Fuzzy Logic User Adaptive Navigation Control System For Mobile Robots In Unknown Environments

Miguel Ángel Olivares Méndez
Computer Vision Group
Universidad Politécnica de Madrid
Madrid, Spain
mig_olivares@hotmail.com

Juan Antonio Fernández Madrigal
ISA
Universidad de Málaga
Málaga, Spain
jafma@ctima.uma.es

Abstract **– This paper presents a software implementation of a user adaptive fuzzy control system for autonomous navigation in mobile robots for unknown environments. This system has been tested in a pioneer mobile robot and on a robotic wheelchair, fitted with PLS laser sensor to detect the obstacles and odometry sensors for localization of robots and the goal positions. The system is able to drive the robots to their goal position avoiding static and dynamic obstacles, without using any pre-built map. Our approach learn from user behaviors in the way it can resolve different situations against obstacles or walls.We propose and implement two updates for the fuzzy system. For the implementation of the learning algorithm we use a weighting scheme giving a value for each fuzzy-rule, this value is based on the synapse-weight idea and represent the contribution of each rule in the system output. We also create of a more important sector in the definition of the fuzzy-variables, based on a statistics system that measure the uses of all the sets of the variables in order to contract the size of the rule-base.**

Keywords **– Fuzzy logic, autonomous navigation, adaptive control, fuzzy control, mobile robots, collision avoidance, reactive navigation**

I. Introduction

The autonomous navigation of mobile robots is one of the most investigated problems in the robotic community. The usual way to face this problem is with reactive, delivered or hybrid methods. One of the biggest problem on those systems are the computational and memory cost. We approach the computational cost avoiding and reducing the use of environment maps, with the additional advantage that it can be used regardless of the place. To approach the computational cost we use fuzzy logic techniques for the interaction between the robot and surroundings. The first publication of fuzzy logic, about characterizing non-probabilistic uncertainties, called "fuzzy sets" was 1965 by Lofti A. Zadeh. Since then, the use of fuzzy logic techniques in the robotic field has become a common approach.

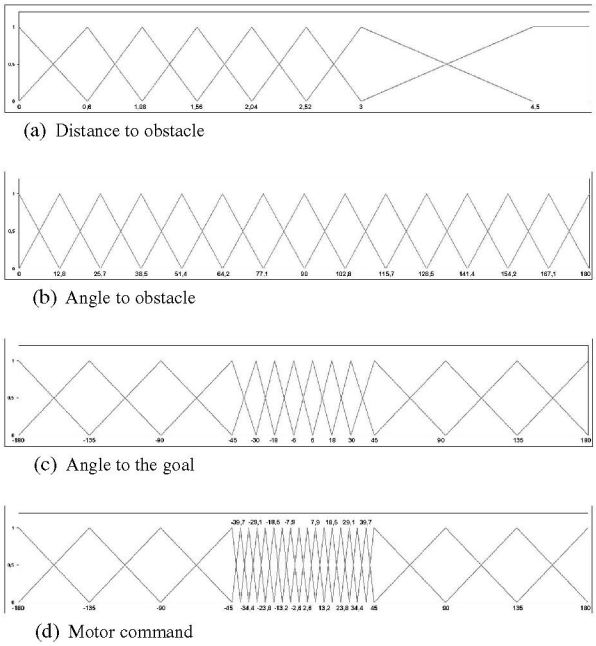
In the following paper we propose a set of improvements like, a more comprehensive sector for the definition of each fuzzy-variable and a user adaptive learning algorithm using the synapses-weight idea of the brain operation. For this purpose we create a versatile software called MOFS (**M**iguel **O**livares **F**uzzy **S**oftware) with fuzzy logic techniques that can learn from experience regardless the intelligent system architecture.

In the next section of this paper the autonomous fuzzy navigation control system is detailed. Section III define the user adaptive learning algorithm developed. In section IV describes the fuzzy-software implementation. Section V show the experimental results. Section VI conclusion and future works.

II. Autonomous Fuzzy Navigation Control System

In order to express in a more flexible way our approach we have developed an autonomous fuzzy navigation system. The structure of the fuzzy control is divided in three inputs variables for interaction with the environment and one output. The data we use to check each situation is defined by the distance (in meters) between the robot and the near obstacle, the angle direction between obstacles and the angle to the goal from the robot position (in degrees). The angles and distance are recovered using a 180 range laser sensor and an odometry system. The system output is the direction and orientation that the robot should follow in order to reach its goal position. Different velocities are declared as a constants depending how dangerous the situation is based on the distance of the near obstacle. The system scans the environment 3 times per second.

