Combining Fuzzy Logic and Neural Networks to Control an Autonomous Vehicle

Bernd Freisleben and Thomas Kunkelmann
Department of Computer Science (FB 20), University of Darmstadt
Alexanderstr. 10, D–6100 Darmstadt, Germany

***Abstract*—In this paper we present an approach to design a controller that enables a simulated car to drive autonomously around a race track. The input to the controller is the current speed of the car and several sensor signals indicating the properties of the race track, and as its output the controller is supposed to determine the car's change of direction and its change of speed in response to the information received. The basic idea of our proposal is to let a fuzzy controller supply the training data for a backpropagation neural network and use the trained network to drive the car on an unknown race track. The implementation of the proposal is described and the driving performance of the car is evaluated. The results indicate that the combined neural/fuzzy approach is superior to solutions where either the fuzzy controller or the neural network alone are used to drive the car.**

I. Introduction

Designing a vehicle for totally autonomous operation is a difficult task which encompasses several aspects common to complex control applications, such as path planning, sensory–motor control and obstacle detection/avoidance. The dynamical properties and real–time requirements of vehicle control, together with the limited success of conventional computational methods, have fostered the development of neural network and fuzzy logic techniques as promising approaches to the problem. For example, neural networks have been used to back up a simulated truck [7], and fuzzy controllers have been developed to control a model race car [1, 5, 9].

In this paper we present an approach to the problem of controlling a car to drive autonomously around an unknown race track. The car is simulated in software; it is assumed to be equipped with a number of sensors pointing at different directions in order to view the road conditions. The information delivered by the sensors and the current speed of the car is used to determine its change of direction and speed. The basic idea of our solution is to train a neural network on several, differently structured race tracks and then evaluate its driving behaviour on an unknown race track. The most obvious way to train the network is to use a supervised learning algorithm, where the network tries to learn the mapping between the input data (the sensor signals and the current speed) and the desired actions it should perform (changes of direction and speed). The training set of input/desired output pairs must be delivered to the network, and the question arises where the information should come from. If a human expert is chosen to create the training set, he or she will be busy for quite some time due to the large number of different cases which must be considered in order to prepare the car for all potentially possible driving situations occuring in an unknown terrain. Furthermore, the human expert will probably not be able to always determine the optimal actions for a given input vector, and in most cases he or she will base the decisions for the desired actions on rules of thumb.

A better approach for creating the training data is to develop a set of basic rules from which the desired driving behaviour can be infered automatically. There are several ways to realize this approach, and the one most promising for our application is the use of a *fuzzy* rule base, together with suitable *fuzzy* operators and inference strategies [10]. Thus, the neural network designed for driving the car is trained with a set of input/output examples which are supplied by a *fuzzy controller* after having driven the car on the track. In other words, the fuzzy controller acts as a training preprocessor to the neural network, in contrast to other combined neural/fuzzy proposals [4], where the neural units consider the incoming signals as fuzzy sets and process them according to the mechanisms employed in fuzzy theory.

The proposal of combining a fuzzy controller with a neural network as described above is compared to an approach where a fuzzy controller alone is used to drive the car and also to an approach where a neural network, trained with hand–coded data, is responsible for performing the task. It will be shown that the combined approach is superior to the other two, both in terms of driving quality and efficiency. This, however, does not imply that a properly designed fuzzy controller cannot achieve the same

performance as the combined approach. Our intention is to demonstrate that the large effort required for developing a high–quality fuzzy controller can be avoided by designing a fuzzy controller in a quick–and–dirty manner and combining it with a simple to implement neural network as described above.

The paper is organized as follows. Section 2 describes the general idea behind our proposal in more detail. In section 3 the design of the fuzzy controller is presented, whereas section 4 is devoted to the neural network model used. The implementation of our proposal and its performance are discussed in section 5. Section 6 concludes the paper and outlines directions for future research.

II. General Approach

The general architecture of our system is shown in Fig. 1.

