International Journal of Engineering & Technology, 7 (4) (2018) 2410-2417

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.14767
Research paper



Robot navigation with obstacle avoidance in unknown environment

Neerendra Kumar[1*] and Zoltán Vámossy[2]

[1,2]*John von Neumann Faculty of Informatics, Óbuda University, Budapest, Hungary.*
[1]*Department of Computer Science & IT, Central University of Jammu, India.*
[*]*neerendra.kumar@phd.uni-obuda.hu*

Abstract

In this paper, a robot navigation model is constructed in MATLAB-Simulink. This robot navigation model make the robot capable for the obstacles avoidance in unknown environment. The navigation model uses two types of controllers: pure pursuit controller and fuzzy logic controller. The role of the pure pursuit controller is to generate linear and angular velocities to drive the robot from its current position to the given goal position. The obstacle avoidance is achieved through the fuzzy logic controller. For the fuzzy controller, two novel fuzzy inference systems (FISs) are developed. Initially, a Mamdani-type fuzzy inference system (FIS) is generated. Using this Mamdani-type FIS in the fuzzy controller, the training data of input and output mapping, is collected. This training data is supplied to the adaptive neuro-fuzzy inference system (ANFIS) to obtain the second FIS as of Sugeno-type. The navigation model, using the proposed FISs, is implemented on the simulated as well as real robots.

Keywords: *ANFIS; fuzzy controller; obstacle avoidance; path planning; robot navigation.*

1. Introduction

Mobile robots have a wide range of applications like space missions, household and office work, receiving-delivering orders of clients in restaurants, transportation of logistics in inventories, inspection and maintenance, agriculture, security and defence, operations in radioactive areas etc. Mobile robots are very useful in the area where either the task is boring due to the repetitiveness of the same operations or the working environment is hazardous for the human being. For mobile robots, the navigation from one location to the other is one of the most desirable operations. Therefore, robot navigation attracted many researcher since past few decades to the recent time [1]. In an unknown environment where very less or no information about the environment is prior available to the robot for the navigation. As a result, robot navigation in unknown environment becomes much more challenging in comparison to the robot navigation in known environment where sufficient information about it is prior available. In the navigation process can be subdivided into two parts: global navigation and local navigation. The global navigation method is used to find globally optimal path on the basis of priory information like map of the environment. However, in the absence of sufficient priory information the global navigation strategy is not applicable. Instead of the priory information, the local navigation is based on the on-line sensory data received from the sensors mounted on the robot. Therefore, the local navigation methods can also be applied when the navigation environment is unknown or partially known. In robot navigation, obstacle avoidance is one of the major tasks [2]. The task of obstacle detection can be more complex in the case of similar kind of obstacles [3]. Obstacles in the robot's path can be categorized as static and dynamic. Static obstacles are those which do not change their position and orientation after the start of the navigation. In this case, the priory information about the obstacles does not change after navigation starts. Therefore, the global navigation methods can provide an optimal path from start to goal. On the other hand, the dynamic obstacle changes their position or orientation after the start of navigation process. So, in the presence of dynamic obstacles, the global navigation methods are not applicable. The local navigation methods can solve the problem of dynamic obstacles. In case of unknown environment, navigation task needs an approach that can work in uncertain situation [4].

Commonly, there are two steps in the design of the controllers for mobile robots. Firstly, the data from sensors are computed into high level and meaningful values of variables. Secondly, some machine learning method is used to produce a controller. Taking these high level variables as input, the controller outputs the control commands for the robot [5]. Humans perform the navigation task without any exact computation and mathematical modelling. The ability of humans to deal with uncertainties can be followed in robot navigation using the neural network and fuzzy logic. Fuzzy logic gives an abstract behaviour of the system in an intuitive form even in the absence of precise mathematical and logical model. In practice, the sensor information is noisy and unreliable. The sensor inputs to the robot can be mapped into the control actions using fuzzy rules. Therefore, fuzzy logic is a suitable tool to achieve robust robot behaviour [6].

In robot navigation, neural network can be employed to map the relationships between inputs and outputs for interpreting the sensory data, obstacle avoidance and path planning. Although, the neural

Copyright © 2018 Author. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

network method is slow and the learning algorithm used may not lead to an optimal solution. Thus, the integrated approaches like neuro-fuzzy technique are much more suitable for the robot navigation task [7]. In complex and unknown environments, simulation experiments indicate better navigation performance using neuro-fuzzy approach. Using neuro-fuzzy approach, the parameters of the controller can be optimized and the structure of the controller can be self-adaptive. To improve automatic learning and adaptation, ANFIS combines fuzzy logic and neural network. ANFIS can be used to predict and model several engineering systems. ANFIS is a fuzzy based model which is trained using some data set. Consequently, ANFIS computes the best suitable parameters of membership functions involved in FIS [8, 9]. A fuzzy controller for robot navigation in unknown static and dynamic environment is implemented in [10]. However, ANFIS was not taken into consideration and controllers were not implemented on real robots.

The remaining part of this paper is organized as follows: Section 2 presents the theoretical background of controllers used in our robot navigation model. Simulink model for the robot navigation is presented in Section 3. Section 4 describes the FISs for the fuzzy controller taken in the model. Section 5 is for experimental set-up and results. Conclusions of the study are presented in Section 6.

