Design of Embedded DSP-based Fuzzy Controllers for Autonomous Mobile Robots

Iluminada Baturone, Francisco J. Moreno-Velo, Víctor Blanco, and Joaquín Ferruz

***Abstract*— Fuzzy controllers are used in many applications because of their rapid design by translating heuristic knowledge, robustness against perturbations, and smoothness in the control action. However, they require parallel processing and special operators (such as fuzzification or defuzzification) which are not available at standard digital signal processors (DSPs), thus complicating its direct implementation. This paper describes an efficient design methodology which allows starting with any kind of fuzzy controller and subsequently transforming it until obtaining a system suitable for an easy DSP implementation. Such methodology is greatly aided by the design environment Xfuzzy 3. The parking problem of an autonomous robot is described to illustrate the steps of this methodology. Real experiments with the autonomous robot ROMEO 4R demonstrate the efficiency of the designed fuzzy controller embedded into a stand-alone card based on a fixed-point DSP from Texas Instruments.**

***Index Terms*— Embedded systems, Fuzzy controllers, DSPs, CAD tools, autonomous robots.**

I. Introduction

Autonomous mobile robots are capable of performing tasks without the intervention of human operators. Hence, they should contain built-in machine intelligence and an on-board control system. Fuzzy logic-based techniques have been applied successfully to build the control system of autonomous intelligent mobile robots: from low-level controllers for sensors and actuators and intermediate modules that carry out simple individual behaviors to high-level modules in charge of integrating and coordinating primitive behaviors [1]-[6]. Advantages of fuzzy logic-based techniques are that they allow building robust and smooth controllers starting from heuristic knowledge and qualitative models, considering imprecise, vague, and unreliable information, and integrating symbolic reasoning and numeric processing in the same framework.

The trend in many autonomous robots (such as planetary exploration vehicles, micro robots, etc.) is that the on-board control system is restricted to a small size, light weight, and low power consumption, which means the need for embedded controllers capable of real-time operation. An embedded system may follow different approaches: (a) embedded software on off-the-shelf components (microprocessors or specific processors such as microcontrollers and DSPs), (b) programmable logic such as FPGAs (Field Programmable Gate Arrays), and (c) full-custom or semi-custom ASICs (Application Specific Integrated Circuits).

Since fuzzy controllers can use complex IF-THEN rules, they are not suitable to be implemented directly on off-the-shelf processors. For example, rules' antecedent parts can contain several antecedents connected by arbitrarily complex connectives and input variables can be related to fuzzy sets by any kind of linguistic hedges. In addition, the use of particular membership functions such as Gaussian functions, which require exponential calculus, and/or defuzzification methods that require sweeping the output universe of discourse may increase the computational cost of the system.

In order to alleviate the limitations of standard embedded processors, two main approaches have been applied. One of them is to expand the processor with a set of dedicated fuzzy logic instructions. This is the solution adopted, for instance, by Motorola (with the 16-bit 68HC12 microcontroller family [6]) and, more recently, by ST Microelectronics (with the ST FIVE family [5]). The other approach is to employ a fuzzy processor that cooperates with the main processor or works alone. Companies such as Togai InfraLogic, Toshiba, Siemens, Omron, and ST Microelectronics launched this kind of coprocessors in the 90s [8]-[9]. Recent research in this area can be seen in [4] and [10].

Another different line of action is to design the fuzzy controller so as to be adequate for the processor capabilities. In this way, several design environments were developed to translate the fuzzy system definition into specific code of microcontrollers from Motorola, Intel, etc. Some examples are the well-known FIDE system (from Aptronix Inc.), MicroFPL (from Togai InfraLogic), and FuzzyTECH (from Inform) [11], and the most recent rFLASH (from Rigel Corporation) [12]. Recent research concerning this approach can be seen in [13]. However, these solutions impose constraints on the designed systems to ease the translation and, hence, the advantages of

Manuscript received May 6, 2005. This work was supported in part by the Spanish CICYT Projects DPI2005-02293 and TEC2005-04359, and by the Projects TIC2006-635 and TEP2006-375 from the Andalusian regional Government.

I. Baturone is with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC) and the Dept. de Electrónica y Electromagnetismo, Univ. de Sevilla, Seville, SPAIN (phone: +34-955-056-666; fax: +34-955-056-686; e-mail: lumi@imse.cnm.es).

F. J. Moreno Velo is with the DIESIA, Esc. Politécnica Superior, Univ. de Huelva, Huelva, SPAIN (e-mail: francisco.moreno@diesia.uhu.es).

V. Blanco and J. Ferruz are with the Dept. de Ing. de Sistemas y Automática, Univ. de Sevilla, Seville, SPAIN (e-mail: vmblanco@cartuja.us.es, ferruz@cartuja.us.es).

fuzzy controllers to be designed rapidly from linguistic and qualitative knowledge can be lost.

The design environment Xfuzzy 3 was developed by some of the authors at the Instituto de Microelectrónica de Sevilla to reduce these limitations. As far as we know, Xfuzzy 3 provides the greatest flexibility in the design and implementation of fuzzy systems. For example, it allows describing hierarchical systems with complex rules (e.g. including rule weights, linguistic hedges, etc.); it can apply more identification algorithms to extract rule bases from numerical data than other specific fuzzy software (such as those in [14]-[15]); it admits a wide set of learning algorithms to tune the designed system; and it is provided with many simplification facilities that are not available at current fuzzy software. As a result, Xfuzzy 3 allows to start the design process with no constraints, that is, with any kind of fuzzy controller, and subsequently facilitates its transformation until obtaining a system suitable for off-the-shelf processor implementation.

The DSP-based solution is addressed in this paper since current DSPs combine the advantageous on-chip peripherals of a typical microcontroller with the math-intensive computation power of a typical DSP, thus being very suitable for robotic applications. In addition, they are provided with development environments which ease building and debugging the real-time software applications of an embedded controller.

