Fuzzy Behaviour Based Mobile Robot Navigation in Static Environment

Pratik Gyawali and Praveen Kumar Agarwal
Department of Mechanical Engineering
Motilal Nehru National Institute of Technology Allahabad
Prayagraj, Uttar Pradesh, India

***Abstract*—Navigation in mobile robotics refers to the motion of a robot from an initial location to the goal location safely by avoiding all the obstacles on its way. In this paper, a method is presented for navigation using a behavioural architecture based on fuzzy controllers. Mamdani type fuzzy logic controllers are used to build up individual behaviours. The control algorithm is developed in Simulink® using the Robotics System Toolbox and Fuzzy logic Toolbox of MATLAB®. The performance of the proposed controller is evaluated in a virtual platform V-REP©, through cross platform simulation. Each of these programs communicates using standard ROS messages and topics. Simulation results for multiple test conditions show the feasibility and good performance of the proposed method.**

Keywords— robot navigation, mobile robotics, fuzzy logic, ROS

I. INTRODUCTION

The ability to navigate autonomously while avoiding both static and dynamic obstacles in a crowded and unpredictable environment has been a hot topic of research in the field of robotics. Traditional mobile robot planning approaches are not robust enough to overcome these challenges [1]. As a result, many reactive approaches along with artificial intelligence techniques were introduced.

Autonomous navigation is associated with large amount of uncertainties in the sensor information as well as the environmental dynamics. Fuzzy theory fits very well in this application because of its ability to cope with uncertain, incomplete and approximate information [2]. The theory of fuzzy logic systems is inspired by the remarkable human capacity to reason with perception-based information. Therefore, the field of robot navigation has seen many applications of Fuzzy logic in recent times.

Teleity, Nossair, Mansour and TagElDein [3] proposed the sensor-based mobile robot navigation in an indoor environment using a fuzzy logic controller. Ren, Wang and Du [4] designed an intelligent fuzzy logic controller to solve the navigation problem of wheeled mobile robot in an unknown and changing environment. Muthu, Gloude Swaminathan and Kumar [5] proposed an Atmega microcontroller based fuzzy logic controller for the wheeled mobile robot. Their controller trained the mobile robot to navigate in an environment without any human intervention.

Behaviour based navigation approach structures the navigation task in order to deal with the navigational complexities and has been used extensively in the field of robot navigation. The behaviour control is a special form of decentralized switching control in which every behaviour is autonomous and drives the robot without dependencies on other behaviours.

This paper addresses the mobile robot navigation issue by combining the ideas of behaviour based control and fuzzy logic control. The rest of this paper is organized as follows; Section 2 presents the mathematical model of the robot. In Section 3, behaviour design will be discussed. Simulation method is presented in Section 4 which is followed by the results and conclusions in Section 5 and Section 6 respectively.

II. KINEMATIC MODEL OF DIFFERENTIAL DRIVE

A differential drive mobile robot is the most common type of mobile robot consisting of a chassis that is driven by two wheels and a supporting castor wheel. The robot moves forward and takes turn depending on the right (v_R) and left wheel velocity (v_L) which can be controlled at different rates by the input signal.

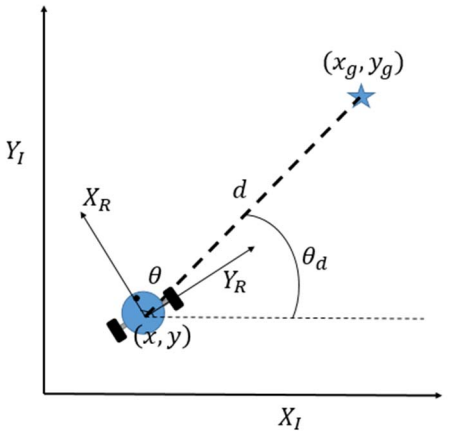

Figure 1: Model of the differential drive robot

Based on this principle, a simple kinematic model for a differential drive robot as shown in Figure 1 is described by the equations (1)-(3).

$$\dot{x} = \frac{R}{2}(v_R + v_L)cos\theta \quad (1)$$

$$\dot{y} = \frac{R}{2}(v_R - v_L)sin\theta \quad (2)$$

$$\dot{\theta} = \frac{R}{L}(v_R - v_L) \quad (3)$$

where (x, y) and θ define the position and the orientation of the robot in the Inertial frame respectively, R is the wheel radius, L is the track width and (x_g, y_g) denote the goal location.