2. Background of controllers used

The path pursuit controller is based on the path pursuit algorithm given in subsection 2.1. Subsections 2.2 and 2.3 present theoretical insight of Mamdani-type and Sugeno-type FIS, respectively.

2.1. Path Pursuit Algorithm

The pure pursuit algorithm [11] calculates the required curvature that can lead the robot from its current position to the goal position. The geometry of the pure pursuit algorithm is given in Fig. 1.

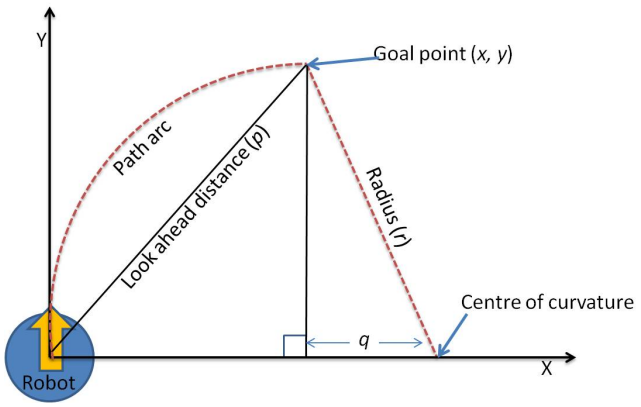


Figure 1: Geometry of the pure pursuit algorithm.

Pythagorean equation for the larger right triangle in Fig. 1, can be written as (1):

$$p^2 = x^2 + y^2. \tag{1}$$

The line segments at the X axis of Fig. 1 hold the relation given in (2).

$$q = r - x. \tag{2}$$

For the smaller right triangle of Fig. 1, Pythagorean equation can be presented by (3).

$$r^2 = q^2 + y^2. \tag{3}$$

By solving (1) to (3), the curvature (ρ) of the path arc can be given by (4) as follows:

$$\rho = \frac{1}{r} = \frac{2x}{p^2}. \tag{4}$$

The curvature of the path arc, obtained in (4), determines the steering wheel angle of the robot.

2.2. Mamdani FIS

Fuzzy inference system maps the input variable(s) to an output by using If-Then rules, membership functions and fuzzy logical operation. Mamdani fuzzy inference includes the following five steps:

1) Convert the crisp numerical values of input(s) into the fuzzy sets using membership functions. This is known as fuzzification of input variables.
2) Employ the fuzzy operator.
3) Employ implication method.
4) To make a decision, combine the fuzzy sets obtained in step 3 into a fuzzy set. This is called aggregation process.
5) Defuzzify the combined fuzzy set to obtain the output as a single scalar quantity.

2.3. Sugeno FIS

Sugeno FIS is similar to the Mamdani method in fuzzifying the inputs and applying the fuzzy operator. However, the output membership functions in Sugeno FIS are either linear or constant. Table 1 shows a typical rule in a Sugeno fuzzy model.

Table 1: Form of a typical rule in Sugeno FIS.

Input 1	Input 2	Output
A	B	$c_1A + c_2B + c_3$. Where, c_1, c_2, and c_3 are constants. For zero order FIS: $c_1 = c_2 = 0$.

3. Robot Navigation Model in Simulink

The robot navigation model is presented using MATLAB-Simulink software. This robot navigation model communicates with the robot through robot operating system (ROS). Using publisher-subscriber strategy, the Simulink model and the robot communicate with each other by passing specified messages on the relevant ROS topics. A publisher of the message sends the message on ROS topic. On the other side, the subscriber receives the message on the same topic from the ROS. In this model we are using two controllers: pure pursuit controller and fuzzy based controller. The pure pursuit controller directs the robot from the current location to goal location without considering the obstacles in the path. In addition, the fuzzy based controller is used for the obstacle avoidance.

Fig. 2 describes the details of the robot navigation model. This Simulink model contains four subsystems namely: *Subscribing topic: '/odom'*, *Subscribing topic: '/scan'*, *Goal Distance Checking*, and *Publishing topic: 'mobile_base/commands/velocity'*. The robot publishes its position and orientation, in the environment, on the */odom* topic of the ROS. The *Subscribing topic: '/odom'* subsystem, in the given model, subscribes the */odom* topic of the ROS to obtain current pose of the robot. The robot publishes its laser sensor scan readings vector on */scan* topic in ROS. A typical message vector on */scan* topic contains distance ranges of obstacles in the scan area. In two dimensional state, the scan area is of 'V' shape. From right to left, a single laser scan may contain some hundred ranges. Using minimum

