Fuzzy/Neural Drive Control of an Autonomous Vehicle

(Extended Abstract)

Bernd Freisleben[1] and Thomas Kunkelmann[2]

[1]Department of Electrical Engineering and Computer Science, University of Siegen
Hölderlinstr. 3, D-57068 Siegen, Germany
[2]Department of Computer Science, University of Darmstadt
Alexanderstr. 10, D–64283 Darmstadt, Germany

1 Introduction

The limited success of conventional approaches to dynamical, noisy, non–linear control problems have fostered the development and use of neural networks and fuzzy logic techniques as promising alternatives for solving the issues associated with complex control applications. An application area which requires to deal with several aspects of difficult, real-time control tasks, such as path planning, sensory–motor control and obstacle detection/avoidance, is vehicle control. Several proposals for enabling a vehicle to drive autonomously in unknown terrain, both based on neural networks [8] and fuzzy controllers [1, 6, 10], have already been suggested.

In this paper we present and compare different approaches to the problem of controlling a car to drive autonomously around an unknown race track. The functionality of the car is simulated in software; in order to view the road conditions, the car is assumed to be equipped with a number of sensors pointing at different directions. Based on the information delivered by the sensors and the current speed of the car the controller's task is to determine the car's change of direction and speed.

In the first approach, a *fuzzy controller*, operating on a fuzzy rule base from which the desired driving behaviour of the car can be deduced by applying suitable fuzzy operators and inference strategies [11], is used.

The second approach is based on training a *neural network* to learn the mapping between the input data and the desired actions it should perform in order to enable the network to generalize to unknown situations. If a reasonable training set of input/desired output pairs is created by a human expert, 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. Thus, a better way of producing the training set is to let a fuzzy controller supply a set of input/output examples after having driven the car on the race track. In this case the fuzzy controller acts as the teacher of the neural network.

In a third approach, the car is controlled by a *fuzzy network*, i.e. another combination of neural and fuzzy techniques, in which the units of a neural network consider the incoming signals as fuzzy sets and process them according to the mechanisms employed in fuzzy theory [5].

The three approaches are compared to each other. It will be shown that the second approach is superior to the other two in the investigated scenario, both in terms of driving quality and efficiency. This, however, does not imply that it is impossible to develop a fuzzy controller or a fuzzy network as in the third approach which both would be able to achieve very good performance in driving the car. Our intention is to demonstrate that the large effort usually required for developing a high–quality fuzzy controller or a fuzzy network can be avoided by designing a fuzzy controller in a quick–and–dirty manner and combining it with a standard neural network as described above to simply exploit the network's generalization abilities.

2 The Fuzzy Controller

The design of the fuzzy controller developed for controlling the car in our application (see Fig. 1) is based on the standard procedure of producing a rule base containing *fuzzy if–then rules*, defining *linguistic variables* and using *defuzzification* heuristics [11].

Since the input and output parameters of the fuzzy controller are treated as linguistic variables, the raw input data (the current speed and the sensor signals) is preprocessed to obtain appropriate fuzzy sets. The fuzzy sets for the input parameter *speed (ISP)* are determined by dividing the range of possible speed

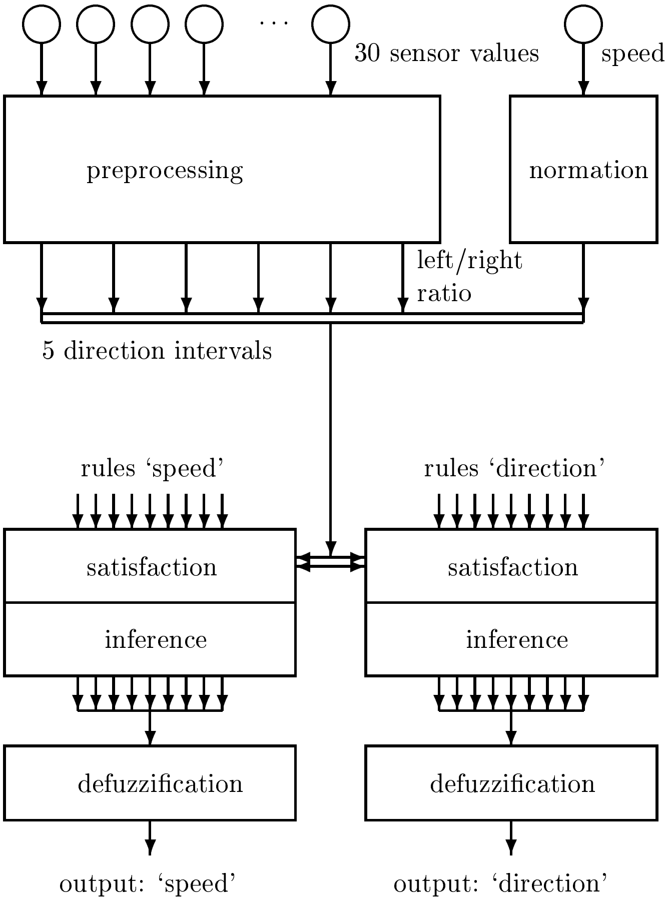

Fig. 1. Architecture of the fuzzy controller

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 %

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 out of 5 different directions, as shown in Fig. 2. 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 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*.

