FACTA UNIVERSITATIS
Series: **Mechanics, Automatic Control and Robotics** Vol. 6, Special Issue, 2007, pp. 123 - 132

NEURO-FUZZY CONTROLERS AND APPLICATION TO AUTONOMUS ROBOTS

UDC 007.52:004.832.28(045)=111

Pavle Boskoski, Mile Stankovski

SS Cyril & Methodius University, Faculty of Electrical Eng. & Info. Technologies,
Karpos II, 1000 Skopje, Republic of Macedonia
e-mail: milestk @feit.ukim.edu.mk

Abstract. *This work presents an analysis of the neuro-fuzzy controllers and algorithms for their generation. The generated neuro-fuzzy controllers are then used for optimal control of autonomous robot (Khepera) in an unknown environment, which is different from the one used for the training of the controller. The combination of neural networks with fuzzy logic offers the opportunity for resolving the difficulties of the proper generation of fuzzy controller. This works covers some of the algorithms for generation of fuzzy controllers with neural networks.*

Key words: *Autonomous robots, fuzzy logic, fuzzy controllers, neural networks, back-propagation*

1. INTRODUCTION

In order to achieve a correct behavior the control system of an autonomous robot must perform many complex information processing tasks in real time. The robots operate in environment where the boundary conditions vary rapidly (1). The fuzzy logic controller is well suited for controlling a robot because it is capable of making inferences even under uncertainty (2). Ever since the fuzzy systems were applied in industrial applications, developers know that the construction of a well performing fuzzy system is not always easy. The problem of finding appropriate membership functions and fuzzy rules are often a tiring process of trial and error.

The learning parameters of neural networks made them a prime target for a combination with fuzzy systems in order to automate or support the process of developing a fuzzy system for a given task (3).

Learning allows autonomous robots to acquire knowledge by interacting with the environment. This kind of behavior learning methods can be used to solve control problems that robots encounter in unfamiliar real-world environment (4).

This paper discusses an experimental neuro-fuzzy controller for sensor based mobile robot navigation in indoor environments. Our work is a succession of previous work done

Received November 15, 2007

in the Institute for Automation and System Engineering at the Faculty of electrical engineering and Information technologies (5).

2. Neuro-fuzzy controller

2.1. Omnidirectional robot

The omnidirectional robot used in our experiments is Khepera II robot. The robot has 8 IR sensors spread on the edge of the robot. These sensors are used for determining the distance between the robot and the surrounding obstacles. d on the edge of the robot. These sensors are used for determining the distance between the robot and the surrounding obstacles. In order to decrease the input space only 6 of the 8 sensors have been used (two rear sensors are not used). Additionally the sensors covering the front i.e. sensors S2 and S3 are grouped with the following relation:

$$S_{front} = \frac{S_2 + S_3}{2} \tag{1}$$

The robot is moved using two motors mounted on each wheel. Each motor can be controlled independently. On each motor there is a linear encoder that generates pulses with the rotation of the wheel.

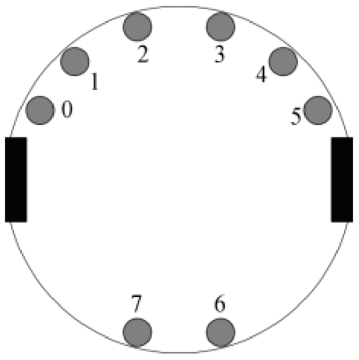


Fig. 1 Mobile robot and sensor arrangement

With this configuration we obtain a system that has 5 inputs and 2 outputs.

2.2. Neuro-Fuzzy controller

A number of results have been presented to demonstrate the equivalence of the fuzzy rule based system and neural networks (6). Additionally it has been shown that a functional equivalence exists between a radial basis function network and fuzzy systems (7). Among the most popular neuro-fuzzy systems can be cited: NEFCON (8), ANFIS (9). In this paper we have used the ANFIS model.