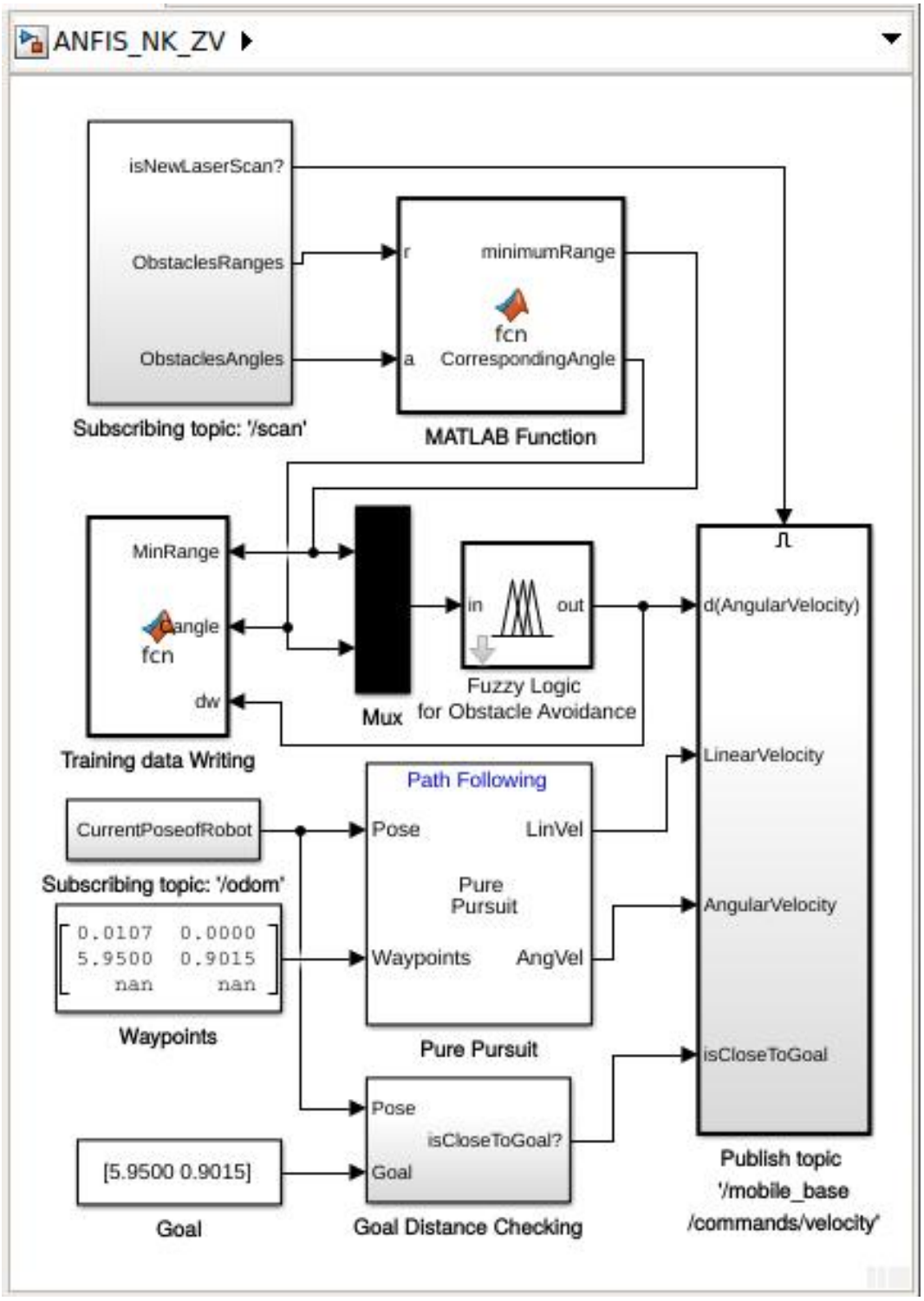


Figure 2: Robot navigation model in Simulink.

angle of scan and angle increment, the corresponding angle to each range in a message can be find out. The obstacle ranges and their corresponding angles of scan, together, give the locations of the obstacles in the robot path. The *Subscribing topic: '/scan'* subsystem is subscribing the ROS topic */scan* to get the vectors of ranges and their corresponding scan angle.

The goal distance (p) between the goal position (x,y) and current position (x',y') of the robot can be expressed by (5) as given below:

$$p = \sqrt{(x-x')^2+(y-y')^2}. \tag{5}$$

Before sending the new velocity commands to the robot, the *Goal Distance Checking* subsystem checks the distance of goal from the current position of the robot. If the distance between the goal and the robot is less than or equal to the given value (in our case, it is 0.1 meter) then the new velocity commands will not be given to robot. As a result, the robot navigation process gets completed and, therefore, terminated. The linear and angular velocities commands are given to the robot by publishing *geometry_msgs/Twist* message on the topic */mobile_base/commands/velocity* of ROS. Robot is subscribing the topic */mobile_base/commands/velocity* of ROS which is being published by the subsystem *Publishing topic: '/mobile_base/commands/velocity'*.