The fuzzification of the inputs and outputs are defined by using a triangular membership function. Given is the one that offer the best computational cost-simplicity ratio. This function has been used in many robotics applications for navigation purposes [2],[9] [6], [5], [10], [3].



(a) Distance to obstacle

(b) Angle to obstacle

(c) Angle to the goal

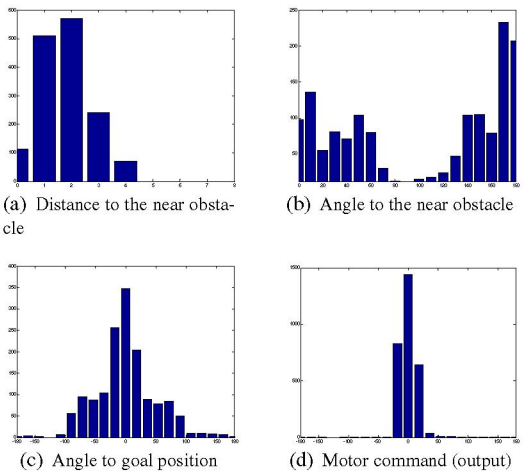
(d) Motor command

Fig. 1. FUZZY SETS AND MEMBERSHIP FUNCTIONS FOR THE MOBILE ROBOT: (A) DISTANCE TO THE NEAR OBSTACLE $(0, 4'5)$ METERS(B) ANGLE BETWEEN ROBOT AND THE NEAR OBSTACLE, $(0, 180)$ (C) ANGLE BETWEEN THE ROBOT AND GOAL POSITION, $(-180, 180)$ (D) MOTOR COMMAND (OUTPUT), THE DIRECTION TO TAKE, $(-180, 180)$

We propose to improve the system defining a more important and comprehensive sector in a variable with the aim of increase the number of fuzzy-sets just in the sector where the variable has been invoked. With this improvement we avoid to increase the number of fuzzy-sets in all space of a variable when a fault occurs in one sector, as shown in Fig. 1. This is very important because there exist situations where the number of fuzzy-rules are increased making the system more slower.

In order to define the most important sector of each variable we trained the system moving it around taking values of the variables to make a statistical study of it (Fig 2). Figure 2 show the relation between the different fuzzy-sets of each variable and the number of times that each variable has been used. To compare the same system response without this new improvement, the rule-based size was increased by 40%, this yielded to a 78% of increment in the computational time, $1.26s$ (improved) vs $2.25s$ (not improved). In regard to the time spent to check each situations the system spent 17% more, $(0, 059s$ (improved) vs $0,06903s$ (not improved).

For the inference process (in the defuzzification) we used an adaptation of the minimum and product classic method [2], [5], [10], [1], [8], [7], with the consideration of the weights assigned at each fuzzy-rules. This idea will be explained in the next



(a) Distance to the near obstacle

(b) Angle to the near obstacle

(c) Angle to goal position

(d) Motor command (output)

Fig. 2. STATISTICS OF THE NUMBER OF TIMES THAT THE DIFFERENT FUZZY-SETS HAS BEEN USED.

section. For the defuzzification part itself, we use the Height Method [2], [10], [4], [5], [8], but with an adaptation with the fuzzy-rules weights (Eq. 1a, 1b), where w^l is the rule weight and W is the maximus weight possible.

$$y = \frac{\sum_{l=1}^{M} \overline{y}^l \; min\big(\mu_{B'}(\overline{y}^l)\big) \frac{w^l}{W}}{\sum_{l=1}^{M} \; min\big(\mu_{B'}(\overline{y}^l)\big) \frac{w^l}{W}} \tag{1a}$$

$$y = \frac{\sum_{l=1}^{M} \overline{y}^l \prod \big(\mu_{B'}(\overline{y}^l)\big) \frac{w^l}{W}}{\sum_{l=1}^{M} \prod \big(\mu_{B'}(\overline{y}^l)\big) \frac{w^l}{W}} \tag{1b}$$

