6th International Workshop, Fuzzy–Neuro–Systems'99, Leipzig

Neurofuzzy Architecture for Local Navigation and Obstacle Avoidance for the Mobile Robot *Milva*

Andreas Erler[1], **Klaus Debes**[2], **Horst-Michael Groß**[3]

University of Technology Ilmenau (Germany)
Department of Computational Neuroscience
98684 Ilmenau

Abstract: *A Neuro–Fuzzy Architecture is used to control the mobile autonomous robot* MILVA. *It enables the robot to perform several basic operations like obstacle avoidance, target following and local navigation in real world environments. The Neuro–Fuzzy Network is used to provide the robot with a base of initial knowledge in order to accelerate the supervised learning process. Linguistic Variables represent the base by Terms of (Fuzzy–Sets) and* IF .. THEN .. *production rules. The Neuro–Fuzzy Network inherits the knowledge base and is optimized by a backpropagation training algorithm to improve its behaviour.*

Key Words: Navigation of autonomous Mobile Robots, Fuzzy Control, Neural Networks

1 Introduction

The task is to design a navigation system that enables the robot to perform a collision free movement in a natural environment. For that purpose, the mobile robot *MILVA* is equipped with a laser scanner, bumpers with tactile sensors, infrared sensors, and 3 cameras. Two of the cameras are mounted on a PU (pan unit) and one on a PTU (pan tilt unit). The navigation

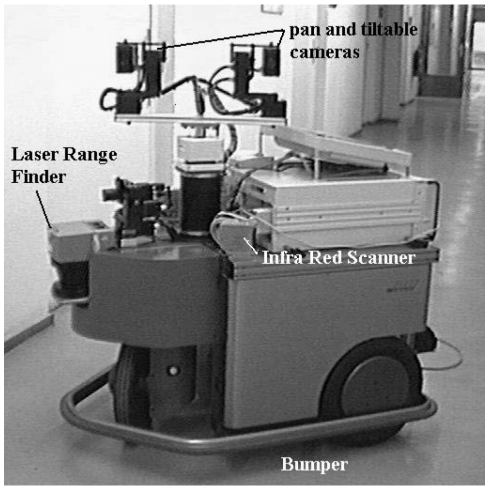

Figure 1: *The mobile Robot MILVA with three cameras, bumpers and a laser scanner*

[1]E-mail: andreas.erler@rz.tu-ilmenau.de
[2]E-mail: klaus.debes@informatik.tu-ilmenau.de
[3]E-mail: homi@informatik.tu-ilmenau.de

system provides the robot with target tracking ability and static and dynamic obstacles can be avoided. Since *MILVA* is a non–holonomic robot, a tractory needs to be considered while turning left or right to avoid collisions at the long sides. Furthermore, the navigation system requires adaptability to different geometric robot shapes to transfer the system onto similar robots but with a different shape. The neuro–fuzzy architecture described here is able to deal with all these issues. The input data consists of a horizontal laser scan profile and the direction and the distance of the movement target, extracted from camera images. The teacher input (steering angle) is provided via joystick.

2 The used Network

The proposed network was first introduced by TRESP [7]. This network architecture is based on RBF–Networks. The used topology consists of four layers. Each input unit represents one input variable. The input layer is followed by the first hidden layer. The neurons of this layer represent the input fuzzy sets of the neuro–fuzzy architecture. Different types of neurons represent gaussian, s-shaped, or z-shaped membership functions of the input fuzzy sets. In this part of the neuro–fuzzy network the fuzzyfication is done and a mapping from crisp input values to internal “fuzzy” values is performed.