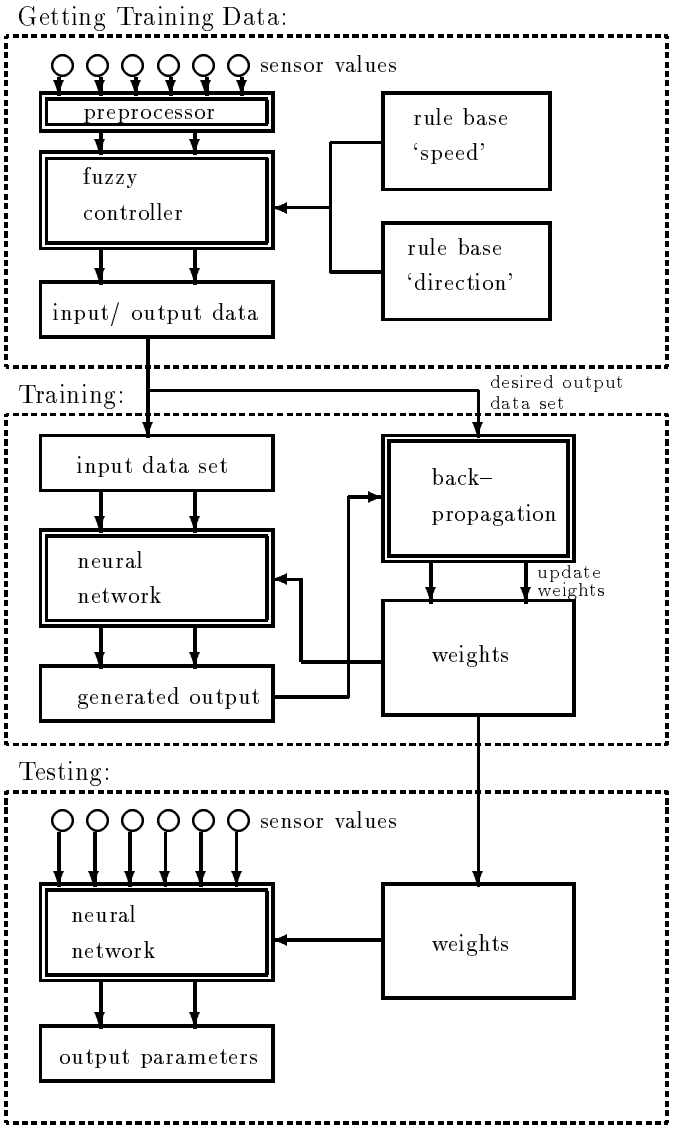

Fig. 1. General architecture of the proposed system

The training data for the neural network is obtained by letting the fuzzy controller drive the car around several sample tracks. The fuzzy controller successively receives the input vectors, suitably preprocessed and transformed into fuzzy sets (see section 3), and processes each of them individually by applying its fuzzy rules, operators and inference strategy to determine what the car should do in response to the input. Since the two possible actions are a change of the direction of movement and a change of the speed (either accelerating the car or slowing it down), the fuzzy controller is internally divided into two nearly identical parts which operate on different rule bases, one for the steering and the other one for the acceleration.

The input/output pairs obtained on the various race tracks are stored and the neural network is then put in charge of driving the car. It is trained with the back-propagation algorithm [8] on the data set created by the fuzzy controller until it has learned the output actions determined by the fuzzy controller, i.e. it basically becomes a "clone" of the fuzzy controller in the sense that its driving behaviour on the training tracks mimics the one of the fuzzy controller. The trained network is then used to drive the car around an unknown race track, the implicit assumption being that the neural network's generalization ability will be superior to the driving capabilities of the fuzzy controller exposed to the same unknown race track.

III. The Fuzzy Controller

In this section we present the functionality of the fuzzy controller developed for our application (see Fig. 2). The design of the fuzzy controller is based on the standard procedure of developing a rule base containing *fuzzy if–then rules*, defining *linguistic variables* and using *defuzzification* heuristics [10]. A linguistic variable can adopt different values; each of these is a fuzzy set which is used to represent a particular interval within the range of possible values for the linguistic variable considered. Usually, all input and output parameters of the fuzzy controller are treated as linguistic variables, and it is therefore required to define appropriate fuzzy sets for each of them.

