NEURO-FUZZY TECHNIQUES TO OPTIMIZE AN FPGA EMBEDDED CONTROLLER FOR ROBOT NAVIGATION

I. BATURONE[1], A. GERSNOVIEZ[2] AND A. BARRIGA[1]

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This paper describes how low-cost embedded controllers for robot navigation can be obtained by using a small number of *if-then* rules (exploiting the connection in cascade of rule bases) that apply Takagi-Sugeno fuzzy inference method and employ fuzzy sets represented by normalized triangular functions. The rules comprise heuristic and fuzzy knowledge together with numerical data obtained from a geometric analysis of the control problem

that considers the kinematic and dynamic constraints of the robot. Numerical data allow tuning the fuzzy symbols used in the rules to optimize the controller performance. From the implementation point of view, very few computational and memory resources are required: standard logical, addition, and multiplication operations and a few data that can be represented by integer values. This is illustrated with the design of a controller for the safe navigation of an autonomous car-like robot among possible obstacles towards a goal configuration. Implementation results of an FPGA embedded system based on a general-purpose soft processor confirm that percentage reduction in clock cycles is drastic thanks to applying the proposed neuro-fuzzy techniques. Simulation and experimental results obtained with the robot confirm the efficiency of the controller designed. Design methodology has been supported by the CAD tools of the environment Xfuzzy 3 and by the Embedded System Tools from Xilinx.

* [1]*Dept. de Electrónica y Electromagnetismo, Univ. of Seville, and the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville (Spain)*
[1] [2]*Dept. de Arquitectura de Computadores, Electrónica y Tecnología Electrónica, Univ. of Cordoba, Cordoba (Spain)*

1 INTRODUCTION

Fuzzy rule-based systems have been successfully used in many control applications thanks to their capability to deal with the heuristic knowledge of an expert that is expressed linguistically [1]-[4]. Linguistic if-then rules including imprecise and maybe ambiguous terms are translated rather easily into if-then rules including fuzzy sets. The capability of fuzzy systems to manage linguistic information facilitates not only the development of controllers but also their debugging and maintenance. Due to the great use of fuzzy systems in control engineering, different types of implementations for these systems have been proposed in the literature [5]. These approaches range from software implementations to hardware realizations by ASICs (Application Specific Integrated Circuits) or FPGAs (Field Programmable Gate Arrays).

The development of complex systems for industrial control applications demands high speed, low power and area consumption, and low cost. The solution to satisfy these requirements is the implementation as embedded systems [6]. Implementation on ASICs satisfies these requirements, but due to the high initial engineering cost, ASICs are adequate when they are manufactured in high quantity. A good alternative is the implementation on FPGAs. An FPGA can be fully programmable to generate a specific hardware that matches the requirements of the user. The great advantages of these devices compared to ASICs are its flexibility, a shorter time-to-market and a lower cost if the number of fabricated devices is small. Modern FPGA families include a high number of specific resources and provide powerful and friendly CAD tools that allow the development of complex embedded systems [7]-[19]. FPGA manufacturers also allow the implementation of efficient 32-bit RISC processors, such as the MicroBlaze system from Xilinx and the Nios processor from Altera.

The continuous evolution of programmable devices has increased the use of FPGAs as platforms for the development of fuzzy systems [12]-[19]. Some of these FPGA implementations of fuzzy controllers are used to control autonomous mobile robots [16]-[19]. Current research in robotics concerns with multi-robot heterogeneous scenarios to execute missions that require safe and reliable cooperation. This means that autonomous robots not only have to navigate safely in real time (low processing time), but also communicate with other robots and participate into collaborative tasks during maybe long time (low power consumption), this is carried out on hardware platforms of maybe limited resources (let us think, for example, in micro-robots or unmanned aerial vehicles with low payload) [20].

This paper describes the use of neuro-fuzzy techniques to optimize the design of an embedded controller for the safe navigation of a car-like autonomous robot among possible obstacles towards a goal destination. The main objective of the proposed approach is to achieve efficient embedded implementations in terms of memory resources and operation speed. In order to reduce implementation costs, the controller contains several modules that, in turn, are composed of simple rule bases connected in cascade; the rules employ normalized triangular membership functions to represent the antecedents; and the inference mechanism employed is Takagi-Sugeno [21].

Many authors exploit the numerical data obtained from trajectories provided by an expert to apply neural-like learning to fuzzy controllers [3]-[4]. However, these trajectories do not usually correspond to the shortest paths. In the proposal herein, numerical data employed correspond to paths of

near minimum length obtained from a geometric analysis of the problem (which considers the nonholonomic constraints of the robot). The use of numerical data to design controllers that meet nonholonomic constraints has been also exploited in other works, such as [22]-[25]. In particular, numerical data associated to the shortest paths affordable for a car-like robot have been employed in [25] to solve a parking problem without obstacles. A similar approach is applied herein to a more complex problem. The data help in generating navigation paths of near minimum lengths when no obstacles are detected and, in presence of obstacles, minimum deviation from these paths. A novelty and advantage of the approach described herein is that automatic learning maintains the linguistic meaning of the fuzzy rules and optimizes the implementation into embedded systems. Interpretable if-then rules are very interesting to facilitate human-robot communication [26].

The paper is organized as follows. Section 2 describes the navigation problem of a car-like autonomous robot. Its subsections summarize the kinematic and dynamic considerations to obtain a controller that generates paths of near minimum length. Section 3 describes how to use this controller as a reference to generate the numerical learning data to adjust the symbols of a fuzzy controller with the objective of obtaining a much more efficient implementation. The design methodology of the proposed neuro-fuzzy solution is aided by the description, identification, verification, and learning CAD tools of Xfuzzy 3 (an environment to design neuro-fuzzy controllers available at [27]). Section 4 evaluates the behavior of the designed neuro-fuzzy controller with simulation and experimental results obtained with a car-like robot designed and built at the Escuela Superior de Ingenieros, University of Seville, Spain. Implementation results of both the reference and the neuro-fuzzy controllers into an embedded system based on MicroBlaze on a Virtex FPGA from Xilinx are compared and evaluated in Section 4. The superiority of hierarchical Takagi-Sugeno fuzzy inference systems to other approximator systems, such as radial basis function (RBF) neural networks, is also illustrated in Section 4. Finally, Section 5 gives the conclusions.

2 THE FIRST DESIGN OF THE FPGA CONTROLLER

The configuration of a car-like robot can be given by the position of the back wheels axle midpoint with regards to a global coordinate system, (x, y); its orientation, ϕ; the curvature defined by the front wheels, γ; and its speed, v. The navigation problem consists of generating a collision-free trajectory from an initial configuration (x, y, ϕ, v, γ) to a goal one, which is $(0, 0, 180°, 0, 0)$ in the global coordinate system that have been selected herein (Figure 1). In the first stage of the controller design, a mathematical description of the collision-free navigation problem is performed to obtain a controller that generates paths of near minimum length. A direct implementation of this controller on an embedded system based on MicroBlaze is developed for its later comparison with the neuro-fuzzy solution.