In addition to these subsystems, there are two constant blocks named

as *Waypoints* and *Goal*, a *Mux* block, a *Pure Pursuit* path following algorithm block, two MATLAB functions and a *fuzzy logic for obstacle avoidance* block. The *Waypoints* constant block is to contain a *Nby2* matrix of way points of navigation path . In case of unknown environment, we are considering that only start and goal positions of the robot are known, initially. The *Goal* constant block contains a two dimensional coordinates of goal position. The *Mux* block is combining the two inputs (*minimumRange* and *CorrespondingAngle* received from the *MATLAB Function* block) in a single output link. So that, these two input values can be passed to the fuzzy controller block by a single link. The *Pure Pursuit* block is used to compute the required linear and angular velocities to drive the robot using inputs from way-points of the path and current pose of the robot. Since the navigation environment is considered as unknown, the way-points of the feasible path are unknown except the two points: start and goal. The function *fcn* in *MATLAB Function* block computes the minimum range and its corresponding angle of scan from the vectors of *ObstaclesRanges* and *ObstaclesAngles* received from *Subscribing topic: '/scan'* subsystem block. The MATLAB function in the *Training data writing* block is used to write the data in three separate columns for minimum range, its corresponding angle and the output from the fuzzy controller. This training data can be used to train a FIS using ANFIS model. The block named *Fuzzy Logic for Obstacle Avoidance* is a fuzzy logic controller block.
The published linear velocity is same as linear velocity given by the pure pursuit block. Since the pure pursuit block does not consider the obstacles in the path, the angular velocity is needed to be adjusted before its publication on the ROS. This required adjustment in the angular velocity is obtained by the *Fuzzy Logic for Obstacle Avoidance* block. The published angular velocity (ω) can be stated by (6) as below:

$$\omega = \omega_1 + \omega_2. \tag{6}$$

where, ω_1 is the angular velocity (i.e. *AngVel*) provided by path pursuit controller. ω_2 is the output (i.e. *d(AngularVelocity)*) of the fuzzy logic controller.
The FIS of the used fuzzy logic controller is considered as of the following two types:

1) Mamdani-type FIS
2) ANFIS trained Sugeno-type FIS

The steps to use a FIS in the Simulink model are as follows:

1) Export the FIS to the MATLAB workspace with a name (e.g. *NK_ZV_Sugeno*).
2) In the block parameters of the fuzzy logic controller in the Simulink model, specify the FIS name as taken in step 1 (i.e. *NK_ZV_Sugeno*).

4. Proposed FIS for the Fuzzy Controller

For the fuzzy controller, the Mamdani and Sugeno-type FIS are evolved in subsections 4.1 and 4.2, respectively.

4.1. Mamdani-type FIS

Fuzzy controller used in the robot navigation model of the section 3 outputs the required change in angular velocity of robot to avoid the obstacles in the path. The fuzzy logic designer of the Mamdani-type FIS is given in Fig. 3. The input variables *minR* and *A(minR)* are the minimum scan range and corresponding angle of the scan, respectively. Using *minR*, it can be find out that whether the obstacle is close enough for its avoidance. In addition, the *A(minR)* variable decides the location side of the closest obstacle in the laser scan area. The controller is using the strategy that if the obstacle is at right side of the robot's scan area then turn the robot left. Similarly, if the

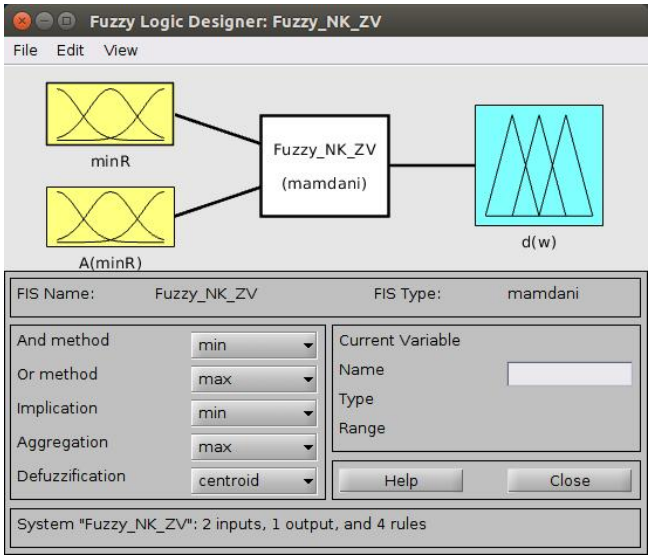


Figure 3: Fuzzy logic designer of Mamdani-type FIS.

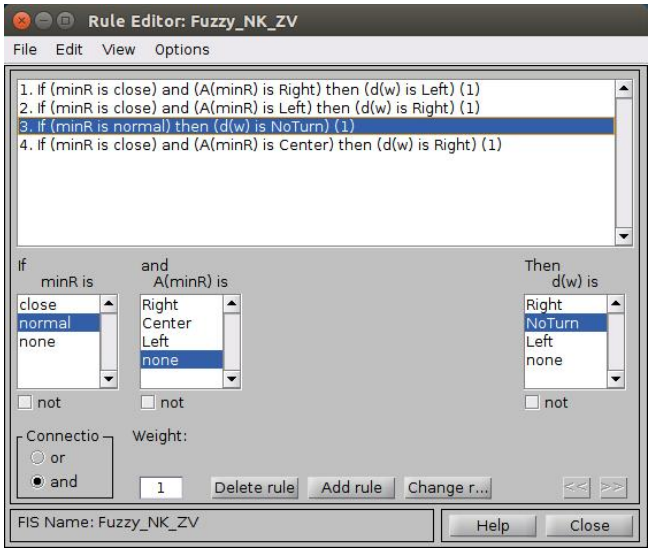


Figure 4: Rule base of the Mamdani-type FIS.

obstacle is at left side then turn the robot to the right. Fig. 4 presents the rule base used in the controller. It can be noted from Fig. 4 that the number of rules taken in the Mamdani-type FIS is 4. There are two membership functions (namely: *close* and *normal*) for the input variable *minR* and three membership functions (i.e. *Right*, *Center*, and *Left*) for the input variable *A(minR)*. For the only output variable *d(w)*, three membership functions, *Right, noTurn, and Left*, are used.

