AUTOMATIC DESIGN OF FUZZY CONTROL SYSTEMS FOR AUTONOMOUS MOBILE ROBOTS

I. Baturone[1], F. J. Moreno-Velo[1], S. Sánchez-Solano[1], R. Martín de Agar[2], A. Ollero[2]

[1] Instituto de Microelectrónica de Sevilla (IMSE). Avd. Reina Mercedes, s/n.
Edif. CICA, 41012, Sevilla, SPAIN

[2] Dep. Ingeniería de Sistemas y Automática. E.S.Ingenieros. Camino
Descubrimientos s/n. 41092, Sevilla, SPAIN

xfuzzy-team@imse.cnm.es

Proc. 28th Annual Conference of the IEEE Industrial Electronics Society (IECON'2002), pp. 2457-2462, Sevilla, November 5-8, 2002.

Automatic Design of Fuzzy Control Systems for Autonomous Mobile Robots

I. Baturone[1], F. J. Moreno-Velo[1], S. Sánchez-Solano[1], R. Martín de Agar[2], A. Ollero[2]

[1] Instituto de Microelectrónica de Sevilla (IMSE-CNM). Avd. Reina Mercedes, s/n. Edif. CICA, 41012, Sevilla, SPAIN

[2] Dep. Ingeniería de Sistemas y Automática. E.S.Ingenieros. Camino Descubrimientos s/n. 41092, Sevilla, SPAIN

xfuzzy-team@imse.cnm.es

***Abstract* - This paper describes the design and implementation of a fuzzy controller for autonomous mobile robots. The tool Xfuzzy 3.0, developed at the IMSE (Instituto de Microelectrónica de Sevilla) has been used to design a controller for the Romeo 4R autonomous vehicle designed and built at the "Escuela Superior de Ingenieros", University of Seville. The paper presents the design of the controller and real experiments with Romeo 4R demonstrating the efficiency of the controller.**

I. INTRODUCTION

Some of the maneuvers that should be performed by an autonomous mobile robot, such as parking in a given place, are easily performed by any human driver with a bit of practice. The way in which human drivers usually express their control actuation to perform a maneuver (brake, steering wheel, etc.) is not quite precise but rather fuzzy. We neither need exact information from the environment or from our vehicle to carry out a successful maneuver. Most of the times, we apply a heuristic knowledge which can be expressed linguistically by more or less chained if-then rules. Fuzzy logic provides a mathematical framework to translate these linguistic and symbolic concepts into numerical data which can be handled by electronic circuits. Many works have been reported in the literature which show the efficiency of fuzzy controllers implemented in software (general-purpose processors) or hardware (application specific processors) [1-2].

As happens to any design process, it is very interesting to employ CAD tools when designing a fuzzy controller. This is particularly true nowadays when reducing the cost and the time-to-market of a product are driving forces of the industry.

In the last few years several CAD tools tailored to the fuzzy system design have been created [3-5]. The CAD environment employed in this paper is Xfuzzy 3.0, which has been developed at the IMSE (Instituto de Microelectrónica de Sevilla) with the objective of being an open environment with the least possible limitations [6]. With this general objective, Xfuzzy 3.0 is based on an specification language (XFL3) that eases the description and manipulation of complex fuzzy systems thanks to the use of user-defined membership functions, fuzzy operators (including linguistic hedges), and rule bases (admitting hierarchical structures) [7]. This objective has also motivated the use of Java as the programming language of Xfuzzy 3.0. This means the use of an advantageous object-oriented methodology and the flexibility of executing Xfuzzy 3.0 in any platform with JRE (Java Runtime Environment) installed.

This paper describes how Xfuzzy 3.0 can help the user to friendly design a fuzzy system for controlling the parking maneuvers of Romeo 4R, an autonomous mobile robot developed at the *Escuela Superior de Ingenieros* (ESI) of the University of Seville [8]. Section II shows the description process of the fuzzy system. Section III explains how the behavior of the controller can be verified by monitoring the inference process as well as simulating the controller in a closed loop with a model of the robot. Once the system has been designed and validated, Xfuzzy 3.0 allows its synthesis into several programming languages. Section IV shows how the controller is synthesized as a C code and integrated into the software executed by the computer that controls Romeo 4R. Several experimental results of diagonal parking maneuvers are included to illustrate the efficiency and robustness of the designed controller.

