Piotr Skrzypczyński, PhD
Technical University of Poznań, Institute of Control and Information Engineering

EXPERIMENTS AND RESULTS IN MOBILE ROBOT NAVIGATION WITH A FUZZY-GENETIC CONTROLLER

Abstract : In this article, a behaviour-based control system for mobile robot navigation is presented. It uses fuzzy rules to map current sensor readings onto robot actions. A genetic classifier system is used to learn rules. The fuzzy rules are evolved in a computer simulation, and then used by controllers implemented on real mobile robots. Results of experiments are provided, showing that the proposed method can find rules for mobile robot navigation.

1. INTRODUCTION

Behaviour-based architectures are widely used to control autonomous mobile robots [1]. They involve decomposition of the whole navigation system into a number of simple units, called behaviours. Each behaviour produces actions aimed at achieving a particular goal, such as obstacle avoidance and goal seeking. One of the frameworks used for practical implementation of individual behaviour units is the fuzzy logic. Fuzzy controllers are a convenient choice, because they are simple, fast, and they do not require an analytical model of the system to be controlled [9]. However, in the case of a mobile robot, the manual design of a controller can be quite complicated, because of the high number of rules needed to define an unambiguous response for all possible combinations of the input stimuli. Because of that difficulty, numerous approaches to learning and adaptation in fuzzy controllers have emerged, mostly focusing on the reinforcement learning for tuning of the fuzzy membership functions [3], or learning the fuzzy rules themselves [8], and on artificial neural networks [7]. An alternative approach to learning in robotic systems is the use of genetic algorithms (GA). Learning and tuning of mobile robot's behaviours by means of GA has been proposed for crisp [10] and fuzzy-type [6] rules.
This article describes an approach to design the core part of mobile robot's navigation system – the reactive controller providing the obstacle avoidance behaviours, thus responsible for the safety of the vehicle. The controller uses fuzzy `if-then` rules to navigate from a given start position to a given goal position, while avoiding collisions with obstacles. The start and goal points are provided by a human operator or a path planner, being a higher-level module of the navigation system. The proposed controller is evolved in a realistic simulation, by using a genetic-based machine learning method, the classifier system with genetic algorithm.

2. FUZZY LOGIC CONTROLLER OF THE MOBILE ROBOT

To control the mobile robot a classical (Mamdani type) fuzzy controller [4] has been introduced. It consists of:

- a fuzzifier, which maps sensor readings onto fuzzy sets,
- a rule base, made of the fuzzy `if-then` rules,
- an inference module, which uses rules and membership functions to map input fuzzy sets onto output fuzzy sets,
- a defuzzifier, which maps fuzzy output sets onto crisp output values.

The controller must coordinate the execution of behaviours aimed at achievement of two possibly conflicting objectives: getting to the goal, and stay away from obstacles. For both of these objectives separate rule bases have been defined. Because of that, the classical fuzzy controller has been extended by adding a behaviour arbitration mechanism. The role of this selector is to choose the rule base appropriate for the current situation of the robot. There are three rule bases: one with rules for getting to the goal (**BG**), and two for obstacle avoidance – one for left side wall following (**BL**), and another one for right side wall following (**BR**). Two alternative selection mechanisms have been implemented. In the first selector, during the robot motion its distance to the goal is memorized. If the robot is getting to the goal, this distance decreases, but when the robot is maneuvering in the presence of obstacles, the distance to the goal usually increases. The selector recognizes this moment and activates one of the obstacle avoidance rule bases. Which one is activated (**BR** or **BL**), depends on the current bearing to the goal. The active base is switched back to **BG**, when the distance to the goal becomes smaller than the last memorized value. The second type of selector analyses only the bearing to the goal. If the bearing is greater than 145^o, the selector switches to obstacle avoidance. When the bearing becomes smaller than 45^o, it switches back to the goal seeking behaviour.