III. USER ADAPTIVE LEARNING ALGORITHM

This algorithm is based on the idea of the synapse-weight, defining a weight variable for each rule, that represent the contribution of each rule in the system output. The default value of the weights is 0.3 and the maximum value is 3. The decision of the user will make the weight increase or decrease of the rules that represent the situation and according with the system output. For each situation 8 rules are selected, given the fact that we use a simple overlap in the variables. A bigger overlap index had been tested without better behavior against the obstacles, but with a time increment of 50% in the creation process of the system, and with a time increment of 8% in the environment evaluation process. In the training period the system compares each rule output and the user decision, which had been fuzzificated by the same way. In case of the rule output and the user decision are different the system will decrease the weight of the rule in 0.35, and in case of the rule weight is negative the system will change the output of the rule by the set of the user decision that has the highest value. Remember that every real value is transformed in two fuzzy-sets by the fuzzification. In case of the rule output is like one of the two sets of the user decision the system will increase the weight by the reason of the fourth part of the set membership value. So with this method the system will adapt

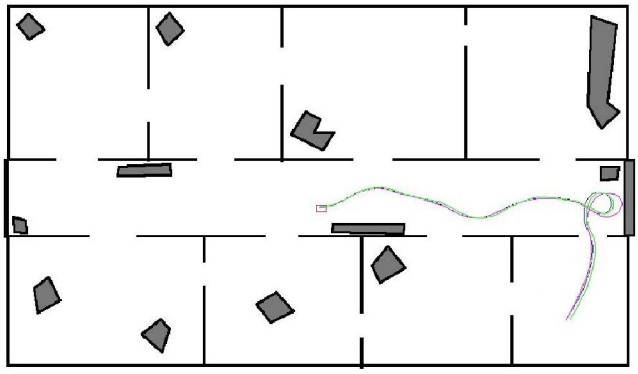


Fig. 3. LEARNING PATHS

its behavior to the user behavior. In the training phase the user just needs to make some tours and the system will learn about this, evaluating all the situations that the user makes in the tour. Taking into account of the difficulty of the tours the system will learn early.

This algorithm will be used in two phases of the system, one is the explained training phase and in the execution phase, when the user isn't making a decision, so the system continues learning based on the rules that have higher weights, which represent the user behavior, in the way that the system compares the output of the situation selected rules with the output of the system, in the same way as in the training phase.

In figures III, 4, 5 a learning example with the frightened user are shown. In the Fig. III the different path of the learning test is shown. At first, the system makes a path very near to the wall (the magenta one, the most right), then we make just one path in training mode (the green one, the second one by the right), where the system modified the weight of the rules and their outputs in order to modify its behavior to make the turn by letting more distance between the robot and the wall.

In Fig. 4 the error between the system output and the user decision (cyan color), and the other colors represent the rules output change is shown, divided by proximity to the obstacle. When the learning algorithm makes the rules' output change the system reduces the error. The more characteristic weight changes are represent in Fig. 5.

IV. SOFTWARE IMPLEMENTATION

The software was totally implemented by our own under the platform of c++. We defined one class for each part of the fuzzy-logic environment in order to facilitate the future updates and to be easy working with it. There are different classes for variables, rules, membership functions and defuzzification modes (Fig. 6). Depending on the system we want to create we can define the number of inputs and outputs that we prefer or make some different system in serial mode, where the output of one system can be the input of another. In the same way we can define the different characteristics of the variables, the type of the fuzzification inference type or the defuzzification mode. At the

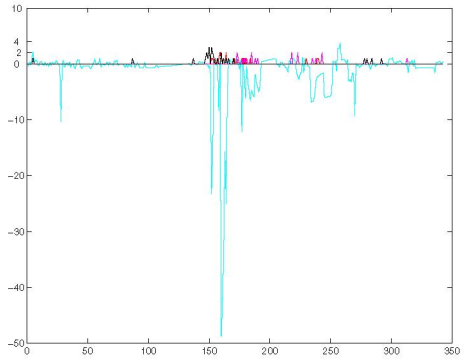


Fig. 4. OUTPUTS MODIFICATIONS.