The raw input data is preprocessed to obtain the appropriate fuzzy sets. The fuzzy sets for the input parameter *speed (ISP)* are relatively straightforward to determine. The idea is to divide the range of possible speed values (in our case integer values between 0 and 30, normalized to [0,1]) into suitably sized intervals; we have decided to use the 6 intervals depicted in Table I.

In order to determine the linguistic variables and their values for the sensors, it is necessary to explain how they are installed on the simulated car and what information they deliver. We assume that sets of 5 or 6 different sensors are grouped together to approximately point to one

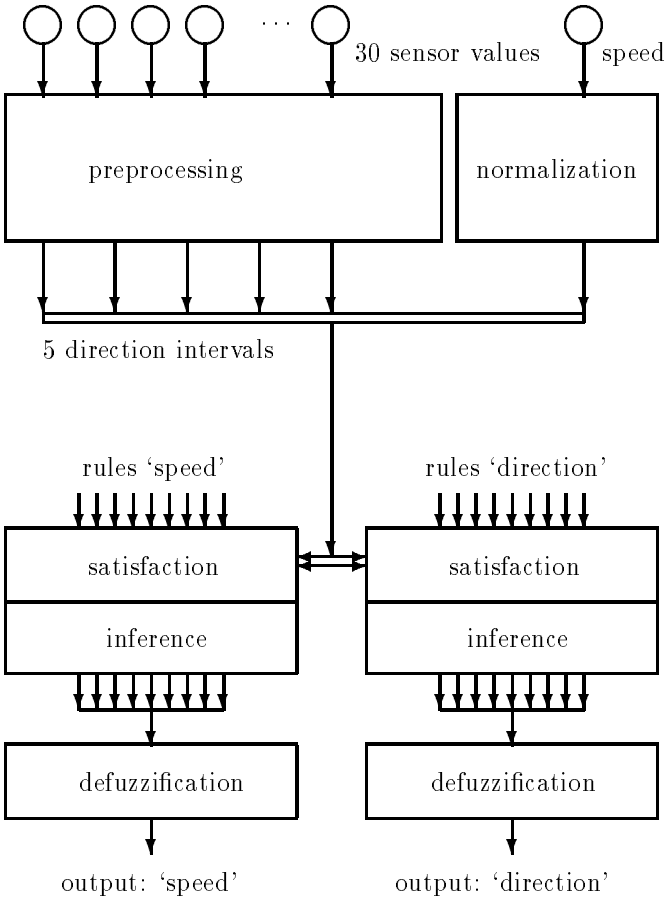

Fig. 2. Architecture of the fuzzy controller

TABLE I
FUZZY SETS FOR INPUT SPEED

ISP	input speed	intervals
HA	halt	0 – 5 %
SD	slide	5 – 20 %
SL	slow	20 – 40 %
NO	normal	40 – 65 %
FS	fast	65 – 85 %
TS	topspeed	85 – 100 %

out of 5 different directions, as shown in Fig. 3. The values on the two axis are distances (in meters).

The 5 directions are denoted as: *Direction Front Left (DFL)*, *Direction Front Middle (DFM)*, *Direction Front Right (DFR)*, *Direction Side Left (DSL)* and *Direction Side Right (DSR)*. The sensors pointing to a particular direction return integer values between 0 and 15 to indicate what they see on the track (pavement, border, obstacle etc.), each of them being responsible for a particular distance in that direction. This distance is the second linguistic variable used in our model, and since there are