Gaussian–function:

$$y^{H1}(x) = e^{-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2} \tag{1}$$

Sigmoid–function:

$$y_i^{H1}(x) = \frac{1}{1+e^{-n(x-\mu_i)\sigma_i}} \tag{2}$$

x	:	input value
μ_i	:	value of membership function
σ_i	:	variance
$y_i^{H1}(x)$	:	output of first hidden layer neuron i

The RBF–network output is computed as (see [10]):

$$y_j^{O}(x) = \sum_{i=1}^{n} w_{ji}\, y_i^{H1}(x) + bias_{out} \tag{3}$$

$y_i^{H1}(x)$	:	output of first hidden layer neuron *i*
$y_j^{O}(x)$	:	output of output neuron *j*
w_{ji}	:	weight of the connection between neuron *i* in the first hidden layer and output neuron *j*

The gaussian activation functions of the neurons in the first hidden layer only have local character. This leads to a disadvantage when the input values show big euclidian distances to the centers of the gaussian functions. To avoid this, the RBF functions are normalized and called Partitioning–To–One (PTO) [8]. Now the network equation looks like this:

$$y_j^O(x) = \frac{\sum_{i=1}^{n} w_{ij}\, y_i^{H1}(x)}{\sum_{i=1}^{n} y_i^{H1}(x)} \tag{4}$$

n	:	number of rule neurons
$y^O j(x)$	:	output of output neuron j
$y_i^{H1}(x)$	:	output of neuron i in the first hidden layer
w_{ij}	:	weight of the connection between neuron i in the first hidden layer and output neuron j

The used network contains an additional layer between the first and the second hidden layer containing a binary matrix. These values indicate whether an input fuzzy set i is part of the rule j or not. The rules are represented by neurons of the second hidden layer.
One more parameter is now added to the neurons of the second hidden layer. This parameter k_i ranges between 0 and 1 and expresses the *degree of support* or the *importance* of each rule. The whole neuro–fuzzy network is shown in figure 2.

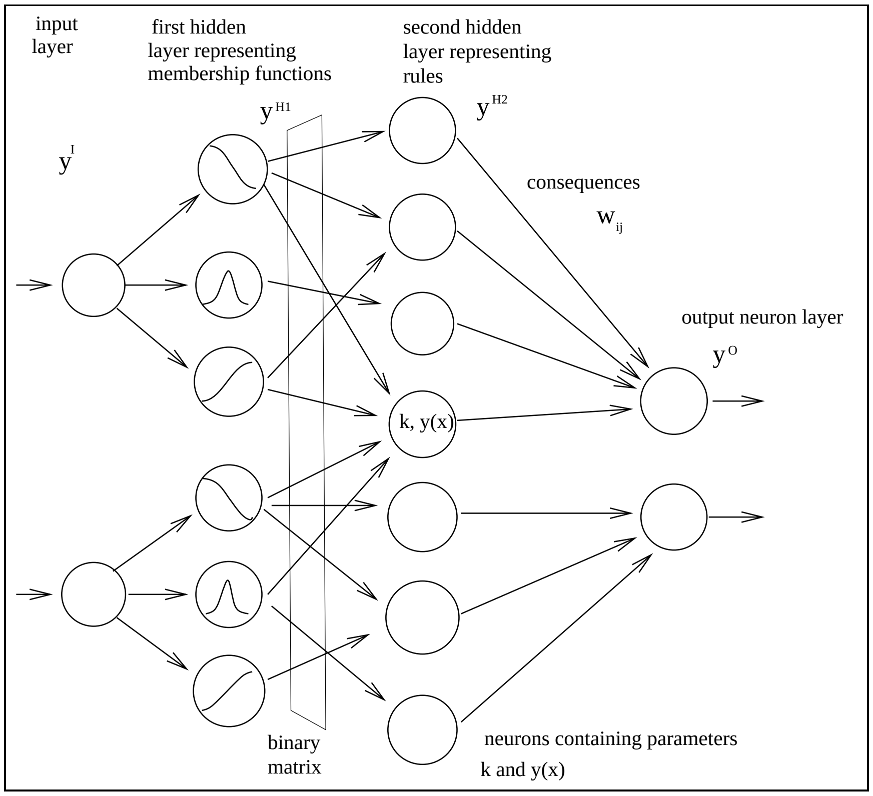
input layer
first hidden layer representing membership functions
second hidden layer representing rules
y^H1
y^H2
y^I
consequences
w_ij
output neuron layer
y^O
k, y(x)
binary matrix
neurons containing parameters k and y(x)

