ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

Neuro-Fuzzy Optimization of Navigation Time for a Differential-Drive Mobile Robot Controlled by a Fuzzy Controller

Ralaivao Harinaivo Hajasoa[1]; Mahera Salala Mandimby Hasina[2]; Herinantenaina Edmond Fils[3]

[1]Polytechnic University of Antananarivo, Madagascar
[2]Polytechnic University of Antananarivo, Madagascar
[3]Polytechnic University of Antananarivo, Madagascar

Publication Date: 2025/10/10

Abstract: This paper presents a comparative study of two control strategies: fuzzy logic and adaptive neuro-fuzzy inference system (ANFIS) for autonomous guidance of a differential-drive mobile robot. The robot executes goal-seeking and reactive obstacle-avoidance tasks in a MATLAB Simulink environment using the Mobile Robotics Simulation Toolbox. Initially, a fuzzy logic controller with expert-defined IF–THEN rules generates linear and angular velocity commands while logging state and control data into a training matrix. These data are then used to train an ANFIS model, which is redeployed under identical simulation conditions. Both controllers are compared based on path-tracking accuracy, obstacle-avoidance robustness, and control-loop execution time. Simulation results indicate that the ANFIS controller reproduces the fuzzy logic decision boundaries with reduced computational latency, demonstrating the effectiveness of data-driven neuro-fuzzy models for real-time mobile robot control.

Keywords: *Fuzzy Logic Control, ANFIS, Differential Drive, Target Reaching, Obstacle Avoiding.*

How to Cite: Ralaivao Harinaivo Hajasoa; Mahera Salala Mandimby Hasina; Herinantenaina Edmond Fils (2025) Neuro-Fuzzy Optimization of Navigation Time for a Differential-Drive Mobile Robot Controlled by a Fuzzy Controller. *International Journal of Innovative Science and Research Technology*, 10(9), 2864-2882. https://doi.org/10.38124/ijisrt/25sep1549

I. INTRODUCTION

The domain of mobile robotics encompasses the development of autonomous systems designed to operate independently in unstructured and dynamic settings. These platforms must navigate through environments filled with uncertainties without requiring direct human oversight. These systems integrate perception, localization [3][7][8][12], planning, and control algorithms to achieve tasks such as path following, obstacle avoidance, and target localization. In practice, mobile robots must cope with imperfect sensors, unmodeled dynamics, and changing terrain, which demand flexible and robust control strategies. In this study, we concentrate on a differential-drive robot architecture, a widely adopted configuration in which two independently actuated wheels provide both linear and angular motion. The following sections detail our modeling framework and parameter identification for this differential drive robot.

II. ROBOT MODELING

In this section, we present the kinematic description of our differential-drive platform. We first identify the key geometric and dynamic parameters (wheel velocities, track width), then derive the continuous-time kinematic model, and finally discuss our implementation in MATLAB.

➢ *Identification of Robot Parameters*

For kinematic modeling, essential parameters of the differential-drive robot are identified. These include the left and right wheel linear velocities (vL and vR), the robot's forward velocity (V), its angular velocity (ω), and the instantaneous center of curvature (ICC) position. Figure 1 illustrates the robot geometry and the ICC location relative to the wheel axis. From the wheel velocities and the track width b, the robot's linear and angular velocities are computed as [3][8]:

$$V = \frac{v_R + v_L}{2}, \quad \omega = \frac{v_R - v_L}{b} \tag{1}$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

These relations form the basis for the continuous-time motion model.

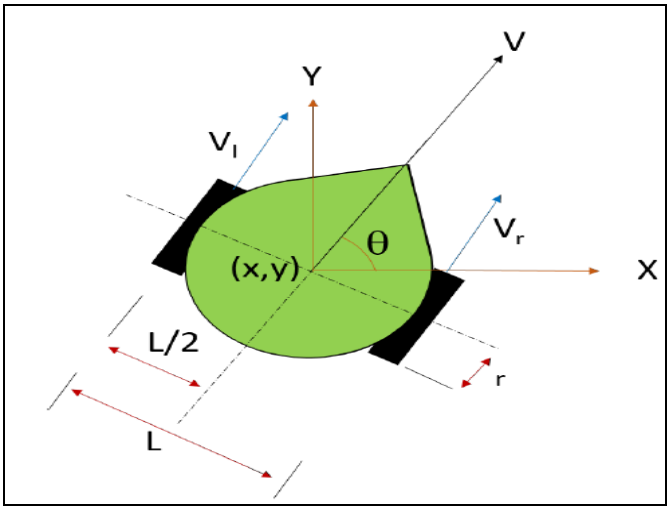


Fig 1 Geometry of the Differential-Drive Robot

➢ *Continuous-Time Kinematic Model*

Rather than a discrete rigid-body update, the robot's pose evolves according to the differential equations[7][12]:

$$\dot{x}(t) = V(t)\cos\theta(t) \tag{2}$$

$$\dot{y}(t) = V(t)\sin\theta(t) \tag{3}$$

$$\dot{\theta}(t) = \omega(t) \tag{4}$$

Integrating these expressions over time yields the robot's trajectory (x(t), y(t), θ(t)).

$$P_x(t) = x(t) = \int V\sin\theta\,dt, \tag{5}$$

$$P_y(t) = y(t) = \int V\cos\theta\,dt, \tag{6}$$

$$\theta(t) = \int \omega\,dt. \tag{7}$$

➢ *MATLAB and the Mobile Robotics Simulation Toolbox*

The modeling and simulation framework is implemented in MATLAB using the Mobile Robotics Simulation Toolbox. This toolbox provides preconfigured robot models, environment definitions, and Simulink blocks for rapid prototyping. Figure 2 shows the differential-drive robot model loaded in the toolbox, along with the virtual environment and the corresponding Simulink diagram used for closed-loop simulation. The toolbox's built-in functions facilitate sensor emulation, path planning, and control loop integration.

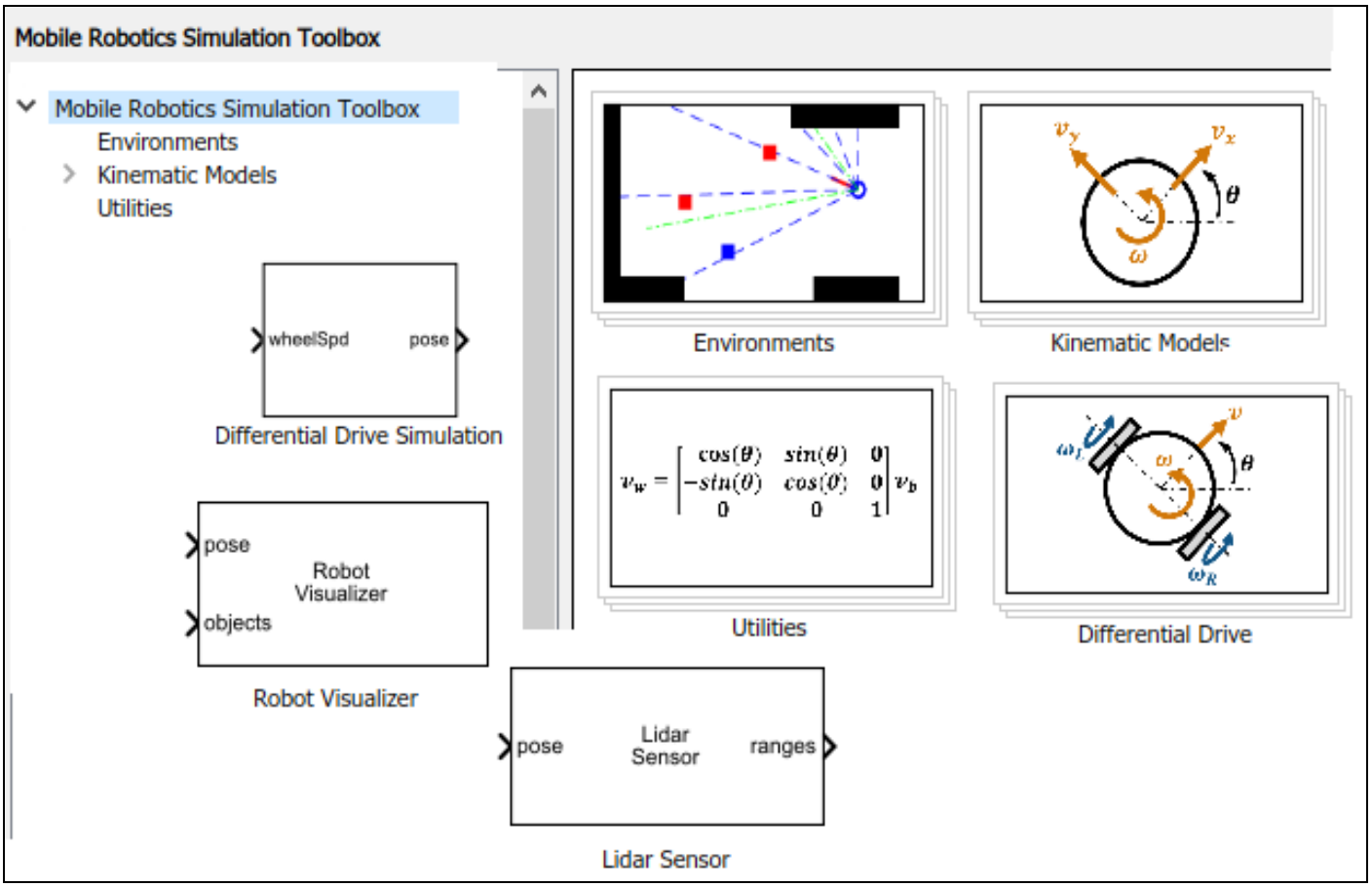


Fig 2 Mobile Robotics Simulation Toolbox: All the Necessary Blocks

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

III. FUZZY LOGIC

First proposed by Lotfi Zadeh in 1965, fuzzy logic extends binary logic by permitting gradual transitions between truth values, thereby handling concepts like 'partially true' or 'moderately high' that are inherent in human reasoning[13]. It is particularly effective in managing uncertainty and imprecision, making it a valuable tool in control systems, decision-making, and artificial intelligence[6][13].

➢ *Fuzzy Set Theory*

Basic Principle A fuzzy set is characterized by a membership function that assigns to each element a grade of membership ranging between 0 and 1. This allows elements to partially belong to a set, unlike classical (crisp) sets5. Membership Function Membership functions define how each point in the input space is mapped to a membership value. The most commonly used shapes include triangular, trapezoidal, and Gaussian. Defining Membership Functions Membership functions can be defined empirically or based on expert knowledge, often represented by linguistic variables such as "small," "medium," and "large"[6].

- *Fuzzy Set Operations*

Fuzzy Logic Operators Operations like AND, OR, and NOT are generalized using T-norms and S-norms:

✓ Fuzzy AND (T-norm): min(μA(x), μB(x))
✓ Fuzzy OR (S-norm): max(μA(x), μB(x))
✓ Fuzzy NOT: 1 − μA(x)

Triangular Norms and Co-norms Triangular norms (T-norms) generalize logical conjunctions, and co-norms generalize disjunctions. Examples include minimum, product, and bounded difference4.