4.2. Sugeno-type FIS

The training data, received from the execution of the navigation model with Mamdani-type FIS in its fuzzy controller, are inputed to ANFIS to find the Sugeno-type FIS. Fig. 5 depicts the plot between the training data and FIS output. The structure of ANFIS model is presented in the Fig. 6. After training, the number of rules and output membership functions are six each. It is observed that for the given number of input variables, output variable(s) and input membership functions for each input variable, the ANFIS has generated input membership functions, rules and output membership functions. Fig. 7–8 presents the resultant input membership functions. The input variables, *input1* and *input2*, are auto generated names after

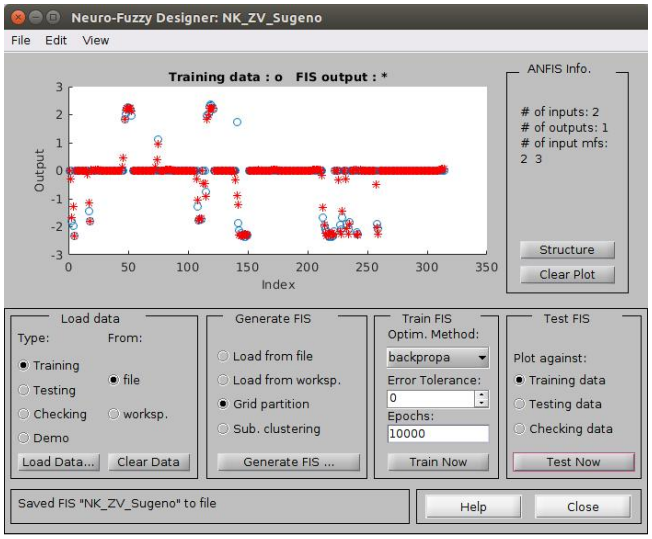


Figure 5: Training data and FIS output in neuro-fuzzy designer.

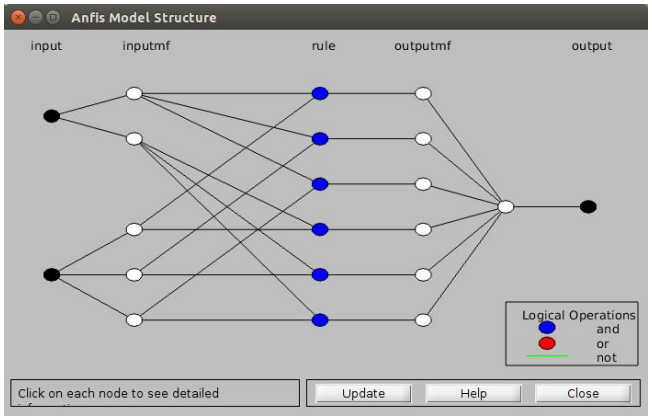


Figure 6: ANFIS model structure.

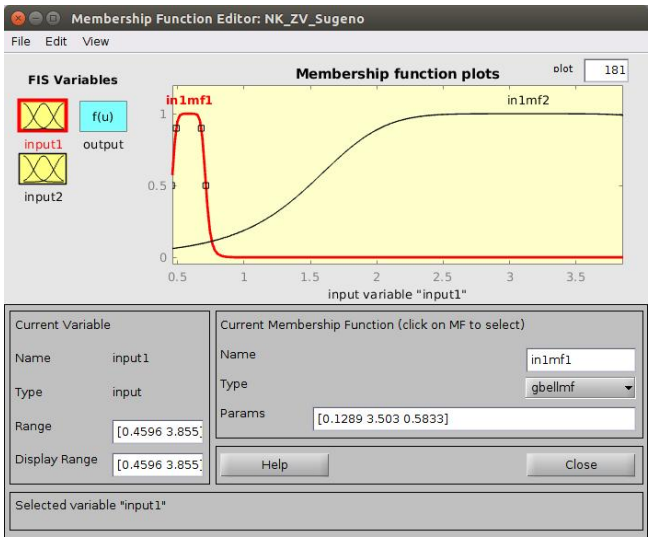


Figure 7: Membership function plots for *input1* in Sugeno-type FIS.

the training through ANFIS and correspond to the input variables *minR* and *A(minR)* of Mamdani-type FIS given in Fig. 4, respectively. Similarly, the output variable *output* is corresponding to the output variable *d(w)* of the Mamdani-type FIS. The membership functions of the only output variable of the produced Sugeno-type FIS, are