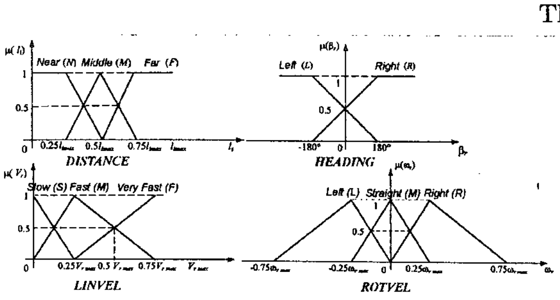

Figure 1: Definitions of the linguistic variables

The vector $\mathbf{X} = (\beta_r, l_1, l_2, \ldots l_n)$, which contains information from the sensors of the robot, i.e. the distances to obstacles l_i given by n sensors and the angle β_r giving the orientation of the goal relative to the robot's heading, is the input of the controller. The distance readings are provided by ultrasonic range finders (sonars), a low-cost sensor type ubiquitous on mobile robots. Also short-range, infra-red (IR) proximity sensors are

used. It is rather hard to use directly all the individual sensor readings as inputs to the fuzzy controller. Thus, the sensors are grouped to reduce the number of inputs, and the number of rules in the controller [11]. All the sensors within a group constitute single input of the controller. For sonars of a given group the shortest measured distance is picked up. This value is then verified by the readings of the IR proximity sensors: if the proximity sensors find an obstacle closer than the value given by sonars, the IR reading is used as the controller input. The IR reading is also used when there is no data from sonars, e.g. due to specular reflections.
For the fuzzification of the crisp values of the input vector $\mathbf{X}$, the controller uses linguistic variables given by the membership functions of γ, L, and t type. The following input linguistic variables have been defined (Fig. 1):

Distance – used for the fuzzification of distances from all groups of sensors. One of two possible definitions can be used alternatively – with two and three fuzzy sets. The membership functions may be adjusted according to the current type of surroundings. If for some given period of time the distance readings from sensor groups in the left and right sides of the robot are shorter than half of the sensor range, then the value l_{max} is set to $l_{max} = \frac{l_{max}}{2}$. This scaling enables more precise steering of the robot in narrow corridors [12].

Heading – used for fuzzification of the bearing to the goal.

The output linguistic variables have been defined as: *LinVel* – linear velocity v_r, and *RotVel* – angular velocity ω_r of the robot. These values are then converted into velocities of the driven wheels. Mapping of the fuzzy sets onto crisp values of the output vector $\mathbf{Y} = (v_r, \omega_r)$ is performed by using the center of gravity method.

3. GENETIC CLASSIFIER SYSTEM

In the proposed mobile robot controller a classifier system with genetic algorithm [5] has been used to generate fuzzy `if-then` rules. Fuzzy logic rules are represented by classifiers. The linguistic variables and membership functions defined in the system do not undergo adaptation. The classifiers are production rules, which take the form of fixed-length strings, encoded with binary values.
The external stimuli transmitted from sensors are also encoded as binary strings. The classifier consists of an antecedent having the same form as the encoded external stimuli (perception), and a message in which information about the consequent part of the fuzzy rule is encoded. If the antecedent of a classifier matches the current perception, this classifier sends it's message to the rule base. The message is divided into two parts. The first part indicates the number of the fuzzy set of the linguistic variable *LinVel*, the second part is the number of the fuzzy set of the variable *RotVel*. The numbers of fuzzy sets are linearly encoded. The whole classifier is a fuzzy rule of the controller, e.g. for a robot with three groups of sensors a classifier can be like this:

$$|0|0|1|0|0|0|1|1|1|0|0|$$

The fuzzy rule resulting from the above classifier has the following form (in square brackets the numbers of fuzzy sets are shown):

$$\begin{aligned}&\textbf{if } (\mathit{Heading} \textbf{ is } \mathit{Left}\ [0] \textbf{ and } \mathit{Distance} \textbf{ is } \mathit{Far}\ [0] \textbf{ and}\\ &\quad \mathit{Distance} \textbf{ is } \mathit{Near}\ [1] \textbf{ and } \mathit{Distance} \textbf{ is } \mathit{Far}\ [0])\\ &\textbf{then } (\mathit{LinVel} \textbf{ is } \mathit{Slow}\ [0] \textbf{ and } \mathit{RotVel} \textbf{ is } \mathit{Right}\ [2])\end{aligned}$$

In a population, there can be several classifiers, with the antecedent matching the current stimuli. Since only one classifier can send it's message to the environment, all classifiers activated as the result of the given stimuli take part in an *auction*, which results in a selection of the winner. The classifiers issue *bids* B, proportional to their *strength* S. The bid value of i-th classifier is computed as:

$$B_i = f_{bid} S_i + N(0, \sigma),$$

where f_{bid} is the *investment factor*, $N(0, \sigma)$ is the Gaussian noise. The noise is needed, because several classifiers with the same S value can coexist in a population. The classifier with the highest bid becomes the winner and activates the actuator, by triggering the corresponding fuzzy rule in the controller. Then, the winning classifier pays the declared B value to the remaining active classifiers. During each cycle of the system activity all classifiers pay also a ***life tax*** which decrease their strength. This scheme facilitates elimination of non-productive classifiers, and enables existence of the subsets of similar classifiers. The utility of a classifier for the robot depends on how the rule generated from this classifier performs during the control step. Getting towards the goal yields a reward, while a behaviour which results in collisions receives no reward. The evaluator module assigns strength value to the classifier according to the reward received from the environment.
The role of the genetic algorithm is the creation of new classifiers (and thus new rules). The new classifiers are created in a process based on three elementary genetic operators: reproduction, crossover and mutation [5]. The probability of being selected as a "parent" is proportional to the strength of the classifier. The fittest classifiers from the new generation replace less fit individuals in the population. The genetic algorithm is triggered every n_{gen} steps of the classifier system activity – this parameter can be adjusted in the simulator. Moreover, the genetic algorithm is called whenever the robot collides with an obstacle, or when it is moving for some period of time, without a progress towards the goal (this is very likely a trap or a loop). In such a case, the genetic algorithm is invoked with a higher mutation probability, to push the robot out of an equilibrium.

4. LEARNING IN SIMULATION

The motivation to use a simulator for learning robot behaviours stems primarily from the fact, that the try-and-error learning on a real robot may be dangerous. Also, the number of simulated runs performed in the given amount of time can be much higher than the number of real experiments.

The simulation and learning system consists of two kinds of programs running in Win32 environment and co-operating through the TCP/IP network. The first program is the *environment server*. This program implements the simulation of the world, the robot vehicles, and sensors. The simulated environment is described by means of 2D vector primitives (polygons and lines). The server simulates such sensors as laser scanner, sonars, IR proximity, and tactile sensors (bumpers). The odometry of the robots is not simulated – the environment server provides the robot-clients with current position and orientation $\mathbf{X}_{rob} = (x, y, \theta)$, with random uncertainty $\mathbf{C}_{(x,y,\theta)}$. This is a simplified equivalent of GPS or a global vision system. The second program is the *robot client*. This program contains the navigation algorithm – in this case the fuzzy logic controller and the learning module. It is possible to configure individually the sensor suite on each robot (e.g the robot can have only sonars and IR sensors, scanner and IR, etc.). Also the parameters of the sensors can be configured individually. In each step of the simulation, the robot-clients receive data simulating their current sensor readings from the server, and they send to the server the current values of the linear and angular velocities, then used by the server to compute the next move of the simulated robot. The distributed simulator enables experiments with multi-agent scenarios, that involve several robots operating in the environment. There is no multi-robot learning possible (as so far), the robots learn individually, but the learned rule bases can be stored in files and then used in testing multi-robot simulations.

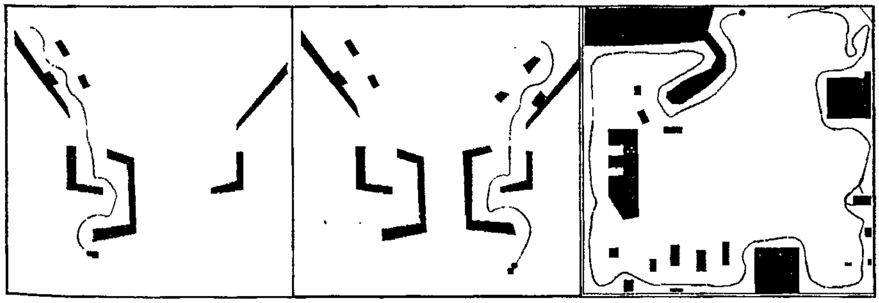

Figure 2: Example environments used for learning of fuzzy rules

The learning procedure is supervised by an operator. At start, he chooses one of the three rule bases of the controller (**BL**, **BR** or **BG**). This base is active while learning, and filled with rules produced by the classifier system. In the learning mode the behaviour selector is not active. Because of that, it is important to use training environments, which are adequate for the behaviour to be learned (Fig. 2). To start the learning, all possible preconditions of the fuzzy rules are generated, as well as an initial population of classifiers. The learning can be stopped by the operator to test the rule bases in the full simulation mode. Then the user can switch back to the learning mode or modify the fuzzy rules manually. It is also possible to start the learning from a set of manually generated rules as the initial