III. BEHAVIOUR DESIGN

The basic idea in behaviour based navigation is to subdivide the navigation task into small, well defined and easy to manage controllers known as behaviours. This divide and conquer approach makes the system modular, which both simplifies the navigation solution as well as offers a possibility to add new behaviours without causing any major increase in complexity in the system.

In order to cope with uncertainty in the sensor measurement and the environment, the behaviours in the proposed method are implemented as Fuzzy Logic Controllers. The use of fuzzy logic controllers provides a model free design based on expert knowledge as well as enables the use of same controller for multiple test and more complex conditions. Mamdani inference engine with the maximum aggregation and centroid defuzzification method is used. These controllers are developed in the MATLAB® environment using the Fuzzy Logic Toolbox. The details about various behaviours are presented next.

A. Go-To-Goal Behaviour

This behaviour comprises of two Mamdani Fuzzy logic controllers, *Orientation Controller* and *Distance Controller*.

The *Orientation Controller* minimizes the error in the angular heading and controls the angular velocity with which the robot tries to orient itself towards the goal. Both the input (heading error) and the output (angular velocity) are characterized by Gaussian membership functions.

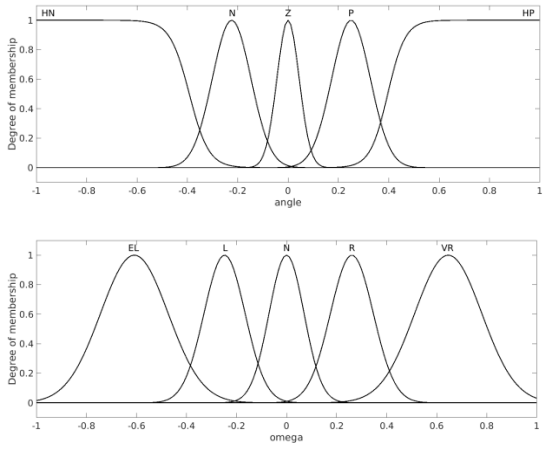

Figure 2: Membership Functions for *Orientation Controller*

Figure 2 shows the membership functions for input (*angle*) and output (*omega*) of the *Orientation Controller*. The input has five linguistic variables namely Highly Negative (HN), Negative (N), Zero (Z), Positive (P) and Highly Positive (HP). The output has five linguist variables which are Extreme Left (EL), Left (L), None (N), Right (R) and Extreme Right (ER). The following set of rules governs this controller:

IF the angular error is Highly Negative THEN steer Extremely Left

IF the angular error is Negative THEN steer Left

IF the angular error is Zero THEN steer Zero

IF the angular error is Positive THEN steer Extremely Right

IF the angular error is Highly Positive THEN steer Extremely Right

Similarly, the *Distance Controller* minimizes the distance to the goal and controls the translational velocity with which the robot moves towards the goal. Both the input (distance to the goal) and the output (linear velocity) are characterized by Gaussian membership functions.