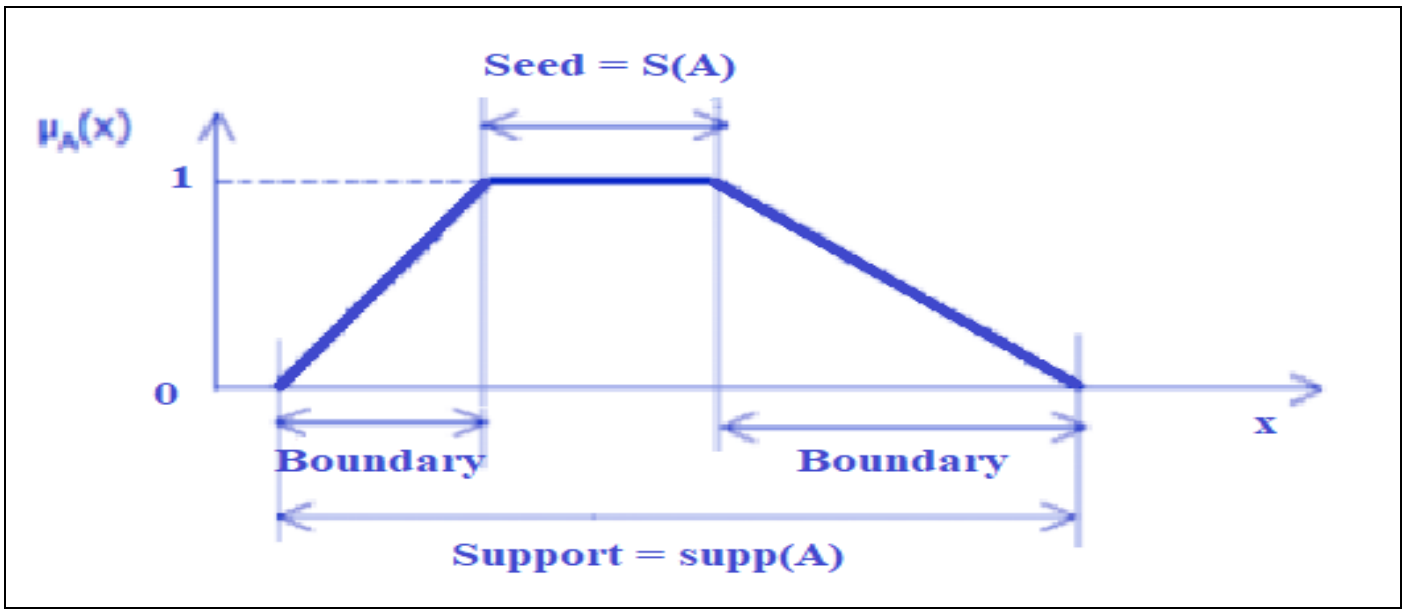


Fig 3 Fuzzy Set Showing Core, Boundary and Support

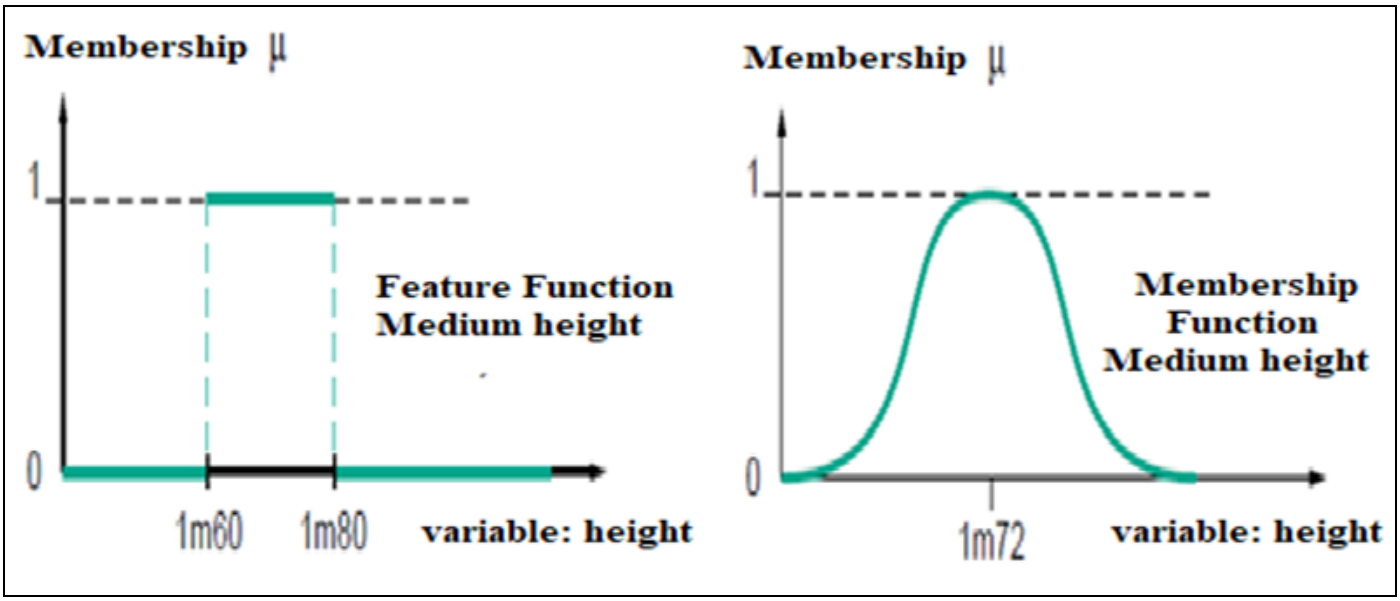


Fig 4 Comparison Between Boolean and Fuzzy Membership Functions for a Person's Height

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

Fuzzy Relations Fuzzy relations extend binary relations using fuzzy sets and can be used in composition and implication rules[5].

Fuzzy Rules (If–Then) Fuzzy rules use linguistic terms to model reasoning processes:

IF temperature is high THEN fan speed is fast.

The generalized Modus Ponens is used to apply these rules even with partial matches[14].

➢ *Fuzzy Inference Systems (FIS)*

Fuzzification This step converts crisp input values into degrees of membership for fuzzy sets[6].

Fuzzy Inference Inference applies fuzzy logic rules to determine the output fuzzy sets.

Implication and Activation Degree The implication step modifies the output set based on the rule's antecedent matching strength[4].

Condition Aggregation Multiple conditions in a rule are combined using fuzzy logic operators.

Defuzzification The final step converts the resulting fuzzy output into a single crisp value. Common methods include the centroid method and mean of maxima[5][6].

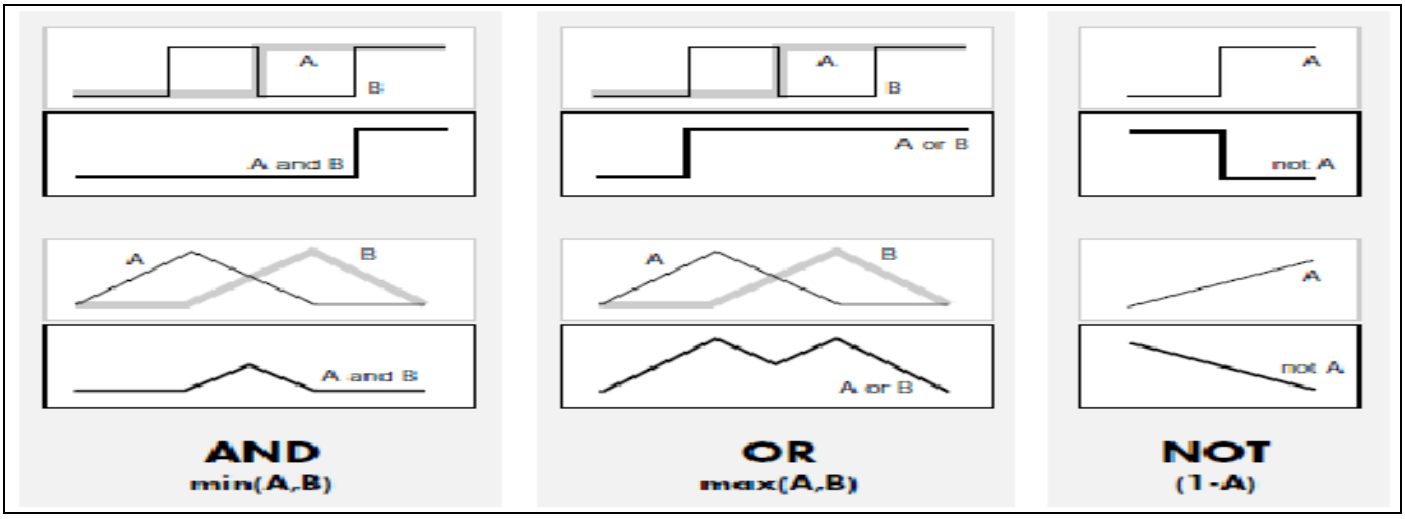


Fig 5 Comparison of AND, OR and NOT Operations in Boolean and Fuzzy Logic

➢ *MATLAB Fuzzy Logic Toolbox*

The MATLAB Fuzzy Logic Toolbox provides a graphical and programmatic environment for designing, simulating, and analyzing fuzzy inference systems. It includes a FIS Editor for defining membership functions and rule bases, visualization tools for performance evaluation, and integration with Simulink for closed-loop control[11].

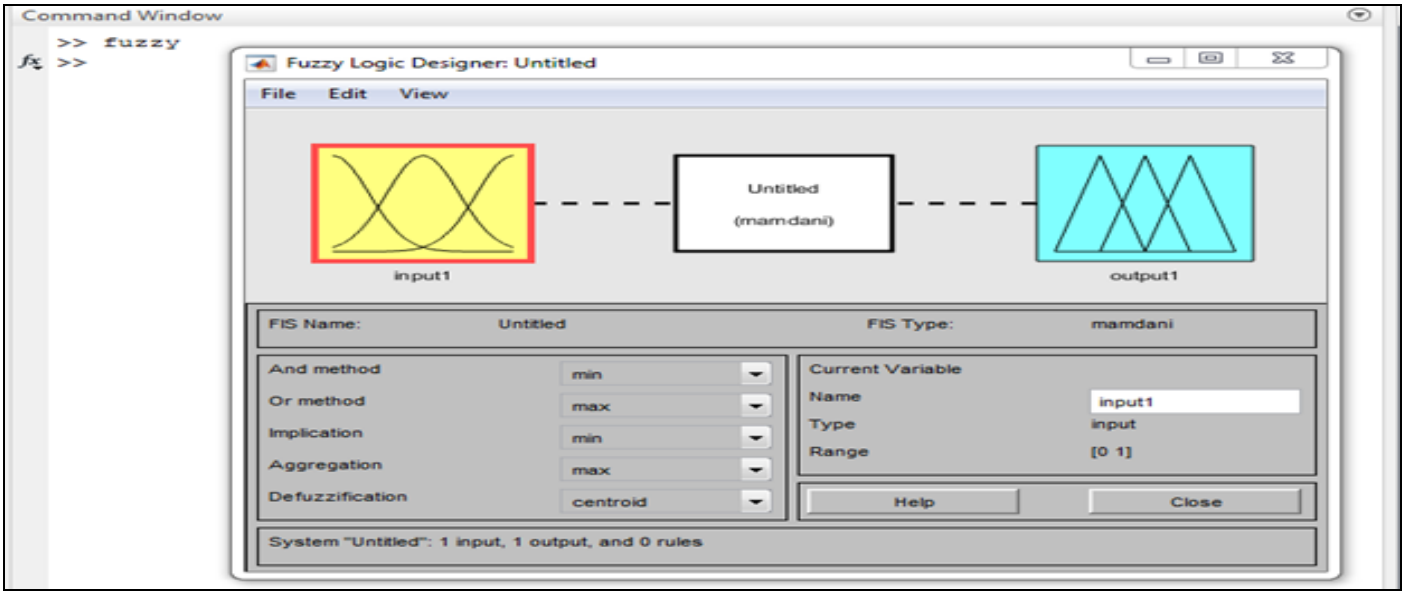


Fig 6 FIS Editor Workspace Showing Membership Functions and Rule View

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

Figure 6 shows the Toolbox welcome interface, where users can access the FIS Editor for defining membership functions and fuzzy rules. In this work, we focus specifically on triangular membership functions due to their intuitive interpretation and low computational complexity.

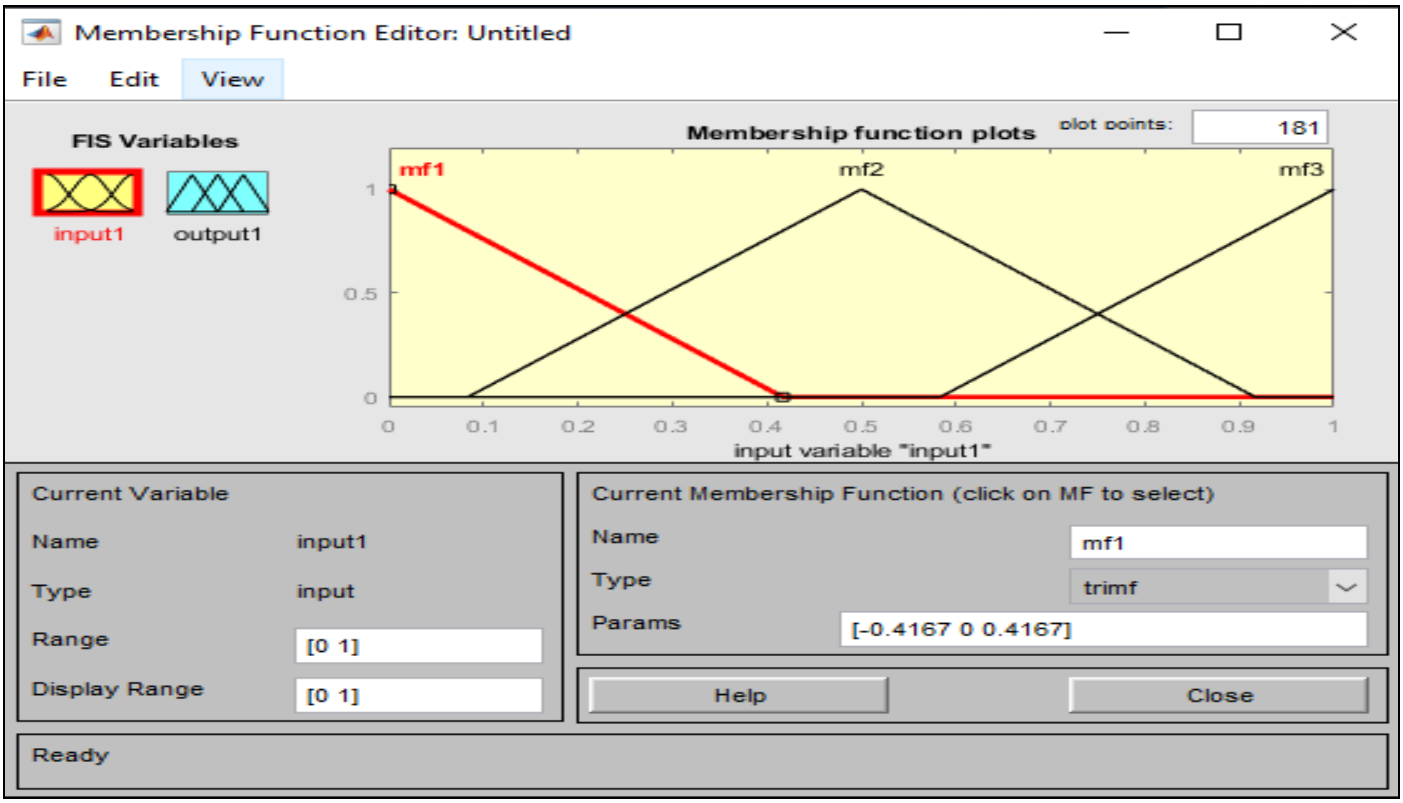


Fig 7 Membership Function Editor with Triangular and Trapezoidal Shapes

Figure 7 illustrates the membership function editor, which supports defining both input and output functions. For example, obstacle sensor readings are categorized into linguistic terms such as near, quite near, and far, each modeled by a triangular function.