This paper describes how Xfuzzy 3 facilitates the rapid and efficient implementation of complex fuzzy controllers into standard DSPs. The paper is structured as follows. Section II summarizes the design flow that can be followed with the aid of Xfuzzy 3. Interesting guidelines to meet the constraints imposed by standard DSPs are summarized in Section III. The design of a fuzzy controller to park an autonomous robot is addressed in Section IV to illustrate the steps of this methodology. Section V describes the inclusion of this fuzzy controller into an embedded DSP platform that controls the autonomous robot ROMEO 4R, a car-like vehicle designed and built at the Escuela Superior de Ingenieros de Sevilla. Experimental results illustrate the efficiency of the controller. Finally, conclusions are included in Section VI.

II. CAD Methodology

The classic inference mechanism of a fuzzy controller is based on three stages: (a) fuzzification, to compute the membership degrees of the input values to the antecedent fuzzy sets; (b) inference mechanism, to obtain a global conclusion of the rule base; and (c) defuzzification, to compute the non fuzzy values of the output variables. Triangular, trapezoidal and Gaussian functions have been used typically as membership functions to implement the first stage. Three steps are performed in the inference mechanism: firstly, each rule activation degree is usually calculated by applying conjunctive operators between the antecedents (such as the minimum or the bounded product); secondly, an implication function (typically the minimum or the product) is used to obtain each rule conclusion; and, finally, the global conclusion is calculated by an aggregation operator (such as the maximum or the sum). The center of area and center of sums are the typical methods employed in the defuzzification stage [1].

In order to describe complex fuzzy systems, this structure can be improved by the use of complex antecedent parts in the rules, that is, by connecting the several antecedents by any type of conjunctive and disjunctive connectives, by relating input variables with any type of fuzzy sets by any kind of linguistic hedges, and by even applying linguistic hedges to some connected antecedents. As a result, the expressiveness and linguistic interpretability of the resulting fuzzy system increase [16]. Current fuzzy software tools do not support this flexibility and this is why Xfuzzy 3 was developed. The formal specification language of Xfuzzy 3, named XFL3, facilitates the translation of complex rules expressed linguistically into a fuzzy system. In addition, Xfuzzy 3 admits a large set of mathematical functions and algorithms, even new ones introduced by the designer, to represent the fuzzy concepts that appear in the fuzzy inference mechanism. Another advantage of Xfuzzy, which is neither supported by current fuzzy software, is the capability of describing hierarchical systems consisting of several modules interconnected, where each module can apply its own and appropriate fuzzy operators and the data they interchange can be either fuzzy or non fuzzy.

The design methodology with Xfuzzy 3 follows the flow chart in Fig. 1. The aim of the first stage (*description*) is to describe the whole fuzzy system. Xfuzzy 3 contains a CAD tool, named *xfedit*, to aid in this stage. The constituent modules of the whole system usually follow the architecture designed by the human expert who wants to solve a particular problem. The structure of each module can also be determined from the expert linguistic knowledge or can be identified automatically from numerical data. In the last case, we distinguish an *identification* stage. Xfuzzy 3 contains a CAD tool, named *xfdm*, to automate this identification stage. It can currently apply five algorithms based on clustering and four algorithms based on grid techniques.

Once the whole system has been described, its behavior should be tested at the verification stage. Three tools have been programmed to facilitate this verification process in sev-

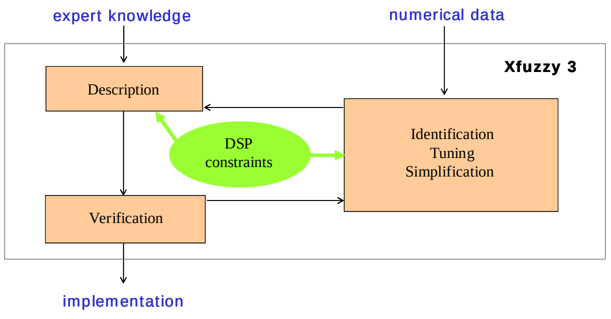


Fig. 1. Design flow aided by Xfuzzy 3.

eral ways. One of them (*xfplot*) allows to show two- and three-dimensional graphics with the input/output behavior of the fuzzy system. Another tool (*xfmt*) allows monitoring the values of the internal and global variables of the system and the activation degrees of the rules of the different modules. The last tool, named *xfsim*, simulates the behavior of the fuzzy system working in line with a model of an external system (a plant in the case of a controller).

One way of correcting the deviations detected at the verification stage is to apply tuning techniques. The tool *xfsl* of Xfuzzy 3 includes a wide set of tuning algorithms: six gradient-descent algorithms, five conjugate gradient, four second-order, two algorithms without derivatives, and two statistical algorithms. It allows tuning hierarchical fuzzy systems with operators defined freely by the user. No other fuzzy software provides such flexibility.

After tuning and identification stages, the resulting module descriptions can be simplified with the tool named *xfsp*. It can be applied to either the variable membership functions or the rule bases. For the first ones, a purge mechanism, five clustering methods, and a merging method based on a similarity measure can be selected. The rules can be pruned according to either a threshold in their activation degree or a number which fixes how many should be maintained or eliminated. Besides, all the rules sharing the same consequent can be merged by connecting their antecedents disjunctively, and, vice versa, the rules can be expanded. Another interesting method available at *xfsp*, which is neither supported by current fuzzy software, is a tabular simplification of the rules based on an extension of the Quine-McCluskey algorithm of Boolean design.

The limitations imposed by the target implementation, standard DSPs in our case, can be considered easily by the designer in the iterations of this design flow. The idea is to subsequently transform the system (which initially is not usually suitable for a direct implementation into a DSP) into a more adequate one. Interesting transformations to consider are summarized in the following section. Once the whole system description has been verified, tuned, and simplified, according to the DSP resources, it can be easily translated to C code and compiled by using the DSP development software.

Further information about Xfuzzy 3 can be found at its official web page (http://www.imse.cnm.es/Xfuzzy/), where its first version is distributed under GNU General Public License.

III. DSP Constraints

Since the central processing unit of standard DSPs is adequate to perform addition, subtraction, multiplication, standard "if-then" conditional statements and standard relational and logical operators, the objective of the design flow is to translate the fuzzy rule bases obtained from expert knowledge and/or numerical data into non fuzzy ones which include these standard operators.