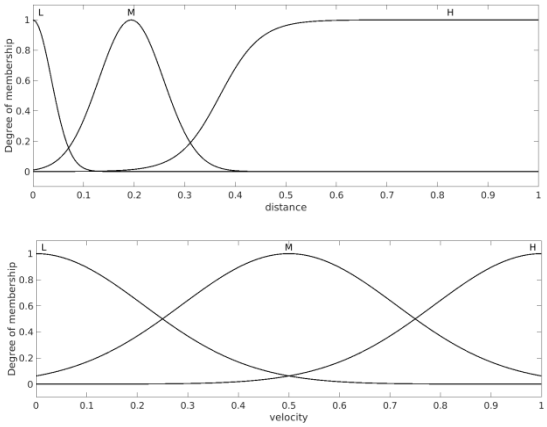

Figure 3: Membership Functions for Distance Controller

Figure 3 shows the membership function of the *Distance Controller*. The input (*distance*) has three linguistic variables namely Low (L), Medium (M) and High (H) as shown in the figure. The output (*velocity*) has three linguist variables which are Low (L), Medium (M) and High (H). The following set of rules governs this controller:

IF distance to goal is Low THEN velocity is Low

IF distance to goal is Medium THEN velocity is Medium

IF distance to goal is High THEN velocity is High

B. Obstacle Avoidance Behaviour

The obstacle avoidance behaviour ensures that the robot never collides with an obstacle in its way. In case the robot senses any obstacle in front of it and if this distance is less than the safe distance, the robot will steer away from the obstacle towards a location where the robot senses no obstacle or the obstacle is quite far away.

This behaviour consists of a single Mamdani Fuzzy Logic Controller with three inputs and a single output. The three inputs to the obstacle avoidance controller are the three sonar (Left, Front and Right) readings whereas the output is the angular velocity of the robot.

Figure 4 shows the Gaussian membership functions used for the *Obstacle Avoidance Controller*. The input (*leftSensor*) has three linguistic variables namely Near (N), Close (C),

and Far (F) whereas the output (*omega*) has five linguist variables which are Very Right (VR), Right (R), Zero (Z), Left (L) and Very Left (VL). The other two inputs (*frontSensor and rightSensor*) also use the similar membership functions as that of input *leftSensor*.

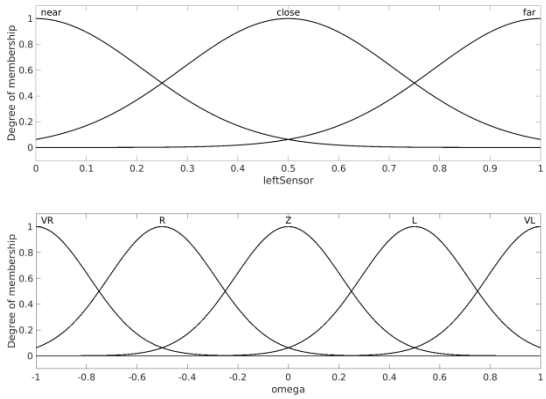

Figure 4: Membership Functions for Obstacle Avoidance Controller

The following set of simple rules governs the obstacle avoidance controller:

IF obstacle is Far to right sensor and Far to front sensor and Far to left sensor THEN steer Zero

IF obstacle is Far to right sensor and Close to front sensor and Near to left sensor THEN steer Very Right

IF obstacle is Far right sensor and Close to front sensor and Close to left sensor THEN steer Right

IF obstacle is Near right sensor and Close to front sensor and Far to left sensor THEN steer Very Left

IF obstacle is Close right sensor and Close to front sensor and Far to left sensor THEN steer Left

C. Behaviour Coordination

Algorithm: Algorithm for Robot Navigation

Input: Goal Location (x_g, y_g), Encoder ticks $(tick)$, IMU heading (γ), Distance to the obstacles (d_{obst})

Output: v_R, v_L

1: Compute the next state (x', y', θ') of the robot:

$$x' = x + D_c\cos(\varphi),\ y' = y + D_c\sin(\varphi) \text{ and } \theta' = \gamma$$

$$\varphi' = \varphi + \frac{D_r - D_l}{L} \text{ where, } D_c = \frac{D_r + D_l}{2},\ D_i = 2\pi R_i \frac{\Delta tick}{N}$$