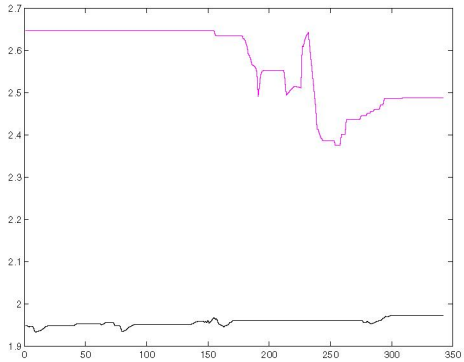


Fig. 5. WEIGHT MODIFICATIONS.

beginning the rule-base is implemented randomly, making this process so fast, and depending on the size of the rule-base the user adaptive learning algorithm in the training mode modified this in a few or little more tours or uses. For this initial training it is possible to introduce a pattern list. In this paper we make an initial rule base and then we use it for adaptation of two users. With this initial rule base, where the weight of all the rules are reset at the default weight, we can make a basic adaptation in 4 or 5 tours, taking into account its complexity, with the robotic-wheelchair.

For the updates of the software we can make it in all the ways of the fuzzy-logic parts, like introducing different membership functions, fuzzy inference types or introducing another kind of defuzzification mode.

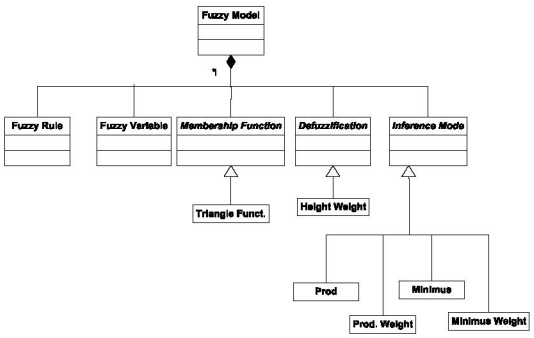


Fig. 6. SOFTWARE DEFINITION.

V. Experiments Results

The test part is divided in two phases; simulation and the real test with two robots.

For the two phases we create a full rule base with random output for each one. With this strategy we obtain a reduction in time for creation of all system components. So we start to do different path, where we control the robot movement and the system doesn't do any action. At the beginning with the user adaptive learning algorithm the system evaluated all the situations but doesn't make any action. Instead of, it learns from the user decisions, it starts to modify the weights of the rules which are wrong, and then the value of the output. In that way, we consider a default weight minor than the decrease value, in order to change, at first, the initial random values of the rules. The velocity that the system learns depends on the size of the rules-base. In the simulation phase, at first, we try to get a

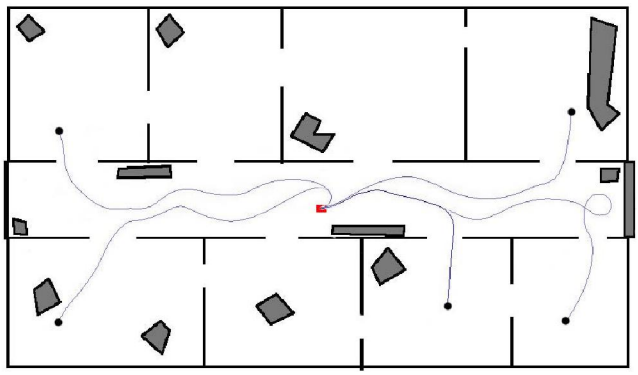


Fig. 7. Frightened user paths under the simulator.

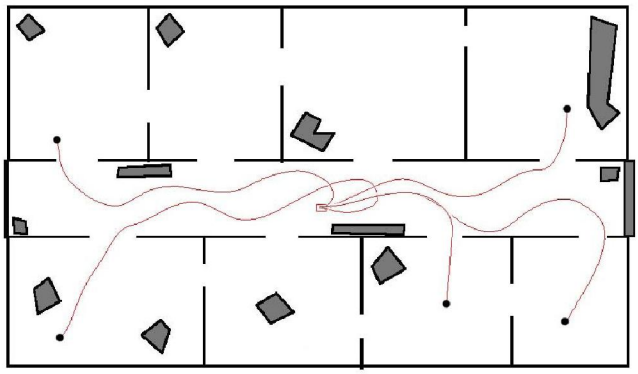


Fig. 8. Reckless user paths under the simulator.

basic rules-base, which the robot can avoid the obstacles and get to the goal. Over this basic rules-base we start to make the adaptation to two different users, the frightened one, who tries to pass as far as possible from the obstacles and walls (Fig 7), and the reckless one, who tries to fit the path very near the obstacles, going through the path more directly (Fig 8).