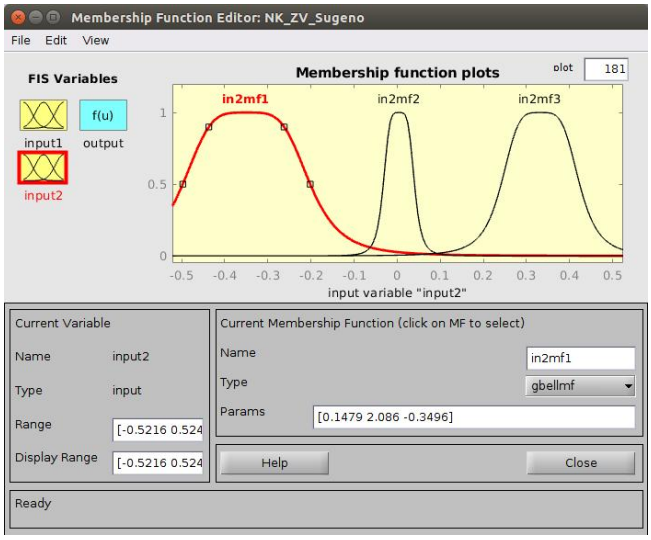


Figure 8: Membership function plots for *input2* in Sugeno-type FIS.

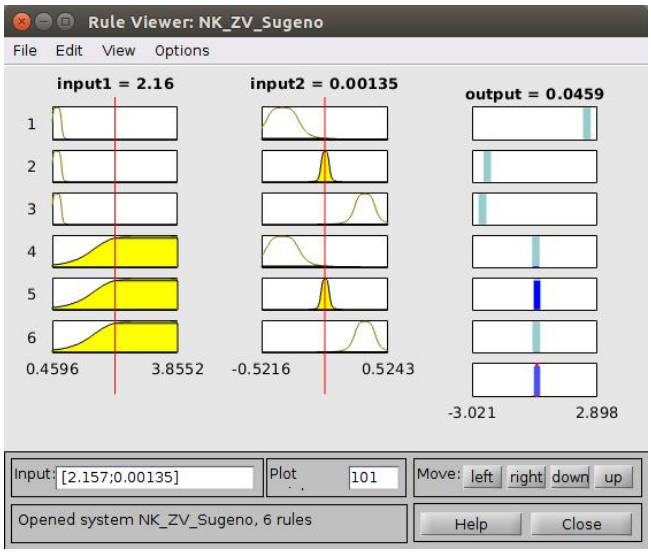


Figure 9: Rule Viewer of the Sugeno-type FIS.

constants. Therefore, it can be find out from the output column of Table 1 that this Sugeno-type FIS is of zero order.
Similarly, the output variable *output* is corresponding to the output variable *d(w)* of the Mamdani-type FIS. The membership functions of the only output variable of the produced Sugeno-type FIS, are constants. Therefore, it can be find out from the output column of Table 1 that this Sugeno-type FIS is of zero order.
The rule viewer of the received Sugeno-type FIS is presented in Fig. 9. The rules in the Fig. 9 are auto generated rules after the successful training from the ANFIS model. It can be noted that the number of rules of this Sugeno-type FIS is different from the number of rules used in the Mamdani-type FIS given in subsection 4.1.

5. Experimental Set-Up and Results

The robot navigation model is implemented on a general purpose laptop with the processor configuration as *4x Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz* and 3952MB of memory. The operating system is considered as *Ubuntu 14.04.5 LTS*. *Turtlebot* robot model in *Gazebo* multi-robot simulator, version 2.2.6, and *Indigo* 1.11.21 robot operating system are taken. For Simulink modeling, the MATLAB version 9.1.0.441655 (R2016b) is used.