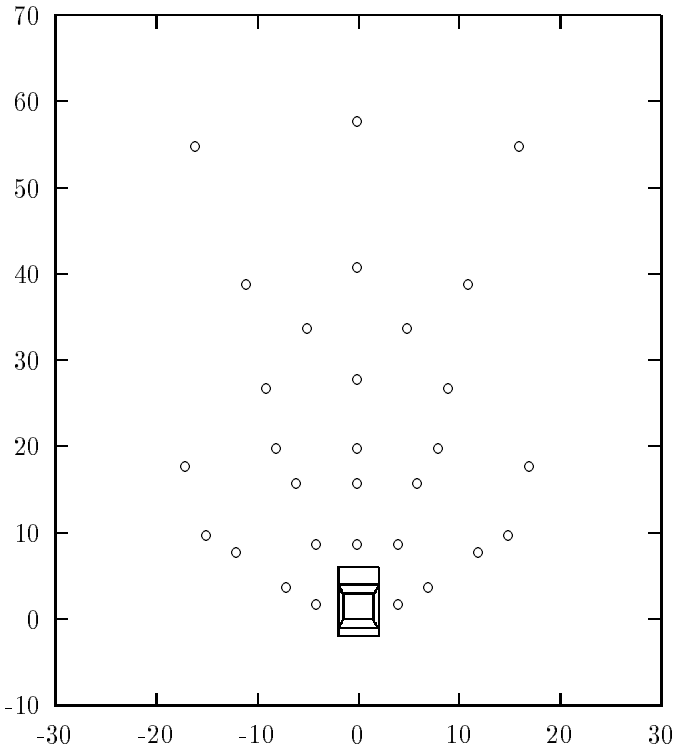

Fig. 3. Sensor positioning

directions emanating from the front and the side of the car, it seems reasonable to distinguish between the two linguistic variables *Frontal Distance (FDI)* and *Side Distance (SDI)*. The fuzzy sets for these linguistic variables have been determined as *Touch (FT/ST)*, *Accidental (FA/SA)*, *Critical (FC/SC)*, *Normal (FN/SN)*, *Far (FF/SF)* and *Infinite (FI/SI)*, where the letter F in the abbreviations indicates *Frontal* and S indicates *Side*. The fuzzy numbers for the linguistic variables associated with the sensors are summarized in Table II.

The two outputs of the fuzzy controller are also represented as linguistic variables. The range of the parameter *Change of Speed (COS)* (integer values between -4 and 2) is divided into 6 intervals, and the range of the parameter *Change of Direction (COD)* (values between -30 and $+30$ degrees, normalized to [-1,1]) is divided into 9 distinct intervals, as shown in Table III. For both of them, the interval ranges defined exceed the total range possible, in order to ensure that the maximal values can be definitely reached.

The first step the fuzzy controller has to perform is to calculate for each rule contained in the rule base the degree to which the rule in discussion is satisfied. In order to do so, the fuzzy controller matches the membership function of the inputs with the linguistic values present in the rule. This is achieved by applying a *fuzzy–AND* operator to the *AND–terms* of the rule, which in our design is the standard

TABLE II

FUZZY SETS FOR THE SENSOR SIGNALS

FDI	frontal distance (*DFL, DFM, DFR*)	values
FT	frontal touch	0 m
FA	frontal accidental	3 m
FC	frontal critical	6 m
FN	frontal near	9 m
FF	frontal far	15 m
FI	frontal infinite	$>$ 15 m

SDI	side distance (*DSL, DSR*)	values
ST	side touch	0 m
SA	side accidental	1 m
SC	side critical	2.5 m
SN	side normal	4.5 m
SF	side far	7 m
SI	side infinite	$>$ 7 m

TABLE III

FUZZY SETS FOR THE OUTPUT PARAMETERS

COS	change of speed	intervals
EMB	emergency brake	-4.5 – -3
STB	strong brake	-3 – -2
SOB	soft brake	-2 – -0.5
CON	continue	-0.5 – 0.5
ACC	accelerate	0.5 – 1.5
FSP	full speed	1.5 – 2.5

COD	change of direction	intervals
LL	left always	$\ll$ -100 %
LB	left big	-100 – -75 %
LM	left medium	-75 – -45 %
LS	left small	-45 – -15 %
ZE	zero	-15 – 15 %
RS	right small	15 – 45 %
RM	right medium	45 – 75 %
RB	right big	75 – 100 %
RR	right always	$\gg$ 100 %

minimum operator.