The ANFIS stands for Adaptive Network-based Fuzzy Inference System. The ANFIS model generates Takagi-Sugeno fuzzy model. The neural network used to generate the fuzzy controller has the architecture shown on the Figure 2.

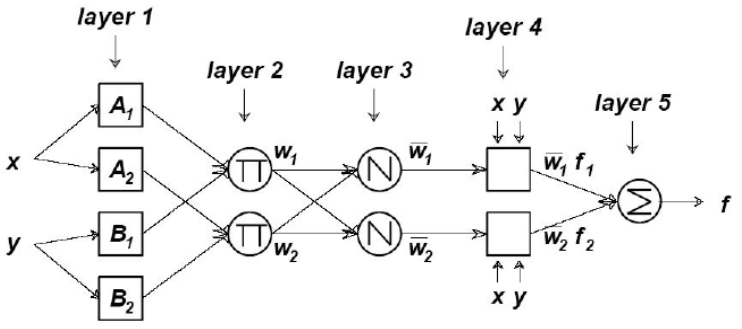


Fig. 2 ANFIS Architecture

The depicted model defines a controller with two inputs and one output. Each input has two membership functions. The adaptive nodes of the neural network are nodes in the layers 1 and 4.

The output of the nodes in layer 1 are the membership values of the premise part i.e.:

$$\begin{aligned} O_{1,i} &= \mu_{A_i}(x),\ for\ i = 1,2\ or \\ O_{1,i} &= \mu_{B_{i-2}}(y),\ for\ i = 3,4 \end{aligned} \tag{2}$$

Every node in layer 2 is a fixed node labeled Π, which multiplies the incoming signals:

$$O_{2,i} = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), i = 1,2 \tag{3}$$

Every node in layer 3 is fixed node labeled *N*. It calculates the ration of the *i*-th rule's firing strength to the sum of all rules firing strengths:

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2},\ i = 1,2 \tag{4}$$

Every node in layer 4 is an adaptive node with the node function

$$O_{3,i} = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i x + r_i) \tag{5}$$

where $\overline{w}_i$ is the output of layer 3 and p_i, q_i, r_i is the parameter set for the first order Sugeno rule.

The overall output of the network can be defined as:

$$O = overalloutput = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{6}$$

3. THE SIMULATION AND RESULTS

We have used the ANFIS toolbox, and Neural Network toolbox from MATLAB for the generation and training of the neuro-fuzzy controller. As a test environment, and environment for gathering the training set we have used the WSU Khepera Simulator (10). Our first task was to create an environment where we can make fast development and testing.Wrapping the MATLAB workspace environment with JMatLink, in conjunction with Jini (Java native interface) allows modules and services implemented as native interpreted MATLAB code to be accessed as remote and distributed objects and to be directly incorporated into behavioral architectures, resulting in the acceleration of the development of an added flexibility of implementation (11).

Using this approach we used the controllers generated by ANFIS and NN toolboxes in MATLAB directly into the WSU Khepera simulator. On the other hand the WSU Khepera simulator was used to gather the training data set i.e. the Braitenberg vehicle implementation, and afterward the same environment was used to test the generated controllers.

3.1. Gathering the training data

In order to train the neuro-fuzzy controller the training data set should be gathered. In general case this can be done by recording the actions of a human expert. In the case of wall following behavior the "teacher" (human expert) must generate the same commands to the robot in the same behavioral situations (12). In order to achieve this task, we have used Braitenberg vehicle (13) as a teacher.