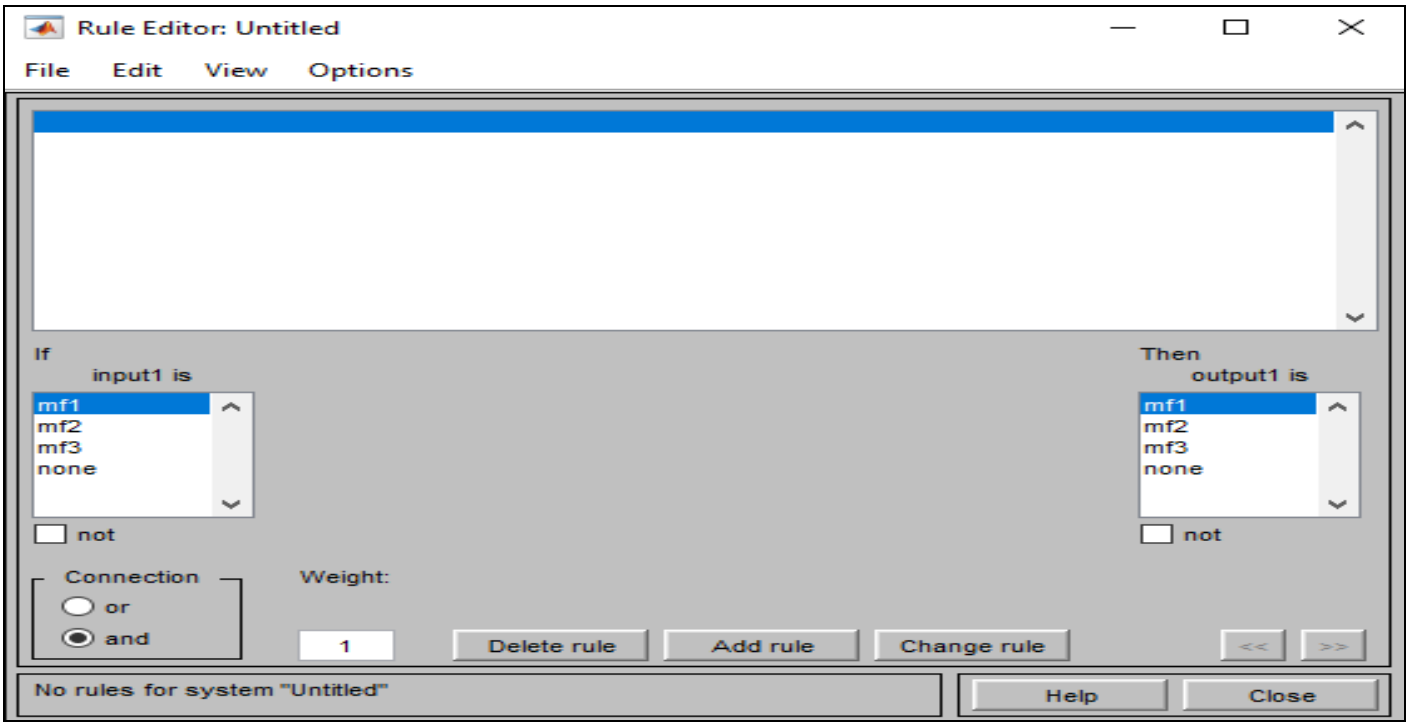


Fig 8 Rule Viewer and Surface Plot for Two input FIS Demonstrating Rule Response

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

Figure 8 presents the fuzzy rule interface, where control logic is encoded as IF–THEN statements. In our controller design, rules follow patterns like "IF distance is near AND bearing error is large THEN rotate in place," enabling human-like decision making.

IV. ANFIS MODELING AND TRAINING

ANFIS architectures combine the transparent, rule-based reasoning of fuzzy systems with the learning capabilities of neural networks, enabling the model to adapt and refine its parameters from data[2]. Two common initialization strategies are detailed below.

➢ *Grid Partition vs. Subtractive Clustering*

- Grid Partition: Uniformly divides each input domain into fuzzy sets, producing a full rule base by Cartesian product. Suitable for few inputs but scales poorly as rule count grows exponentially[2].
- Subtractive Clustering: Detects data clusters based on density in the input space, creating rules where data are dense. This yields a compact, data-driven rule set[1].

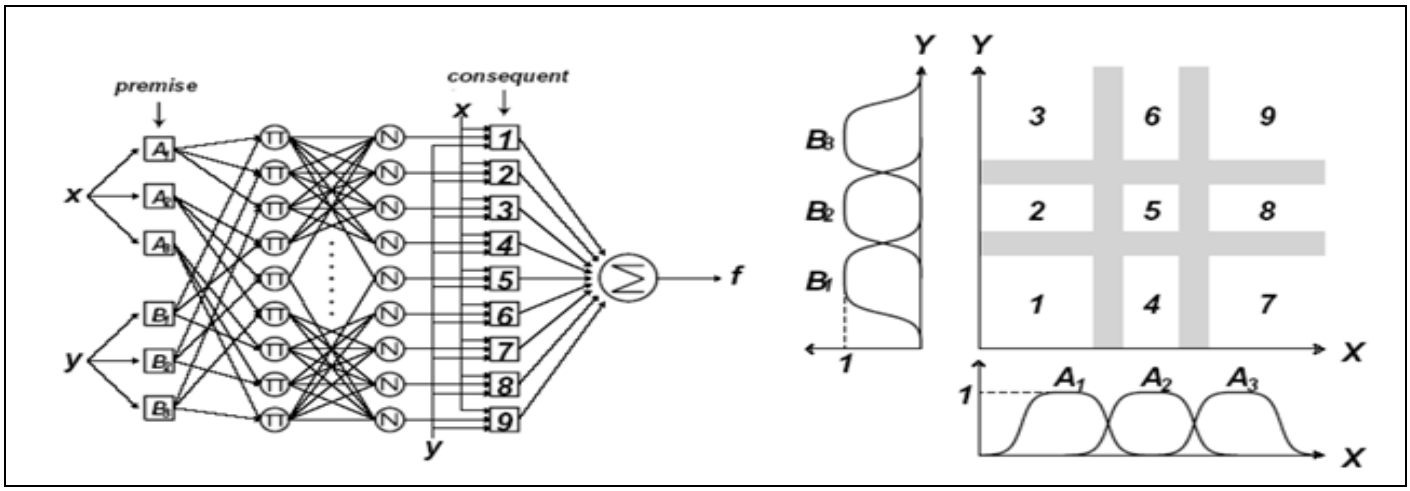


Fig 9 ANFIS Model Initialized via Grid Partition

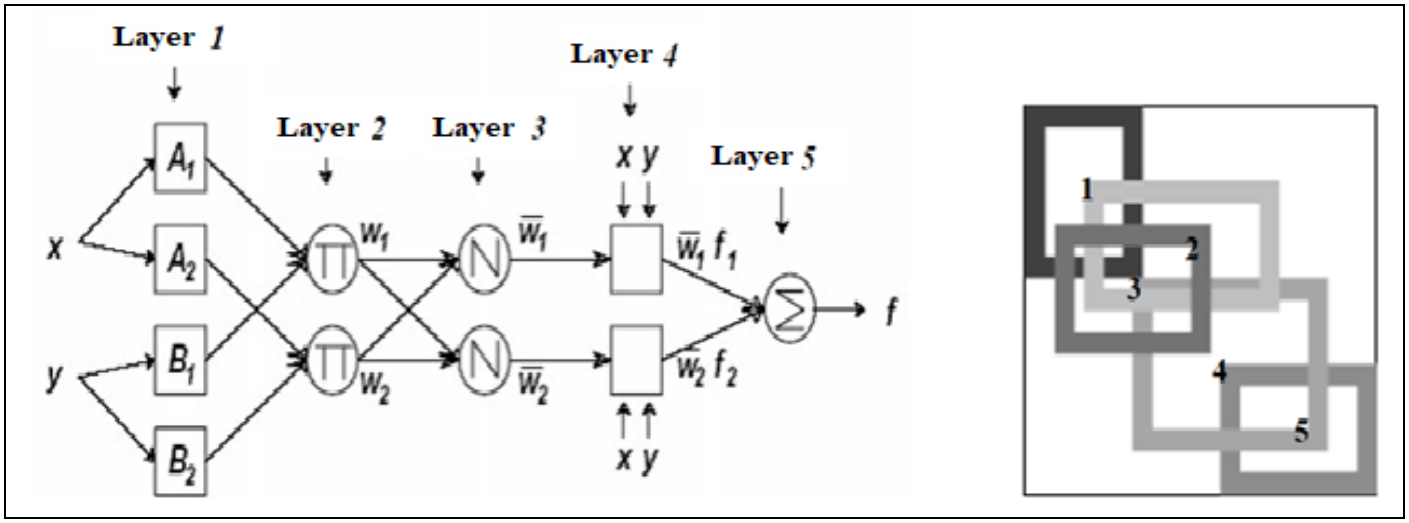


Fig 10 ANFIS Model Initialized via Substractive Clustering

➢ *Neuro-Fuzzy Designer in MATLAB*

The MATLAB Neuro-Fuzzy Designer app streamlines ANFIS model creation. After importing the collected data, users select Grid or scatter initialization, define membership functions, and configure training parameters while monitoring training error and epoch convergence[10].

➢ *Training Data Collection*

We deployed the fuzzy logic controller in Simulink to gather a dataset comprising:

- Wheel velocities (vL, vR) and resulting linear velocity (V).
- Robot pose (x, y, θ) from odometry.
- Proximity sensor measurements (three sensors) for obstacle distance.