II. FIRST STEP: DESCRIPTION PROCESS OF THE FUZZY CONTROLLER

Parking a vehicle at a given place has been a problem usually addressed in the literature to illustrate the capabilities of neural and fuzzy controllers. A typical goal is to back up a vehicle so as to arrive at a desired loading dock at a right angle with the horizontal [9]. The input variables considered in these reported controllers are the x position of the vehicle and the vehicles´s orientation angle with the horizontal. The output control variable is the required steering angle (see Fig-

This work has been partially supported by the Spanish CICYT Projects TAP99-0926-C04-01 and TIC2001-1726.

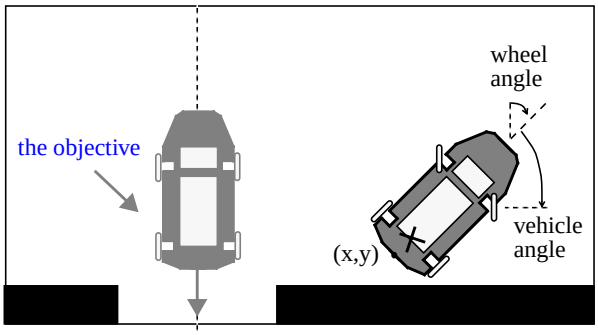


Fig. 1: Example of the diagonal parking problem.

ure 1). The speed magnitude as well as the backward direction of driving are constant. These reported controllers are efficient whenever the vehicle is rather far from the loading dock but fail if it is near the dock and with a bad orientation angle, like that in Figure 1.

The parking problem that we address in this paper is more complex and realistic: our autonomous robot has to park at a desired place, arriving backward and at a right angle, but it could drive backward and forward to achieve success from any starting position and orientation.

The approximation we have taken to design this controller is to directly emulate what we would do as drivers. In this sense, our 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 decision is dynamic because it takes into account not only the current position and orientation of the vehicle but also its previous speed. This knowledge is included into a rule base that we call "direction".

In addition, the constraints imposed by Romeo 4R have to be considered when deciding the new speed. For example, Romeo has not an electronically controlled brake currently, and it is important to ensure that the driving direction changes softly. This means that 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 are considered by a rule base that we call "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 that will be adopted by Romeo.

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".

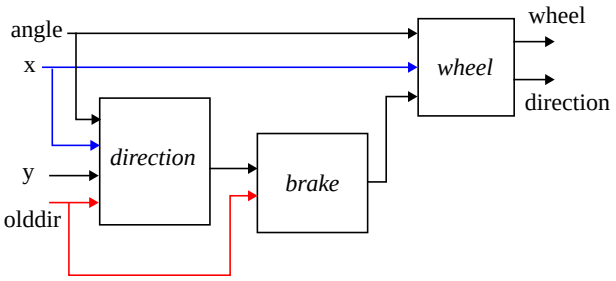


Fig. 2: Structure of the designed controller.

As a result of our knowledge emulation, the fuzzy controller that we have obtained is a hierarchical system with the structure shown in Figure 2. The global input variables are the position (x, y), orientation (angle), and previous speed (olddir) of the robot; and the output variables are the steering wheel angle (wheel) and the new speed (direction).

We have employed Xfuzzy 3.0 to describe this controller. Xfuzzy 3.0 divides the description of a fuzzy system into two parts. One part is the logical definition of the system (its structure, the membership functions that represent the fuzzy sets, and the rules of each rule base). This part can be defined via graphical user interfaces by using the tool *xfedit* or by editing directly a ".xfl" file. The other part is the mathematical definition of the different functions that appear in the logical definition (membership functions, connective operators, defuzzification methods, linguistic hedges, etc.). This part can be defined via graphical user interfaces by using the tool *xfpkg* or by editing directly a ".pkg" file.