The resulting match value is used to compute the inference result. The method used in our design is to determine the minimum between the match value and the result of the rule, i.e. the fuzzy set. The fuzzy sets originating from the inference computation are then used to determine the final result. A *fuzzy–OR* operator, realized by the *maximum* operator, is applied to all fuzzy sets to obtain the final result. The result is again a fuzzy set which is transformed into a particular crisp value (*defuzzification*) by computing the center of the area below the membership function.

The two rule bases developed consist of about 100 *if–then* rules each, which were defined on the grounds of plausibility and were successively extended or refined to yield the desired behaviour of the car. An example of such an intuitive rule for the change of speed is:

```
ISP = NO & DFM = FI & DFL = FI : FSP
```

which indicates that if the car drives at normal speed (`ISP = NO`) *and* if no obstacle has been recognized by the sensors positioned at the middle front (`DFM = FI`) *and* the left front (`DFL = FI`) of the car, then it should accelerate as much as possible (`FSP: Full Speed`). Once a basic set of such rules has been set up to model the car's fundamental capabilities, the car will make its way through the race track. In our application, about 50 basic rules were required in each of the rule bases, but these rules are not sufficient to achieve a satisfactory driving performance, particularly in somewhat extreme situations (U–turns etc.). Refining the control strategy requires modifying, deleting or adding rules in a trial–and–error fashion, which often is a time–consuming process. Several proposals have been made to remedy this problem [1, 2, 6].

IV. THE NEURAL NETWORK

The neural network used in our proposal is a standard three–layer feedforward architecture, where the input layer consist of 31 neural units (one for each of the 30 sensors and one for the current speed of the car), and the output layer has two units to determine the change of direction and the change of speed, respectively (Fig. 4).

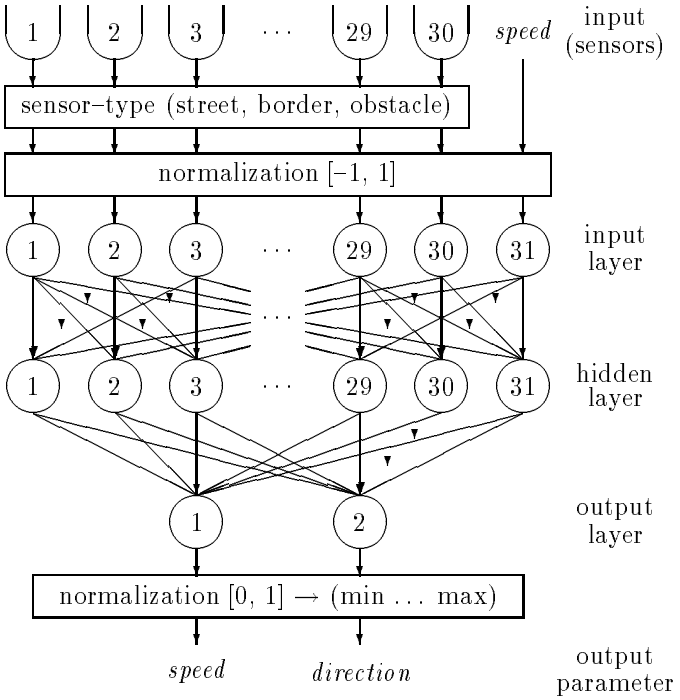

Fig. 4. Neural network architecture

The number of units in the hidden layer has been determined empirically, and the best results were obtained with 31 hidden units. The backpropagation algorithm [8] with momentum term [3] is used for training the network. Since the sigmoid activation functions of the output units produce values between 0 and 1, the network outputs are converted to the ranges adopted for the change of speed and the change of direction.

Since the car is not only supposed to drive safely through the race track and avoid any fixed obstacles, but also should compete against another car simultaneously on the track, the opponent car, which effectively constitutes a moving obstacle, must be appropriately encoded. This is achieved by determining the sensor which has detected the opponent car and check whether the nearby sensors signal the left or right border of the lane; the opponent's car is then treated like a border.