In the fuzzy modules that form a fuzzy system, each input universe of discourse, I_i, is partitioned into L_i linguistic labels, so that $L_1 x \ldots x L_u$ rules can be defined for *u* inputs (L^u in the case of equal partitions). If λ is the maximum overlapping degree between the membership functions of the linguistic labels, $L_i+1-\lambda$ intervals can be distinguished per input and each input x_o can activate λ^u rules at a maximum. For example, five labels with an overlapping degree of two (four intervals) are shown in Fig. 2.

If the fuzzy module corresponds to a decision making-type rule base, the output is usually calculated as the consequent of the rule with maximum activation degree, which is adequate for DSP implementation if these consequents are represented by constant or symbolic values. Our objective when transforming these rule bases will focus on obtaining: (a) a complete rule base, that is, containing all the possible rules so as to ensure that the decision regions can be defined as intersections of input intervals and to exploit complementarity, and (b) rules' antecedents whose expressions admit a translation to standard relational and Boolean operators (such as "and", "or", and "not"). With this transformation, each decision making-type rule base of the initial fuzzy system can also be described by a set of standard if-then statements in which the *if* parts may contain relational and Boolean operators while the *then* parts contain constant or symbolic values associated to the decisions.

If the fuzzy module corresponds to an interpolation-type rule base, such as Takagi-Sugeno's, the output is calculated as:

$$y(x_o) = \frac{\sum_{k=1}^{\lambda^u} h_k(x_o) \cdot c_k(x_o)}{\sum_{k=1}^{\lambda^u} h_k(x_o)} \quad (1)$$

where h_k is the activation degree of one of the λ^u active rules and c_k is its corresponding consequent function.

When considering the implementation of fuzzy systems with DSPs, multi linear expressions are preferred for the output $y(x_o)$ so as to require only addition and multiplication operations. Hence, our objective when transforming an interpolation-type rule base is to end with the following features: (a) linear and normalized membership functions to represent the linguistic labels of the rules' antecedents (such as those in Fig. 2); (b) normalized product to represent the linguistic "and" as the unique antecedents' connective, and (c) constant values for the rules' consequents [17]. This means that, given two inputs, x_1 and x_2, the output expression of the final rule base we are looking for, will be as follows:

$$y = c_{rr}\mu_{r1}\mu_{r2} + c_{rl}\mu_{r1}\mu_{l2} + c_{ll}\mu_{l1}\mu_{l2} + c_{lr}\mu_{l1}\mu_{r2} \quad (2)$$

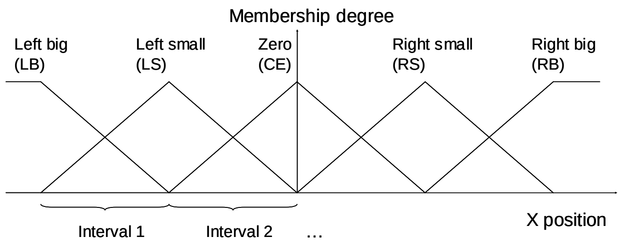


Fig. 2. Normalized triangular membership functions.

where μ_{ri} and $\mu_{li} = 1-\mu_{ri}$ are the non-zero membership degrees of the input x_i to the two overlapping labels.

Since the membership functions are linear, (2) can also be expressed as:

$$y = ax_1x_2 + bx_2 + cx_1 + d = x_2(ax_1 + b) + cx_1 + d \qquad (3)$$

where a, b, c, d are constants related to the parameters of the antecedents' membership functions and the consequents' singleton values.

With this transformation, each interpolation-type rule base of the initial fuzzy system can also be described by a set of standard if-then statements in which the *if* parts evaluate the pertinence of the input to each of the (L-1) input intervals and the *then* parts assign the output as expressed in (3).

Additional transformations which are interesting for any rule base are the use of hierarchical structures and exploitation of symmetry. All these transformations are clarified in the following with examples of different fuzzy modules.

IV. Application Example

A. *The problem of diagonal parking*

Let us consider the diagonal parking problem of a car-like autonomous robot in a constrained space. Starting from any given position (x, y), and orientation (ϕ), the robot can drive forward and backward but has to arrive backward at the desired parking place at a right angle with the horizontal and to stop there ($x = y = \phi = v = 0$). Once known its current configuration (x, y, ϕ, v), the robot has to control the values of its new speed (magnitude and driving direction) and its new curvature (v, γ) to achieve a good parking maneuver.

The goal when driving forward is to lead the robot towards a configuration with $x = 0$ and $\phi = 0$ (the vertical through the center of the parking place), so as to finish the parking maneuver by driving backward with an almost zero curvature. The approaching to the vertical is wanted to be done by approximating a short trajectory made up of arcs of circles of minimum turning radius and straight lines parallel to the line of cars (to avoid collisions and reduce the y distance traveled). These trajectories are shown in Fig. 3a.

The goal when driving backward is that the robot reaches the parking place with the target final configuration. Similarly to the forward maneuvers, the desired trajectories are made up of arcs of circles of minimum radius and straight line segments parallel to the line of cars, as shown in Fig. 3b.

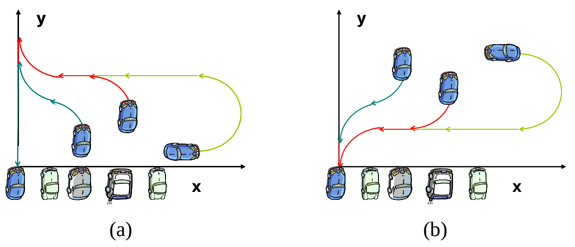
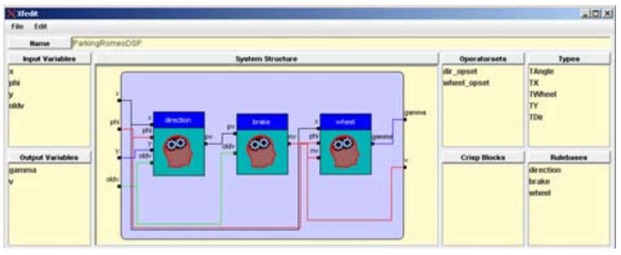


Fig. 3. Example of ideal trajectories to be performed by the robot when driving (a) forward and (b) backward.