2: Compute $d = \sqrt{(x_g - x)^2 + (y_g - y)^2}$

$$\theta_d = \tan^{-1}\left(\frac{y_g - y}{x_g - x}\right)$$

3: **while** $d \leq \in$ do

4: If $d_{obst} \leq \sigma$ then,

$v, \omega = FuzzyAvoidObstacle\ (d_{obst}, d, \theta_d)$

5. Else $v, \omega = FuzzyGoToGoal\ (d, \theta_d)$

6: $v_R = \frac{2v + \omega L}{2R}, v_L = \frac{2v - \omega L}{2R}$

7. **end while**

8: **Return** v_R, v_L

Through the coordination between multiple behaviours, autonomy in the navigation can be achieved. In the proposed method, the distance measured by the ultrasonic sensors play the vital role in the switching criterion.

The three ultrasonic sensors are constantly giving the information about the presence of the obstacles in the environment. These measurements are constantly compared against the safe distance of 200 mm to the obstacles. If the robot doesn't detect any obstacle in the danger range (<200 mm), it simply moves towards the goal. In case an obstacle shows up, the robot makes corresponding manoeuvres depending upon the directional location of the obstacles.

IV. SIMULATION METHOD

Three softwares are used for testing the developed method. In order to keep the simulation as realistic as possible and to make the visualizations more interactive, Virtual Robot Experimentation Platform (V-REP) is used.

The entire control algorithm is developed in the MATLAB® & Simulink® environment as discussed in the previous sections. Robotics Operating System (ROS) is used as pipeline for the data exchange. ROS is the standard framework used in robotics research across industry and academics. Moreover, ROS enables the possibility to use the same algorithm in the hardware with few modifications.

Figure 5 shows the coordination and the flow of information across multiple softwares used in the simulation.

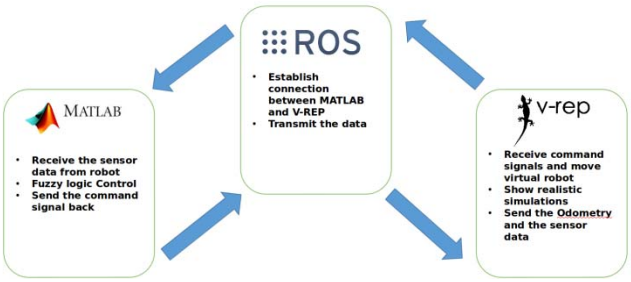

Figure 5: Information exchange across the softwares used

V. RESULTS AND DISCUSSIONS

Multiple test conditions are developed in V-REP. After the simulations, data are extracted in MATLAB® and the robot motion is studied. In this section, the results from the simulations are presented.

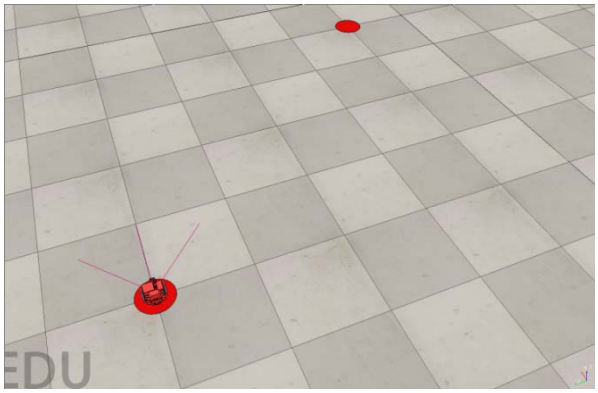

Figure 6: Simulation environment without obstacles

In each of the test conditions, the robot is placed at the point [0, 0] and is commanded to reach the goal at [2,-2] (distance in meters). The linear and angular velocities of the robot are constrained to be 0.15m/s and 5rad/s respectively.

Initially, no obstacles are placed in the environment. Figure 6 shows the test condition in V-REP without any obstacles in the environment. The start and the goal locations are marked in red.