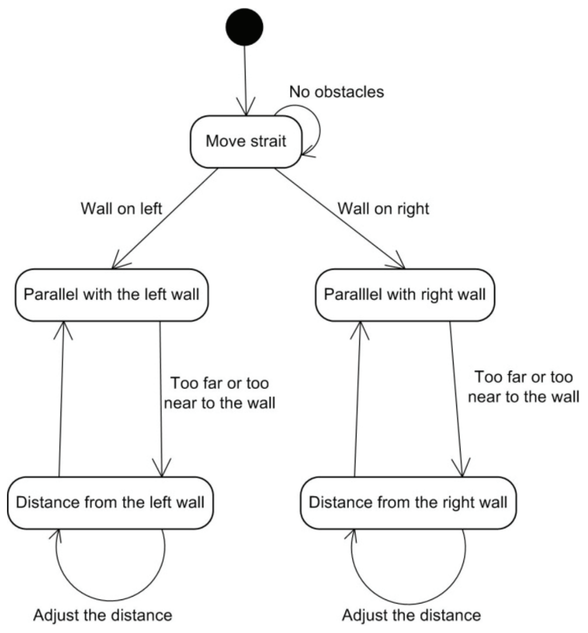

No obstacles
Move strait
Wall on left
Wall on right
Parallel with the left wall
Paralllel with right wall
Too far or too near to the wall
Too far or too near to the wall
Distance from the left wall
Distance from the right wall
Adjust the distance
Adjust the distance

Fig. 3 States in the Braitenber vehicle controller

A simple 4 state controller has been implemented in order to controll the Braitenberg vehicle for the wall following task. As shown on Figure 3 the controller for the braitenberg vehicle is quite simple. It moves straight until an obstacle is spotted. Depending whether the first obstacle is from the left or from the right side the robot will start its wall following on the appropriate side. The explained behavior is simple but it can be used to show the algorithm for generation of the neurofuzzy controllers.

The value of the front six sensors and the values of the two motors have been recorded as a training set. The training set contains 5100 points. The training set has been normalized, before presenting it to the neural network (14). The behavior of the Braitenberg vehicle in the training environment is presented in the Figure 4.

3.2. Generating the Neuro-Fuzzy controller

For the wall following task two variations of fuzzy controllers have been generated.

- fuzzy controller with two membership functions per input, and
- fuzzy controller with three membership functions per input

Both fuzzy controllers are of Takagi-Sugeno zero order type. This choice has been done in order to decrease the number of parameters that need to be trained.

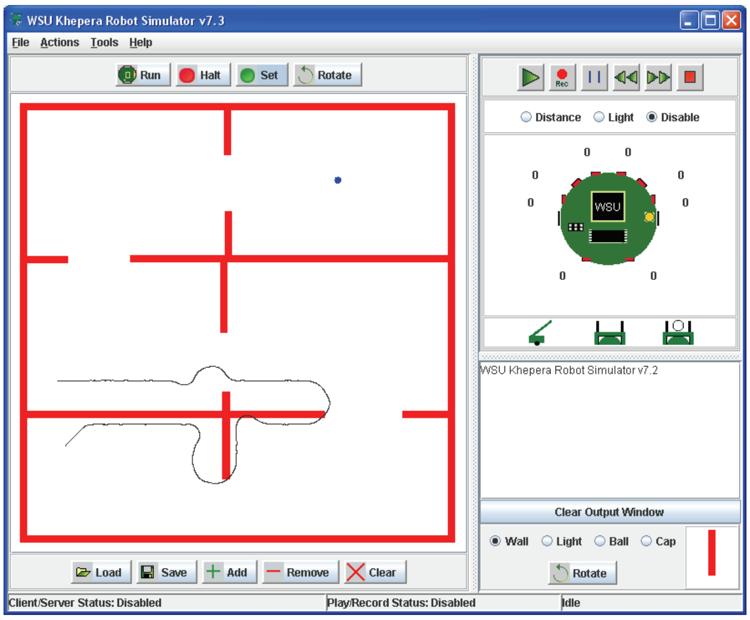

WSU Khepera Robot Simulator v7.3
File Actions Tools Help
Run Halt Set Rotate
Rec
Distance Light Disable
0 0 0 0 0 0 0 0
WSU
WSU Khepera Robot Simulator v7.2
Clear Output Window
Wall Light Ball Cap
Rotate
Load Save Add Remove Clear
Client/Server Status: Disabled
Play/Record Status: Disabled
Idle