This input–output matrix serves as the ANFIS training set. Once trained, ANFIS and fuzzy controllers are compared on target acquisition time and obstacle-avoidance robustness[9]

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

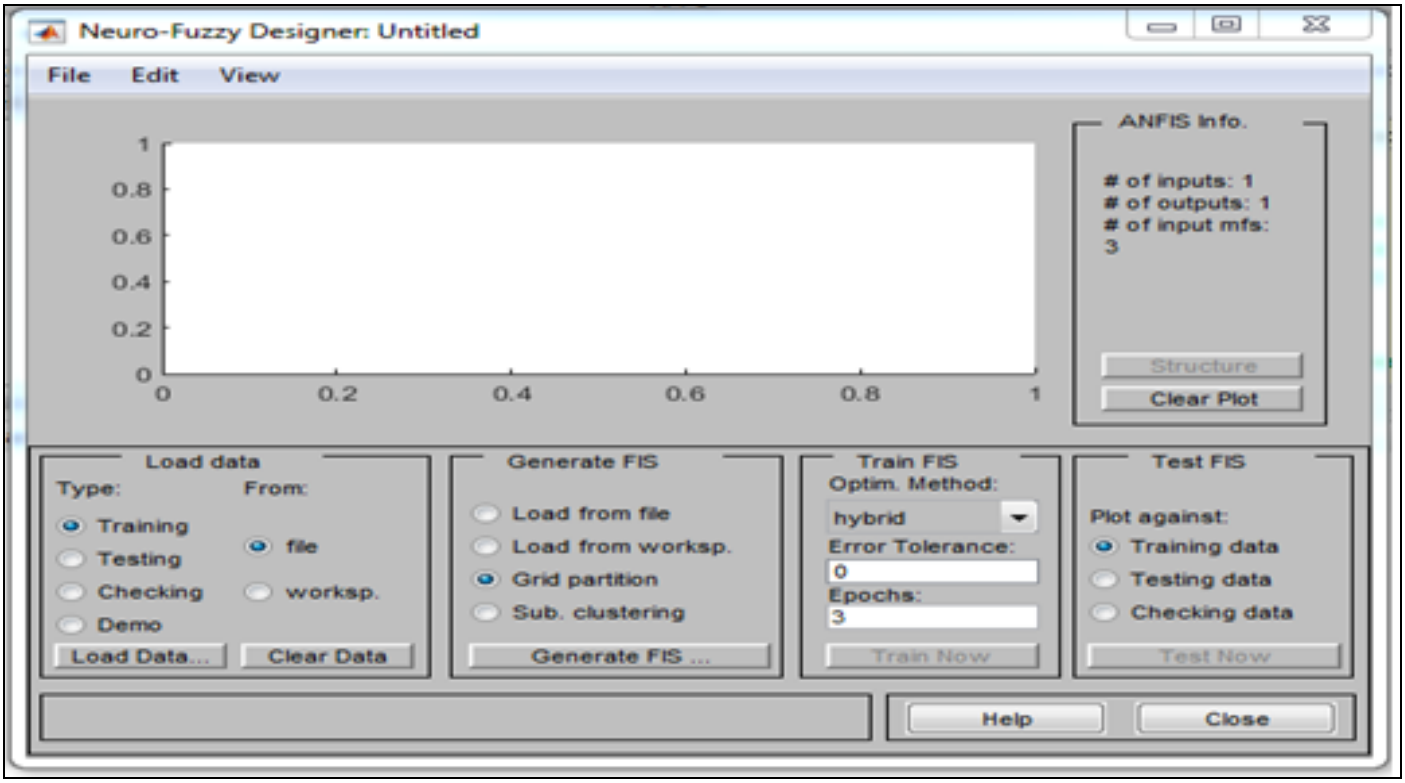


Fig 11 Neuro-Fuzzy Designer Interface Displaying initial FIS Structure and Training Settings

V. ANFIS MODELING AND TRAINING

To evaluate the fuzzy logic controller's performance, the complete control architecture was implemented in Simulink. The robot's dynamic model, sensor subsystems, and fuzzy inference engine were interconnected to form a closed-loop simulation. Data acquisition blocks logged wheel velocities, robot pose, and proximity sensor readings at each simulation step.

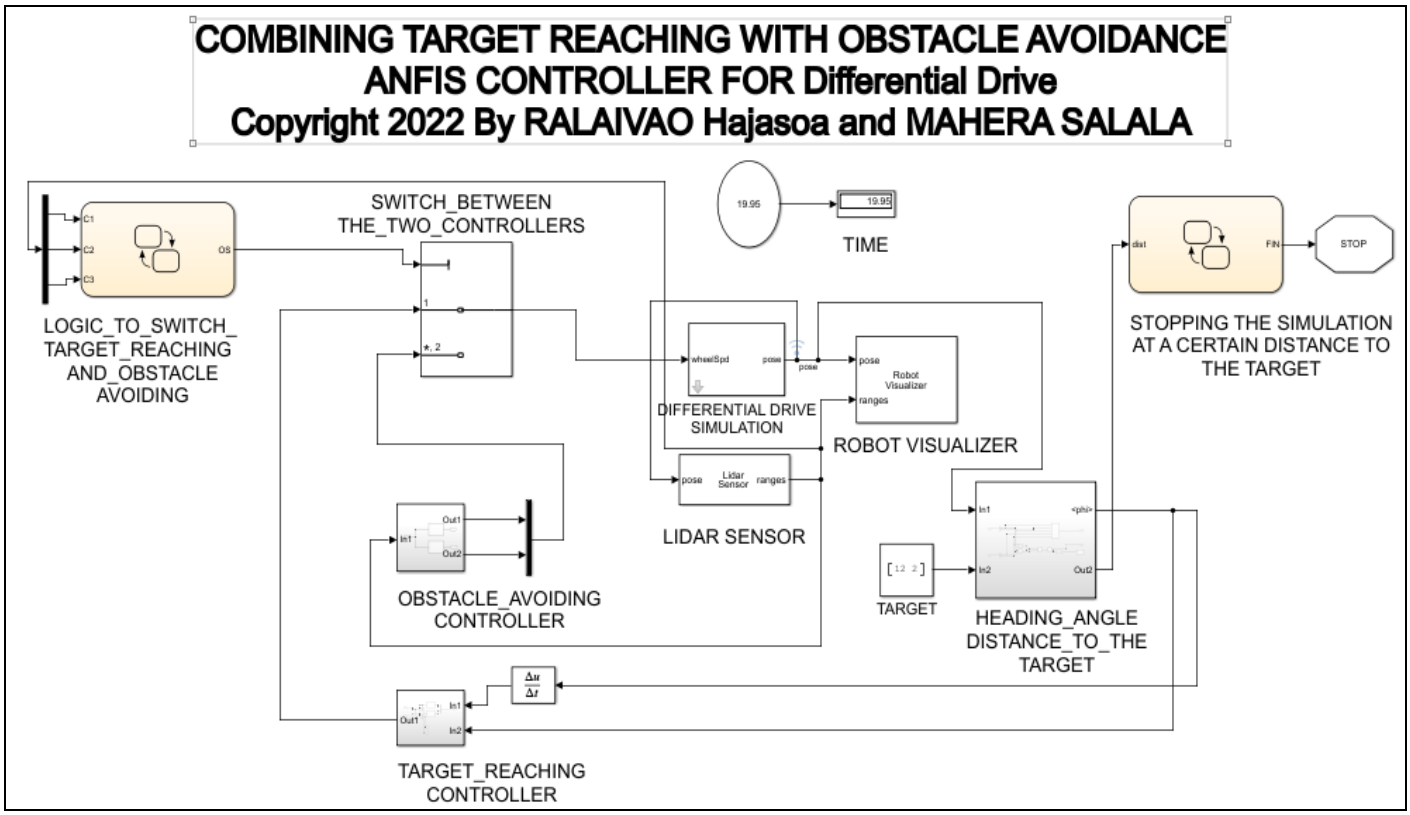


Fig 12 Complete Simulink Model Depicting Robot Kinematics, Sensors and Fuzzy Controller

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

➢ *During Navigation, the Fuzzy Controller was Tested in Various Scenarios:*

- Obstacle avoidance in cluttered environments
- Dynamic target tracking

➢ *The Following Signals were Recorded for the ANFIS Training Dataset:*

- Inputs: wheel velocities (vL, vR), linear velocity (V), robot pose (x, y, θ), and three proximity sensor readings.
- Outputs: fuzzy controller commands adjusting left and right wheel speeds.

Fuzzy Controller Implementation Steps.

➢ *The Fuzzy Controller Design is Divided into Four Steps (Figures 13–16):*

- Input Membership Functions (Figure 13): The inputs are categorized into two variables: the angle between the target and the robot's orientation, and the change in that angle over two time instants. The angle variable is divided into seven fuzzy subsets: Negative Very Large Angle (NVLA), Negative Large Angle (NLA), Negative Medium Angle (NMA), Positive Small Angle (PSA), Positive Medium Angle (PMA), Positive Large Angle (PLA), and Positive Very Large Angle (PVLA). The variation variable is divided into three fuzzy subsets: Negative, Zero, and Positive.
- Output Membership Functions (Figure 14): The outputs (wheel speeds) are divided into five fuzzy subsets: Negative Large (NL), Negative (N), Zero (Z), Positive (P), and Positive Large (PL).
- Rule Viewer and Inference (Figure 15): Visualization of the fuzzy rule base matrix and activation levels during inference.
- Surface Plots of Wheel Speeds (Figure 16): 3D surface curves showing how input variables map to right and left wheel speeds.

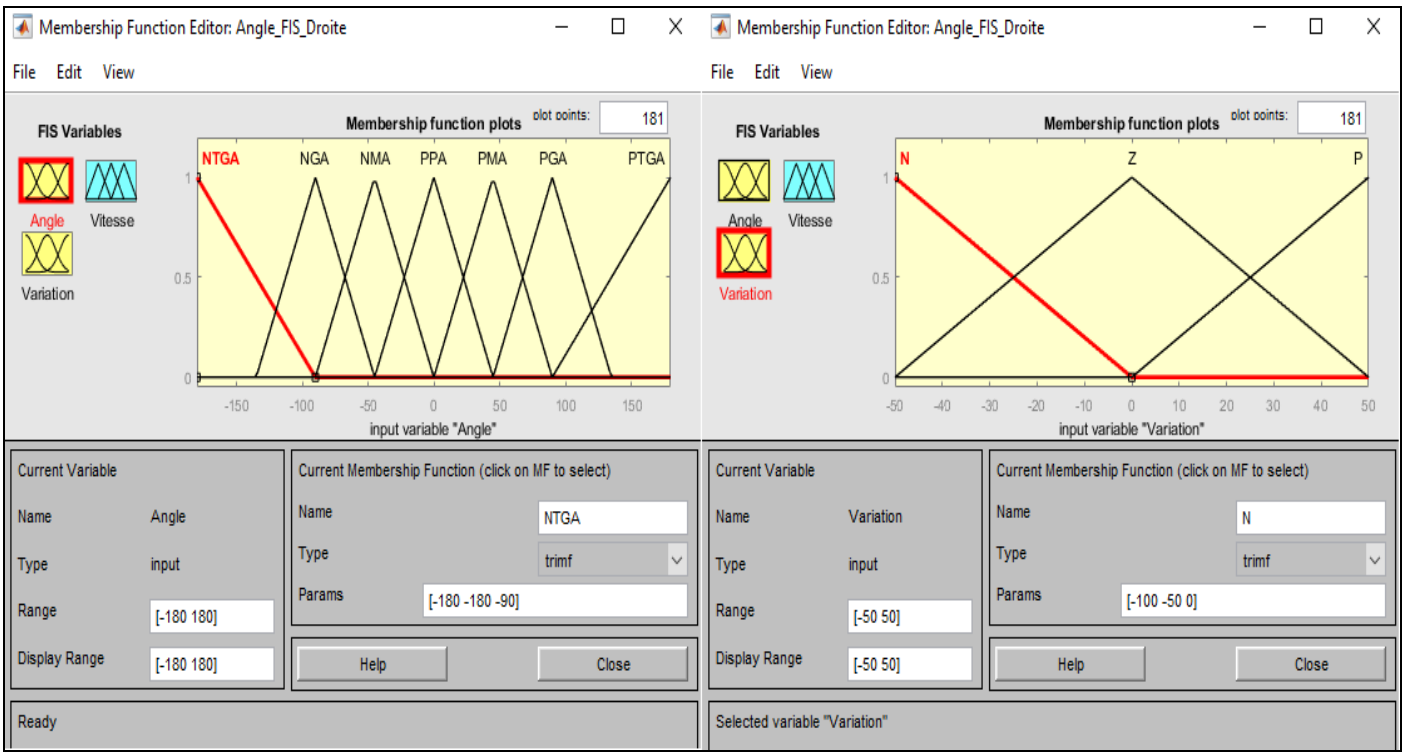


Fig 13 Input Membership Functions

- *Comment:*

Shows the seven fuzzy subsets for the angle variable and three subsets for its variation, defining the linguistic terms for both inputs.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