As seen in Figure 7, the shortest path from [0, 0] to [2,-2] is a straight line joining these points. The robot is initially facing front, hence it maintains the desired heading by steering towards the straight line. After achieving the desired heading (motion in a straight line), the robot then continues to move till it reaches the goal.

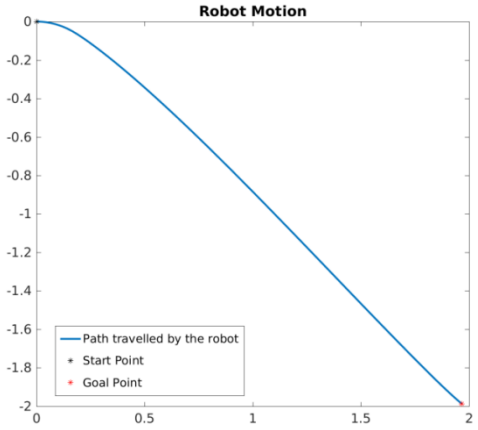

Figure 7: Robot motion without obstacles

Now, a wall like obstacle is developed in V-REP as shown in Figure 8. Due to the presence of obstacle in the shortest path between the points, the robot is unable to proceed with the previous path.

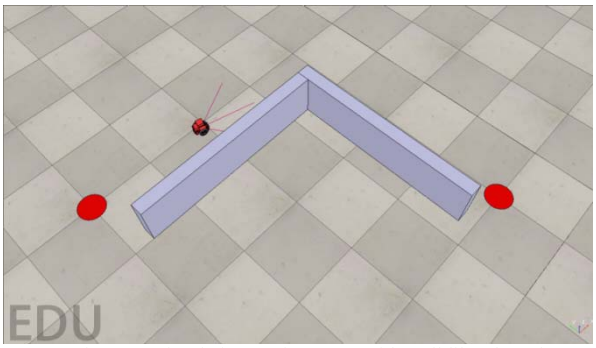

Figure 8: Simulation environment with wall like obstacles

Figure 9 shows the trajectory of the robot in this condition. It can be seen that the robot moves along the wall initially. This behaviour occurs up to point A in the trajectory until the robot finds itself free from the wall obstacle. After point A, the robot maintains the desired heading and finally proceeds towards the goal.

Likewise, a maze like environment is developed as shown in Figure 10. This is a peculiar condition as the robot can have multiple paths to the goal and can often fall in the trap of the local minima.