2.1 The navigation algorithm

A basic task to be performed by an autonomous robot is to navigate safely among possible obstacles towards a goal destination. Navigation problems are usually solved in two steps. Firstly, a reference trajectory is provided by a path planner, and, secondly, a path-tracking controller tries to keep

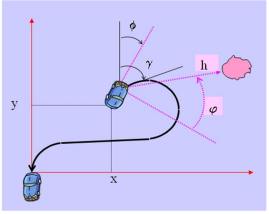

Figure 1: The navigation problem

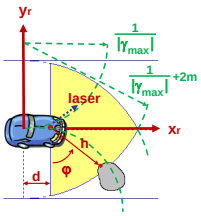

Figure 2: Area of unavoidable obstacles

the robot as close as possible to the reference trajectory. A large number of methods for solving the path planning problem have been reported in the literature (such as roadmap, cell decomposition and potential field methods) [28]. Global path planners offer the advantage of generating reference paths that optimize particular criterion (e.g. minimum time, minimum energy, and shortest length), since they employ global information about the environment. Their main disadvantage is that they usually apply computationally costly algorithms and, hence, they cannot accommodate rapid changes in the environment. In the other side, reactive planners execute simple behaviors in direct response to sensory information. Since they deal with local instead of global information, reactive methods are widely used for implementing real-time navigation through partially unknown and dynamic environments.

Another issue is that global path planners usually provide reference paths made up of directly connected straight lines that could not be easily followed by car-like robots, which always should connect straight lines by arcs of circle.

The controller designed herein addresses a twofold objective: it is reactive to cope with dynamically changing environment in real time and acts as a path-tracking controller that keeps the robot as close as possible to a car-like feasible reference path.

In the typical decomposition of a mobile robot control system into functional modules (sensors, perception, modeling, planning, task execution, motor control and actuators) [29], the module designed herein would be in charge of reactive obstacle avoidance and path-tracking tasks. It can be combined with a global path planner in charge of modeling and planning in order to optimize global criterion (taking into account the overall obstacles spatial configuration and characteristics of the whole space where the robot can move). A review of such hybrid architectures that combine global (deliberative) with local (reactive) modules can be seen in [30]. For example, a global fuzzy behavior control is proposed in [17] to integrate several local modules. However, this is out of the scope of this paper, which will focus on the design of the reactive obstacle avoidance and path-tracking module.

No model about the environment is used nor constructed by the designed reactive controller but only the information provided by the robot sensors. In particular, the sensor considered is a 2-D laser placed at the robot front part. The laser performs a scan of up to 180 degrees of the space in the front of the robot and identifies the points of possible obstacles by their distance, h, and sweeping angle, φ (Figure 1). Of course, the robot could drive backward in a safe way by making use of sensors placed at the robot back part that scan the space in the back of the robot. However, for the sake of clarity, the robot is considered herein to drive always forward or to stop.

The coarse structure of the reactive controller is obtained from heuristic knowledge expressed linguistically, as follows:

1. If there is an obstacle *very close* to the robot, the robot should stop to avoid collision[1].
2. If there is an obstacle *close* to the robot, a maneuver to avoid it should be carried out, as follows[2]:
   a) If the obstacle is *on the right (left)* then *turn to the left (right)*.
   b) If the obstacle is *in front of* the robot then:
      - Turn to the same side as in the previous control cycle (provided the obstacle was already detected).
      - If this is the first time the obstacle is detected, apply the turning sign as no obstacle is detected.
3. If the obstacles are *far* from the robot, there is no need to avoid them (by the moment) and the robot should navigate towards the goal configuration by the *shortest* path.

Several symbolic concepts appear in these rules (depicted in italic fonts) that need specification. They might be defined by membership functions based on heuristics as well as trial and error approach. Doing that, the paths obtained will not be as good as they could be. Our proposal is to consider the dynamic and kinematic constraints of the robot so as to optimally define the symbolic concepts. In particular, the car-like nonholonomic constraints make that the movement direction must always be tangent to the trajectory and the turning radius is mechanically limited to a minimum value, which is equivalent to say that the robot curvature is upper bounded. In the absence of obstacles, the shortest paths for a car-like robot consist of a finite sequence of two elementary components: arcs of circle (with minimum turning radius) and straight line segments [31]-[32]. In an environment cluttered by obstacles, it is proven in [33] that shortest paths consist only of arcs of a circle of minimum turning radius, line segments, and pieces of boundaries. Hence, an obstacle will be *very close* if the radius of the arc of circle to avoid it should be smaller than the minimum. An obstacle will be *close* and should be avoided if it will be inside the area that will be swept by the robot when it travels along its path. In other case, the obstacle is *far* and the controller has to provide the shortest path made of a finite sequence of arcs of circle with minimum turning radius and straight line segments (as illustrated in Figure 1). This is analyzed more in detail in the following.

2.2 Very close obstacles

Let us consider a coordinate system (x_r, y_r) attached to the robot, whose origin is placed at the back wheels axle midpoint (Figure 2). The coordinates on an obstacle referred to that system are the following:

$$x_{ro} = d + h \cdot \sin\varphi \qquad y_{ro} = -h \cdot \cos\varphi \tag{1}$$

where d is the distance of the laser to the back wheel axle (1.65m in the robot considered in the experiments).

1 If the robot would be provided with sensors in its rear, this would not be the final action and the robot would drive backward, as commented above.

2 In case this local module is combined with a global controller, the turning sign of the maneuver defined by rules 2.a and 2.b could be determined by the global controller.

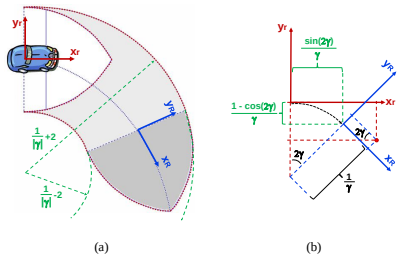

Figure 3: (a) Area of *close* obstacles. (b) Transformation between coordinate systems (x_r, y_r) and (x_R, y_R).

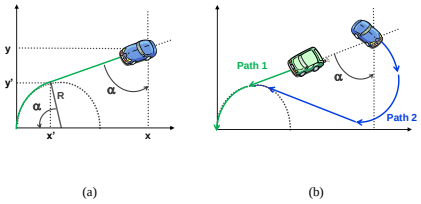

Figure 4: (a) Minimum-length paths of car-like robots. (b) Relation between the angle α and the robot position.

Taking into account that the robot considered in our analysis has a width of 1m and the maximum speed considered is 1m/s, 2m at both sides of the robot should be free of obstacles for safety purposes. An obstacle will be *very close* and, hence, unavoidable, if it enters the forbidden area (shadowed area in Figure 2). Let us consider an obstacle placed on the right of the robot ($\varphi \in [0°, 90°]$), as shown in Figure 2. It will be *very close* if the robot cannot avoid it by turning to the left with an arc of circle with minimum turning radius ($R = 1/|\gamma_{max}|$), leaving 2m of free space:

$$x_{ro}^2 + \left(y_{ro} - \frac{1}{|\gamma_{max}|}\right)^2 < \left(\frac{1}{|\gamma_{max}|} + 2\right)^2 \qquad \text{if } \varphi \in [0°, 90°] \tag{2}$$

Similarly, if the obstacle is placed on the left of the robot ($\varphi \in [90°, 180°]$), it will be *very close* if the robot cannot avoid it by turning to the right:

$$x_{ro}^2 + \left(y_{ro} + \frac{1}{|\gamma_{max}|}\right)^2 < \left(\frac{1}{|\gamma_{max}|} + 2\right)^2 \qquad \text{if } \varphi \in [90°, 180°] \tag{3}$$

Grouping equations (2) and (3), and substituting x_{ro} and y_{ro} by equation (1), the area of *very close* or unavoidable obstacles is defined by:

$$(d + h \cdot |\sin\varphi|)^2 + \left(h \cdot |\cos\varphi| + \frac{1}{|\gamma_{max}|}\right)^2 < \left(\frac{1}{|\gamma_{max}|} + 2\right)^2 \tag{4}$$

Equation (4) has been selected to describe *very close* obstacles, since it uses the variables h and φ provided directly by the laser. Hence, no processing of laser data is required.