Fig. 4 Track of the Braitenberg vehicle

The general form of fuzzy rule generated by the neuro-fuzzy controller (zero order Takagi Sugeno) has the following form:

$$\begin{aligned}&IF\ x\ is\ A_i\ AND\ y\ is\ B_i\ THEN\ lm_k = c_1,\ rm_k = c_2\\ &for\ i = 1,\dots,L;\ j = 1,\dots,M;\\ &k = 1,\dots,N;\ N = L \times M\end{aligned} \tag{7}$$

where x and y are the linguistic variables, lm and rm are left motor and right motor output respectively for the k-th rule, L is the size of the fuzzy set A, M is the size of the fuzzy set B, and N is the size of the rule base.

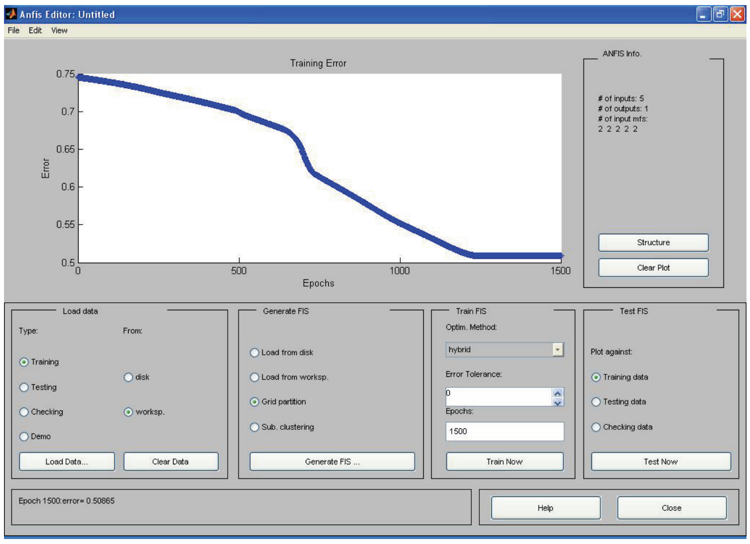
From the general rule (7) one can notice that each rule has two outputs. The approach to solve this MIMO system is to divide the rule base into two independent fuzzy controllers with 5 inputs and one output.

The controllers have been trained using hybrid training algorithm (15). The input spaced has been partitioned using grid partitioning.

3.2.1. Controller No 1

The first controller has two membership functions per input. The membership functions are triangular and initially uniformly spread across the universe of discourse. The initial coefficients of the consequent part of the rules are initiated to 0 (zero). According to the (7) the controller No. 1 has 5 inputs with A = {*near, far*} the number of rules is 32 ($2^5 = 32$).d across the universe of discourse. The initial coefficients of the consequent part of the rules are initiated to 0 (zero). According to the (7) the controller No. 1 has 5 inputs with A = {near, far} the number of rules is 32 ($2^5 = 32$).

The training ends after 1200 epochs. The training process has adjusted 32 linear parameters and 30 nonlinear parameters (adjusted by backpropagation). The training process is shown on Figure 5.

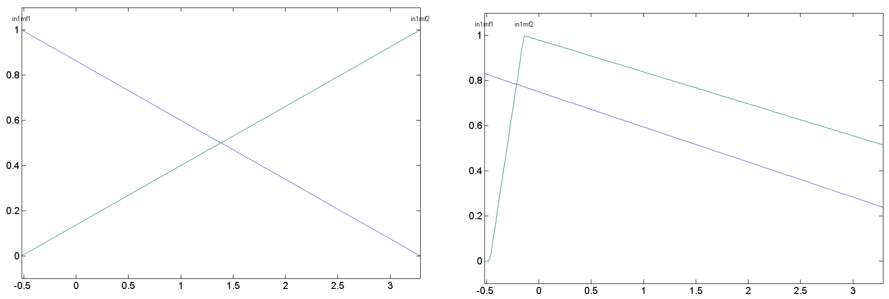


ANFIS Info.
# of inputs: 5
# of outputs: 1
# of input mfs:
2 2 2 2 2

Load data — Type: [✓ Training] Testing Checking Demo; From: disk [✓ worksp.]
Generate FIS — Load from disk; Load from worksp.; [✓ Grid partition]; Sub. clustering
Train FIS — Optim. Method: hybrid; Error Tolerance: 0; Epochs: 1500
Test FIS — Plot against: [✓ Training data] Testing data Checking data