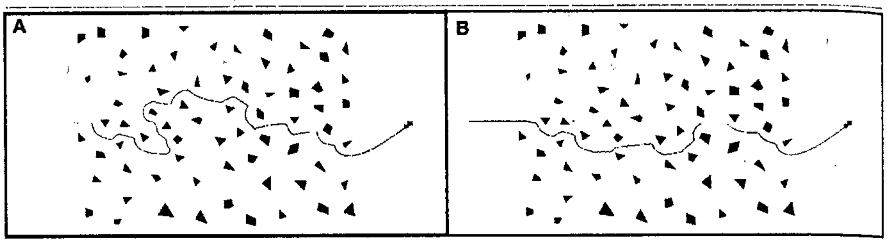

Figure 3: Results of using learned (A) and manually configured rules (B)

population.
The results of simulations show, that the performance of the robot's fuzzy controller with the rules generated by the classifier system is quite similar to the performance of the robot with manually defined rules (Fig. 3). In the case of environments with many obstacles, the path of the robot with learned rules is usually longer than the path of the manually configured robot. Also the time needed to complete the task is longer when the learned rules are used. This is because the evaluator module rewards the rules which keep the robot far from obstacles, and there is no reward directly proportional to the speed of the robot. Thus, in the learned rule bases more "cautious" rules are preferred.
Figure 4 shows screenshots from the environment server program during a multi-robot simulation with four robots trying to pass a crossing simultaneously. The robots detect each other with their range sensors, and treat as dynamic obstacles.

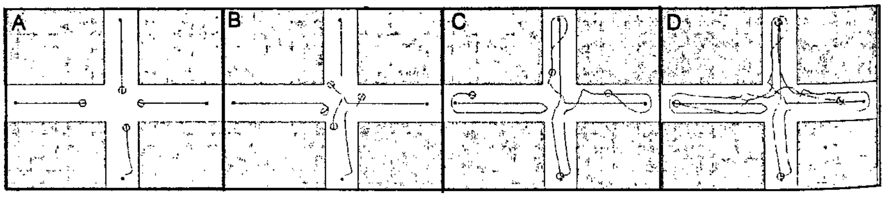

Figure 4: A multi-robot simulation – robots use learned fuzzy rules

5. REAL ROBOT IMPLEMENTATION AND RESULTS

To show the feasibility of the learned rules for real-world robot navigation, the fuzzy logic controller described in this article has been implemented on two mobile robots, *Labmate* and *Pioneer 2DX* (Fig. 5). The controller can use fuzzy rule bases learned in the simulator. Both robots are of differential drive type, but they have different sensing capabilities. The *Labmate* has a belt of 24 ultrasonic sensors and short-range IR proximity sensors. The *Pioneer 2DX* is equipped with 8 forward-looking sonars. The aim of the experiments was to:

- confirm the ability of the GA-based learning to generate rules useful for real robots,
- test how the performance of the fuzzy controller depends on the quality of the sensory data,
- compare the learned rule bases to rule bases created by a human expert, in a number of different environments.

Figure 6 is a sequence of images from the overhead camera mounted to the ceiling of the laboratory room, showing an obstacle avoidance experiment performed with the *Labmate* robot. The robot used fuzzy rules learned in simulation. It has been provided with position and orientation $X_{rob} = (x, y, \theta)$, $C_{(x,y,\theta)}$ by the overhead camera, acting as an external localization system [2]. In this experiment, the robot manoeuvred smoothly in narrow spaces between static obstacles, however it had to stop in several points to contact the vision system and obtain current position and orientation.

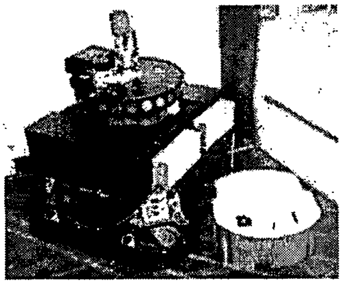
Figure 5: Mobile robots used in the experiments

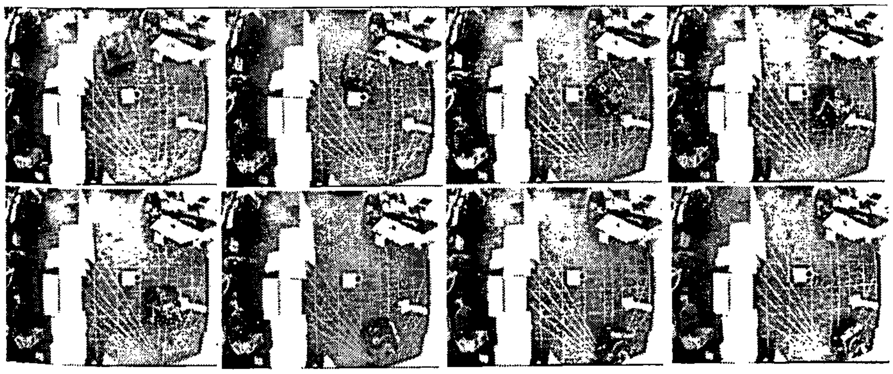
Figure 6: Overhead camera images from an obstacle avoidance experiment with the *Labmate* robot