This twofold definition allows us to create and use our own membership functions, defuzzification methods, etc. In our case, all the mathematical functions employed are described in the xfl.pkg file provided with Xfuzzy 3.0. For example, the defuzzification method employed in the rule base "wheel" is the Fuzzy Mean method, which calculates the weighted average of the consequent singleton values. This provides a soft interpolation among the 7 singleton values considered to represent the wheel angle. On the other side, the defuzzification method that we employ in the rule bases "direction" and "brake" is a method that we call "MaxLabel". It selects the singleton consequent of the rule whose activation degree is maximum, because the decision made by these rule bases has to be crisp: forward or backward but not an average of both. The description of the MaxLabel method can be seen in Figure 3. This figure illustrates the graphical interface of the tool *xfpkg* wherein the Xfuzzy user can define new fuzzy operators (logical connective, linguistic hedges, membership functions, or defuzzification methods).

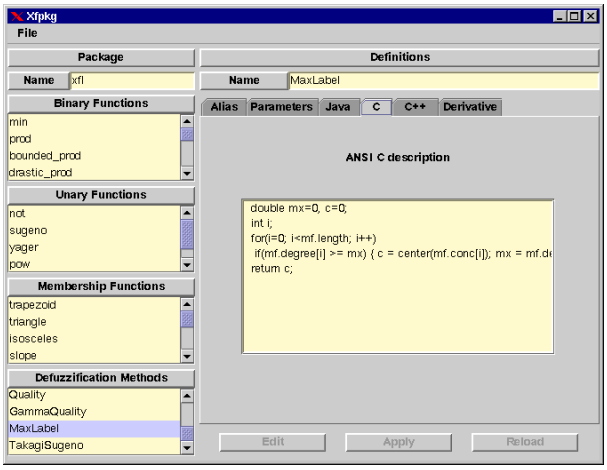


Fig. 3: Graphical user interface of the tool *xfpkg*.

We have used the tool *xfedit* to specify the logical definition of our controller, as can be seen in Figure 4. The membership functions employed are: 5 fuzzy sets to cover the x position; 1 singleton value and 3 fuzzy sets to cover the y position; 7 fuzzy sets to cover the orientation; 5 singleton values to cover the speed (its sign reflects the driving direction); and 7 singleton values to cover the steering wheel angle. The speed values are rather slow as corresponds to parking maneuvers (between -1m/s and 1m/s). The wheel angle values are limited by the maximum curvature that Romeo can apply. Figure 5 shows the window of *xfedit* wherein we have defined the membership functions of the orientation variable.

The rule bases can employ different mathematical functions to represent the fuzzy operators. For example, the rule bases “direction” and “wheel” use different defuzzification methods, as commented above. Figure 6 illustrates the window of *xfedit* wherein we have selected the mathematical functions of the fuzzy operators in the rule base “direction”. An advantage of Xfuzzy 3.0 is that the user can freely modify the mathematical functions that describe these linguistic operators (with the tool *xfpkg* mentioned above).

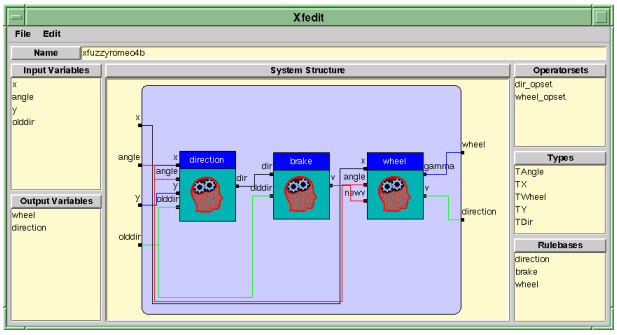


Fig. 4: Main window of *xfedit*.