Epoch 1500:error= 0.50865

Fig. 5 Training epochs for Controller No. 1

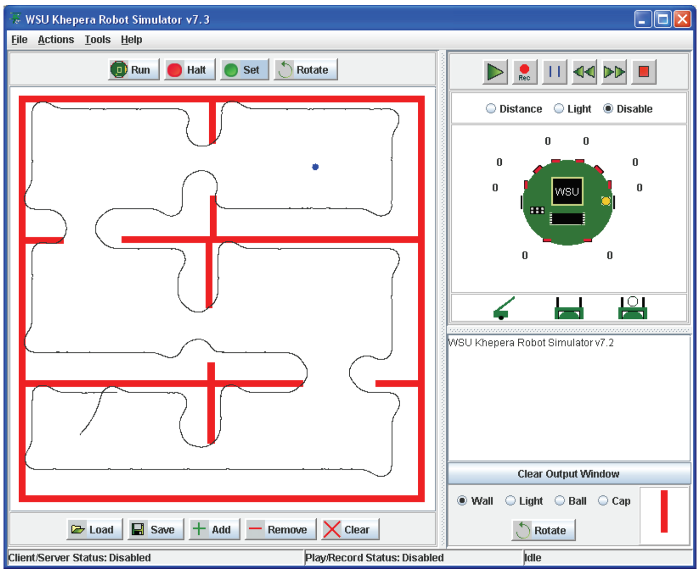


(a) Before the training (b) After the training

Fig 6. Membership functions for the input front sensor

Figure 6 shows the resulting membership functions for the input covering the front sensor. It may be interesting to note that, although the training started with symmetric membership functions, their shape eventually became asymmetric after the training.

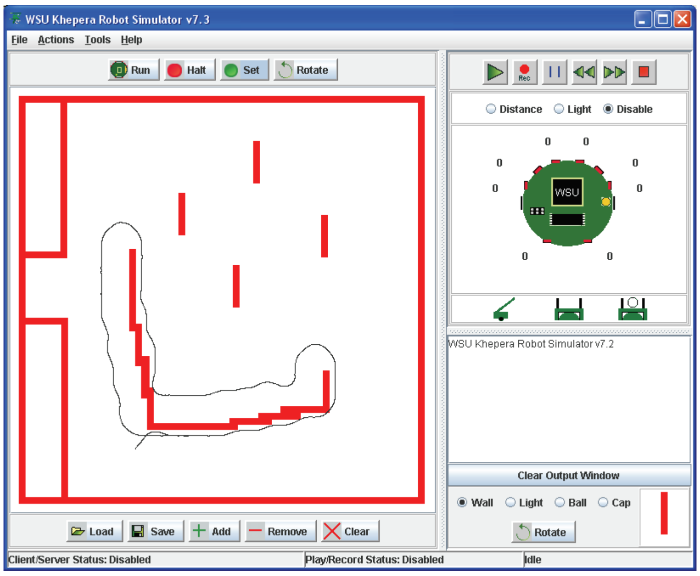
After the training the robot was tested in the same environment as the training and in the different environment. The Figure 7(a) shows the robot behavior in the environment the same as the training environment, and the Figure 7(b) shows the robot behavior in the environment different from the training one.