In addition, there is a further linguistic variable *Orientation (ORI)*, which is used to find out whether the car is moving towards the left or right side of the track. Without this variable it would be impossible for the controller to decide what direction should be taken if, for example, the car is moving orthogonally towards a border of the track. A wrong decision may result traversing the race track in the opposite direction. In order to provide appropriate information for solving this problem, the number of sensor directions recognizing the right border is subtracted from the number of directions recognizing the left border. Since there are 5 main directions the sensors point to, the values obtained with this method are between -5 and 5. The appropriate fuzzy sets for the linguistic variable *Orientation (ORI)* have consequently been determined as *Left Dominant (LD)*, *Left (LE)*, *Equal (EQ)*, *Right (RI)* and *Right Dominant (RD)*. The fuzzy numbers for the linguistic variables associated with the sensors are summarized in Table II.

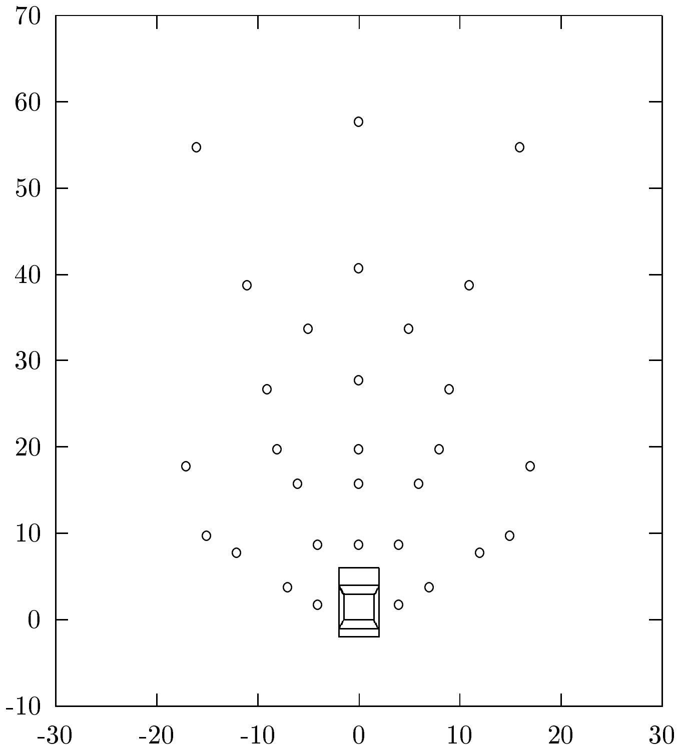

Fig. 2. Sensor positioning

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

ORI	orientation	values
LD	left dominant	≤ -3
LE	left	-2 – -1
EQ	equal	0
RI	right	1 – 2
RD	right dominant	≥ 3

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	≪ -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	≫ 100 %

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 *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.

Since the two possible actions of the car 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 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.

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 were 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 is a quite time–consuming process. Several proposals have been made to remedy this problem [1, 2, 7].

3 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. 3).

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 [9] with momentum term [4] 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.

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 2), 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. 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 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.

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.

4 The Fuzzy Network

In the proposal made in [5], a neural network is used to simulate the operation of a fuzzy controller. The idea is transfer the behaviour of a fuzzy controller via learning to a neural network and to model the functionality of the individual components of a fuzzy controller within the different layers of a suitable