Name: ParkingRomeoDSP

Input Variables: x, phi, y, oldv

Output Variables: gamma, v

System Structure

Operatorsets: dir_opset, wheel_opset

Crisp Blocks

Types: TAngle, TX, TWheel, TY, TDir

Rulebases: direction, brake, wheel

Fig. 4. Structure of the fuzzy controller described with Xfuzzy 3.

Since a human driver has heuristic knowledge to solve this problem, our approximation has been to design a fuzzy controller from this knowledge. In this sense, the first control action is to decide the direction of driving (the sign of the speed): backward or forward, and the magnitude of the speed. This knowledge is included into a rule base that we call "direction". Constraints imposed when deciding the new speed are that the change of driving direction has to be soft, that is, the controller should never decide to go forward at a rather high speed if previously, the vehicle was driving backward at a rather high speed. This kind of constraints is considered by a rule base called "brake". The input variables of this rule base are the speed decided by the rule base "direction" and the previous speed. Its output is the new speed to be adopted.

The second decision is to select the proper angle of the wheels once we have decided to drive backward or forward. The speed selected by the rule base "brake" together with the x position and the orientation of the vehicle are the input variables of another rule base that we call "wheel".

The description tool *xfedit* of Xfuzzy 3 has been employed to describe the resulting hierarchical fuzzy controller, as shown in Fig. 4.

B. *Decision making-type rule bases*

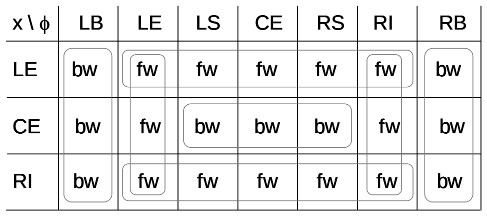
The rules of the modules "direction" and "brake" are easily defined from our heuristic knowledge. In particular, the first description of the rule base "direction" contained 17 rules, such as the following:

"If the y coordinate is *near* the cars of the parking place and the x coordinate is not *approximately zero* and the robot orientation ϕ is greater than *approximately* –90º and smaller than *approximately* 90º, the driving direction should be *forward* to avoid collision with the parked cars".

Table I shows how the above rule and other five ones related to the situation 'y near' are easily expressed with the XFL3 language of Xfuzzy 3. These rules are fuzzy because our knowledge is uncertain about concepts such as *near*, *approximately zero*, etc., and, hence, they are represented by fuzzy sets. As a matter of fact, the universes of discourse of the input variables x, ϕ, y, and *oldv* are covered initially by 3, 7, 5, and 5 membership functions, respectively. In the other side, the output variable can take three singleton or constant

TABLE I
SIX RULES OF THE MODULE "DIRECTION" EXPRESSED WITH XFL3

```
1) if(y == near & x != CE & phi >= LE & phi <= RI)
   -> pv = forward;
2) if(y == near & x != CE & (phi < LE | phi > RI))
   -> pv = backward;
3) if(y == near & x == CE & (phi < LE | phi > RI))
   -> pv = backward;
4) if(y == near & x == CE & (phi == LE | phi == RI))
   -> pv = forward;
5) if(y == near & x == CE & oldv < stop & phi >= LS & phi <= RS)
   -> pv = backward;
6) if(y == near & x == CE & oldv >= stop & phi >= LS & phi <= RS)
   -> pv = forward;
```



$x \setminus \phi$	LB	LE	LS	CE	RS	RI	RB
LE	bw	fw	fw	fw	fw	fw	bw
CE	bw	fw	bw	bw	bw	fw	bw
RI	bw	fw	fw	fw	fw	fw	bw

Fig. 5. Matricial representation of the rules derived from those in Table I.

values: *stop* (0 m/s), *forward* (1 m/s) or *backward* (-1 m/s). The global output of the rule bases is obtained by applying a maximum-type defuzzification process.

Some of the initial and heuristic rules may not be adequate for the control objective. For instance, simulation results obtained with the tool *xfsim* revealed that when the robot was driving forward, it stopped too far before changing the driving direction to backward. The monitoring tool *xfmt* showed that the rule 5 and, mainly, rule 6 in Table I were responsible of that behavior. Hence, rule 6 was eliminated as well as the antecedent 'oldv<stop' in rule 5. The resulting five rules cover all the possible situations in *x* and ϕ as shown in Fig. 5 (y==near). If the tabular simplification method of the tool *xfsp* is now applied, the groupings depicted with gray lines in Fig. 5 are formed, thus resulting the two rules shown in Table II. These rules are complementary (their activation degrees always sum the unity) if the "and" and "or" connective are represented by the normalized product and sum, respectively.

If this simplification process is applied to the whole module "direction": (a) the initial 17 rules obtained from our heuristic knowledge are reduced to 14 by using the tools *xfsim* and *xfmt*, (b) the 14 rules are reduced to 8 by using the tool *xfsp*, and (c) the 8 rules are reduced to 6 by exploiting the complementarity between some of them. These 6 rules, which are shown in Table III, meet the constraints commented in Section III: they form a complete rule base and their

TABLE II
RESULT AFTER SIMPLIFYING THE RULES IN TABLE I

```
1) if(y==near & ((x==CE & phi>= LS & phi <= RS) | phi<LE | phi>RI))
   -> pv=backward;
2) if(y==near & (x!=CE & (phi>=LE & phi<=RI) | phi==LE | phi==RI)
   -> pv= forward;
```

TABLE III
RULES OF THE MODULE "DIRECTION" (FINAL ITERATION)

```
1) if(y > far) -> pv = backward;
2) if(y >= far & oldv <= stop) -> pv = backward;
3) if(y == near & (phi < LE | phi > RI)) -> pv = backward;
4) if(y >= near & x == CE & phi >= LS & phi <= RS) -> pv = backward;
5) if(y < near) -> pv = stop;
6) else -> pv = forward;
```