WSU Khepera Robot Simulator v7.2

Client/Server Status: Disabled | Play/Record Status: Disabled | Idle

(a) Robot behavior in the test environment



WSU Khepera Robot Simulator v7.2

Client/Server Status: Disabled | Play/Record Status: Disabled | Idle

(b) Robot behavior in an unknown environment

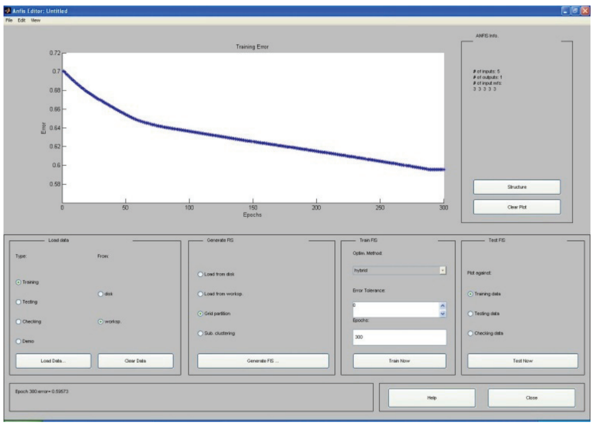
Fig. 7. Robot behavior with Controller No. 1

3.2.2. Controller No 2

The controller No. 2 has three membership functions per input. The same as the controller No. 1 the membership functions are triangular and uniformly spread through the universe of discourse. The consequent parts of the rule base are set to 0 (zero). The same training set is used to train the controller No.2.

According to the (7) the controller No. 2 has 5 inputs for distance with A = {*near, medium, far*} the number of rules is 243 ($3^5 = 243$).

In this case the training process took 270 epochs. The training process has adjusted 243 linear parameters and 45 nonlinear parameters (adjusted by backpropagation). The Figure 8 shows the error during the training process for this controller.



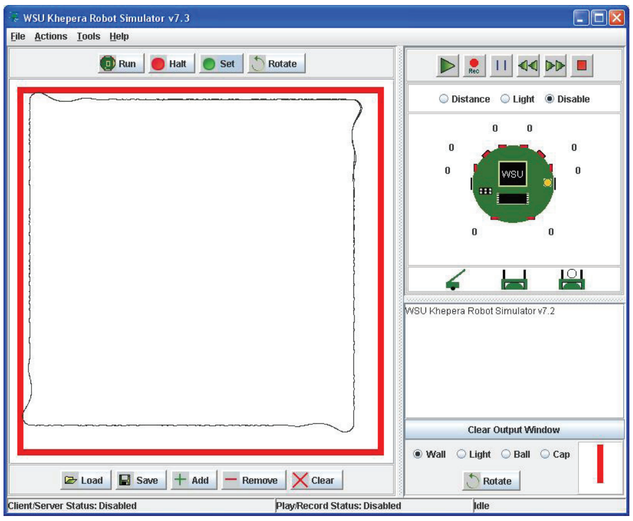
# of inputs: 5
# of outputs: 1
# of input mfs:
3 3 3 3 3

Optim. Method: hybrid
Error Tolerance: 0
Epochs: 300

Epoch 300:error= 0.59573

Fig. 8. Training epoch for controller No. 2

When we observed the robot behavior in an empty environment, as shown on Figure 9, we have spotted an overfitting behavior of the network. After the robot exits a curve small oscillations can be observed.



WSU Khepera Robot Simulator v7.2

Client/Server Status: Disabled
Play/Record Status: Disabled
Idle

Fig. 9. Robot behavior with controller No. 2

4. CONCLUSION

The conducted experiments show the chosen algorithm for the generation of fuzzy controllers can be used for controll of autonomous robots in unknown environments. The learning procedure allows the robot to generalize the knowledge condensed in the training set.

The number of the membership functions per input directly influences the number of the parameters that need to be adjusted, and so it increases the time needed for training the controller. One additional point that has to be taken into consideration is the uniform distribution of the special cases within the training set. With inadequate training set the robot may show unwanted behavior in some situations that were not taken in consideration during the training process.