2.3 Close obstacles

An obstacle will be *close* if it will be inside the safety area that will be swept by the robot when it travels along its path. Taking into account the current robot curvature, γ, a first condition for an obstacle to be *close* is to be less than 2m on the right or on the left along the arc of circle to be described by the robot (Figure 3a), as follows:

$$\left(\frac{1}{|\gamma|} - 2\right)^2 < x_{ro}^2 + \left(y_{ro} + \frac{1}{|\gamma|}\right)^2 < \left(\frac{1}{|\gamma|} + 2\right)^2 \tag{5}$$

The safety area should also contain 2m ahead in the driving direction. Hence, let us consider a coordinate system (x_R, y_R) displaced 2m in the

driving direction and rotated an angle of 2γ with respect to the coordinate system (x_r, y_r), as illustrated in Figure 3b. The coordinates of an obstacle referred to this system are the following:

$$\begin{aligned} x_{Ro} &= \left(x_{ro} - \frac{\sin 2\gamma}{\gamma}\right)\cos 2\gamma - \left(y_{ro} + \frac{1-\cos 2\gamma}{\gamma}\right)\sin 2\gamma \\ y_{Ro} &= \left(x_{ro} - \frac{\sin 2\gamma}{\gamma}\right)\sin 2\gamma + \left(y_{ro} + \frac{1-\cos 2\gamma}{\gamma}\right)\cos 2\gamma \end{aligned} \tag{6}$$

A second condition for an obstacle to be *close* and, hence, to be avoided is that it becomes a *very close* obstacle in the future (dark shadowed area in Figure 3a). This condition is easier to be expressed in the coordinate system (x_R, y_R). Similarly to equations (2) and (3), it will be *very close* if their position with regards to the displaced and rotated coordinate system verifies that:

$$x_{Ro}^2 + \left(y_{Ro} + \frac{1}{|\gamma_{max}|}\right)^2 < \left(\frac{1}{|\gamma_{max}|} + 2\right)^2 \tag{7}$$

Finally, the third condition for a *close* obstacle is that, currently, it is not *very close* and, hence, it does not verify equation (4).

2.4 Obstacle avoidance

The minimum magnitude of the curvature to avoid an obstacle identified as *close* is determined by the closest point of that obstacle (whose coordinates will be named h_M and φ_M herein). Taking into account that a reference curvature is not adopted instantaneously by the robot but has a delay, a 4-m safety corridor has been considered for selecting the minimum value of the curvature. With such selection, the dynamic features of the robot considered in the experiments allow the 2-m width corridor free of obstacles. Hence, the minimum magnitude of the curvature (similarly to equation (4)) is given by:

$$(d + h_M \cdot |\sin\varphi_M|)^2 + \left(h_M \cdot |\cos\varphi_M| + \frac{1}{|\gamma|}\right)^2 = \left(\frac{1}{|\gamma|} + 4\right)^2 \tag{8}$$

From the formula above, the value of $|\gamma|$ can be obtained as:

$$|\gamma| = \frac{8 - 2h_M \cdot |\cos\varphi_M|}{d^2 + h_M^2 + 2h_M \cdot d \cdot |\sin\varphi_M| - 16} \tag{9}$$

The sign of the curvature when an obstacle is avoided is given by the second if-then rule explained in Subsection 2.1 (the criterion adopted is that $|\gamma| = \gamma$, in the case of turning to the right, and $|\gamma| = -\gamma$, in the case of turning to the left).

2.5 Navigation toward the goal

If there are no obstacles, the robot should navigate toward the goal (0,0,180°,0,0) by the shortest path. The shortest paths for a car-like vehicle consist of a finite sequence of two elementary components: arcs of circle (with minimum turning radius) and straight line segments, as was proved by Dubins [31]. Analyzing the shortest paths geometrically, it can be found that there is an angle, α, which defines the orientation of the straight segment which contains the (x, y) point of the current robot configuration and is tangent to the arc of circle that defines the end of the path (Figure 4a). Depending on the

difference between this angle and the current robot orientation, the robot will turn to right or left or will not turn. The value of the angle α depends on x and y as follows (as can be seen in Figure 4b):

$$\begin{array}{l} \tan\alpha = \frac{x-x'}{y-y'} = \frac{x-(R-R\cdot\cos\alpha)}{y-R\cdot\sin\alpha} \\ \text{with } \ (|x|-R)^2+y^2 \geqslant R^2 \end{array} \tag{10}$$

where R is the minimum turning radius.

If the formula above is elaborated, the value of α can be obtained as follows:

$$\alpha = (x)\cdot\arccos\left(\frac{R(R-|x|)+y\sqrt{x^2+y^2-2R|x|}}{x^2+y^2-2R|x|+R^2}\right) \tag{11}$$

3 OPTIMIZING THE DESIGN WITH NEURO-FUZZY TECHNIQUES

The direct implementation of equations (1) to (11) requires trigonometric functions (sin, cos and arccos), square root, and division. As discussed in Section 4, such equations are computationally costly for FPGA embedded controllers (when carried out by MicroBlaze processor using software routines). One solution is to accelerate by hardware the computation of such functions. Another approach to alleviate this cost is to simplify the equations to compute the safety areas. The latter is the approach proposed in [34], which is a little conservative solution that constructs a polygonal over-approximation of such areas and checks the emptiness of the constructed convex polygon with any obstacle by solving a linear programming problem. The novel approach proposed herein is to employ neuro-fuzzy techniques to simplify the computation of all the equations not only of the safety areas, and, in the case of safety areas, not providing a coarse over-approximation but a finer approximation. This is analyzed more in detail in the following.