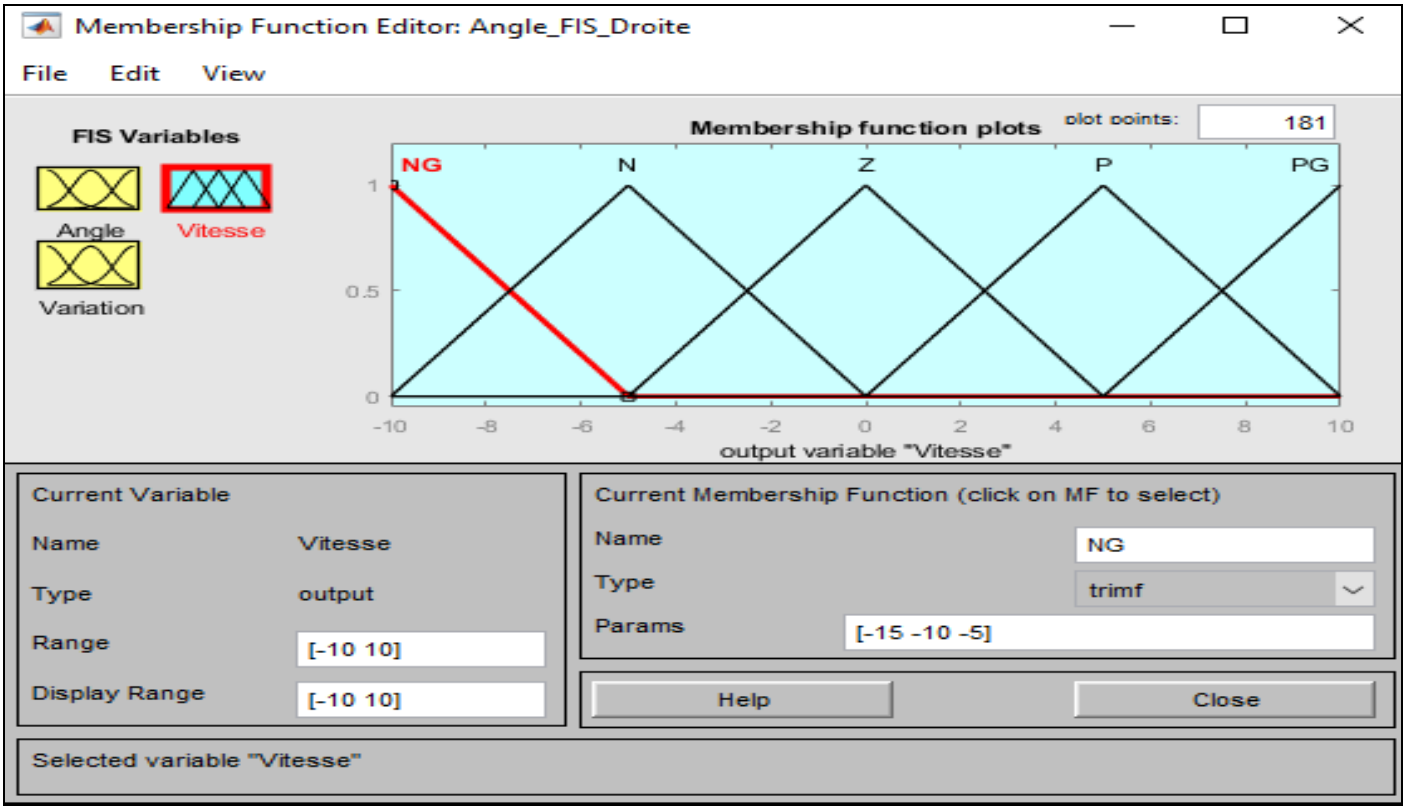


Fig 14 Output Membership Functions (Wheel Speeds)

- *Comment:*
  Depicts the five fuzzy subsets for wheel speed outputs used to translate fuzzy control signals into actuator commands.

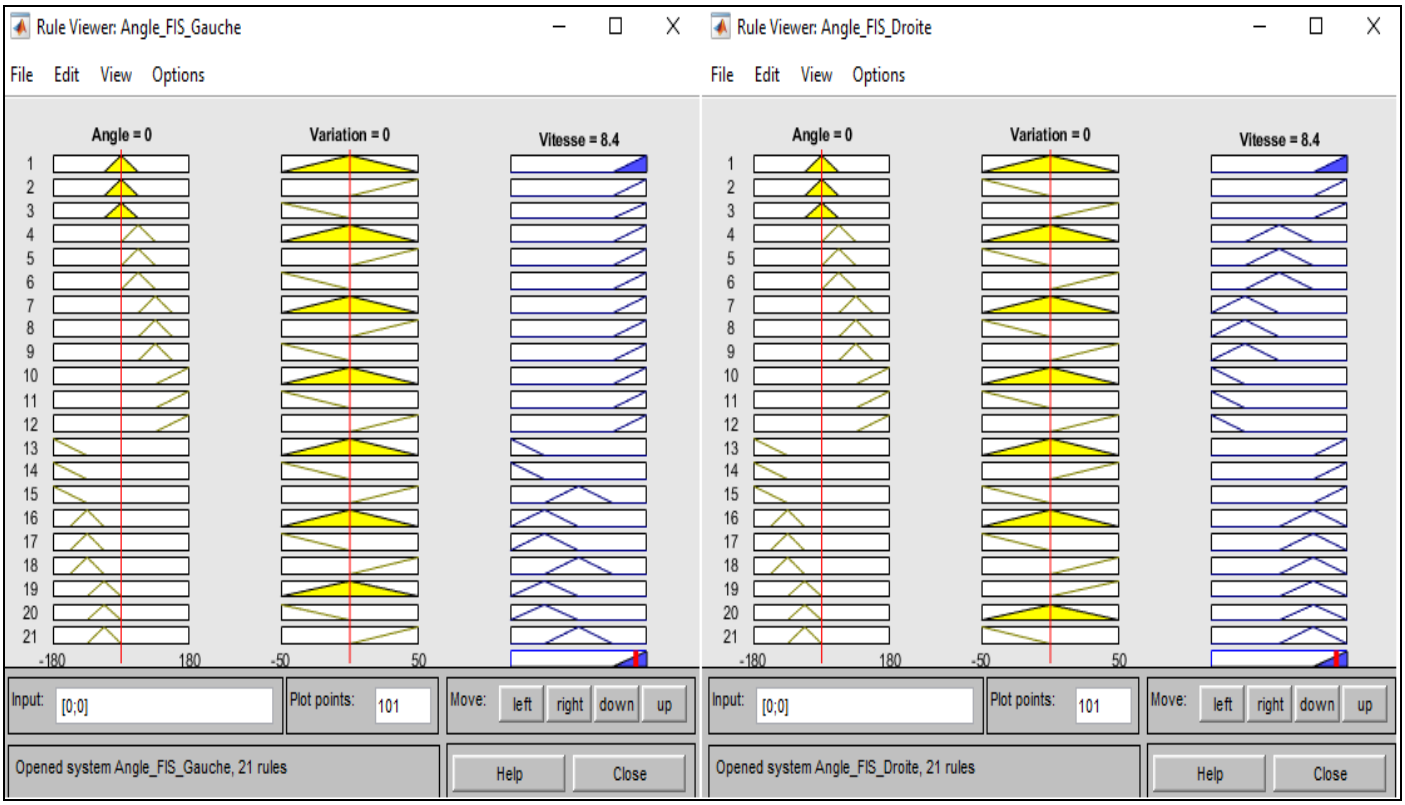


Fig 15 Rule Viewer Showing Fuzzy Base and Inference Activations

- *Comment:*
  Visualizes the full set of fuzzy if–then rules and their activation levels for given input cases.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

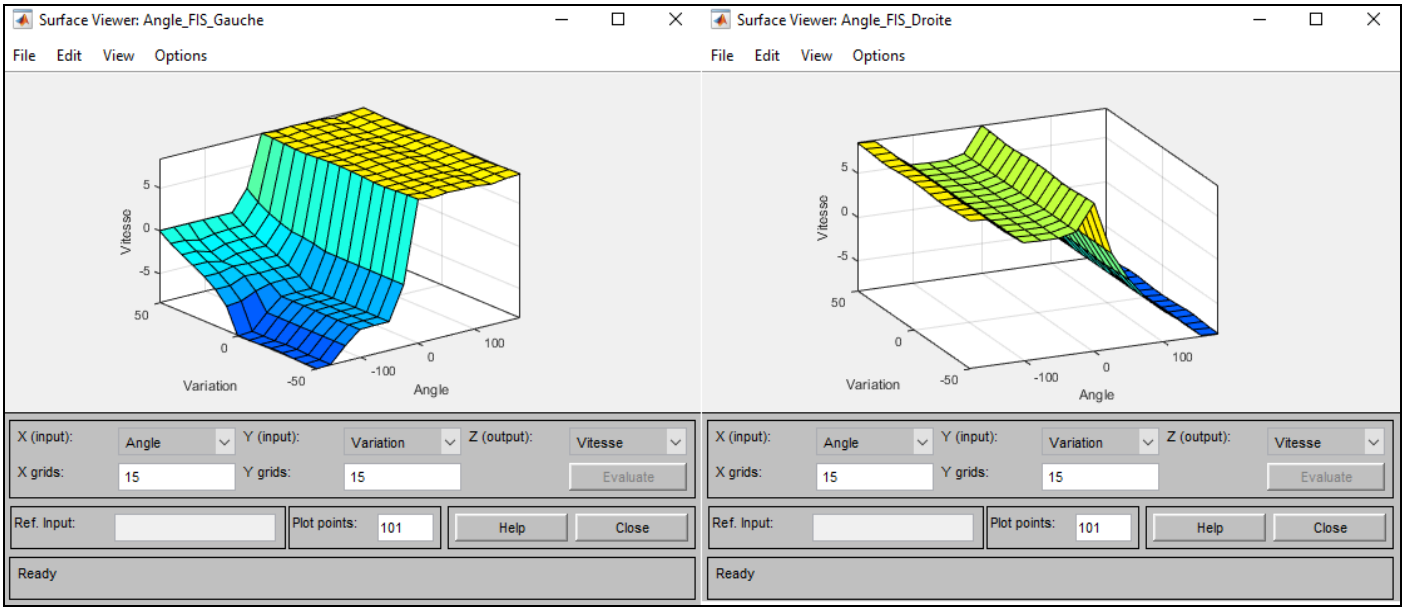


Fig 16 3D Surface Plots of Wheel Speeds as Functions of input Variables

- *Comment:*

Illustrates how combinations of input angles and variations produce specific left and right wheel speeds through 3D surface mappings.

VI. FUZZIFICATION, INFERENCE AND DEFUZZIFICATION FOR OBSTACLE AVOIDANCE

To detail the obstacle avoidance module, the fuzzification, Sugeno inference, and defuzzification processes are presented in four figures (Figures 17–19).

➢ *Fuzzification:*

The obstacle distance and relative bearing inputs are fuzzified into linguistic terms to handle sensor noise and nonlinearity. A typical Sugeno model uses crisp outputs computed from linear functions of the inputs.

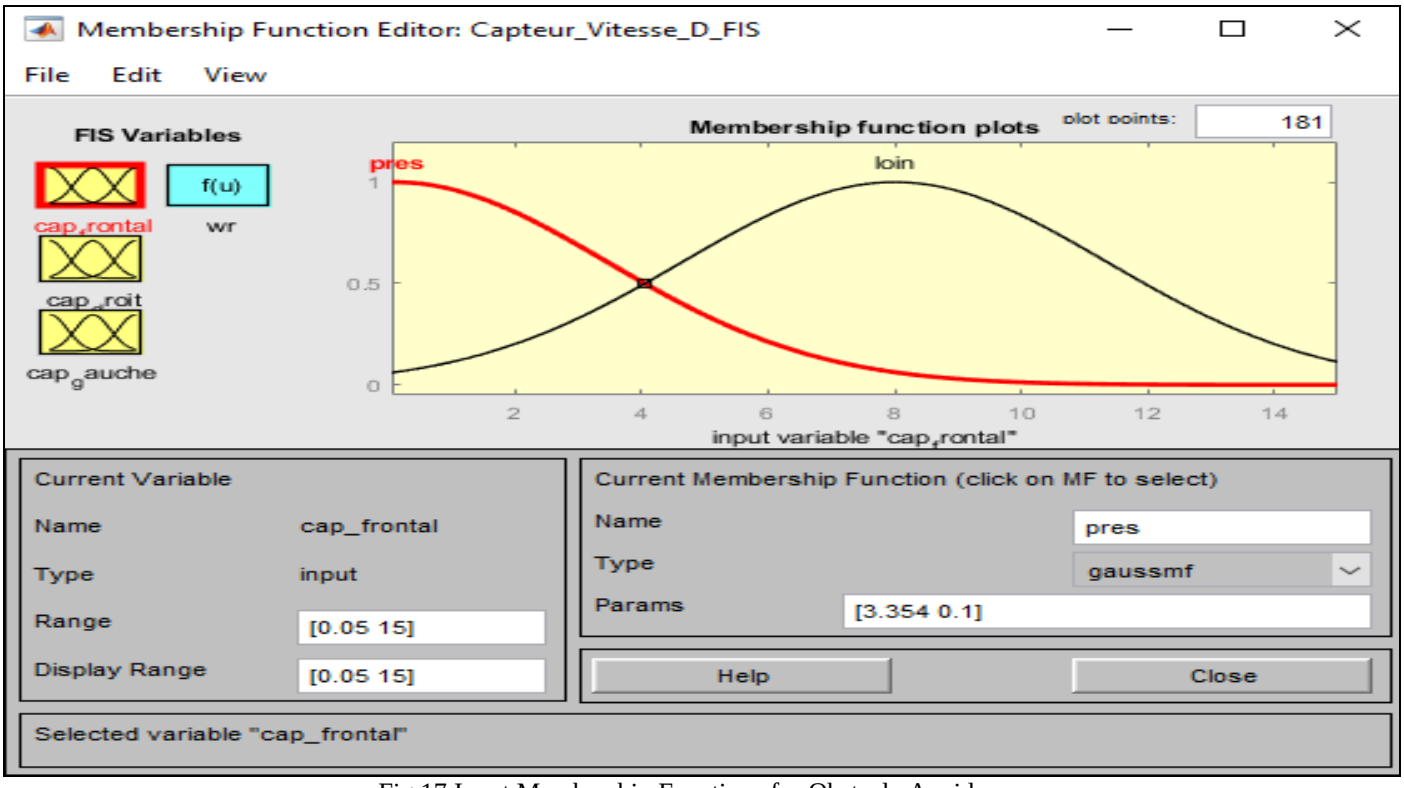


Fig 17 Input Membership Functions for Obstacle Avoidance

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

- *Comment:*

Defines fuzzy subsets for obstacle distance (Near, Far) and relative bearing (Left, Center, Right).