TABLE IV
FINAL RULES OF THE MODULE "DIRECTION" TRANSLATED INTO C CODE

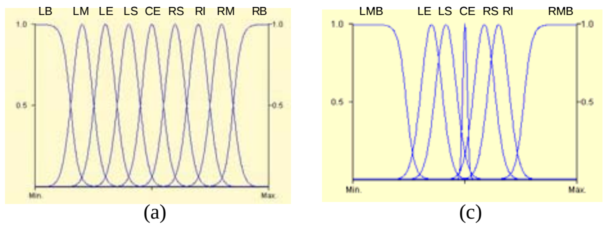
```
if( y > 12.5 ) pv = -1;
   else if( (y > 3.5) && (oldv <= 0) ) pv = -1;
   else if( (y > 0.0) && (y <= 3.5) && ((phi < -90) | (phi > 90)) ) pv = -1;
   else if( (y > 0.0) && (x > -1.25) && (x < 1.25) && (phi > -12) &&
   (phi < 12) ) pv = -1;
   else if( y <= 0.0 ) pv = 0;
else pv = 1;
```

antecedent parts admit a translation to standard relational and logical operators. Hence, they can also be described in a non fuzzy way, as shown in Table IV. This non fuzzy description, expressed in C code, is easily compiled into a DSP.

Applying the same simplification process to the module "brake", it could be translated into 4 standard if-then statements.

C. *Interpolation-type rule bases*

Designing the rule base "wheel" to provide the short trajectories depicted in Fig. 3 is a difficult task by applying heuristic knowledge only. Hence, the first iteration in the design process was to identify two single modules (one for driving forward and another for backward) from numerical data corresponding to the geometric analysis of desired trajectories. The Wang&Mendel algorithm [1] available at the tool *xfdm* was employed to identify these two zero-order Takagi-Sugeno modules with two inputs (*x*, and ϕ) and one output (γ). In order to allow enough flexibility and smoothness, they were selected with 7 linguistic labels to cover the input variable *x* and 9 labels for the vehicle angle (Fig. 6a), both of them represented by Gaussian membership functions. The rule base obtained for controlling the forward driving is shown in Fig. 6b. The labels *f04*, *f0* and *f04n* represent the three singleton values (0.4, 0 and -0.4 m^{-1}) that the output variable can take. The learning and simplification tools, *xfsl* and *xfsp*, were applied in the subsequent iteration to the identified modules. One result of this simplification is that the membership functions of *x* and ϕ were reduced to 5 and 7, respectively (because, for example, the output takes the same value for 'ϕ == RM' and 'ϕ == RB', as can be seen in Fig. 6b, so that the labels RM and RB can be merged into a unique label, RMB). Fig. 6c shows these 7 tuned functions of the variable ϕ. The resulting control surface when driving forward obtained with the tool *xfplot* is shown in Fig. 7. From this result, we extracted that the curvature value could be defined



(a) (c)

x \ ϕ	LB	LM	LE	LS	CE	RS	RI	RM	RB
LB	f04	f04	f04	f04	f04	f04	f04	f04n	f04n
LE	f04	f04	f04	f04	f04	f04	f04	f04n	f04n
LS	f04	f04	f04	f04	f04	f04n	f04n	f04n	f04n
CE	f04	f04	f04	f04	f0	f04n	f04n	f04n	f04n
RS	f04	f04	f04	f04n	f04n	f04n	f04n	f04n	f04n
RI	f04	f04	f04n	f04n	f04n	f04n	f04n	f04n	f04n
RB	f04	f04	f04n	f04n	f04n	f04n	f04n	f04n	f04n

(b)

Fig. 6. (a) Initial membership functions of the ϕ variable. (b) Identified rules for driving forward. (c) Tuned and simplified functions of the ϕ variable.

better depending on the relation between ϕ and *x* rather than on the values of ϕ and *x* separately. Hence, the next iteration to further simplify the module "wheel" was to select a hierarchical structure made up of two rule bases (Fig. 8). The first rule base ("interpolation") provides a value α depending on *x*. The second rule base ("smoothing") generates the value of γ depending on the value of the difference (ϕ– α), when driving forward, and (–ϕ– α), when driving backward, because it was detected the symmetry of the problem regarding ϕ (this difference is named *sfu*). Four rules are included in the rule base "interpolation" and other four ones in the rule base "smoothing", and normalized triangular membership functions are selected now. The eight rules are adjusted with the same training data as before. The capability of the tool *xfsl* to train hierarchical systems is exploited at this point. The resulting control surface when driving forward is shown in Fig. 9. It can be appreciated its similarity with the surface in Fig. 7.

As a result, the module "wheel", which was initially designed as two rule bases, each with 63 rules, has been finally translated into two rule bases, each with 4 rules. In addition, since these final rule bases use normalized triangular membership functions in the antecedents, their outputs are a piecewise linear function of their inputs, and, hence, they can be translated easily into non fuzzy rule bases, as commented in Section III. For example, the rule base "smoothing", can be expressed in C code as shown in Table V. The whole module "wheel" can be expressed in C code as 11 standard if-then conditional statements that only include simple additions and multiplications.

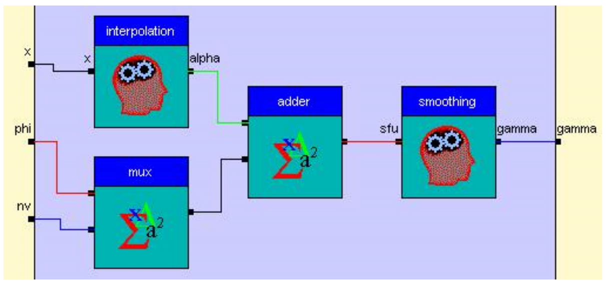


Fig. 8. Structure of the module "wheel" described by Xfuzzy 3.