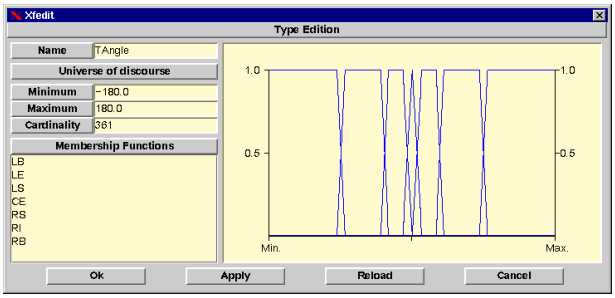


Fig. 5: Membership functions for the vehicle orientation.

We have also used *xfedit* to define the rules of each rule base. The XFL3 language employed by Xfuzzy 3.0 eases the translation of linguistically expressed rules because admits linguistic hedges like “more or less equal to”, “slightly equal to”, etc., and relations like “greater than” or smaller than”. For instance, one of the rules in the rule base “direction” is:

‘if (y is “equal to” *near* and x is “strongly equal to” *center* and angle is “equal or greater than” *left small* and angle is “equal or smaller than” *right small*) then dir is *backward*’.

III. SECOND STEP: OFF-LINE VERIFICATION PROCESS OF THE FUZZY CONTROLLER

Although the definition of the fuzzy controller translates our expert knowledge, we might have forgotten to consider some situations or not consider properly other ones. This is why performance of the controller has to be verified prior to experiment with Romeo 4R. For this purpose, we have employed three verification tools of Xfuzzy 3.0: *xf3dplot*, *xfmt* and *xfsim*.

The tool *xf3dplot* allows us to visualize the behavior of one of the control variables versus two other ones.

Fig. 6: Window of *xfedit* to select the fuzzy operators.

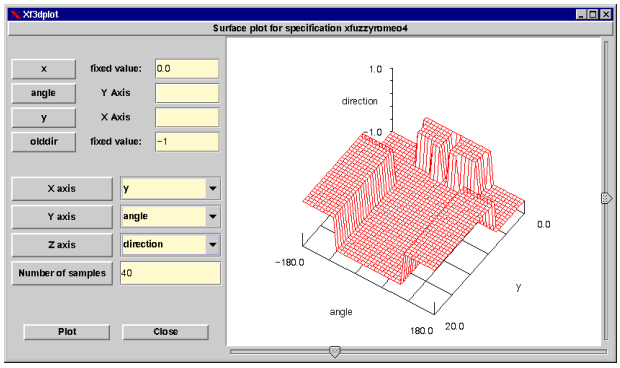


Fig. 7: Studying the behavior of the variable “direction”.

This is very useful to study, for instance, if our controller is safe enough to avoid crashes with a possible pavement at y=0. Figure 7 shows the surface corresponding to the new speed decided by the controller against the y position and the orientation of Romeo, when the x-position coordinate is zero and the previous speed was -1m/s. We can see that if the y position is near zero and the angle is not quite zero, the controller decides to stop to better straighten the car by driving forward in subsequent steps.

The tool *xfmt* is very useful to monitor how is working the inference process. For instance, if we want to know why the new speed is -1m/s when the x position is zero, the previous speed was -1m/s, the y position is 2 m and the angle is 180º, we can use *xfmt* as shown in Figure 8 to discover that the rule responsible of this decision is the rule 30 of the rule base “direction”:

‘if (y is “equal to” *near* and x is “equal to” *center* and (angle is “smaller than” *left* or “greater than” *right*)) then dir is *backward*’.

Although with the previously mentioned tools we can analyze the controller itself, a very important step

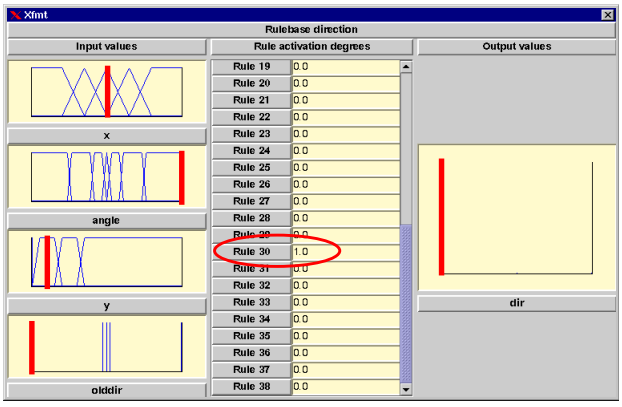


Fig. 8: Monitoring the inference process.