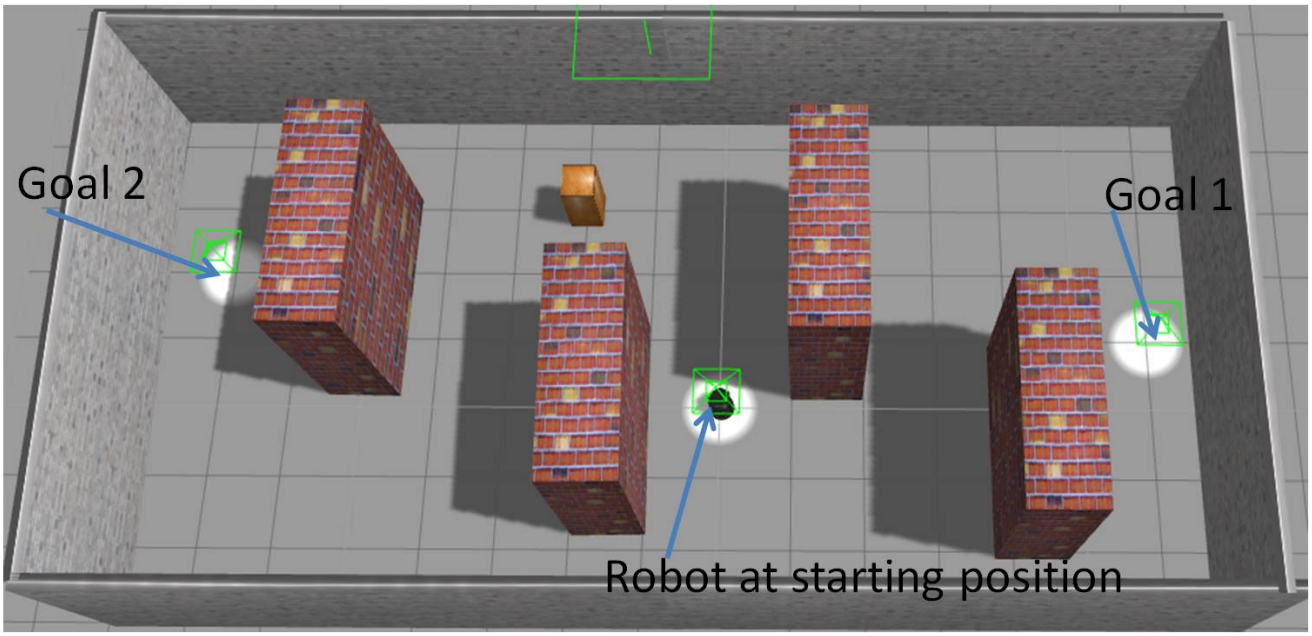


Figure 10: Simulated world in Gazebo simulator.

5.1. Implementation in Gazebo Simulator

The robot navigation model (as given in Section 3) is tested on the simulated world shown by Fig. 10. This simulated world has been constructed by using the Gazebo simulator.

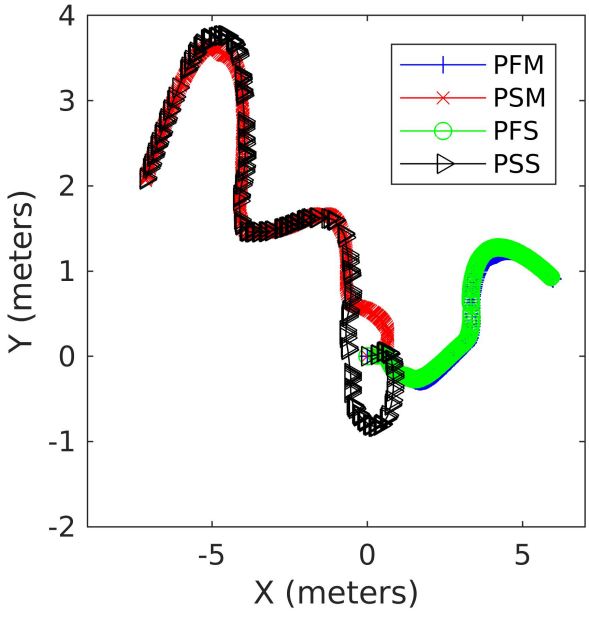


Figure 11: Paths followed by the robot using Mamdani and Sugeno-type FISs in fuzzy controller.

In the simulated world of Fig. 10, three spot lights are used to show one starting and two goal positions. Using the robot navigation model, presented in Fig. 2, the simulated Turtlebot robot navigates from the starting position to each goal positions (i.e. goal 1 and goal 2), successfully. During its navigation from start to goal 1 or goal 2, the robot avoids the obstacles present in each of the path. In our case, the two dimensional coordinates of the start, goal 1 and goal 2 positions are given as (0.0007, 0.0000), (5.9500, 0.9015) and (7.1586, 2.0189), respectively. In addition, at start, the robot is heading towards positive X axis direction. In the navigation model (presented by Fig. 2), the fuzzy logic controller block provides the required change in the angular velocity for obstacle avoidance. Consequently, the robot avoids the obstacles encountered in the path. The position coordinates of the robot, in the navigation environment, can be received by subscribing the ROS topic */odom*. The (X,Y) coordinate positions of the robot, in the Gazebo world (as taken in Fig. 10), are plotted in Fig. 11 to depict the paths followed by the robot using Mamdani and Sugeno-type FISs in the fuzzy controller. Abbreviations of the acronyms used in the legend of the Fig. 11 are given in Table 2.

Table 2: Abbreviations of the acronyms in the legend of the Fig. 11.

Abbreviations	Acronyms
Path to first goal using Mamdani-type FIS	PFM
Path to second goal using Mamdani-type FIS	PSM
Path to first goal using Sugeno-type FIS	PFS
Path to second goal using Sugeno-type FIS	PSS

In case of Mamdani-type FIS, the name of FIS is specified as *Fuzzy_NK_ZV* (the Mamdani-type FIS as taken in subsection 4.1) in block parameters of the fuzzy logic controller of the navigation model in Fig. 2. On the other hand, in case of Sugeno-type FIS, the name of FIS in block parameters of the fuzzy logic controller, of the navigation model, is taken as *NK_ZV_Sugeno* (the Sugeno-type FIS obtained from ANFIS model as taken in subsection 4.2).

5.2. Implementation on Real Robot

The proposed model for the robot navigation is implemented on real Turtlebot robot equipped with Microsoft *Kinect XBOX 360* sensor. The robot is navigated from starting position to the goal position in the environment presented by Fig. 12. The linear velocity of the robot is taken as 0.5 *meter per second*. It is evident from Fig. 12 that the robot has to avoid obstacles on both of its sides during its drive from start to goal. In addition, some of the portion of the corridor is fenced by the iron railing instead of solid brick wall fence. Therefore, this real environment is an excellent case for testing the proposed



Figure 12: Real world environment for the real robot.

navigation model and FISs.

The Sugeno-type FIS, as obtained in subsection 4.2, is used as the FIS of the fuzzy controller block of the robot navigation model.