3.1 Very close obstacles

The shadowed area in Figure 5b illustrates those obstacle positions (h, φ) that satisfy the nonlinear constraint (4). The proposal herein is to describe those positions with fuzzy rules, that is, to design a fuzzy classifier with two inputs, h and φ , and one output that specifies if the obstacle is *very close* or not adjusted to the training data provided by (4). The simplest structure that has been found for this fuzzy classifier is the hierarchical structure shown in Figure 5a. The fuzzy rule base, named *Interpolation* in the figure, is a zero-order Takagi-Sugeno system that interpolates, for each angle φ, the minimum value of h, h_{min}, that an obstacle should have to be considered as *not very close*. This rule base has been trained by the numerical data (φ, h_{min}) that verify (4) when condition *smaller than* is substituted by *equal to*. Levenberg-Marquardt algorithm (a second-order training algorithm that does not compute the Hessian matrix [35]) has been applied with the supervised learning tool of the Xfuzzy environment [36]. The rule base obtained contains 5 rules whose antecedents are: φ is *quite right*, *right*, *in front of*, *left*, and *quite left*; and whose consequents are, respectively: h_{min} is 1.7m, 1.5m, 2.1m, 1.5m, and 1.7m, as shown in Figure 5c. The upper part of Figure 5c illustrates how the designed fuzzy classifier approximates the

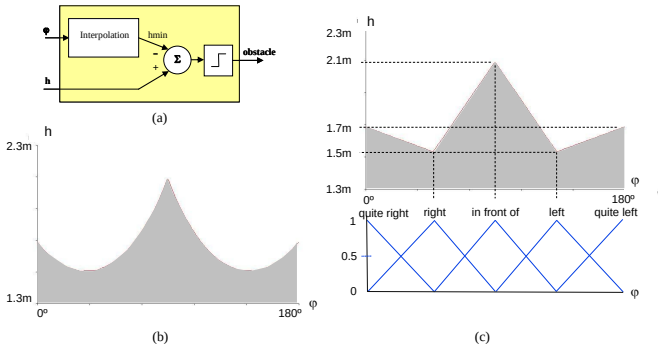

Figure 5: (a) Structure of the fuzzy classifier for *very close* obstacles. (b) The gray area is the area of *very close* obstacles according to (4). (c) Result provided by the fuzzy classifier.

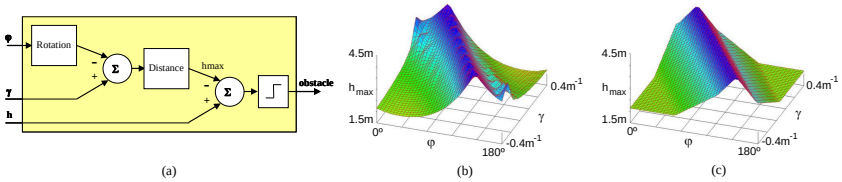

Figure 6: (a) Hierarchical system to classify obstacles as *close* or not. (b) Maximum value of h, h_{max}, of obstacles considering as *close* according to (5)-(7). (c) Approximation of h_{max} provided by the fuzzy classifier.

nonlinear area of *very close* obstacles by a polygonal area. The piecewise linear frontier in the upper part of the polygonal area is the output, h_{min}, provided by *Interpolation* rule base.

3.2 Close obstacles

Let us call h_{max} the maximum value of h that an obstacle can have to be considered as *close*. Depending on the angular position of the obstacle, φ, and the current curvature of the robot, γ, the value of h_{max} can be obtained from equations (5) and (7) when condition *smaller than* is substituted by *equal to*. Figure 6b illustrates the nonlinear relation between h_{max}, φ and γ. The proposal herein is to describe such relation with fuzzy rules and design a fuzzy classifier with three inputs (h, φ and γ) and one output, which specifies if the obstacle is *close* or not adjusted to the nonlinear constraints (5) to (7).

The structure selected for this fuzzy classifier is the hierarchical scheme shown in Figure 6a. This is very simple since it combines zero-order Takagi-Sugeno rule bases with only one input. Looking for a good trade-off between approximation and simplicity, 2 rules have been extracted for the rule base *Rotation* and 5 ones for the rule base *Distance*. These rule bases have been trained with the numerical data $(\varphi, \gamma, h_{max})$ illustrated in Figure 6b, using the CAD tools of Xfuzzy. The capability of the learning tool of Xfuzzy to adjust hierarchical systems has been exploited in this case [36]. The rule base named *Rotation* provides an output which is a linear function of the input φ. The rule base named *Distance* provides a piecewise linear interpolation of h_{max}, depending on the difference between the current curvature of the robot and the output of the rule base *Rotation*, $\gamma - \text{Rotation}(\varphi)$. Such

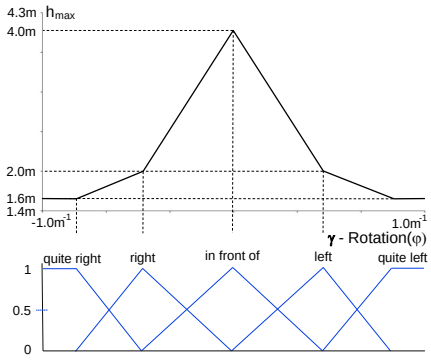

Figure 7: Input-output behavior of the rule base *Distance* in Figure 6a.

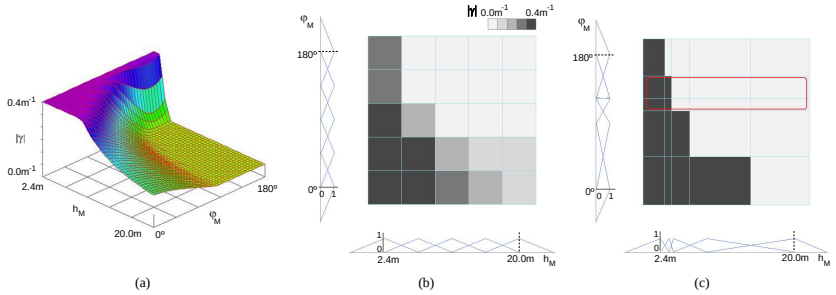

Figure 8: (a) Curvature to avoid obstacles according to (9). (b) 25 rules extracted by Wang-Mendel algorithm with a regular 5x5 grid. (c) The 5x5 grid is adjusted to the data by supervised learning.

difference can be understood as the angular position referred to the robot that the obstacle will have in the future. This is why the antecedents in the 5 rules of *Distance* rule base have been named as *quite right*, *right*, *in front of*, *left*, and *quite left*. Their corresponding consequents are: h_{max} is 1.6m, 2m, 4m, 2m, and 1.6m, as shown in Figure 7.

Figure 6c illustrates how the designed fuzzy classifier approximates the nonlinear relation between h_{max}, φ and γ with piecewise linear relations.

3.3 Obstacle avoidance

The minimum magnitude of the curvature to avoid an obstacle is a nonlinear function, f, of the closest point of the obstacle (h_M and φ_M), as stated in (9). Exploiting the symmetry of the problem, it can be considered that $\gamma = f(\varphi_M, h_M)$, when turning to right and $\gamma = f(180 - \varphi_M, h_M)$, when turning to left. Figure 8a shows the γ values (when turning to right) versus φ_M and h_M. The proposal herein is to approximate such nonlinear relation with fuzzy rules, that is, to design a fuzzy approximator with two inputs, h_M and φ_M, and one output that specifies the magnitude of the curvature. In this case, a non-hierarchical system has been selected because its complexity is somewhat lower (5 instead of the 9 rules extracted in the hierarchical system) achieving a better approximation. The procedure to obtain the non-hierarchical system follows the techniques described in [37]. Firstly, 5 membership functions have been selected as a good initial number to cover uniformly the universes of discourse of the h_M and φ_M variables, as shown in Figure 8b. By using the numerical data (h_M, φ_M, γ) shown