In Fig.7 an experiment with the *Pioneer 2DX* is shown. The robot avoided obstacles travelling to the goal point marked with the white "X" on the floor. The goal point has been described by it's x and y position in the global frame. The *Pioneer* is small, agile, and reliable, thus most of the experiments have been performed with this robot. A disadvantage is the inability to use the overhead camera-based localization system. In experiments with the *Pioneer* robot the overhead camera was used only to obtain images of the scene for the documentation purposes.

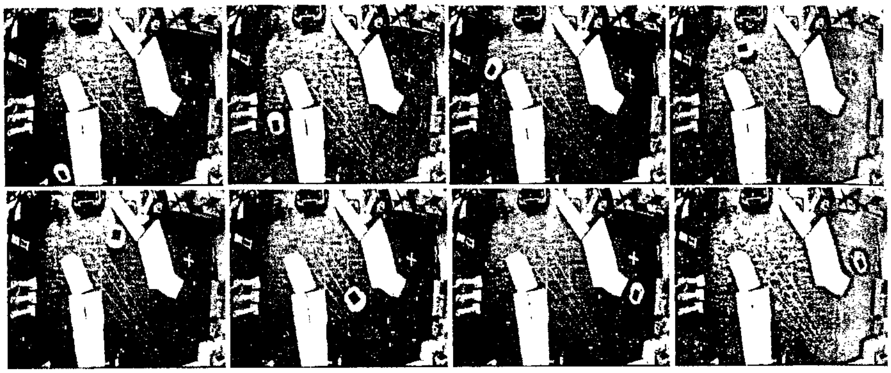

Figure 7: Overhead camera images from an experiment with the *Pioneer* robot

Figures 8 and 9 show two series of images obtained during one of the experiments aimed at the comparison of the learned and hand-crafted rules in a real environment. In these experiments the path obtained with the expert's rules (Fig. 8) was longer than the path resulting from the learned rules. This result can be attributed to the better balance between the two objectives (getting to the goal and obstacle avoidance) in the learned rules. The expert's rules were more "greedy" in moving to the goal, thus the robot initially chose to go to the right, then moving through the narrow passage. The robot with learned rules took the path to the left, moving through more open space.

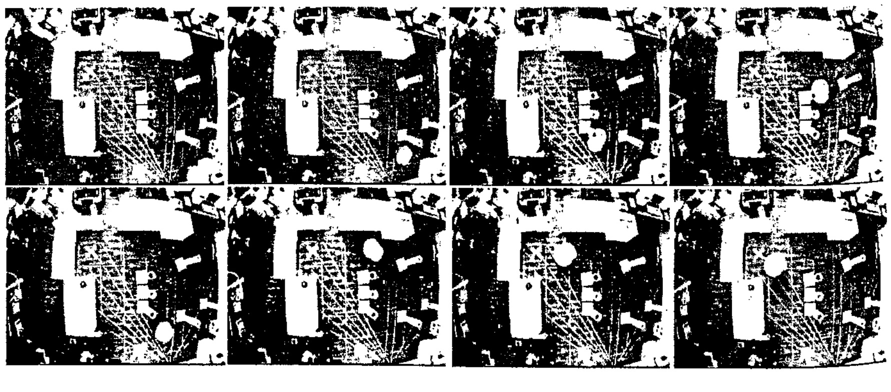

Figure 8: Obstacle avoidance with the hand-crafted rules on the *Pioneer* robot

The performance of the learned and manually defined controllers has been tested in several different environments, from simple (like in Fig. 7) to more difficult, with small obstacles and tighter passages between them. The experimental results confirmed the results of the simulations – the learned rules result in longer paths, but they provide better collision avoidance, and offer better balance between the conflicting objectives. What is interesting, the advantage of the learned rules is more visible in more challenging environments, where the robot receives much more spurious sonar readings. We attribute this fact to the learning in simulation, where the simulated sensor readings are more noisy than the real sensors. Experiments with the *Pioneer* robot have also shown the crucial role of reliable localization. In some of the experimental runs the robot, using only it's odometry, had serious problems to find the goal. These situations were caused by collisions with the "foots" of the white tubes used as obstacles, which are invisible to the sonars, but can cause slippage when the robot runs over one of them.