Figure 2: *The enhanced neuro–fuzzy topology with the binary matrix. The weights between the first and second hidden layer are binary. The used network contains 9 input neurons and 2 output neurons (steering angle and speed).*

The last layer of the neuro–fuzzy architecture is the output layer where defuzzification is done. The output singletons are mapped to a crisp output value for each output variable, using the method of SUGENO [11]:

$$y_j^O(x) = \frac{\sum_{i=1}^{n} w_{ij}\, k_i\, y_i^{H2}(x)}{\sum_{i=1}^{n} k_i\, y_i^{H2}(x)} \tag{5}$$

n	:	number of rule neurons
$y_j^{H2}(x)$	:	output of neuron j of the second hidden layer (rule neurons)
$y_j^O(x)$	:	output of the output neuron j
w_{ij}	:	weights between rule neuron i (second hidden layer) and output neuron j
k_i	:	degree of support of each rule i

The neuro–fuzzy network is trained by a gradient based learning algorithm in a supervised mode. During the networks training four parameters are optimized:

- the parameters of the input fuzzy sets (σ, μ)
- the parameter k_i of each production rule, expressing the *importance* of each individual rule
- the parameter μ of the output fuzzy sets, represented by singletons

The formulae for the training algorithm can be found in [7].

2.1 The equivalence of the modified RBF–Network to Fuzzy–Systems

Since Fuzzy–Systems and the modified RBF–Networks are equivalent, it is possible to transform them into each other. This transformation is given by:

- the input variables correspond to the neurons of the input layer
- each fuzzy set (membership function) is represented by a neuron of the first hidden layer
- the parameters of the membership functions are stored in the weights between the input layer and the first hidden layer (the “membership function layer”).
- the weights of the binary matrix between the first and the second hidden layer indicate whether the premisse (first hidden layer) is part of the rule (second hidden layer) or not
- each fuzzy–rule is equivalent to a neuron of the second hidden layer
- the *degree of support* of each fuzzy rule corresponds to the parameter k_i of the neurons in the second hidden layer
- the parameters of the output fuzzy sets (singletons) are stored in the weights between the second hidden layer and the output layer
- the output variables are represented by the neurons of the output layer

3 Providing the network with input data

The navigation system of the robot *MILVA* is provided with sensory data from the laser scanner, the target angle and the target distance. The teacher input (steering angle) is provided by joystick.

3.1 The Laser Scanner

The laser scanner scans the environment in a horizontal plane within 180° with a resolution of 361 distance values. This scan vector is clustered into 5 major parts, indicating the distance to obstacles on the *right*, *few_right*, *straight_ahead*, *few_left*, and *left*. These input values represent linguistic variables with two terms each.

3.2 The Target Angle and Target Distance

A major task of the robot is to follow targets that are presented in front of the cameras. The cameras mounted on the Pan-Unit detect a target with a characteristic color using filter algorithms. Both cameras are tilted towards the target. The target angle and target distance can easily be calculated by triangulation. The "target angle" is used as an input value with five terms. The "target distance" is considered to be a linguistic variable with two terms.

3.3 The Tractory

Considering the tractory, the input data from the laser scanner is used to compute the minimum distances to obstacles on the long sides of the robot. The tractory can be described by the *Tractrix Algorithm*. This calculation needs to be done since there are no distance sensors on the long sides of the robot. The robots odometry data is supplied to the navigation modul. The measured distances (x–y coordinates) on the left and right sides of the robot are stored in two lists. The minimum distances to obstacles on the left and right side of the robot can then be computed by processing these list elements through a rotation and a translation matrix. As soon as the robot has passed the obstacles, the list elements are deleted (depending on the length of the robot and the odometry data). The minimum distances from the robot to obstacles on both sides are represented by two linguistic variables *lmin* and *rmin* with two terms each (*near* and *far*).