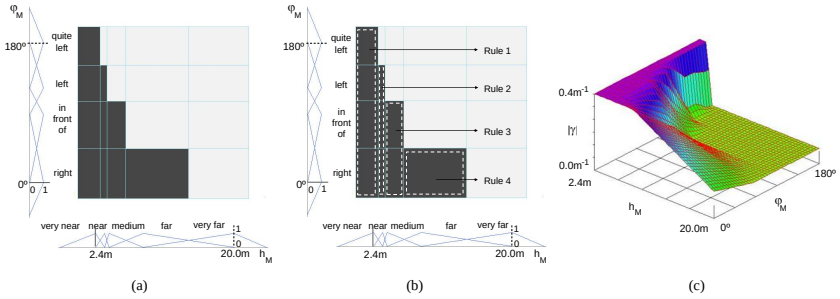

Figure 9: (a) The 25 rules are simplified to 20 rules by merging membership functions in φ_M. (b) The 20 rules are simplified to 5 rules by applying Fuzzy Tabular Simplification method [37]. (c) Approximation of $|\gamma|$ to avoid obstacles provided by the 5-rule fuzzy system.

in Figure 8a and applying Wang-Mendel identification algorithm with the identification tool of Xfuzzy [37], 25 zero-order Takagi-Sugeno rules (illustrated as a matrix in Figure 8b) have been obtained. Again the numerical data (h_M, φ_M, γ) have been employed to apply supervised learning to adjust the membership functions in the antecedents and the singleton values in the consequents of these 25 fuzzy rules so as to reduce the approximation error. The result of such learning is illustrated in Figure 8c. By using the simplification tool of Xfuzzy, two membership functions covering φ_M can be merged, as shown in Figure 8c, so that the 25 rules are reduced to 20. Finally, by applying the Fuzzy Tabular Simplification method available in the simplification tool of Xfuzzy [37], the 20 rules can be reduced to 5 rules, 4 of them describing when turning right at maximum and the other concluding no turning for the rest of situations. This is illustrated in Figure 9a and b. The obtained zero-order Takagi-Sugeno system, which decides how much turning to right (once turning to right has been decided), forces the robot to turn more as more dangerous the obstacle is for the way the robot is going to take immediately. The closer the obstacle is, the more the robot will turn to right. The fuzzy rules express the following conditions to turn right at maximum (imagine the robot is going ahead or to its right):

1. If obstacle is *very near*.
2. If obstacle is *near* and it is not *quite on the left*.
3. If obstacle is at *medium* distance and it is *in front of* or *on the right*.
4. If obstacle is *far* and it is *on the right*.

In other cases, the robot should keep straight ahead.

Since these rules employ fuzzy concepts, the γ values provided by this system do not switch abruptly but they vary smoothly between maximum turning to right and zero, as shown in Figure 9c.

3.4 Navigation toward the goal

If no obstacles are detected, the robot will turn to right or left or will not turn depending on the difference between the current robot orientation and the angle α (defined in (11) as a nonlinear function of the robot position, x and y). The approach herein is to use a fuzzy system that approximates the value of the robot curvature, as described in [25]. A hierarchical structure

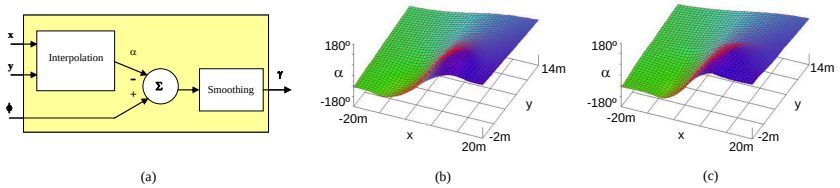

Figure 10: (a) Fuzzy system to navigate without obstacles. (b) Angle α according to (10)-(11). (c) Result provided by the fuzzy system.

Table 1: Rules in the rule base *Interpolation* of Figure 10

IF x **is**	**AND** y **is**	**THEN** α **is**
zero	zero	$51.3 + 8.2x + 2.8y$
zero	near	$8.8x$
zero	far	$-1.6x$
positive-near	zero	$103.3 + 4.9x + 7.8y$
positive-near	near	$29.2 + 5.3x - 1.5y$
positive-near	far	$-9.4 + 0.3x + 1.6y$
positive-far	zero	$-51.3 + 8.2x - 2.8y$
positive-far	near	$-11.2 + 4.1x + 1.9y$
positive-far	far	$3.7 - 0.7x + 2.8y$

with two rule bases connected in cascade, as shown in Figure 10a, has been selected. The first rule base provides approximately the value of the angle α, depending on the input variables x and y. This module has been adjusted by using numerical data (x, y, α), obtained from (11) and from the following equation (which achieves continuity for the rest of positions):

$$\alpha = (x) \cdot \arccos\left(1 - \frac{|x|}{R}\right) \qquad \text{if } \; (|x| - R)^2 + y^2 \leqslant R^2 \tag{12}$$

Exploiting the symmetry of the problem, $\alpha(x, y) = -\alpha(-x, y)$, the neuro-fuzzy system has been trained for positive values of x. The best system found in terms of approximation error and simplicity is a first-order Takagi-Sugeno system with 9 rules. These rules are shown in Table 1, where the fuzzy sets (*zero*, *near*, etc.) in the antecedents are represented by triangular membership functions similar to those in Figure 5c and Figure 7. Figure 10b shows the values of α versus x and y according to (11) and (12), and Figure 10c shows the approximation provided by the rule base *Interpolation*.

The best option found for the second rule base *Smoothing* is a zero-order Takagi-Sugeno system with 2 rules, exploiting symmetry, $\gamma(\phi - \alpha) = -\gamma(\alpha - \phi)$. These two rules are the following:

1. If $(\phi - \alpha)$ is *small-positive* then γ is $0.4 m^{-1}$.
2. If $(\phi - \alpha)$ is *large-positive* then γ is $0.07 m^{-1}$.

4 RESULTS AND DISCUSSION

4.1 Simulation and experimental results

The initial and neuro-fuzzy-based controllers have been described with the tool *xfedit* of Xfuzzy 3, which allows connecting fuzzy and non fuzzy rule bases as well as arithmetic modules.

Simulations have been carried out with the tool *xfsim*. It simulates the controller working in a closed loop with a model of the robot that considers its kinematic and dynamic features. The robot model (which contains the models of its sensors) provides the new configuration of the robot and the new information given by the laser. This model is introduced in *xfsim* as a Java class.

Figure 11 shows several examples of how the robot is controlled to reach the goal configuration $(x, y, \phi, \gamma, v) = (0, 0, 180°, 0, 0)$ by both the initial and neuro-fuzzy controllers. Paths provided by both controllers are similar, thus confirming that approximation achieved by the neuro-fuzzy controller is adequate. An important difference between both controllers is that the neuro-fuzzy controller imposes much smoother changes in the curvature and, hence, their commands are better followed by the robot actuators. This means that the neuro-fuzzy controller performs better than the original one if the robot moves at higher speed and/or the path to perform contains more different curvatures. For example, with the obstacle distribution illustrated in Figure 11c, it has been proven that the robot is able to avoid the obstacles at higher speed when controlled by the neuro-fuzzy solution than by the original one.