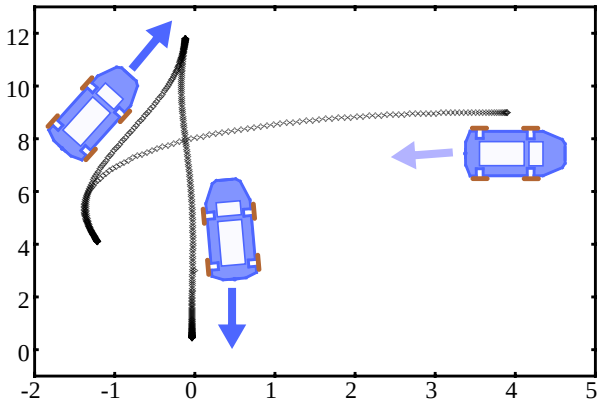


Fig. 9: Simulating the controller with a model of Romeo 4R.

in any control design is to simulate the controller working in a closed loop with the plant. For this simulation, we have employed the tool *xfsim* of Xfuzzy 3.0. The behavior of our plant, Romeo 4R, has been described by the bicycle kinematic model [10], considering a first-order dynamic response in the settling of the speed and the wheel angle imposed by the controller. The outputs of the simulation performed by *xfsim* can be saved to a log file for posterior graphical representation. As an example, Figure 9 illustrates the simulated behavior of Romeo 4R when it starts (with speed 0) at x=3.9m, and y=9m, with an angle of 90º. The arrows indicate the driving direction and the shaded one marks the starting point.

With this off-line simulation, we can analyze the robustness of our controller against perturbations. For instance, if the true speed taken by Romeo 4R is 40% greater than that imposed by the controller and we repeat the simulation of Figure 9, the results obtained are shown in Figure 10.

IV. THIRD STEP: ON-LINE VERIFICATION PROCESS OF THE FUZZY CONTROLLER

Once checked that our system is robust enough, we have verified its control behavior with the true plant, Romeo 4R. This robot is an electrical vehicle provided with a set of sensors and actuators that make it capable of autonomous navigation (Figure 11). The information collected by the sensors and that required by the actuators is centralized by a computer placed at the back of the robot and which also implements the control algorithms. In our parking application, the computer has to govern a motor control card which in turn governs, independently, the steering and traction electrical motors of Romeo. These electrical motors has to receive, respectively, the wheel angle and new speed commands from our controller. In addition, the motor

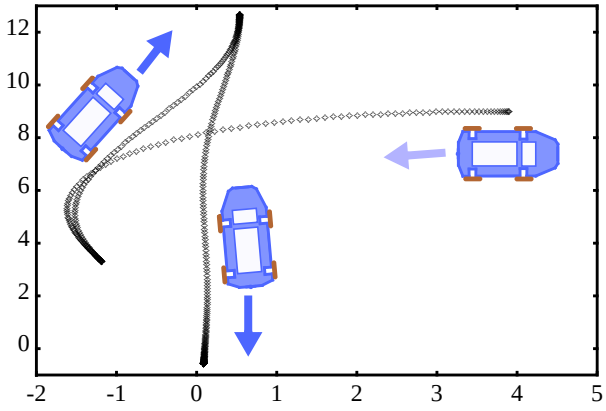


Fig. 10: Simulating the controller with perturbations.