The path followed by the robot is shown by the Fig. 13. It is clear from the Fig. 13 that the robot takes turn whenever the fence wall or railing comes in the path, otherwise, the robot directly navigates towards the goal.

6. Conclusion

The training data obtained from the Mamdani-type FIS is quit suitable for ANFIS model. It is observed that the ranges and types of input membership functions of Sugeno-type FIS can successfully obtained using ANFIS model. Further, the number of required rules and the number of output membership functions generated from the ANFIS model are differ from the Mamdani-type FIS. However, the resultant Sugeno-type FIS gives successful obstacle avoidance during robot navigation and it can be noted from the results that the robot follows the same paths using Mamdani and Sugeno FIS except few situations. The FIS generated through the simulator are capable of obstacle avoidance in real world environment for the real robot. For better results, it has been observed through rigorous experimental work that the linear velocity of the real robot should be close to the linear velocity taken for the simulated robot.

Except starting and goal positions, the navigation model does not require any other prior information about the navigation environment. Therefore, in addition to the static obstacles avoidance, the avoidance for dynamic obstacles is achieved.

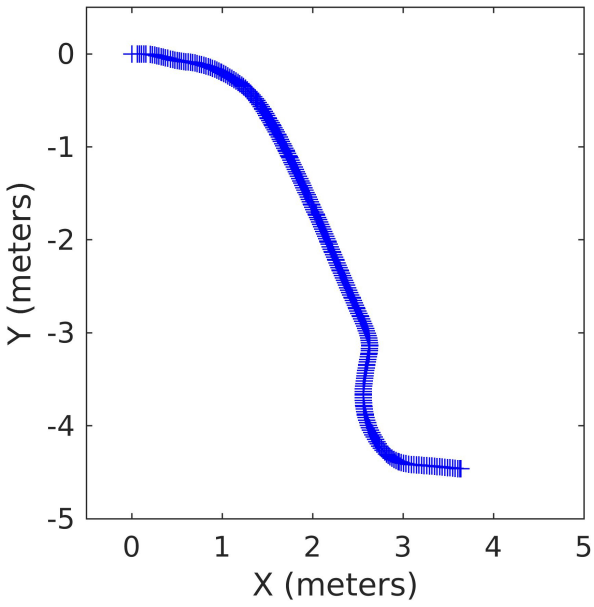


Figure 13: Path followed by the real robot.

Acknowledgements

This work is a part of the undergoing *Teki control* research project of John von Neumann Faculty of Informatics, Óbuda University,

Budapest, Hungary. The authors are highly thankful to the Doctoral School of Applied Informatics and Applied Mathematics Óbuda University, Budapest, Hungary, for the financial support in this study.

References

[1] A. Narayan, E. Tuci, F. Labrosse, and M. H. M. Alkilabi, "A dynamic colour perception system for autonomous robot navigation on unmarked roads," *Neurocomputing*, vol. 275, pp. 2251 – 2263, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217317228

[2] N. Kumar and Z. Vámossy, "Obstacle recognition and avoidance during robot navigation in unknown environment," *International Journal of Engineering & Technology*, vol. 7, no. 3, pp. 1400 – 1404, 2018. [Online]. Available: https://www.sciencepubco.com/index.php/ijet/article/view/13926

[3] G. Kertész, S. Szénási, and Z. Vámossy, "Multi-directional image projections with fixed resolution for object matching," *Acta Polytechnica Hungarica*, vol. 15, no. 2, pp. 211 – 229, 2018. [Online]. Available: https://www.uni-obuda.hu/journal/Kertesz_Szenasi_Vamossy_81.pdf

[4] I. Arvanitakis, A. Tzes, and K. Giannousakis, "Mobile robot navigation under pose uncertainty in unknown environments." *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 710 – 12 714, 2017, 20th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317330628

[5] I. Rodríguez-Fdez, M. Mucientes, and A. Bugarín, "Learning fuzzy controllers in mobile robotics with embedded preprocessing," *Applied Soft Computing*, vol. 26, pp. 123 – 142, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494614004748

[6] N. H. Singh and K. Thongam, "Mobile robot navigation using fuzzy logic in static environments," *Procedia Computer Science*, vol. 125, pp. 11 – 17, 2018, the 6th International Conference on Smart Computing and Communications. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917327680

[7] T. T. Mac, C. Copot, D. T. Tran, and R. D. Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13 – 28, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889015300671

[8] M. Algabri, H. Mathkour, H. Ramdane, and M. Alsulaiman, "Comparative study of soft computing techniques for mobile robot navigation in an unknown environment," *Computers in Human Behavior*, vol. 50, pp. 42 – 56, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0747563215002605

[9] M. S. Masmoudi, N. Krichen, M. Masmoudi, and N. Derbel, "Fuzzy logic controllers design for omnidirectional mobile robot navigation," *Applied Soft Computing*, vol. 49, pp. 901 – 919, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494616304598

[10] N. Kumar, M. Takács, and Z. Vámossy, "Robot navigation in unknown environment using fuzzy logic," in *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Jan 2017, pp. 279 – 284. [Online]. Available: https://ieeexplore.ieee.org/document/7880317/

[11] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, January 1992. [Online]. Available: https://www.ri.cmu.edu/publications/implementation-of-the-pure-pursuit-path-tracking-algorithm/