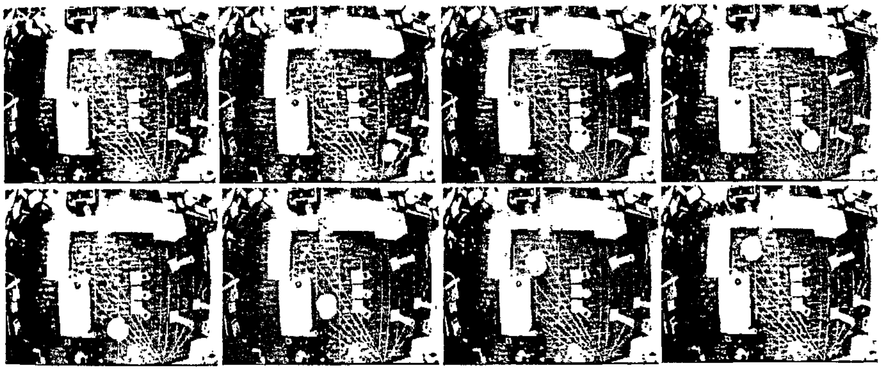

Figure 9: Obstacle avoidance with the learned rules on the *Pioneer* robot

6. CONCLUSIONS

In this article, a method using GA for obtaining fuzzy control rules has been proposed. The resulting controllers have been tested on two real mobile robots, of different sizes and sensing capabilities. Unlike other fuzzy-genetic systems the controller proposed here uses Mamdani-type fuzzy rules, which are intuitive for a human expert. Because of that, the learned rules can be easily understand and modified (if necessary) by the user. In contrary to many other GA-based systems, the one presented here does not evolve different robots over time. It rather improves the existing rule bases by applying GA learning to the population of rules treated as competing individuals. Results of experiments show, that the learning in a realistic simulation yields stimulus-response control rules which are applicable to real mobile robots.

Acknowledgement :
Thanks are due to the students involved in the experiments presented in this paper.

References

[1] Arkin R., *Behavior-based Robotics*, The MIT Press, 1998.

[2] Bączyk R., Skrzypczyński P., *Mobile Robot Localization by Means of an Overhead Camera*, Proc. AUTOMATION 2001, Warsaw, 2001, 220-229.

[3] Beom H. R., Cho H. S., *A Sensor-Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning*, IEEE Trans. on Systems, Man, and Cybernetics, 25(3), 1995, 464-477.

[4] Driankov D., Hellendoorn H., Reinfrank M., *An Introduction to Fuzzy Control*, Springer-Verlag, 1993.

[5] Goldberg D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[6] Hoffmann F., *The Role of Fuzzy Logic Control in Evolutionary Robotics*, Fuzzy Logic Techniques for Autonomous Vehicle Navigation, (Saffiotti and Driankov, Eds.), Physica-Verlag, 2001, 119-147.

[7] S. Nefti, J. Pontnau, M. Soufian, K. Djouani. *Practical Adaptative Navigation and Control of a Mobile Robot Using Neuro–Fuzzy Systems*, IFAC Workshop on Adaptive Systems in Control and Signal Processing, Glasgow, 1998, 64-69.

[8] Reignier P., *Supervised Incremental Learning of Fuzzy Rules*, Robotics and Autonomous Systems, 16, 1995, 57-71.

[9] Saffiotti A., *Fuzzy Logic in Autonomous Navigation*, Fuzzy Logic Techniques for Autonomous Vehicle Navigation, (Saffiotti and Driankov, Eds.), Physica-Verlag, 2001, 3-24.

[10] Schultz A., Grefenstette J., *Using a Genetic Algorithm to Learn Behaviours for Autonomous Vehicles*, Proc. Navigation and Control Conference, Hilton Head, 1992, 739-749.

[11] Skrzypczyński P., Rudziński D., *An Adaptive, Behaviour-based Navigation System of a Mobile Robot* (in Polish), Proc. VII National Conf. on Robotics, Wroclaw, 2001, 159-168.

[12] Skrzypczyński P., *Practical Application of Fuzzy Logic and Genetic Learning to Control a Mobile Robot*, Proc. Conf. on Fuzzy Systems (KSR'02), Krakow, 2002, 233-240.