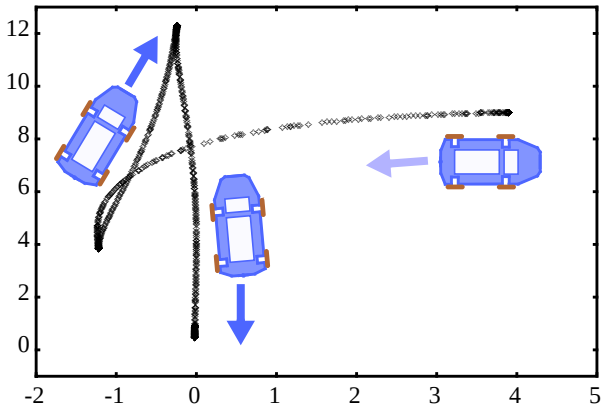
control card reads the direction and traction encoders of the engines. These measures, together with the information provided by a gyroscope have to be processed by the computer to estimate the current position, orientation and speed of the robot, which are the input variables required by our controller.

The computer operates with Linux and all the drivers to the sensors and actuators have been written in C code. In addition, an interface has been developed that include a wide set of functions (programmed in C++) to work easily with sensors and actuators. Having this interface, the inclusion of our controller is as simple as generating its C code. For this task, we have employed the synthesis tool *xfc* of Xfuzzy.

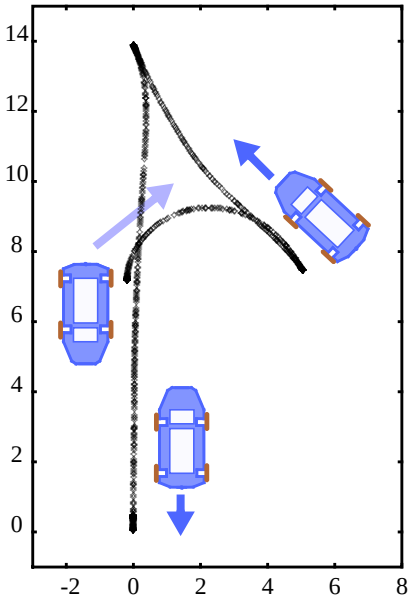
Despite executing the control code and all the other required routines, the computer operates at real time without problems because a control cycle period of 100 ms is enough for our application.

Figure 12 shows two examples of experimental trajectories followed by Romeo when starting at different positions (marked by the shaded arrows) and with different orientations. Comparing Figure 12a with Figure 9 we can see that experimental results do not differ very much from simulated results.

Fig. 11: Romeo 4R successfully parked.



(a)



(b)

Fig. 12: Experimental results.

Figure 13a illustrates the evolution in time of the wheel angle reference given by the fuzzy controller (in solid line) and the real angle taken by Romeo (in dashed line), for the experiment in Figure 12a. We can see how the angle reference changes softly (as a consequence of the Fuzzy Mean defuzzification method applied by the rule base “wheel”) and how it is followed rapidly by the real angle.

Figure 13b compares the evolution in time of both the speed reference (in solid line) and the real speed (in dashed line) corresponding to the experiment in Figure 12a. In this case, the reference changes abruptly (because of the MaxLabel defuzzification method applied) and the dynamic of the real speed is slower