Inference (Sugeno Model) Each fuzzy rule produces a crisp output as a linear function of the inputs. The weighted average of all rule outputs yields the final control action. Rule Base A set of if–then rules maps fuzzified sensor inputs to control outputs, enabling reactive obstacle avoidance.

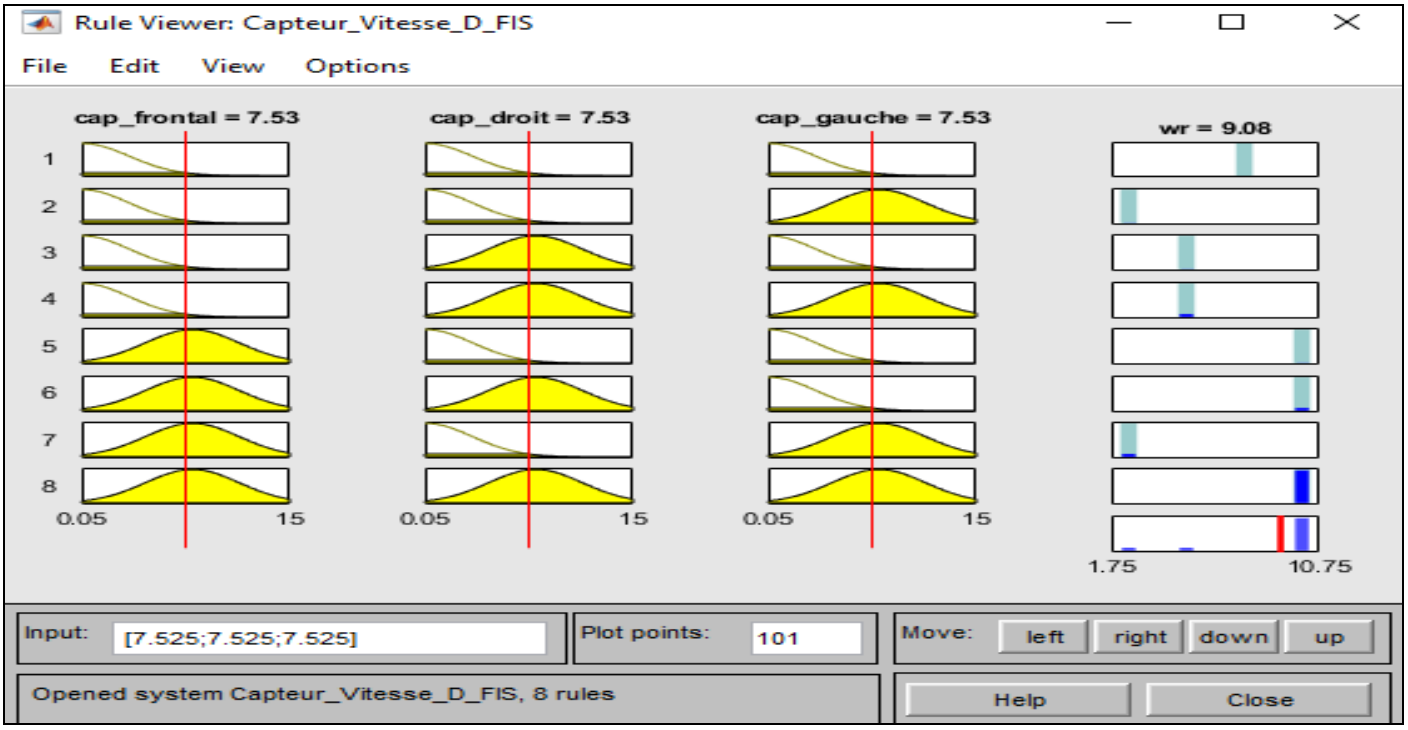


Fig 18 Fuzzy Rule Base for Obstacle Avoidance

- *Comment:*

Lists the obstacle avoidance rules, e.g., ”If distance is Near and bearing is Left, then turn right fast.”

➢ *Defuzzification:*

The Sugeno output is computed via a weighted average of rule consequents, providing a crisp wheel velocity command.

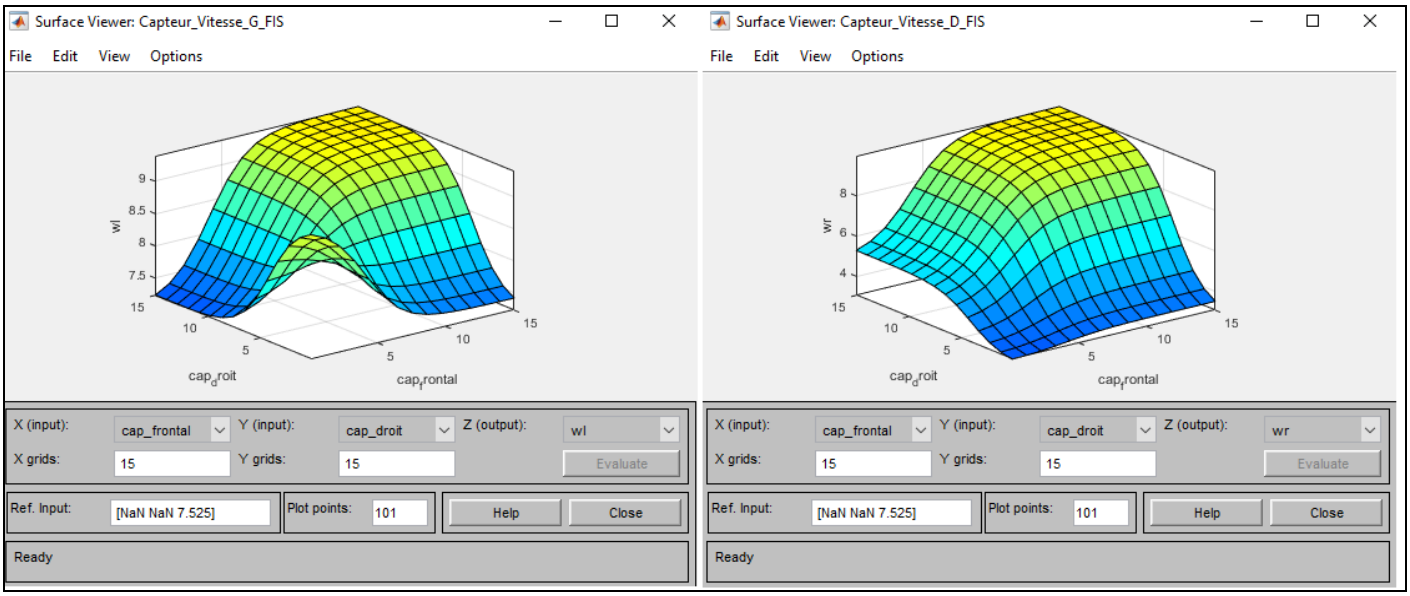


Fig 19 Surface Plot of Obstacle Avoidance Control Output

- *Comment:*

Visualizes how combinations of obstacle distance and bearing map to left/right wheel speed commands via the Sugeno model.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

VII. FUZZY CONTROLLER SIMULATION

We evaluated the standalone performance of the fuzzy logic controller on three scenarios: pure target-reaching, pure obstacle-avoidance, and the combined task of navigating from (2, 2) to (8, 8) while avoiding obstacles.

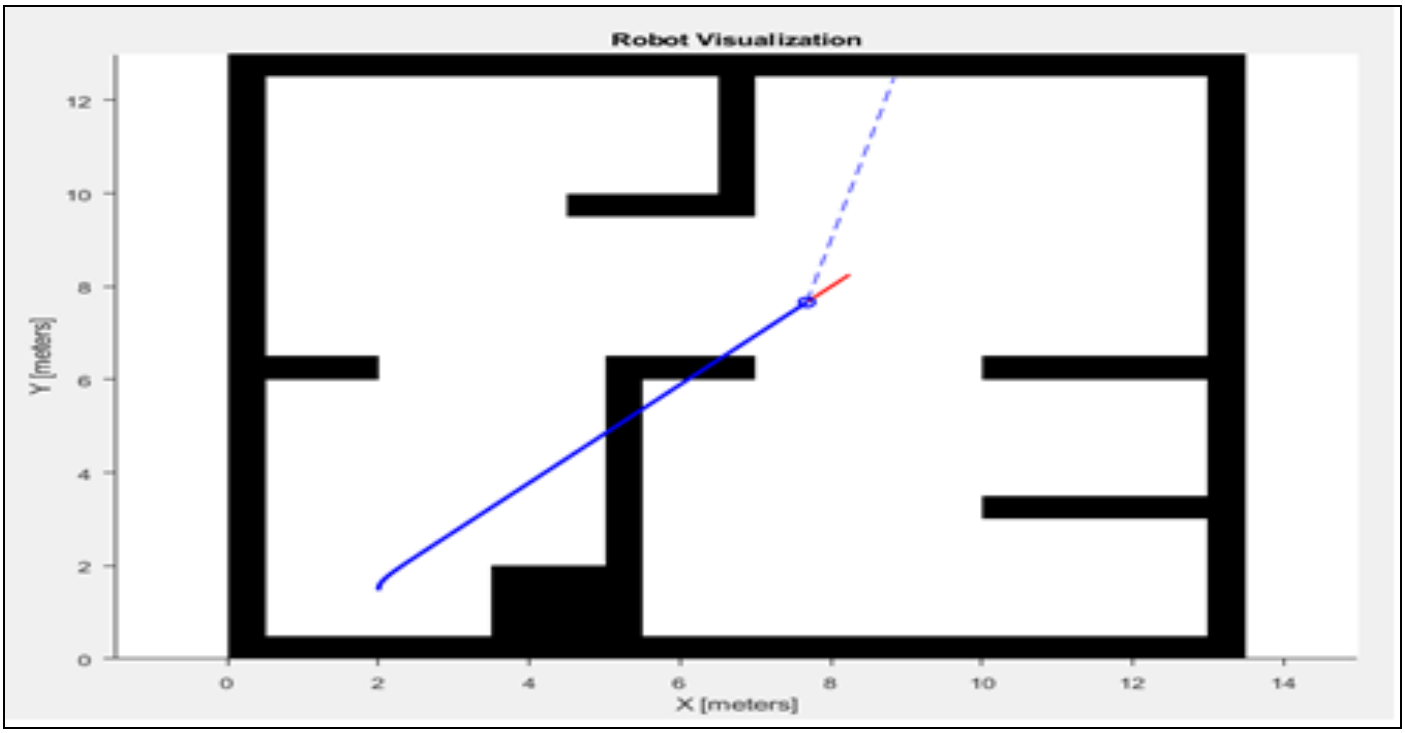


Fig 20 Simulation of the Fuzzy Controller Driving the Robot from its Start to Target (8,8) without Obstacles

Figure 20 demonstrates that the controller reliably guides the robot to the specified goal (8, 8) along a smooth trajectory, achieving the target as expected.