4 The Rule System

The final rule system of the navigation module consists of 38 rules based on heuristic knowledge. This rule base includes 9 rules for speed control, 29 rules for obstacle avoidance, target tracking, and the tractory.

Rules for Speed Control Speed control is necessary to adapt the current driving speed of the robot according to the present situation. If there are no obstacles around the robot it may travel at full speed. While approaching obstacles in any direction, the robot should slow down. Finally, a rule is required that causes the robot to stop at its destination.

Rules for Obstacle Avoidance If an obstacle is ahead the robot should swerve. The rule system selects the best alternative direction. If there is no obstacle around, the robot may go straight to the desired target.

Rules for Wall Following These rules describe the behavior while approaching a wall on the left or right side, with target also on the left (or right) side. The robot has to drive along the wall without causing a collision. Close to the wall, the robot has to steer away from the wall. If the distance between the robot and the wall is large, the robot will return to its original driving direction. These rules are evaluated parallel, resulting in a wall following behavior without causing a collision.

Rules for the Tractory Finally there are some rules to consider the tractory. They are essential for the robot for turns in narrow areas or lanes. If there is an obstacle close to the sides of the robot turns are not allowed.

5 Network training

5.1 Training data

The initial network is optimized by the training algorithm. The training data should reflect a large number of different situations so that our optimized system can adapt to as many situations as possible. The training data was taken from the situations listed in figure 3.

5.2 The network optimization

The neuro–fuzzy network is trained by a gradient based learning algorithm in the supervised mode. The teacher input (steering angle) was provided by joystick. During the network training the parameters of the input fuzzy sets (σ, μ), the parameter k_i of each production rule, and the parameter μ of the output fuzzy sets were optimized. For each of the 4 parameters an individual learning rate was implemented to prevent instability during the training process. Each training pattern contained the inputs from the laser scanner, target distance, target angle, tractory, steering angle and speed. The data set for network optimisation was split into a training data set and a testing data set consisting approximately 400 pattern each. The network training consisted of 500 training cycles for all types of parameter.

6 Results

The results of the network training show that the performance of the optimized network strongly depends on the quality of the training data. A particular situation not included in the training data will not be performed well. The reason is, that the parameter k_i indicating the importance of those rules is getting adapted to zero. These rules influence the optimized network only in a slight way. A pruning of rules with a low importance was not done in order to keep some redundancy.
The result of the network training improved the robots navigation behaviour. After the training the robot was able to navigate in a narrower area than prior optimization. Another result was the