In the training mode, each input vector, stemming from one out of three rounds on three different race tracks, has been presented to the network about 100 times until the network had reduced the error below a predefined threshold. The training times on a SUN Sparcstation were about 60 minutes for processing the whole training set consisting of about 2500 input vectors.

V. Implementation and Performance

Both the fuzzy controller and the neural network have been implemented in *C*. The neural network was trained on a SUN Sparcstation, but the trained network used to control the car was run on an IBM–PC, since the race track was simulated by using the graphics features of the PC. Several artificial race tracks were generated to train the network. An example of such a race track is shown in Fig. 5.

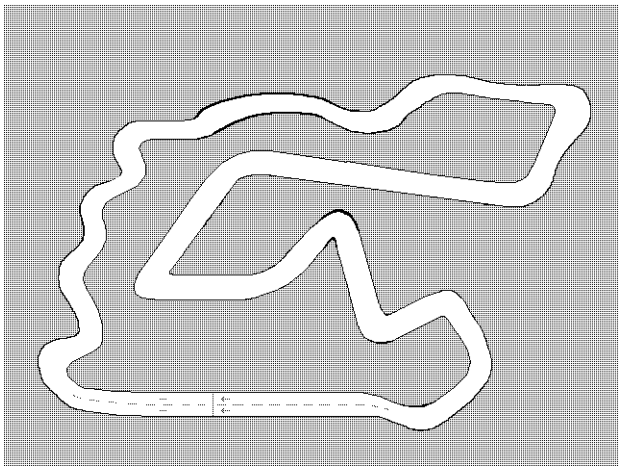

Fig. 5. A race track used for training

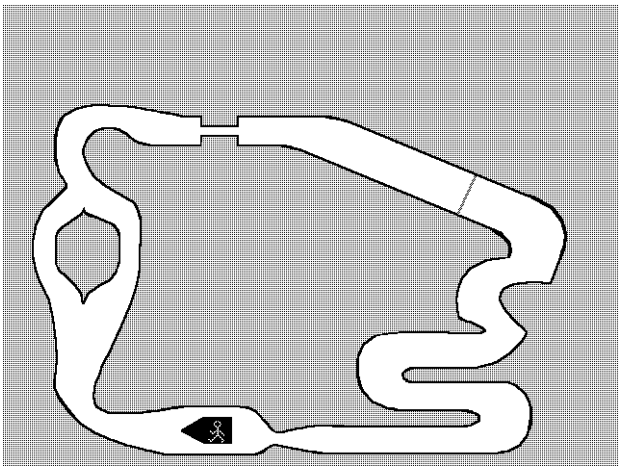

Fig. 6. A race track used for testing

The performance of the three controllers developed for driving the car, the fuzzy controller, the neural network trained with hand–coded training data and the combined fuzzy/neural approach, was measured in several rounds on several unknown test tracks. One of these is graphically shown in Fig. 6.

The first set of experiments conducted was to evaluate the driving capabilities of the three different controllers on a training track. In each experiment, two cars, driven by two different controllers, were competing against each other by permanently switching between the two after each of them has performed one simulation step to update the car's position. Whenever a car ran over the border of the track or touched the other car or an obstacle, a handicap, represented by some simulation steps of forced inactivity, was burdened on the car. All possible combinations of competing cars (fuzzy controller vs. hand–trained neural network, fuzzy controller vs. fuzzy/neural network, hand–trained neural network vs. fuzzy/neural network) were investigated two times. The second set of experiments was conducted analogously on an unknown test track. The performance obtained in the experiments is summarized in Table IV.

The values shown for the total number of simulation steps used and the number of steps considered as handicaps are average values of all races performed. On the training track, the driving performance of the controllers is not significantly different. All three controllers won and lost two races each; the neural network drives the car faster than the other two, but produces more accidents. The best driving quality is achieved by the fuzzy/neural network, since it never caused a crash.