TABLE V
FINAL RULES OF THE MODULE "SMOOTHING" TRANSLATED INTO C CODE

```
if( sfu<=-30.0 ) gamma = 0.4;
if( (sfu>-30.0) && (sfu<=-4.0) ) gamma= 0.039 - 0.361*(sfu+4.0)/26.0;
if( (sfu>-4.0) && (sfu<4.0) ) gamma = -0.039*sfu/4.0;
if( (sfu>=4.0) && (sfu<30.0) ) gamma= -0.039 - 0.361*(sfu-4.0)/26.0;
if( sfu>=30.0 ) gamma = -0.4;
```

V. EXPERIMENTAL RESULTS

The above described fuzzy controller has been employed to park the robot ROMEO 4R, an electrical car-like vehicle designed and built at the Escuela Superior de Ingenieros de Sevilla (Fig. 10) [18]. An embedded controller, based on the fixed-point TMS320LF2407 DSP from Texas Instruments, has been used to implement the low-level control loops as well as the high-level fuzzy controller [19]. Fig. 11 shows the components of this embedded controller. It consists of: (a) an EVM2407 board from Spectrum Digital [20], which includes the DSP chip, 128 K of external SRAM, four 12-bit D/A channels and interface circuitry; (b) expansion boards that

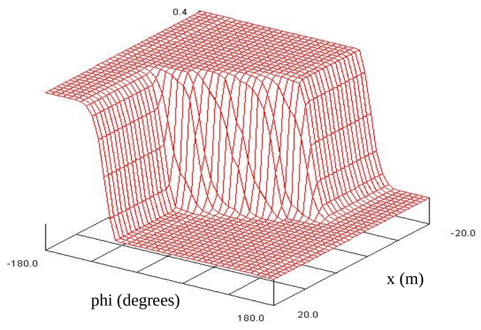


Fig. 7. Control surface of the module "wheel" (at the first iterations).

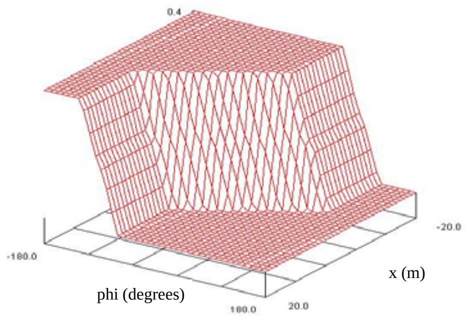


Fig. 9. Control surface of the module "wheel" (final iteration).

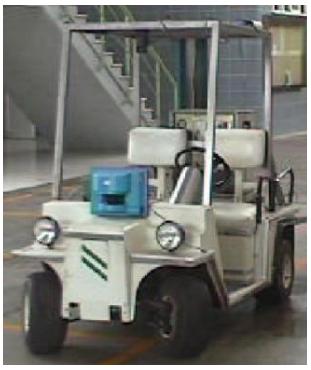


Fig. 10. The autonomous robot ROMEO 4R.

provide signal conditioning, overvoltage protection and four additional serial ports; and (c) a compact GPS module (not used for this specific application).

The D/A channels of the EVM board are used to communicate the outputs (v, γ) of the fuzzy controller to the traction and steering motor control boards. One of the RS-232 ports of the expansion board is used to communicate with a gyroscope. The information from the gyro is combined by the DSP chip with the measurements taken by the traction and steering encoders to compute the robot position, orientation, speed, and curvature, that is, the variables (x, y, ϕ, v, γ) involved in this parking problem. The on-chip DSP hardware offers direct interface for quadrature encoders as well as an RS-232 port, which is used as communication link to an external PC. At the beginning of the maneuver, this PC sends to the controller the situation of the target parking place. During the maneuver, it collects from the controller the data of the path followed, as well as the potential error messages.

Texas Instruments provides several development tools for its DSPs; in particular, ANSI C compiler, assembler/ linker, and the Code Composer Studio^{TM} environment. The latter has been used to develop and load into the DSP a run-time environment [21] which includes the low-level peripheral drivers necessary to access the sensors and actuators of the robot; it also contains a simple scheduler that allows to define and run a number of time-triggered, pre-emptable software modules, following a fixed priority scheme. Among such modules are the low-level steering and driving PID controllers, the position estimation algorithms and the fuzzy controller itself. This means that the fuzzy controller was also programmed and loaded into the DSP by using the Code Composer Studio. Fixed-point data formats were employed in the C code so as to maximize the accuracy (avoiding data conversions) and minimize the size and execution time of the code (avoiding software emulation of floating point). Care was taken to avoid numerical errors due to truncation, rounding, and overflow. Under this run-time environment, the high-level fuzzy control algorithm is relieved of the task of synchronizing the read and write processes of the different robot devices.

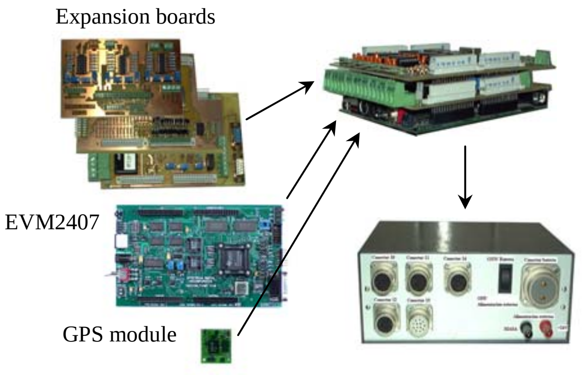


Fig. 11. Components of the embedded controller.

The DSP core processor uses a Harvard-type architecture, runs at 33 MHz and is capable of single-cycle 16-bit multiplication. The on chip memory includes 32 K words of 16 bits of Flash EEPROM and 2.5 K words of 16 bits of Data/Program RAM. The fuzzy controller described herein occupies 851 words of 16 bits of Data/Program RAM. Once the robot configuration is available, the time spent by the DSP in implementing the fuzzy controller ranges from 19.3 to 21.4 μs (the total time spent in computing the robot configuration, implementing the fuzzy controller, and assigning the new values of speed and curvature ranges from 35.0 to 37.1 μs). It is interesting to notice that the implementation of the fuzzy controller designed at the first iteration (without considering any DSP constraints) in a Pentium processor at 100 MHz takes about 2 ms, that is, 2 order of magnitude slower.