For the real test phase we use two robots, the pioneer-2 and a robotic wheel-chair. Both of them have a 180 laser scanner (*PLS*), which is the only sensor we use. We have no problems using the different rules-bases from the simulation phase, but in the same way than the simulation phase we learn from the beginning using the controller and with the system in listening mode, without taking any action, just hearing the user decisions and learning from them. Then we make the adaptation to the same two users. In Fig. 10 and 9 two paths made by the robotic-wheelchair by the two users are shown. Those maps have been catched for the representation of the paths and for the odometry control, but they are not used by the autonomous navigation control system. A zoom on the most characteristic part of

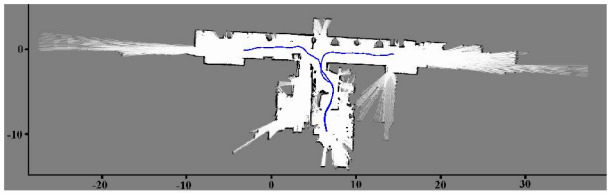


Fig. 9. Real test with the frightened user. Map in meters.

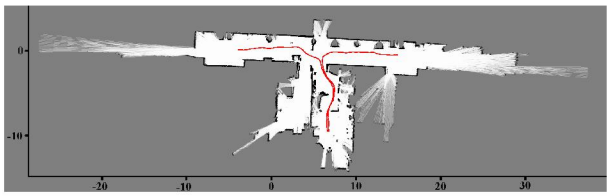
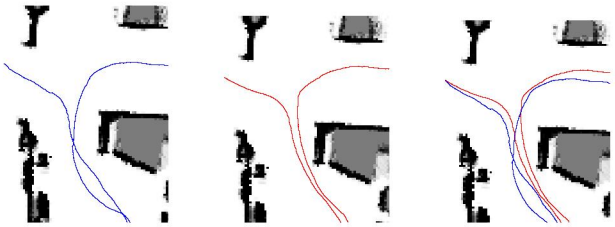


Fig. 10. Real test with the reckless user. Map in meters.

the paths, the door crossing, is shown in figures 11(a), 11(b) and a comparison between them is shown in figure 11(c). The proximity to the door is measured by the charts shown in figures 12(a), for the frightened user, and 12(b), for the reckless one. In those graphics it have to taken in account that the width of the wheelchair is $50cm$, the PLS-laser is positioned in the middle of it, and the width of the door is $110cm$. So to understand the graphics, we must said that the distances between the wheelchair and the two sides of the door are, for the frightened user, $31cm$ and $24cm$, nevertheless, for the reckless user, the distances are $4cm$ and $51cm$ to the door. In the Fig 13 are shown two pictures about this precise moment.

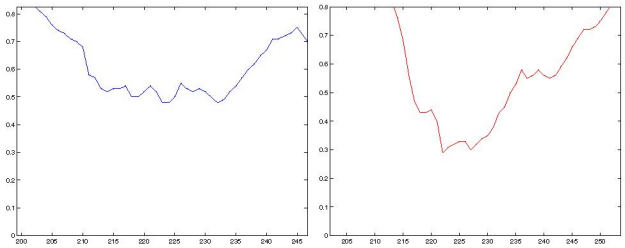
In this phase we can test the dynamic obstacles avoidance without any problem with the pioneer mobile robot for the two users. Those paths are shown in figures 15 and 14. Where the dynamic obstacles is a person walking against the robot direction in the map point $(0, 9)$ for the frightened user and in the point $(0, 10)$ for the other one.

Finally, the efficient of the system is evaluated taking into account the efficiency of the system, it evaluated each situation and gives the action to do in $0,059$ seconds and create the system (create variables and the rule-base) in $1,26$ seconds.



(a) Zoom of the Fig 9 (b) Zoom of the Fig 10 (c) Compared zoom of the two users

Fig. 11. REAL TEST PATHS' ZOOM.



(a) Frightened user (b) Reckless user

Fig. 12. MEASURES OF THE PROXIMITY BETWEEN THE ROBOT AND THE DOOR (FIG. 11(B) AND FIG. 11(A)). THE Y-AXIS REPRESENTS THE DISTANCE BETWEEN THE LASER OF THE WHEELCHAIR AND THE NEARLY OBSTACLE, AND THE X-AXIS REPRESENTS THE SAMPLES.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, a fuzzy-based navigator system for mobile robots has been proposed for obstacle avoidance and navigation problems in unknown environments. The systems has a learning algorithm that is capable to adapt to different users. The approach has been tested in two robots in different environments with static and dynamics obstacles. The main goal of this approach was the reduction of the computational and memory costs of the system in the evaluation of each situation. Our system has achieved, in regard to others reactive navigation control tested in the same platform, an average computational time of 0.59 seconds, 90% less when compared with previous

(a) Frightened user (b) Reckless user

Fig. 13. DOOR CROSSING MOMENT.