As illustrated in Figure 11a, if no obstacles are detected, the robot goes to the goal by a path of near minimum length because the path is made up of arcs of circles with almost minimum turning radius and straight line segments. Only the obstacles that interrupt the path the robot is going to take are avoided. They are avoided by minimum deviations from the original path, as shown in Figure 11b, c and d. These figures also illustrate that, once obstacles have been avoided, the controller recalculates the new path of near minimum length to the goal. The scenario of Figure 11c is particularly difficult to be solved with such a path of near minimum length. If the controller would not have taken into account the car-like features of the robot, either the path generated would have been of longer length or even no free path would have been found among the obstacles.

Another interesting case is considering moving obstacles. Figure 12 shows the simulation case of Figure 11d adding an emergent and moving obstacle.

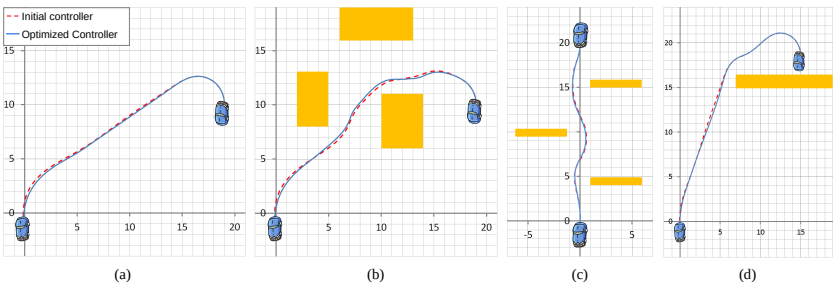

Figure 11: Simulation results of navigation with and without obstacles, provided by initial and neuro-fuzzy (optimized) controllers (axis units are in meters).

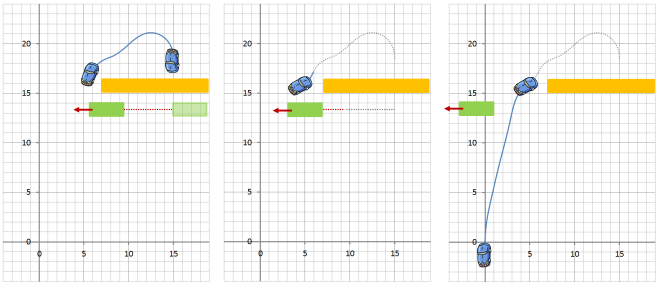

Figure 12: Simulation results of navigation with an emergent moving obstacle.

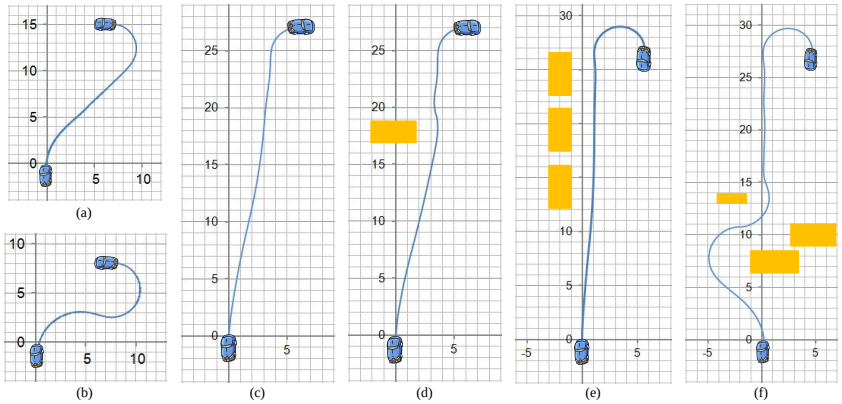

Figure 13: Experimental results of navigation with and without obstacles obtained with the neuro-fuzzy controller (axis units are in meters).

As can be seen in Figure 12a, the robot does not see the moving obstacle untill it passes the wall. Figure 12b shows that the robot tries to avoid the obstacle but it is too late to perform an avoiding maneuver and the *very close obstacles* module makes the robot stop. In Figure 12c, once the obstacle is gone, the robot continues to the configuration goal.

We have verified the behavior of the designed controller with the car-like robot shown in Figure 15. This robot is an electrical vehicle provided with a set of sensors and actuators that make it capable of autonomous navigation. The inputs x, y, and ϕ, required by the controller are obtained by odometry from encoders and a gyroscope. The inputs h and φ are obtained from a 2-D laser LMS 220-30106. In the control example, the electrical motors of the robot have to receive the curvature and speed commands from the controller. In addition, the motor control card reads the direction and traction encoders of the engines to estimate the variables v and γ of the robot configuration. Safe navigation among obstacles is particularly challenging with this robot since it does not have an electrically controlled brake, but only its speed can be controlled. In our experiments, speed is simply set to zero when unavoidable obstacles are detected or goal configuration is going to be reached, otherwise speed is constant.

Figure 13a-c confirms experimentally that the robot carries out paths of near minimum length in the absence of obstacles (combination of arcs of circle of near minimum radius and straight lines). The experiment in Figure 13b is particularly challenging because the robot is very close to the goal position and has to concatenate two arcs of circle of almost minimum turning radius. A controller that does not take into account the car-like features of

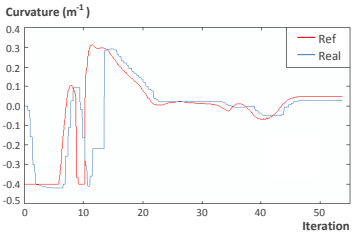

Figure 14: Evolution of reference curvature (Ref) provided by the neuro-fuzzy controller and actual curvature (Real) of the robot.

Figure 15: Experimental results of navigation with moving obstacles.

the robot would provide a longer path or even would fail in this scenario. Figure 13d-e shows the minimum deviation from the original path when an obstacle is detected. Such minimum deviation allows the robot navigating safely along narrow free spaces, as illustrated in Figure 13f. A successful navigation between the obstacles shown in Figure 13f and a successful consecution of the goal configuration is very difficult for a controller not optimized for car-like robots. It is achieved by the neuro-fuzzy controller because it provides a control action that can be tracked by the robot actuator. This can be seen in Figure 14, which details a typical evolution of robot curvature when avoiding an obstacle (this case corresponds to the trajectory in Figure 13d).

Finally, Figure 15 shows two cases of navigation with moving obstacles. In Figures 15a-f a pedestrian walks toward the path of the robot. As can be seen, the robot avoids the pedestrian and then it returns to the original path. In Figures 15g-l a pedestrian suddenly appears in front of the robot. The *very close obstacles* module makes the robot stop, avoiding the collision (Figure 15h). Once the pedestrian walks away, the robot returns to the path (Figures 15i-l).