The environment that we have used for the training and testing of the robot can be easily modified to allow usage of the real robots connected either via serial link or via wireless communication.

REFERENCES

1. *A robust layered control system for a mobile robot.* Brooks, R. 1986, IEEE Journal of Robotics and Automation, pp. 14-23.
2. *The uses of fuzzy logic in autonomous robot navigation.* Saffiotti, A. 1997, Soft Computing, Vol. 1, pp. 180-197.
3. *Neuro-fuzzy systems: Review and prospects.* Nauck, D. 1997. Fifth European Congress on Intelligent Techniques and Soft Computing(EUFIT'97). pp. 1044–1053.
4. *Behavior-based neuro-fuzzy controller for mobile robot navigation.* Rusu, Petru, Emil M. Petriu, Thom E. Whalen, Aurel Cornell and Hans J. W. Spoelder. 2003, IEEE Transactionon Instrumentation nad Measurment, Vol. 52, pp. 1335–1340.
5. *Mobile robot path tracking usingfuzzy logic.* Stankovski, M., T. Kolemishevska-Gugulovska, D. Stankovski, P. Boshkoski and B. Mileva. 2004. World Automation Congress.
6. *Evolutionary neuro-fuzzy systems and applications.* Castellano, G., C. Castiello, A. M. Fanelli and L. Jain. IEEE Journal of Robitocs and automation.
7. *A robust layered control system for a mobile robot.* Jang, J.-S. R. and C.-T. Sun. 1993, IEEE Trans. on Neural Networks, pp. 156-159.
8. *A neurofuzzy controller learning by fuzzy error propagation.* Nauck, Detelf and Rudolf Kruse. 1992. Proc. of Conf. of the North Amer. Fuzzy Inf. Proc. Soc. pp. 388–398.
9. *Adaptive-networkbased fuzzy inference systems.* Jang, J.-S. R. 1993, IEEE Trans. on Systems, Man, and Cybernetics, Vol. 23, pp. 665-685.
10. Wsu ksuite 1.1. *WSU, Wright State University.* [Online] 2005. http://carl.cs.wright.edu/reg/ksim/downloads.
11. *A case study of fuzzy-logicbasedrobot navigation.* Valavanis, K.P., L. Doitsidis, M. Long and R.R. Murphy. 2004, Robotics & Automation Magazine, IEEE, Vol. 13, pp. 93–107.
12. *Neurofuzzy control of a mobile robot.* Godjevac, Jelena and Nigel Steele. 1999, Neurocomputing, Vol. 28, pp. 127-143.
13. Braitenberg, Valentino. *Vehicles: Experiments in Synthetic Psychology.* s.l. : The MIT Press., 1986.
14. Bishop, Chirstopher M. *Neural Networks for pattern recognition.* s.l. : Oxford university press, 1995.
15. Jang, J.S. Roger, Chuen-Tsai Sun and Eiji Mizutani. *Neuro-Fuzzy and Soft Computing, A Computational Approach to Learning and Machine Intelligence.* s.l. : Prentice Hall, 1997.

NEURO-FAZI UPRAVLJACI I NJIHOVA PRIMENA NA AUTONOMNE ROBOTE

Pavle Boskoski, Mile Stankovski

U ovom radu je data analiza neuro-fazi upravljanja i algoritama za njihovo generisanje. Generativni neuro-fazi upravljači se onda koriste za optimalno upravljanje autonomnih robota (Khepera) u nepoznatoj sredini koja se razlikuje od one korišćene za obuku upravljača. Kombinacija neuronskih mreža sa fazi logikom - nudi mogućnost razrešavanja problema odgovarajućeg proširivanja u praktičnu realizaciju fazi upravljača. Ovaj rad pokriva neke algoritme za praktičnu realizaciju fazi-upravljača sa neuronskim mrežama.

Ključne reči: *fazi logika ,fazi kontrolori, neuronske mreže, realizaciju sa povratnom spregom*