5] R. Martínez-Soto, O. Castillo, J. R. Castro, Genetic algorithm optimization for type-2 non-singleton fuzzy logic controllers, Recent Advances on Hybrid Approaches for Designing Intelligent Systems (2014) 3–18.

[6] M. Umar Suleman, M. Awais, Learning from demonstration in robots: Experimental comparison of neural architectures, Robotics and Computer-Integrated Manufacturing 27 (4) (2011) 794–801.

[7] A. Agostini, E. Celaya Llover, Reinforcement learning for robot control using probability density estimations, in: Proceedings of the 7th International Conference on Informatics in Control (ICINCO), 2010, pp. 160–168.

[8] C. W. Lo, K. L. Wu, Y. C. Lin, J. S. Liu, An intelligent control system for mobile robot navigation tasks in surveillance, in: Robot Intelligence Technology and Applications 2, Springer, 2014, pp. 449–462.

[9] T. Kondo, Evolutionary design and behavior analysis of neuromodulatory neural networks for mobile robots control, Applied Soft Computing 7 (1) (2007) 189–202.

[10] K. Samsudin, F. Ahmad, S. Mashohor, A highly interpretable fuzzy rule base using ordinal structure for obstacle avoidance of mobile robot, Applied Soft Computing 11 (2) (2011) 1631–1637.

[11] S. Mabu, A. Tjahjadi, S. Sendari, K. Hirasawa, Evaluation on the robustness of genetic network programming with reinforcement learning, in: Proceedings of the IEEE International Conference on Systems Man and Cybernetics (SMC), 2010, pp. 1659–1664.

[12] K. Senthilkumar, K. Bharadwaj, Hybrid genetic-fuzzy approach to autonomous mobile robot, in: Proceedings of the IEEE International Conference on Technologies for Practical Robot Applications (TePRA), 2009, pp. 29–34.

[13] M. Mucientes, D. Moreno, A. Bugarín, S. Barro, Design of a fuzzy controller in mobile robotics using genetic algorithms, Applied Soft Computing 7 (2) (2007) 540–546.

[14] M. Mucientes, R. Alcalá, J. Alcalá-Fdez, J. Casillas, Learning weighted linguistic rules to control an autonomous robot, International Journal of Intelligent Systems 24 (3) (2009) 226–251.

[15] M. Mucientes, J. Alcalá-Fdez, R. Alcalá, J. Casillas, A case study for learning behaviors in mobile robotics by evolutionary fuzzy systems, Expert Systems With Applications 37 (2010) 1471–1493.

[16] J. Kuo, Y. Ou, An evolutionary fuzzy behaviour controller using genetic algorithm in robocup soccer game, in: Proceedings of the Ninth International Conference on Hybrid Intelligent Systems (HIS), Vol. 1, 2009, pp. 281–286.

[17] M. Khanian, A. Fakharian, M. Chegini, B. Jozi, An intelligent fuzzy controller based on genetic algorithms, in: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), 2009, pp. 486–491.

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

[19] C. Juang, C. Hsu, Reinforcement ant optimized fuzzy controller for mobile-robot wall-following control, IEEE Transactions on Industrial Electronics 56 (10) (2009) 3931–3940.

[20] C. Hsu, C. Juang, Evolutionary robot wall-following control using type-2 fuzzy controller with species-DE-activated continuous ACO, IEEE Transactions on Fuzzy Systems 21 (1) (2013) 100–112.

[21] C. Juang, Y. Chang, Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments, IEEE Transactions on Fuzzy Systems 19 (2) (2011) 379–392.

[22] M. Mucientes, I. Rodríguez-Fdez, A. Bugarín, Evolutionary learning of quantified fuzzy rules for hierarchical grouping of laser sensor data in intelligent control, in: Proceedings of the IFSA-EUSFLAT 2009 conference, 2009, pp. 1559–1564.

[23] L. Zadeh, A computational approach to fuzzy quantifiers in natural languages, Computers & Mathematics with Applications 9 (1) (1983) 149–184.

[24] O. Cordón, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases, Vol. 19, World Scientific, 2001.

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

[26] D. Greene, S. Smith, Competition-based induction of decision models from examples, Machine Learning 13 (2) (1993) 229–257.

[27] A. Eiben, J. Smith, Introduction to evolutionary computing, Springer-Verlag, 2003.

[28] R. Scozzafava, B. Vantaggi, Fuzzy inclusion and similarity through coherent conditional probability, Fuzzy Sets and Systems 160 (3) (2009) 292–305.

[29] H. Lourenço, O. Martin, T. Stützle, Handbook of Metaheuristics, Kluwer Academic Publishers, 2003, Ch. Iterated Local Search, pp. 321–353.

[30] B. Gerkey, R. Vaughan, A. Howard, The player/stage project: Tools for multi-robot and distributed sensor systems, in: Proceedings of the 11th International Conference on Advanced Robotics (ICAR), 2003, pp. 317–323.

[31] O. Cordón, F. Herrera, A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples, International Journal of Approximate Reasoning 17 (4) (1997) 369–407.

[32] J. Alcalá-Fdez, L. Sánchez, S. García, M. Del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, et al., Keel: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing 13 (3) (2009) 307–318.

[33] R. Setiono, L. Hui, Use of a quasi-newton method in a feedforward neural network construction algorithm, IEEE Transactions on Neural Networks 6 (1) (1995) 273–277.

[34] W. Venables, B. Ripley, Modern applied statistics with S, Springer-Verlag, 2002.

[35] A. Karatzoglou, A. Smola, K. Hornik, A. Zeileis, kernlab - An S4 Package for Kernel Methods in R, Journal of Statistical Software 11 (9) (2004) 1–20.

[36] B. Schölkopf, A. Smola, R. Williamson, P. Bartlett, New support vector algorithms, Neural Computation 12 (5) (2000) 1207–1245.

[37] G. Kim, W. Chung, K.-R. Kim, M. Kim, S. Han, R. H. Shinn, The autonomous tour-guide robot jinny, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), Vol. 4, IEEE, 2004, pp. 3450–3455.

[38] M.-Y. Shieh, J. Hsieh, C. Cheng, Design of an intelligent hospital service robot and its applications, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 5, IEEE, 2004, pp. 4377–4382.

[39] J. López, D. Pérez, E. Paz, A. Santana, Watchbot: A building maintenance and surveillance system based on autonomous robots, Robotics and Autonomous Systems 61 (12) (2013) 1559–1571.

[40] C. Gamallo, C. V. Regueiro, P. Quintía, M. Mucientes, Omnivision-based kld-monte carlo localization, Robotics and Autonomous Systems 58 (3) (2010) 295–305.

[41] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, W. Burgard, A

navigation system for robots operating in crowded urban environments, in: Proccedings of the IEEE International Conference on Robotics & Automation (ICRA), 2013.

[42] A. González-Sieira, M. Mucientes, A. Bugarín, A state lattice approach for motion planning under control and sensor uncertainty, in: Proceedings of the First Iberian Robotics Conference (ROBOT), Madrid (Spain), 2013, pp. 247–260.

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

[44] A. Ben-David, A lot of randomness is hiding in accuracy, Engineering Applications of Artificial Intelligence 20 (7) (2007) 875–885.