4.2 Implementation results

The initial controller described in Section 2 that applies three navigation fuzzy rules whose symbols are described by equations (1) to (11) has been executed by an embedded system based on MicroBlaze and implemented in a Virtex II-Pro FPGA from Xilinx [38]. MicroBlaze is a 32-bit RISC Harvard architecture soft processor core with a rich instruction set optimized for embedded applications. In the system designed, MicroBlaze processor includes a floating point unit so as to carry out equations (1) to (11). A timer is added as peripheral in order to evaluate the number of clock cycles invested by the processor in the four main modules of the control algorithm: evaluating *very close* obstacles (module 1) and *close* obstacles (module 2), avoiding

Table 2: Number of clock cycles required by the initial and optimized (neuro-fuzzy) embedded controllers

Controller	Module 1	Module 2	Module 3	Module 4
Initial (float)	58642	236157	65024	47145
Neuro-fuzzy (float)	2123	5370	4580	21035
Neuro-fuzzy (int)	44	67	141	142

Table 3: Computation time (in seconds) of embedded controllers using 50MHz of working frequency

Controller	Module 1	Module 2	Module 3	Module 4
Initial (float)	1.17e-3	4.72e-3	1.30e-3	0.94e-3
Neuro-fuzzy (float)	42.46e-6	107.4e-6	91.6e-6	420.7e-6
Neuro-fuzzy (int)	0.88e-6	1.34e-6	2.82e-6	2.84e-6

close obstacles (module 3), and navigating toward the goal (module 4). The input variables considered by each module (for example h and φ in module 1, or φ, γ and h in module 2) have been swept considering their universes of discourse (for example φ has been swept from 0° to 180°) in order to evaluate the number of clock cycles required by all the possible situations (all possible values at their inputs). The averages of those numbers for each module are shown in the first row of Table 2.

The proposed neuro-fuzzy approach allows a good trade-off between memory resources and computation time. Regarding memory, the parameters required to be stored are the partition values of the input universes of discourse and the consequent values, which is a small number since there is a small number of rules (28 rules counting the four modules). This is much more efficient than storing in a look-up table the possible control actions for all the possible input values. Regarding computation time, the designed neuro-fuzzy controller requires only logical, addition, and multiplication operations, while the direct implementation of the equations (1) to (11) (as shown in Section 2) also requires more complex mathematical functions. For any kind of processor, our proposal requires much fewer clock cycles to compute the control value, especially if the processor has low computational resources. Table 2 supports this affirmation by showing the number of clock cycles of the same embedded system based on MicroBlaze processor designed for the initial controller. The C code corresponding to the four modules required by the neuro-fuzzy controller has been executed on the processor using float variables. The second row in Table 2 shows the cycles (also the average calculated with all the possible combinations of input values) when the modules are implemented as neuro-fuzzy systems. The percentage in clock cycle reduction is higher than 93% in three of the modules. In the Module 4, reduction is inferior (55.4%) mainly because one of the rule bases of the neuro-fuzzy solution uses a first-order (instead of a zero-order) Takagi-Sugeno inference method.

A further relevant advantage of our proposal is that computation can be performed with integer variables without a significant lost of precision (MicroBlaze works with 32 bits). The third row of Table 2 shows that using integer instead of float variables reduces clock cycles in average in two orders of magnitude (more than 96%). Hence our approach can be implemented efficiently in processors with fixed-point architectures. Table 3 shows a sim-

ilar comparison but in terms of computation time, considering a clock cycle of 20ns (50MHz of working frequency).

4.3 Discussion

Previous work on neuro-fuzzy and fuzzy approaches implemented on FPGAs to control autonomous car-like robots, such as [16], [17] and [19], learn or emulate the expert knowledge of a human driver and focus on parking problems (parking without obstacles in [16] and parking in a structured environment in [17]) or focus on path tracking without obstacles [19]. Other FPGA-based neuro-fuzzy approaches, such as [18], do not take into account the constraints of car-like robots. The approach proposed herein is able to provide paths of smaller length for car-like robots in unstructured environments with obstacles.

Previous work on neuro-fuzzy approaches that learn to provide paths of small length, such as [22], consider structured environments and are less suitable for implementation into embedded systems, because they employ Gaussian membership functions. Others, which are more suitable for embedded systems, such as [24] and [25], do not avoid obstacles. The solution proposed in [25], which is implemented in a Pentium processor working at 100MHz, takes about 2ms to compute the reference curvature. The solution proposed in [24], which is implemented in a DSP (Digital Signal Processor) working at 33MHz, takes about 20 s for a similar computation. In the approach proposed herein, Module 4, whose task is comparable to the solutions in [24] and [25], takes 2.84µs at 50MHz, which means the fastest solution based on an embedded processor. In addition, the other modules of the controller designed herein are even faster. This provides the robot with the capability of performing more complex tasks with the same hardware platform or, similarly, allows using platforms with limited resources to carry out safe navigation. In case that a fast solution is not required by the application, the operation frequency can be very low and, hence, a solution of very low power consumption can be obtained.

The neuro-fuzzy tecnique proposed herein approximates the equations of a reference controller. Other approximators could have been employed, such as neural networks based on radial basis functions (RBF NNs). Table 4 compares the number of rules and approximation error (expressed as root mean square error, RMSE) of the four modules designed herein with the number of neurons in the hidden layer of RBF NNs and their approximation error. The radial basis functions considered have been the product of isosceles triangles (to be similar in complexity to the triangular membership functions and the product connective considered in the neuro-fuzzy approach). The RBF NNs have been trained with the same numerical data, using also the supervised learning tool of the Xfuzzy environment. In the case of modules 1 and 3, which deal with two inputs, the RBF NNs need 30 and 16 neurons, respectively, to obtain the same approximation error, which means an increase in complexity of approximately 6 and 3.2 times more, respectively. In the case of modules 2 and 4, which deal with three inputs, the curse of dimensionality, typical of RBF NNs, begins to appear. Although using 150 and 180 neurons (in case of module 2 and 4, respectively), which means an increase in complexity of approximately 21.4 and 16.4 times, respectively, the approximation errors are worse.

Further research will involve the integration of the designed local controller into a hybrid architecture that contains global and other local mod-

Table 4: Comparison of the neuro-fuzzy proposal with neural networks based on radial basis functions (RBF NNs).

	Proposal		**RBF NNs**	
Module	No. rules	RMSE	No. hidden neurons	RMSE
Module 1	5	0.14	30	0.14
Module 2	7	0.09	150	0.15
Module 3	5	0.07	16	0.07
Module 4	11	0.08	180	0.15

ules so as to cope with complex navigation problems (including driving backward) and to deal with the considerable levels of noise present in complex robotic systems.

5 CONCLUSIONS

Neuro-fuzzy techniques have been proven very effective to optimize the design of an embedded controller for the free collision navigation of an autonomous car-like robot. The control algorithm can be executed rapidly or with low power consumption by even systems based on fixed-point processors because the rule bases employed contain a few number of rules, the number of antecedents is low, and Takagi-Sugeno inference is applied (with antecedents represented by normalized triangular functions). Despite its simplicity, the controller action is smooth and provides paths of near minimum length, as has been proven by simulation and experimental results. The methodology to design the neuro-fuzzy controller has been automated thanks to the CAD tools of the environment Xfuzzy 3, while the FPGA embedded system has been developed with the CAD tools from Xilinx.

Acknowledgements

This work was supported in part by the Spanish Ministerio de Economía y Competitividad under the Project TEC2011-24319 and by Junta de Andalucía under the Project P09-TEP-4479 (both with support from FEDER). The authors would like to thank the Department of *Ingeniería de Sistemas y Automática*, University of Seville, in particular to A. Ollero, J. Ferruz, and F. Leal for advice and collaboration to carry out the experiments with the car-like robot.

REFERENCES

[1] T.J. Ross, Fuzzy logic with engineering applications, 3rd Edition, Wiley, 2010.