Many successful experiments were performed by ROMEO 4R being controlled by the embedded DSP-based controller. The control loop has a period of 50 ms in the experiments. In each period, the DSP computes: (a) the new robot configuration by using the information provided by the encoders and the gyroscope and applying odometry, and (b) the new curvature and speed by applying the fuzzy controller. Three examples of experimental trajectories of different complexity are shown in Fig. 12 and 13. All the experimental results obtained with the embedded controller were very similar to the simulation results obtained by Xfuzzy 3 with a model of the robot, as illustrated in Fig. 13.

VI. Conclusions

The CAD tools of the environment Xfuzzy 3 offer more description, identification, verification, tuning, and simplification facilities than any other fuzzy software, thus allowing a rapid design of complex fuzzy controllers starting from linguistically expressed knowledge and/or numerical data as well as easing its translation to be efficiently implemented in low-cost DSP-based embedded platforms. A set of guidelines have been provided to direct the efficient translation of the different rule bases which constitute a whole fuzzy controller. The memory resources and the processing time required by the obtained controllers are so low that standard DSPs can be used not only to implement the low-level control tasks, as usual, but also the high-level ones required by real-time embedded applications. This has been proven by fulfilling the whole design process (from the

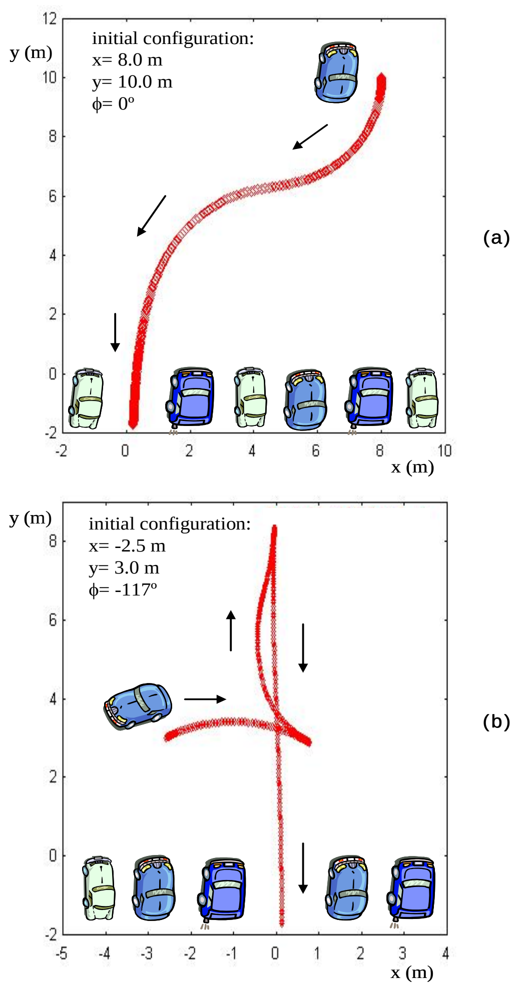


Fig. 12. Experimental results.

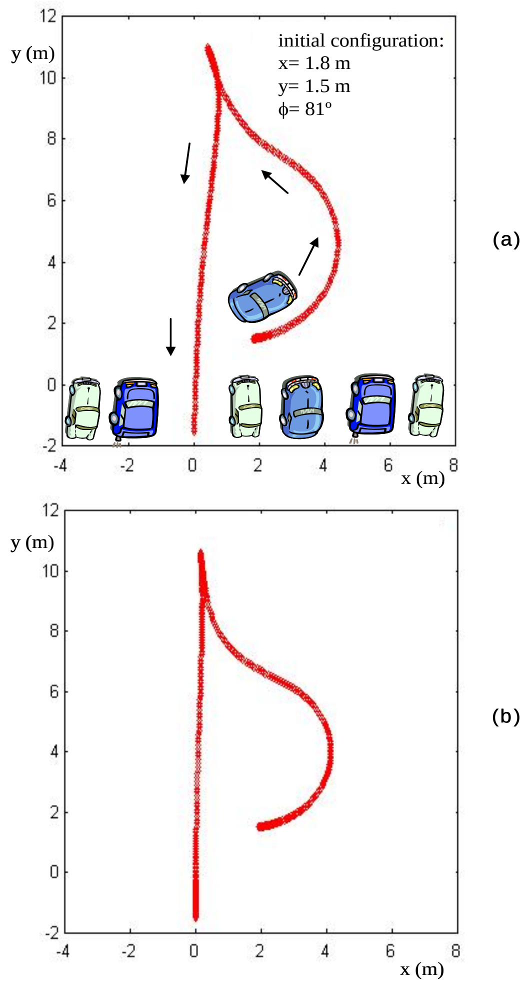


Fig. 13. (a) Experimental and (b) simulated results.

conception of the controller to the consecution of successful experiments) to solve a typical navigation problem in robotics.

REFERENCES

[1] L-X. Wang, *A course in fuzzy systems and control*, Prentice Hall, Englewood Cliffs, New Jersey, 1996.

[2] K.-B. Sim, K.-S. Byun, F. Harashima, "Internet-based teleoperation of an intelligent robot with optimal two-layer fuzzy controller," *IEEE Trans. on Industrial Electronics*, vol. 53, no. 4, pp. 1362- 1372, June 2006.

[3] C.-L. Hwang, L.-J. Chang, Y.-S. Yu, "Network-based fuzzy decentralized sliding-mode control for car-like mobile robots", *IEEE Trans. on Industrial Electronics*, vol. 54, no. 1, pp. 574-585, Feb. 2007.

[4] S. Sánchez-Solano, A. J. Cabrera, I. Baturone, F. J. Moreno-Velo, M. Brox, "FPGA implementation of embedded fuzzy controllers for robotic applications", *IEEE Trans. on Industrial Electronics*, vol. 54, no. 4, pp. 1937-1945, Aug. 2007.

[5] L. Fortuna, M. Lo Presti, C. Vinci, A. Cucuccio, "Recent trends in fuzzy control of electrical drives: an industry point of view", *Proc. Int. Symp. Circuits and Systems*, ISCAS'2003, vol. 3 pp. 459-461, Bangkok, 2003.