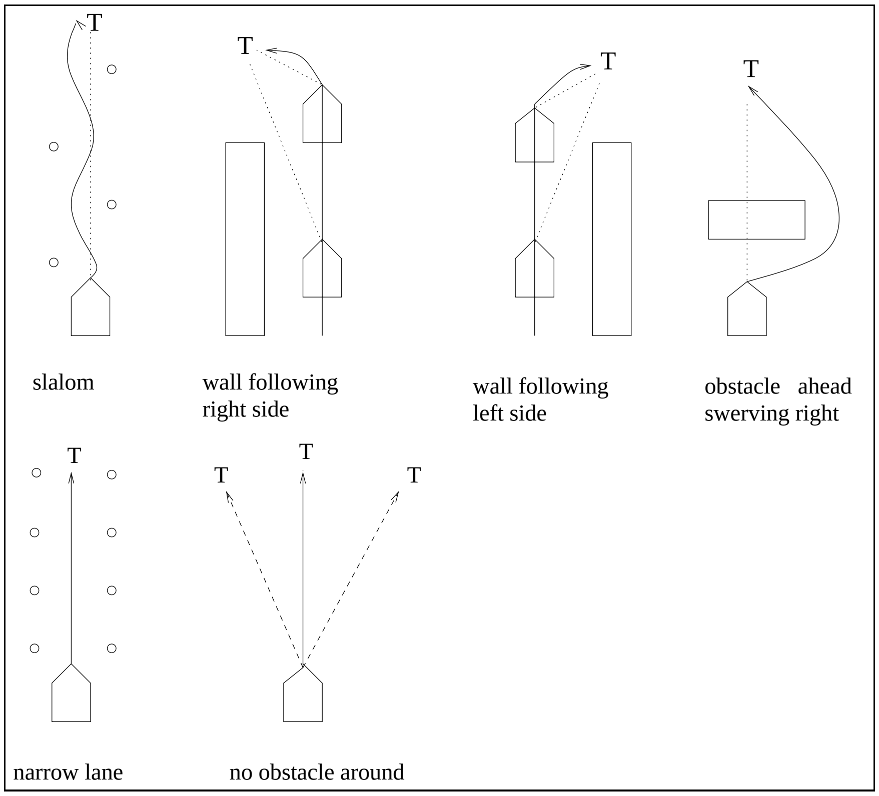

Figure 3: *The training includes the following situations: slalom, obstacle in the front, left or right, narrow lane, and an empty environment. The letter "T" markes the target.*

reliability of avoiding obstacles. After optimization the robot did no longer cause any collisions with obstacles on its sides when making narrow turns.

7 Conclusion

Using a neuro–fuzzy network for navigation tasks shows the efficiency of the combination of Fuzzy–Logic and Neural Networks. Fuzzy–Logic provides the system with initial knowledge consisting of input and output fuzzy sets and a rule base containing *IF..THEN..* production rules. On the other hand Neural Networks allow optimization of the network parameters by a backpropagation learning algorithm. This feature is used to optimize the parameters of the input and output fuzzy sets, the degree of support (importance) of each production rule, and the conclusions of the rules represented by output singletons. All these features allow the design of an adaptable navigation system. Such a system can be optimized to different geometric sizes of robots and tractories. Therefore the navigation module can be transferred to another robot with the same rules.

8 Acknowledgements

We thank Rolf Nestler and Volker Stephan for their good proposals and cooperative work concerning the hardware of the robot, the software basis and some helpful advice for the used algorithms.

references

[1] M. Krabbes, H.-J. Boehme, V. Stephan, H.-M. Gross: *Extention of the ALVINN-Architecture for Robust Visual Guidance of a Miniature Robot* In: EUROBOT'97 -2nd EUROMICRO Workshop on Advanced Mobile Robots, Brescia, Oktober 1997.

[2] D.A. Pomerbleau: *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Boston/Dortrecht/London, 1993.

[3] M. Krabbes, H.-J. Boehme, V. Stephan, H.-M. Gross: *Handlungsorganisation intentionaler neuronaler Agenten*.In: Fortschrittsberichte VDI, Reihe 8, Nr. 663, VDI Verlag GmbH 1997.

[4] Erler A.: *Realisierung einer Hindernisvermeidung und lokaler Navigationsaufgaben mittels Fuzzy-Algorithmen für den Roboter MILVA*. TU Ilmenau, Fachgebiet Neuroinformatik, Studienarbeit, 1998.

[5] Kühn T., Wernstedt J.: *Entwurf einer Fuzzy-Steuerung für den mobilen Miniroboter Khepera*. Praktikum Fuzzy und Neuro Control, Fachgebiet Systemanalyse, TU Ilmenau, 1997.

[6] James C. Bezdek: *Analysis of fuzzy Information*. Boca Raton, Fla, CRC PR., 1997.

[7] V. Tresp, J.Hollatz, S.Ahmad: *Network structuring and training using rule based knowledge*. Advances in Neural Information Processing Systems 5, 1993.

[8] J.Moody, S.Darken: *Fast learning in Networks for locally tuned processing units*. Neural Computation, Edition 1.

[9] Nauck D., Klawonn F., Kruse R.: *Neuronale Netze und Fuzzy-Systeme*. Computational Intelligence, 2.Auflage Vieweg-Verlag, 1996.

[10] A. Zell: *Simulation Neuronaler Netze*. Addison-Wesley, 1994

[11] H.-H.Bothe: *Neuro–Fuzzy–Methoden*. Springer–Verlag, 1998.