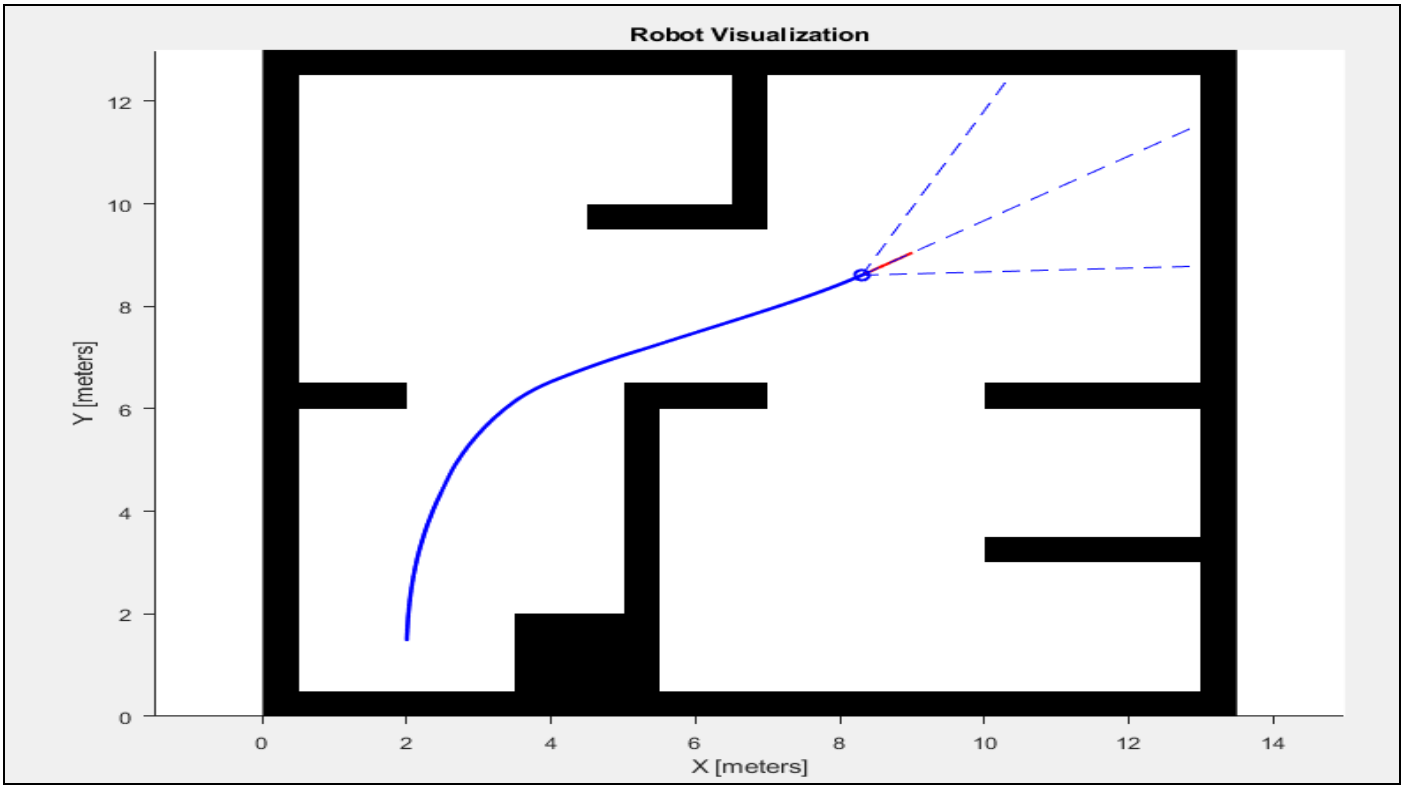


Fig 21 Simulation of the Fuzzy Obstacle Avoidance Controller with Multiple Static Obstacles

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

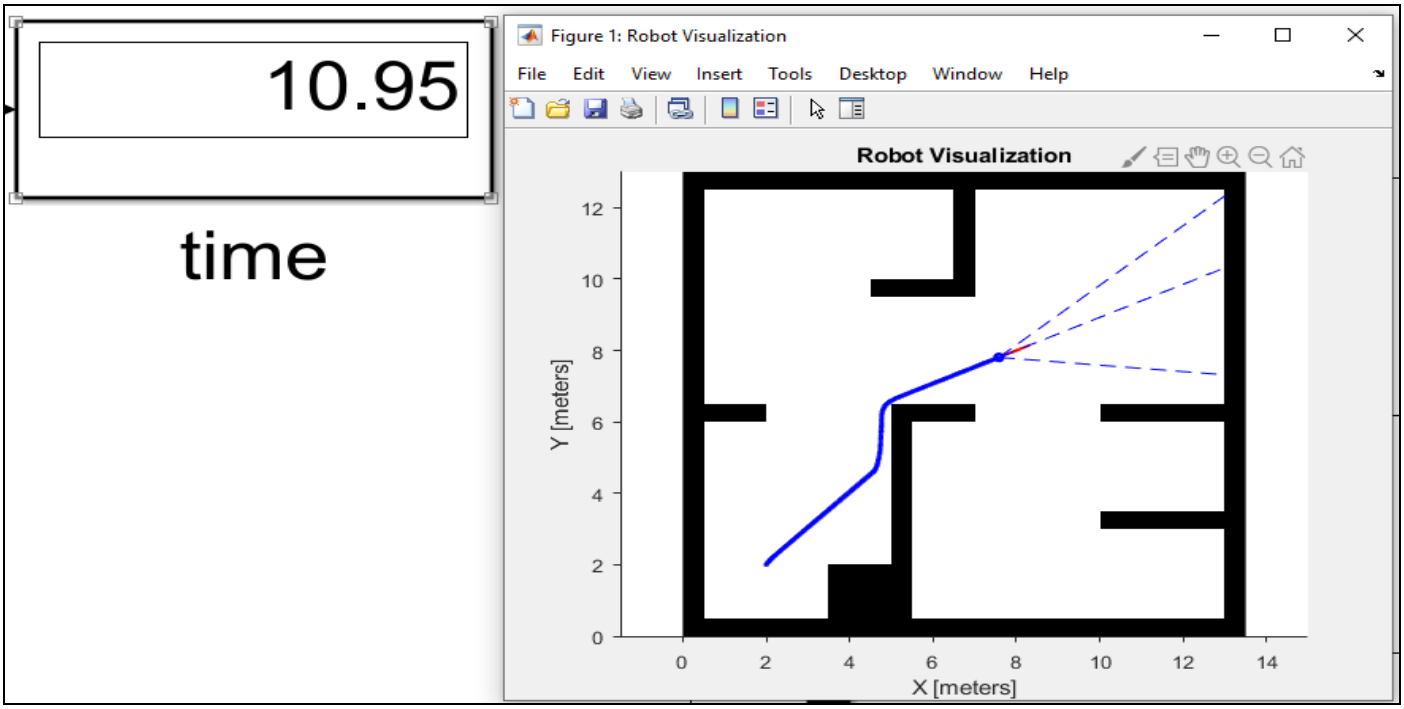


Fig 22 Combined Simulation from (2, 2) to (8,8)

Figure 22 illustrates the first combined-task scenario. The fuzzy controller merges goal seeking and obstacle-avoidance, yielding an optimal, collision-free trajectory from the start point (2, 2) to the goal (8, 8), with a traveling time of 10.95 seconds.

These are some more additional navigation scenarios.

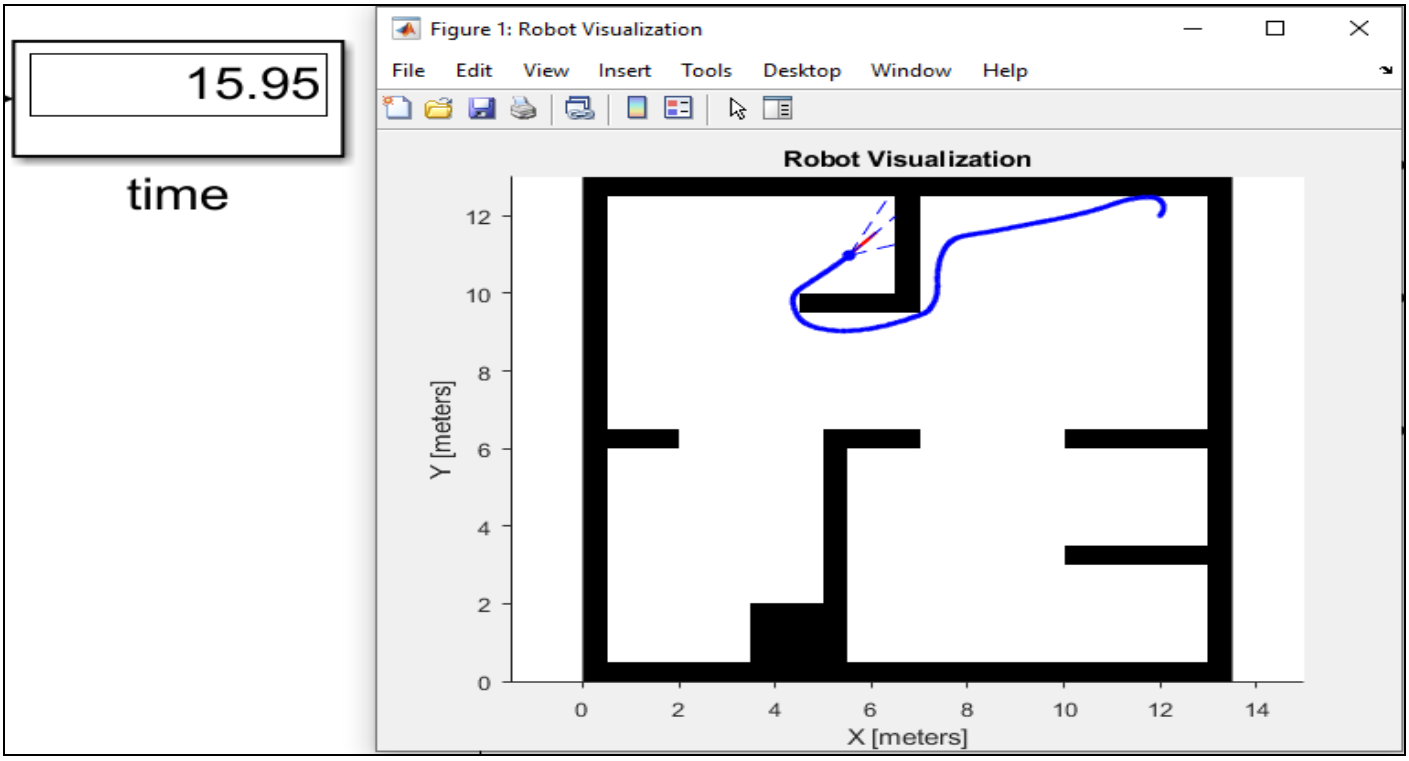


Fig 23 Scenario 2 from (12, 12) to (6, 11)

Figure 23 shows the robot navigating from (12, 12) to (6, 11). The fuzzy controller completes the journey in 15.95 s while avoiding all obstacles.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

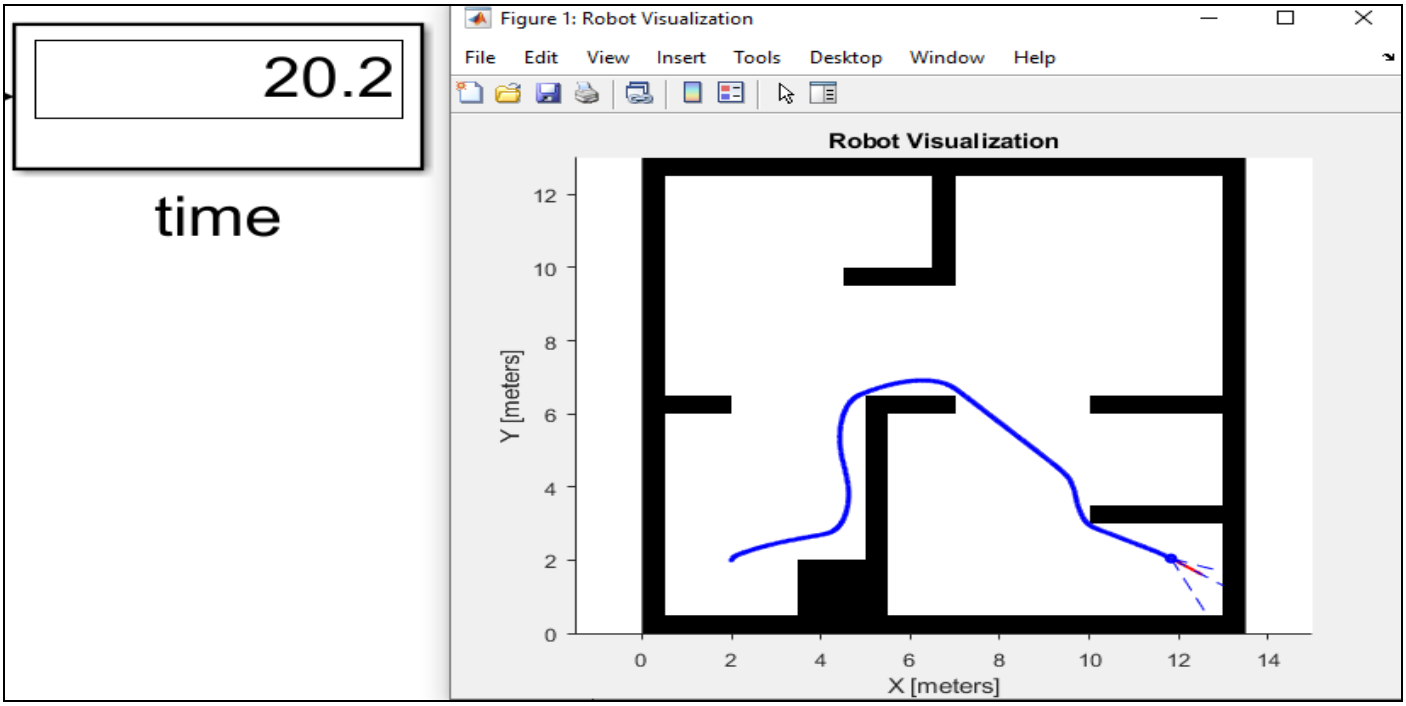


Fig 24 Scenario 3 from (2, 2) to (12, 2)

Figure 24 illustrates the path from (2, 2) to (12, 2). The controller achieves the target in 20.02s, demonstrating efficient straight-line travel with obstacle avoidance where needed.