[2] H.-P. Huang, J.-L. Yan and T.-H. Cheng, Development and fuzzy control of a pipe inspection robot, IEEE Transactions on Industrial Electronics 57 (3) (2010) 1088-1095.

[3] P. Rusu, E.M. Petriu, T.E. Whalen, A. Cornell and H.J.W. Spoelder, Behavior-based neuro-fuzzy controller for mobile robot navigation, IEEE Transactions on Instrumentation and Measurement 52 (4) (2003) 1335-1340.

[4] X. Wang and S.X. Yang, A neuro-fuzzy approach to obstacle avoidance for a nonholonomic mobile robot, in: Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM'03), 20-24 July 2003, vol. 1, pp. 29-34.

[5] K. Basterretxea and I. del Campo, Electronic hardware for fuzzy computation. In A. Laurent and M-J. Lesot (Eds.), Scalable fuzzy algorithms for data management and analysis: Methods and design, pp. 1-30, IGI Global, 2010.

[6] A. Malinowski and H. Yu, Comparison of embedded system design for industrial applications, IEEE Transactions on Industrial Informatics 7 (2) (2011) 244-254.

[7] E. Monmasson, L. Idkhajine, M.N. Cirstea, I. Bahri, A. Tisan and M.W. Naouar, FPGAs in industrial control applications, IEEE Transactions on Industrial Informatics 7 (2) (2011) 224-243.

[8] J.J. Rodriguez-Andina, M.J. Moure and M.D. Valdes, Features, Design tools and application domains of FPGAs, IEEE Transactions on Industrial Electronics 54 (4) (2007) 1810-1823.

[9] D. Majoe, L. Widmer, L. Ling, J. Kao and J. Gutknecht, A reconfigurable multi-core computing platform for robotics and e-Health applications, in: Proceedings of the 2012 IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS'12), 30 May - 1 June 2012, pp. 451-456.

[10] M.A. Cavuslu, C. Karakuzu and F. Karakaya, Neural identification of dynamic systems on FPGA with improved PSO learning, Applied Soft Computing 12 (9) (2012) 2707-2718.

[11] C.-Y. Chen, R.-C. Hwang and Y.-J. Chen, A passive auto-focus camera control system, Applied Soft Computing 10 (1) (2010) 296-303.

[12] P. Brox, I. Baturone and S. Sánchez-Solano, Fuzzy logic-based embedded system for video de-interlacing, Applied Soft Computing 14 (C) (2014) 338-346.

[13] F. Montesino-Pouzols, A. Barriga-Barros, D.R. López and S. Sánchez-Solano, Enabling fuzzy technologies in high performance networking via an open FPGA-based development platform, Applied Soft Computing 12 (4) (2012) 1440-1450.

[14] S. Sánchez-Solano, M. Brox, E. del Toro, P. Brox and I. Baturone, Model-based design methodology for rapid development of fuzzy controllers on FPGAs, IEEE Transactions on Industrial Informatics 9 (3) (2013) 1361-1370.

[15] R. Sepúlveda, O. Montiel, O. Castillo and P. Melin, Embedding a high speed interval type-2 fuzzy controller for a real plant into an FPGA, Applied Soft Computing 12 (3) (2012) 988-998.

[16] S. Sánchez-Solano, A.J. Cabrera, I. Baturone, F.J. Moreno-Velo and M. Brox, FPGA implementation of embedded fuzzy controllers for robotic applications, IEEE Transactions on Industrial Electronics 54 (4) (2007) 1937-1945.

[17] T.S. Li, S.-J. Chang and Y.-X. Chen, Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot, IEEE Transactions on Industrial Electronics 50 (5) (2003) 867-880.

[18] M.N. Mahyuddin, C.Z. Wei and M.R. Arshad, Neuro-fuzzy algorithm implemented in Altera's FPGA for mobile robot's obstacle avoidance mission, in: Proceedings of the 2009 IEEE Region 10 Conference (TENCON'09), 23-26 January 2009, pp. 1-6.

[19] S.G. Tzafestas, K.M. Deliparaschos and G.P. Moustris, Fuzzy logic path tracking control for autonomous non-holonomic mobile robots: Design of System on a Chip, Robotics and Autonomous Systems 58 (8) (2010) 1017-1027.

[20] J. Ferruz, V.M. Vega, A. Ollero and V. Blanco, Reconfigurable control architecture for distributed systems in the HERO autonomous helicopter, IEEE Transactions on Industrial Electronics 58 (12) (2011) 5311-5318.

[21] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man, and Cybernetics 15 (1) (1985) 116-132.

[22] K. Demirli and M. Khoshnejad, Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy sensor-based controller, Fuzzy Sets and Systems 160 (19) (2009) 2876-2891.

[23] N. Uchiyama, T. Hashimo, S. Sano and S. Takagi, Model-reference control approach to obstacle avoidance for a human-operated mobile robot, IEEE Transactions on Industrial Electronics 56 (10) (2009) 3892-3896.

[24] I. Baturone, F.J. Moreno-Velo, V. Blanco and J. Ferruz, Design of embedded DSP-based fuzzy controllers for autonomous mobile robots, IEEE Transactions on Industrial Electronics 55 (2) (2008) 928-936.

[25] I. Baturone, F.J. Moreno-Velo, S. Sánchez-Solano and A. Ollero, Automatic design of fuzzy controllers for car-like autonomous robots, IEEE Transactions on Fuzzy Systems 12 (4) (2004) 447-465.

[26] K. Samsudin, F.A. Ahmad and 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.

[27] Xfuzzy: Fuzzy logic design tools. Available at: http://www.imse-cnm.csic.es/Xfuzzy

[28] J.C. Latombe, Robot motion planning, Kluwer Academic Publishers, 1991.

[29] R.A. Brooks, A robust layered control system for a mobile robot, IEEE Journal on Robotics and Automation 2 (1) (1986) 14-23.

[30] M.J. Wooldridge, An introduction to multiagent systems, John Wiley & Sons Ltd., 2009.

[31] L.E. Dubins, On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents, American Journal of Mathematics 79 (3) (1957) 497-516.

[32] J.A. Reeds and R.A. Shepp, Optimal path for a car that goes both forward and backward, Pacific Journal of Mathematics 145 (2) (1990) 367-393.

[33] G. Desaulniers, On shortest paths for a car-like robot maneuvering around obstacles, Robotics and Autonomous Systems 17 (3) (1996) 139-148.

[34] N. Ghita and M. Kloetzer, Trajectory planning for a car-like robot by environment abstraction, Robotics and Autonomous Systems 60 (4) (2012) 609-619.

[35] R. Battiti, First- and second-order methods for learning: between steepest descent and Newton's method, Neural Computation 4 (2) (1992) 141-166.

[36] F.J. Moreno-Velo, I. Baturone, A. Barriga and S. Sánchez-Solano, Automatic tuning of complex fuzzy systems with Xfuzzy, Fuzzy Sets and Systems 158 (18) (2007) 2026-2038.

[37] I. Baturone, F.J. Moreno-Velo and A. Gersnoviez, A CAD approach to simplify fuzzy system description, in: Proceedings of the 2006 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'06), 16-21 July 2006, pp. 2392-2399.

[38] Virtex-II Pro and Virtex-II Pro X FPGA user guide, Xilinx, 2007.