[6] F. Sun, L. Li, H.-X. Li, H. Liu, "Neuro-fuzzy dynamic-inversion-based adaptive control for robotic manipulators—discrete time case," *IEEE Trans. on Industrial Electronics*, vol. 54, no. 3, pp. 1342-1351, June 2007.

[7] R. Bannatyne, "Motorola's 68HC12 an evolution from 8-bit to 16-bit", *Embedded-System Engineering*, vol. 4, no. 4, pp. 32-3, June-July 1996.

[8] H. Eichfeld, T. Kunemund, M. Menke, "A 12b general-purpose fuzzy logic controller chip", *IEEE Trans. on Fuzzy Systems*, vol. 4, no. 4, pp. 460-475, Nov. 1996.

[9] A. Pagni, "Digital approaches", in *Handbook of Fuzzy Computation*, Institute of Physics Publishing, 1998.

[10] C.-F. Juang, J.-S. Chen, "Water bath temperature control by a recurrent fuzzy controller and its FPGA implementation," *IEEE Trans. on Industrial Electronics*, vol. 53, no. 3, pp. 941- 949, June 2006.

[11] C. Von Altrok, "Adapting existing hardware for fuzzy computation", in *Handbook of Fuzzy Computation*, Institute of Physics Publishing, 1998.

[12] *rFLASH User's Guide*, Rigel Corporation, 2002.

[13] F. Betin, A. Sivert, A. Yazidi, G.-A. Capolino, "Determination of scaling factors for fuzzy logic control using the sliding-mode approach: application to control of a DC machine drive", *IEEE Trans. on Industrial Electronics*, vol. 54, no. 1, pp 296-309, Feb. 2007.

[14] D. D. Nauck, "Measuring interpretability in rule-based classification systems", *Proc. IEEE International Conference on Fuzzy Systems*, FUZZ-IEEE'2003, vol. 1, pp. 196-201, St. Louis, May 2003.

[15] J. M. Alonso, L. Magdalena, S. Guillaume, "KBCT: A knowledge extraction and representation tool for fuzzy logic based systems", *Proc. IEEE International Conference on Fuzzy Systems*, FUZZ-IEEE'2004, vol. 2, pp. 989-994, Budapest, July 2004.

[16] M. Sugeno, T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling", *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, 1993.

[17] I. Baturone, S. Sánchez Solano, "Microelectronic design of universal fuzzy controllers", *Mathware & Soft Computing*, vol. 8, pp. 303- 319, Dec. 2001.

[18] F. Cuesta , F. Gómez-Bravo, A. Ollero, “Parking maneuvers of industrial-like electrical vehicles with and without trailer", *IEEE Trans. on Industrial Electronics*, vol. 51, no. 2, pp 257-269, April 2004.
[19] *TMS320LF2407 Reference Guides*, Texas Instruments, Inc, 2000.
[20] *TMS320LF2407 Evaluation Module. Technical Reference*, Spectrum Digital, Inc., 2000.
[21] J. Ferruz, V. Blanco, A. Ollero, J. V. Acevedo, “An embedded DSP-based controller for the ROMEO-4R vehicle”, *Proc. 5th IFAC Int. Symp. on Intelligent Components and Instruments for Control Applications,* SICICA’2003, pp. 101-106, Aveiro (Portugal), 2003.

Iluminada Baturone received the Licenciado en Físicas degree (with honors) and the Doctor en Físicas degree (with honors) in 1991 and 1996, respectively, both from the University of Seville, Spain.

Since 1990 she has been with the Instituto de Microelectrónica de Sevilla. From 1990 to 1991 she had a fellowship of introduction to research for undergraduated students and during 1992 she was a Postgraduated Research Fellow. She also had a fellowship from the regional government in the Dept. de Electrónica y Electromagnetismo at the University of Seville and later a post-doctoral grant from the Consejo Superior de Investigaciones Científicas. Currently, she is an Associate Professor at the University of Seville.

Dr. Baturone is co-author of one book, co-editor of another book and a journal, and author or co-author of more than 100 publications. She was the recipient of a Best Paper Award at IECON’2002 Conf. of IEEE Industrial Electronics Society. Her current research interests include hardware/software codesign, neuro-fuzzy systems, autonomous robots and image processing applications.

Francisco J. Moreno-Velo received the B.S. degree in Physics from the University of Seville, Spain, in 1995, and the B.S and Ph.D. degrees in Computer Science from the University of Seville, Spain, in 1996 and 2003, respectively.

From 1996 to 1999, he was an Assistant Professor with the Department of Applied Physics and Electrical Engineering, University of Huelva. From 2000 to 2003, he was a Postgraduated Research Fellow at the Instituto de Microelectrónica de Sevilla. Currently, he is an Assistant Professor with the Department of Information Technologies, University of Huelva. His current research interests include fuzzy systems, soft-computing techniques, development of CAD tools for fuzzy systems, and compiler design.

Víctor Blanco was born in Algeciras, Spain, in 1977. There, he achieved his basic and medium studies focused towards the science field. He began his higher studies at the University of Seville, where he obtained the Telecommunication Engineering degree in 2003.

From 2003 to February 2006 he was working as a researcher at Systems and Automation Engineering Department at the University of Seville, especially focused on embedded systems for motion control and artificial perception in the scope of aerial and terrestrial robotics. Currently, he develops his main activity in the aeronautical industry, situation that he combines with his Ph.D. studies at the University of Seville.

Joaquín Ferruz was born in Seville, Spain, in 1961. He received the Industrial Engineering degree and the Ph.D. degree in Engineering from the University of Seville in 1986 and 1997, respectively.

From 1985 to 1992 he worked as a Hardware Design Engineer at the R&D department of Fujitsu España in Malaga, Spain, where he was involved in computer and microelectronics design activities. In 1992 he joined the Systems and Automation Department of the University of Seville as a Lecturer. In 1999 he was appointed Associate Professor. His main research interests are aerial and mobile robotics, computer vision and real-time embedded systems.