On the test track, the situation is different. Although the relationships between the number of active steps are similar to those obtained on the training track, the driving quality of the fuzzy/neural network outperforms the other

TABLE IV

DRIVING PERFORMANCE

car driver	*training track* #_steps + #_handicap = #_total	won : lost	*test track* #_steps + #_handicap =#_total	won : lost
fuzzy controller	87 + 6 = 93	2 : 2	176 + 50 = 226	3 : 1
neural net	77 + 23 = 100	2 : 2	153 + 93 = 246	0 : 4
fuzzy/neural net	96 + 0 = 96	2 : 2	186 + 14 = 200	3 : 1

controllers, due to the fact that it produces far less accidents and therefore achieves the best result for the total number of simulation steps needed. It did, however, not succeed in winning all races, but lost in one race against the fuzzy controller.

Considering that in the combined fuzzy/neural approach the neural network was supposed to learn from the fuzzy controller how to drive, it seems somewhat surprising that the neural network is better than its teacher. One possible explanation for this is that the network is probably able to generalize better than the fuzzy controller. It should be noted that this does not imply that a fuzzy controller or a neural network cannot be designed in a way such that their individual driving performance outperforms the combined approach. However, supposing that the results obtained in our experiments do also hold in other driving environments, it might be reasonable to assume that a well designed fuzzy controller employed as the teacher of a neural network will lead to further performance improvements in a combined fuzzy/neural approach.

VI. CONCLUSIONS

In this paper we have presented a combined neural/fuzzy approach to drive a simulated car around a race track. Assuming that the car is equipped with a set of sensors recognizing the properties of the track and the current speed is known, the task of the controller is to determine the car's change of direction and change of speed in response to the information received. The basic idea of our proposal was to develop a fuzzy controller for driving the car and use its outputs for creating the set of training patterns required to let a standard backpropagation neural network learn how to drive. We have implemented both the fuzzy controller and the neural network and have evaluated the driving performance obtained on several unknown race tracks simulated on an IBM–PC. The results have shown that the combined approach is superior to solutions where either the fuzzy controller or the network alone were used to drive the car. There are several issues for future research, such as using more sophisticated fuzzy design techniques than the simple *if–then* rules and *MAX/MIN* operators [1], testing other neural architectures and learning algorithms, and investigating if a combination of a fuzzy controller and a neural network in the manner proposed will be beneficial for other control applications.

REFERENCES

[1] C. von Altrock, B. Krause, and H.J. Zimmermann, "Advanced fuzzy logic control of a model car in extreme situations," *Fuzzy Sets and Systems*, 48(1):41–52, 1992.

[2] J.C. Fodor, "On fuzzy implication operators," *Fuzzy Sets and Systems*, 42:293–300, 1991.

[3] J.A. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*, Addison–Wesley, Reading, Massachusetts, 1991.

[4] C. Lin and C.S. Lee, "Neural–network–based fuzzy logic control and decision system," *IEEE Transactions on Computers*, 40(12):1320–1336, 1991.

[5] M. Maeda, Y. Maeda, and S. Murakami, "Fuzzy drive control of an autonomous mobile robot," *Fuzzy Sets and Systems*, 39:195–204, 1991.

[6] M. Mitsumoto and H.-J. Zimmermann, "Comparison of fuzzy reasoning methods," *Fuzzy Sets and Systems*, 8:253–285, 1992.

[7] D. Nguyen and B. Widrow, "The truck backer–upper: An example of self–learning in neural networks," *Proc. of the Int. Joint Conference on Neural Networks*, vol. 2, pp. 357–364, 1989.

[8] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," In: D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing*, vol. 1, pp. 318–362, MIT Press, Cambridge, 1986.

[9] M. Sugeno, T. Murofushi, T. Mori, T. Tatematsu, and J. Tanaka, "Fuzzy algorithmic control of a model car by oral instructions," *Fuzzy Sets and Systems*, 32:207–219, 1989.

[10] H.-J. Zimmermann, *Fuzzy Set Theory – and its Applications*, Kluwer Academic, Boston, 1991.