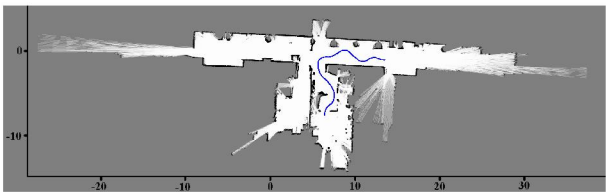


Fig. 14. REAL TEST WITH DYNAMICS OBSTACLES AND FRIGHTENED USER. MAP IN METERS.

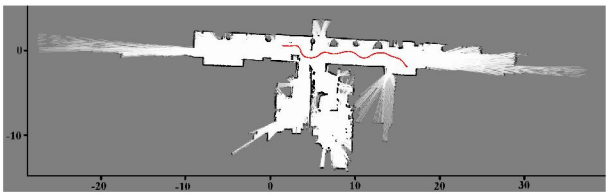


Fig. 15. REAL TEST WITH DYNAMICS OBSTACLES AND RECKLESS USER. MAP IN METERS.

systems. We have achieved an important capability for this system which is the quick adaptation to different users. The approach developed has proven its versatility and scalability in the way it can be used for different intelligent system that learn about the experience and/or to adapt to different users. We just redefined the system by the inputs and outputs that we need, adding modules or code segments where is needed.

For future works, we have planned to scale more the software with different methods of fuzzy-inference, defuzzification or membership functions. We plan to include new behaviors to the autonomous navigation control system like following walls, scape from blind roads, etc. This can be achieved by storing different rule-base and including more capabilities for recognition of these situations or just changing of the rules when the system has too many continuous errors. We are considering to include more sensors and/or real-time map creation that stores repetitive paths, using SLAM techniques.

Additionally we intend to use the software in a different type of robotic system that have no navigation task as a main objective, like medical robots, car shock absorber systems or other security car systems.

ACKNOWLEDGMENT

The authors would like to thanks Dr. Pascual Campoy, Juan Fernando Correa, Ivan Mondragón, Dr. Luis Mejias, Carol Martinez and Jan Treiber for support and help in the development of this work. We also thanks José Luis Blanco, C. Galindo and M. E. Cruz Martín for the initial collaboration during this work.

REFERENCES

[1] ARAUJO, R., NUNES, U., AND DE ALMEIDA, A. Fuzzy-based reinforcement learning of a robot force control skill. In *in Proc. of IEEE International Symposium on Industrial Electronics (ISIE'96)* (Warsaw, Poland, 17–20 1996), pp. 453–458.

[2] CASTELLANOA, G., AND FANELLI, A. A self-organizing neural fuzzy inference network. In *in Proc. of IEEE-INNS-ENNS International*

Joint Conference on Neural Networks (IJCNN 2000) (Como, Italy, 24–27 2000), pp. 14–19.

[3] DE OLIVEIRA, J. V., AND LEMOS, J. M. "a comparision of some adaptive-predictive fuzzy-control strategies". *IEEE. Transactions on Systems, Man and Cybernetics* (2000).

[4] DEL BRO, B. M., AND MOLINA, A. S. *Redes Neuronales y Sistemas Borrosos*. Ed. Ra-Ma, 2001.

[5] HUANG, S.-J., AND HU, C.-F. "predictive fuzzy controller for robotics motion control". *IEEE* (1995).

[6] KASABOV, N. K. "learning fuzzy rules throught neural networks". *IEEE* (1993).

[7] PESONEN, A., AND WOLSKI, A. "quatified and temporal fuzzy reasoning for active monitoring in rapidbase". *TOOLMET* (2000).

[8] THONGCHAI, S. "behavior-based learning fuzzy rules for mobile robots". *IEEE* (2000).

[9] WU, C.-J. "fuzzy robot navigation in unknown enviroments". *National Yunlin Institute of Technology* (1995).

[10] ZAVLANGAS, P. G., TZAFESTAS, S. G., AND ALTHOEFER, K. "fuzzy obstacle avoidance and navigation for omnidirectional mobile robots". *IEEE* (2000).