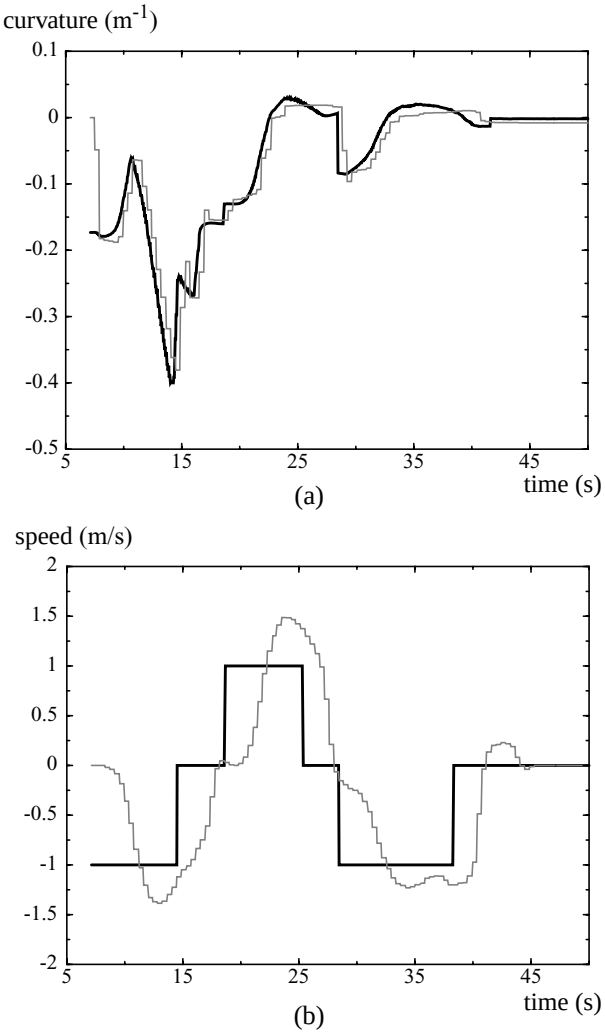


Fig. 13: Evolution of: (a) wheel angle reference (solid line) and real angle (dashed line), (b) speed reference (solid line) and real speed (dashed line).

than that of the real angle. The worthwhile fact shown by this figure is that the robot is controlled efficiently to commute softly between driving backward and forward, thus meeting the imposed requirements.

V. CONCLUSIONS

The automatic design and implementation of fuzzy control systems involves a number of activities related to the definition of the controller, including its logical structure and the mathematical definition of functions, the off-line verification of the controller by means of simulation and the on-line verification and testing with the real process.

This paper presents the application of the Xfuzzy 3.0 tool to the design of the fuzzy controller of autonomous vehicles. Particularly the design and implementation of a controller for the Romeo 4R is described. The results obtained with the real vehicle are similar to the simulation results and Romeo 4R is able to perform successfully a parking maneuver even when the vehicle is initially close to the parking position, with a bad orientation, and having to maneuver autonomously for parking.

Future work will include a comparison of the presented fuzzy logic method with other techniques based on the consecutive execution of path planning, generation and control techniques in different practical cases.

VI REFERENCES

[1] Munakata, T., Jani, Y., Fuzzy systems: an overview, Communications of the ACM, Vol. 37, N. 3, 1994.

[2] Sugeno, M., Ed., Industrial Application of Fuzzy Control, North-Holland, pp. 19-40, 1985.

[3] Home page of FuzzyTech: http://www.fuzzytech.com

[4] Home page of FIDE: http://www.aptronix.com/fide/

[5] Home page of TILShell: http://www.ortech engr.com/fuzzy/TilShell.html

[6] F. J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, A. Barriga, "Xfuzzy 3.0: A Development Environment for Fuzzy Systems", Proc. 2nd IEEE Int. Conf. on Fuzzy Logic and Technology (EUSFLAT'2001), pp. 93-96, Leicester, 2001.

[7] F. J. Moreno-Velo, S. Sánchez-Solano, A. Barriga, I. Baturone, D. R. López, "An Specification Language for Fuzzy Systems", Mathware & Soft Computing, Vol. 8, No. 3, pp. 239-253, 2001.

[8] A. Ollero, B. C. Arrue, J. Ferruz, G. Heredia, F. Cuesta, F. López-Pichaco and C. Nogales. "Control and perception components for autonomous vehicles guidance. Application to the Romeo Vehicles". Control Engineering Practice, Vol. 7, No. 10, pp 1291-1299, October 1999.

[9] S.-G. Kong, B. Kosko, "Comparison of Fuzzy and Neural Truck Backer-Upper Control Systems", Chapter 9 in Neural Networks and Fuzzy Systems, B. Kosko, Prentice Hall, 1992.

[10] Y. Zhao, S. L. Bement, "Kinematics, Dynamics and Control of Wheeled Mobile Robots", Proc. IEEE Int. Conf. on Robotics and Automation, pp. 91-96, Nice, 1992.