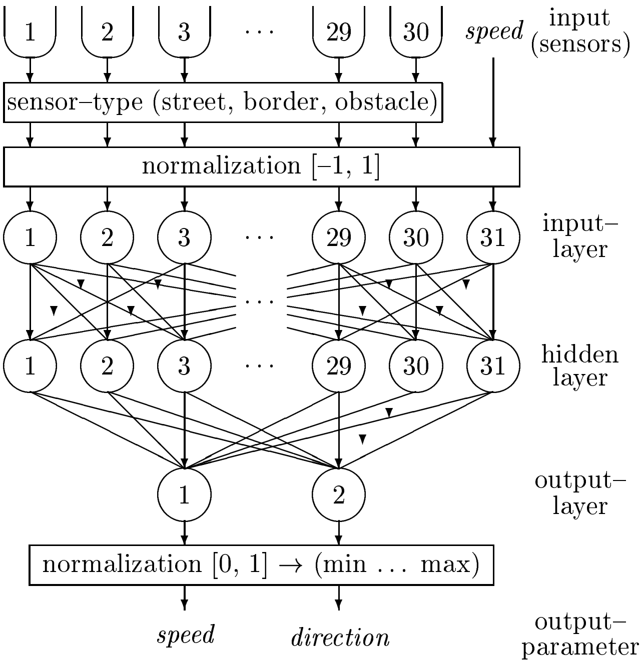

Fig. 3. Neural network architecture

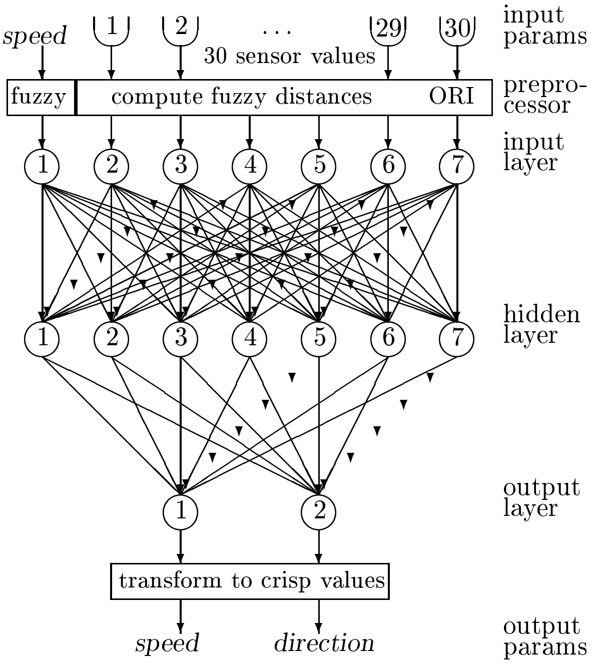

Fig. 4. Fuzzy network architecture

network architecture. Another approach where the units of a standard neural network process the incoming information by fuzzy operators, has been presented in [3]. We have combined the two approaches to design a fuzzy neural network architecture which operates on fuzzy sets.

In contrast to the approach described in the previous section, the neural network should not only learn the driving skills of the fuzzy controller, but also should treat the input information as fuzzy sets. Thus, the preprocessing stage in which the raw input is transformed into fuzzy sets is identical to that of the fuzzy controller. In contrast to the fuzzy controller, where the resulting distance values of the sensors are directly used as inputs to the controller, linguistic terms are used to assign the distance values to the different sensors in the fuzzy network. The architecture of the fuzzy network is shown in Fig. 4. It consists of 7 input units (for input speed, orientation and 5 distance intervals), 7 hidden units and 2 output units; backpropagation is used as the learning algorithm. The weights and thresholds of the units are the usual crisp values, i.e. weight updates require to transform fuzzy into crisp values when the input layer is concerned. The operations performed by the units (addition, exponentiation, division) are the standard operations on fuzzy numbers. The defuzzification heuristics for the network outputs are identical to those employed in the fuzzy controller.

5 Implementation and Performance

The fuzzy controller, the neural network and the fuzzy network have been implemented in *C*. The neural network and the fuzzy network were trained on a SUN Sparcstation, but the trained networks used to control the car were 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 networks.

The performance of the three controllers developed for driving the car, the fuzzy controller, the neural network trained on the data supplied by the fuzzy controller and the fuzzy network, was measured in several rounds on several unknown test tracks.

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. neural network, fuzzy controller vs. fuzzy network, neural network vs. fuzzy network) were investigated 20 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

TABLE IV
Driving Performance

	training track		test track	
car driver	#_S + #_H = #_T	won : lost	#_S + #_H =#_T	won : lost
fuzzy controller	220 + 9 = 229	23 : 17	96 + 3 = 99	14 : 26
neural net	209 + 7 = 216	25 : 15	85 + 3 = 88	30 : 10
fuzzy net	241 + 31 = 272	12 : 28	90 + 5 = 95	16 : 24

#_S : number of steps per round, #_H : number of handicaps, #_T : total number of steps

as handicaps are average values of all races performed. On the training track, the driving performance of the fuzzy controller and the neural network is not significantly different (each of them won ten races when competing against each other). The fuzzy network could not achieve that performance, since it only won 5 races against the neural network and 7 against the fuzzy controller. The best driving quality is obtained by the neural network, since it produced the least number of accidents, indicated by the smallest number of handicaps.

On the test track, the situation is different. The neural network clearly outperforms the other controllers, due to the fact that it requires the smallest number of simulation steps and it won 15 out of 20 races against each of the two controllers. The performances of the fuzzy controller and the fuzzy network are approximately the same.

Considering that 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. The neural network generalization abilities also seem to explain why the fuzzy network could improve its performance on the test track when driving against the fuzzy controller. It should be noted that these results do not suggest that it is impossible to design a fuzzy controller or a fuzzy network in a way such that their individual driving performance outperforms the neural network. 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 the neural network approach.

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] T.H. Goh, P.Z. Wang, and H.C. Lui, "Learning algorithm for the enhanced fuzzy perceptron," *Advances in Neural Information Processing Systems 5*, 1993.

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

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

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

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

[8] 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.

[9] 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.

[10] 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.

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