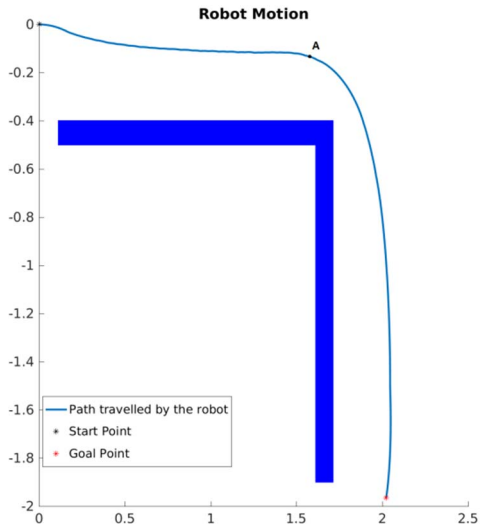

Figure 9: Robot motion in presence of wall like obstacles

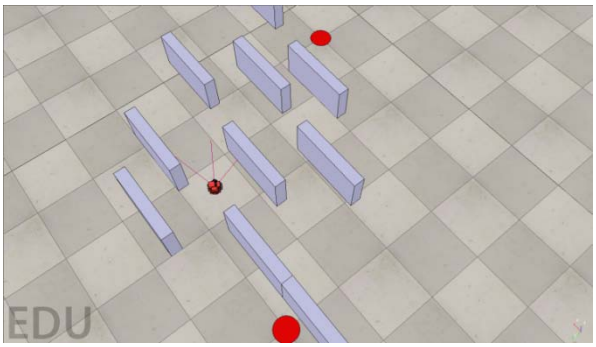

Figure 10: Simulation environment with Maze like obstacles

Figure 11 shows the trajectory of the robot in a maze like environment. Noticeable bends are seen at the points A, B and C. These are the points where *Go-To-Goal Controller* is switching over to *Obstacle Avoidance Controller*. At these points, the robot sees the obstacles ahead and steers away from them.

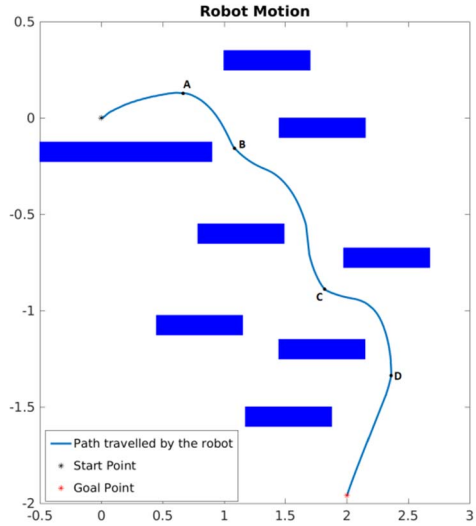

Figure 11: Robot motion in presence of Maze like obstacles

At point D, *Go-To-Goal Controller* again takes action and drives the robot towards the goal. Since this is a reactive architecture and no prior information of the world is known, the robot entirely works on the *Sense-Act* method. Despite the Maze like situation and multiple local minimas, the robot is able to reach the goal in a smooth path without significant jitters in the trajectory. Finally, a cluttered environment with large number of cylindrical obstacles is developed as shown in Figure 12. This type of environment is much complex as characterized by multiple ways to reach the goal location.

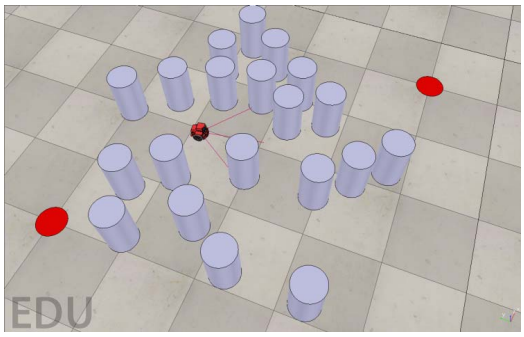

Figure 12: Simulation environment with cluttered obstacles

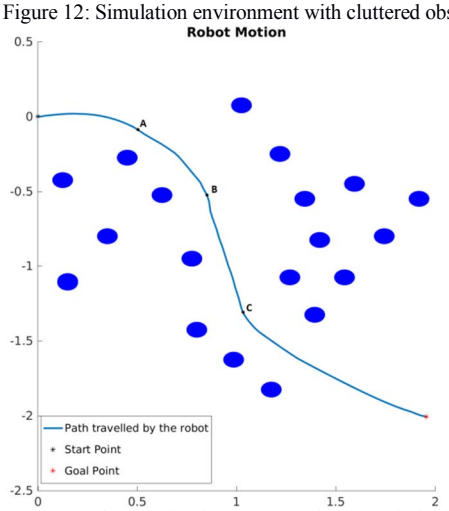

Figure 13: Robot motion in presence of cluttered obstacles

Figure 13 shows the trajectory of the robot in the cluttered environment. The *Obstacle Avoidance Controller* is active from point A to C. After point C, only *Go-To-Goal Controller* is in play.

Similarly a disturbance condition is simulated by presenting two moving robots in the path of our robot motion as shown in Figure 14. The robot is placed at [0,0] and is commanded to move at the point [1,-2]. As shown in Figure 15, at point A and B the robot switches to the *Obstacle Avoidance Controller* and deviates from the path.

Table I summarizes the performance of the proposed controller in multiple test conditions.

TABLE I. SUMMARY OF ROBOT PERFORMANCE

S. No.	Environment	Simulation Time	Path Length
1	No obstacles	24.80 s	2.82 m
2	Wall like obstacles	53.20 s	3.60 m
3	Maze like obstacles	109.25 s	3.86 m
4	Cluttered obstacles	92.42 s	3.12 m
5	Disturbance	37.50 s	2.75 m

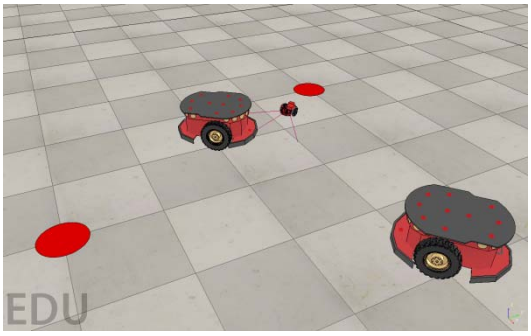

Figure 14: Simulation environment in presence of disturbance

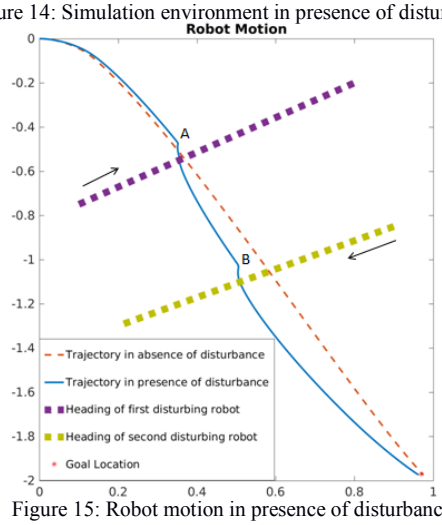

Figure 15: Robot motion in presence of disturbance

VI. CONCLUSIONS

The paper presents the design and implementation of a navigation method for a differential drive mobile robot utilizing the behaviour based control and fuzzy logic theory. The behavioural approach simplifies the navigational complexity whereas the use of fuzzy logic control ensures robust performance despite the uncertain sensor information and environmental dynamics. Realistic simulations in a virtual robot simulator are carried out in multiple test conditions. Simulation results show the smooth traversal of the robot with significantly less bends and jitters. In future, it is planned to further enhance the obstacle avoidance behaviour for dynamic obstacles, incorporate other behaviours and carry out experiments on the hardware ROS enabled robot.

REFERENCES

[1] Latombe, J. C., *Robot motion planning*. Springer, 1991.

[2] Ross, T. J., *Fuzzy logic with engineering applications*, John Wiley & Sons, 2009.

[3] El-Teleity, S. A., Nossair, Z. B., Abdel-Kader Mansour H. M. and TagElDein, A., *Fuzzy logic control of an autonomous mobile robot*, 16th International Conference on Methods & Models in Automation & Robotics, Miedzyzdroje, pp. 188-193, 2011.

[4] Ren, L., Wang, W. and Du, Z., *A new fuzzy intelligent obstacle avoidance control strategy for wheeled mobile robot*, 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, pp. 1732-1737, 2012.

[5] Muthu T., Thierry Gloude R., Swaminathan S. and Satish Kumar L., *Fuzzy logic controller for autonomous navigation*, 2012 International Conference on Communication and Signal Processing, Chennai, pp. 81-92, 2012.

[6] Tsai, C., Chen, C., Chan, C., & Li, Y.Y., *Behavior-Based Navigation Using Heuristic Fuzzy Kohonen Clustering Network for Mobile Service Robots*, International Journal of Fuzzy Systems, Vol. 12, No. 1, pp. 25-32, 2010.