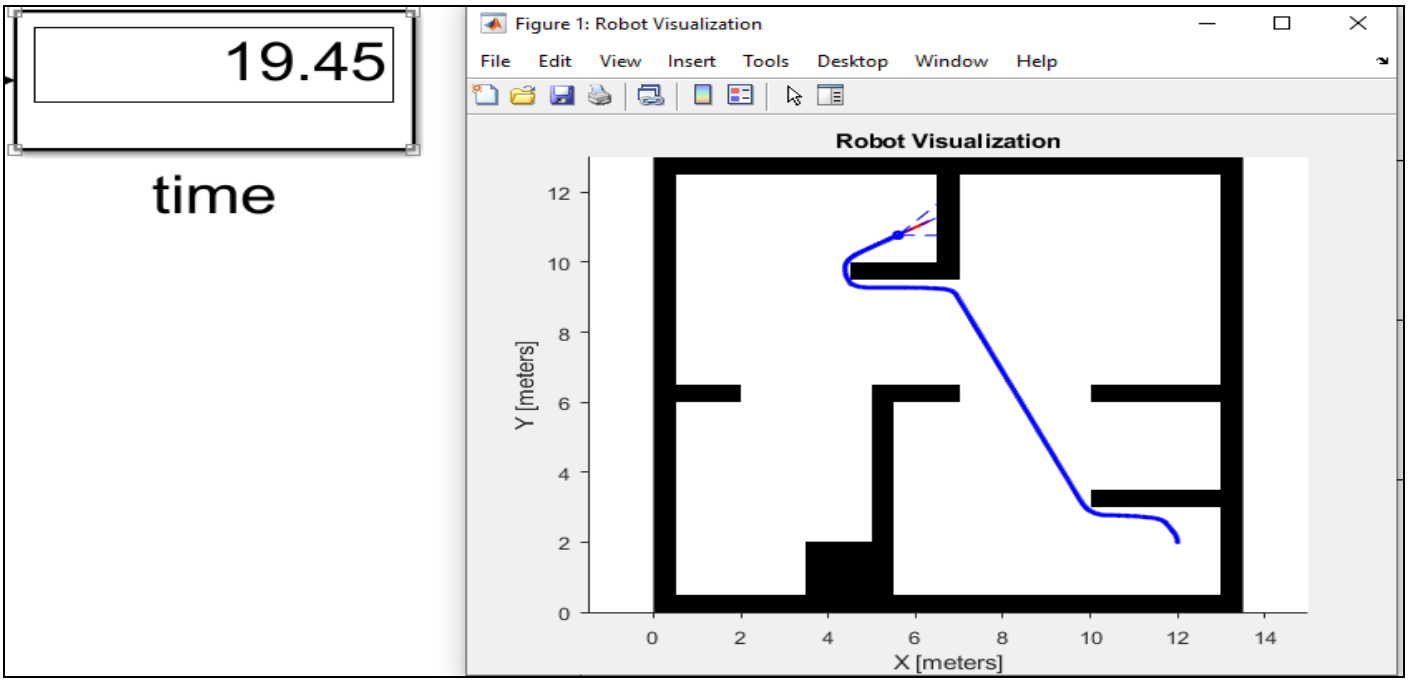


Fig 25 Scenario 4 from (12, 2) to (6, 11)

Figure 25 depicts the route from (12, 2) to (6,11). Due to the more complex obstacle layout, the controller requires 19.45 s to negotiate obstacles and arrive at the goal.

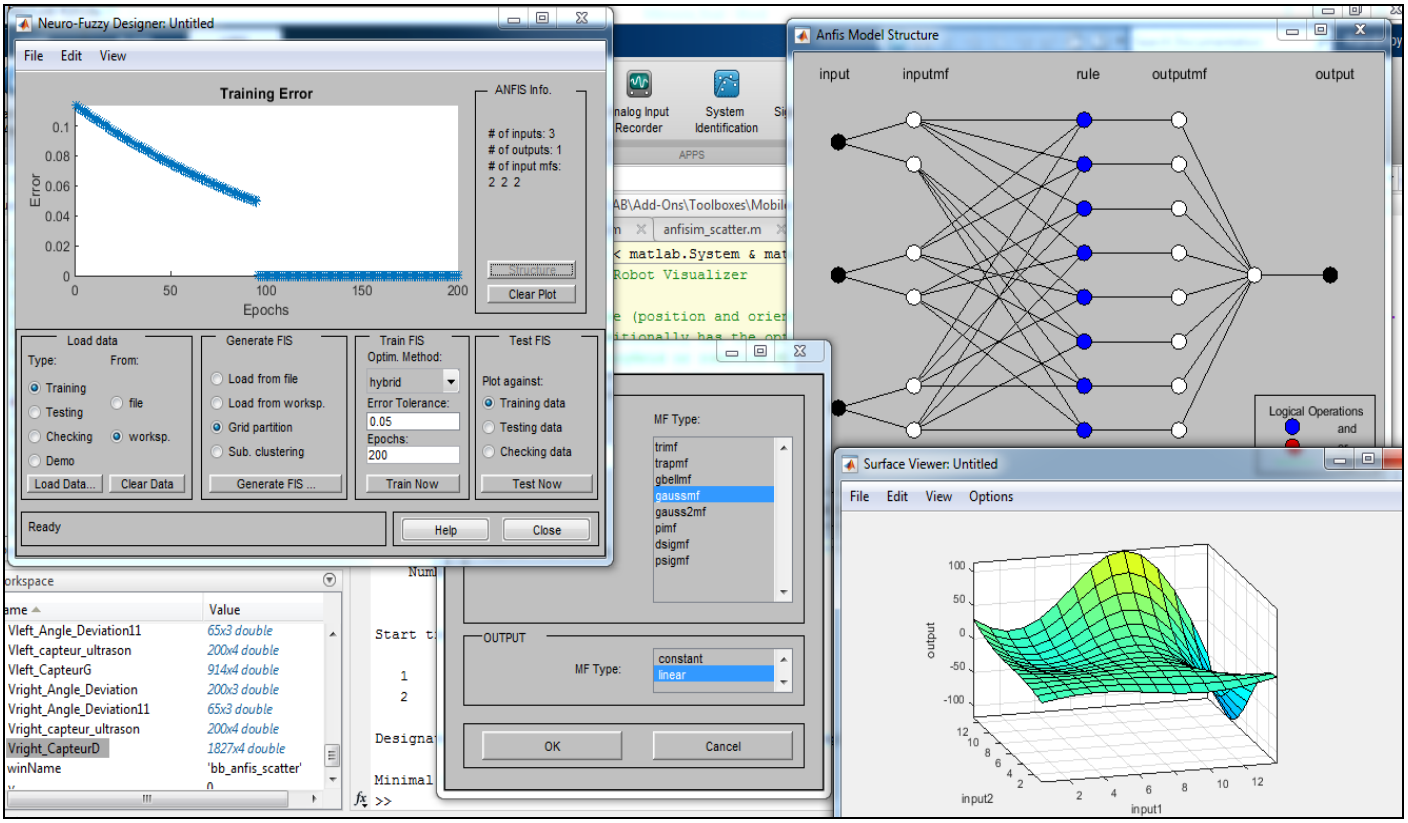
VIII. ANFIS TRAINING AND PARAMETERIZATION

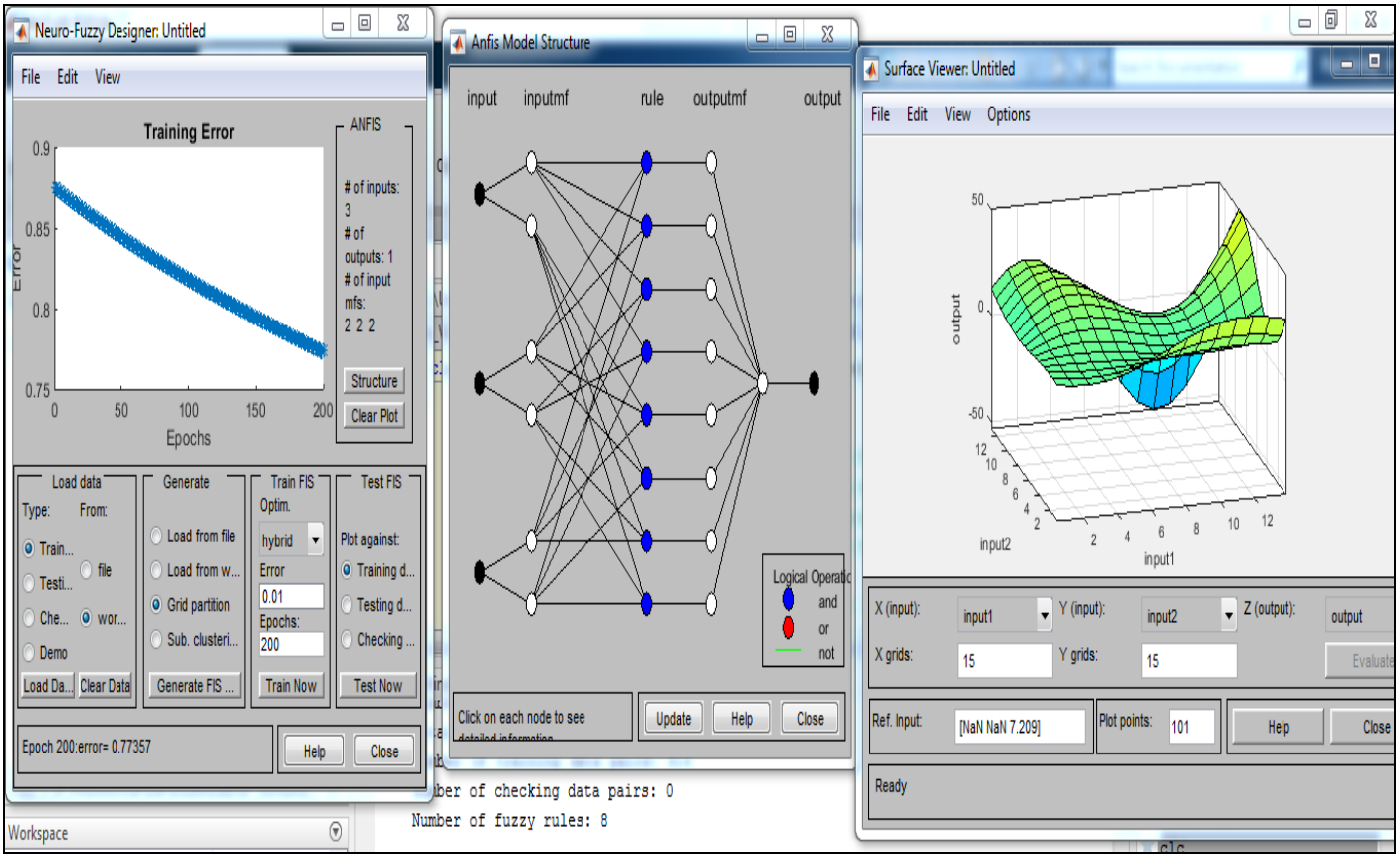
Upon completion of the robot deployments with the fuzzy controllers, all data from each scenario including left and right wheel velocities, sensor readings, and heading angle was recorded into a training matrix. This dataset was then used to train ANFIS models, yielding neuro-fuzzy controllers for each wheel and each task (target reaching and obstacle avoidance). Subsequently, the robot was redeployed to compare the performance of these controllers.

In this first part of the section, Figures 26a–26d illustrate the step by step parameterization and training of the ANFIS models:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

- Obstacle avoidance controller training for the left wheel (Figure 26a),
- Obstacle avoidance controller training for the right wheel (Figure 26b),
- Target reaching controller training for the left wheel (Figure 26c),
- Target reaching controller training for the right wheel (Figure 26d).





ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

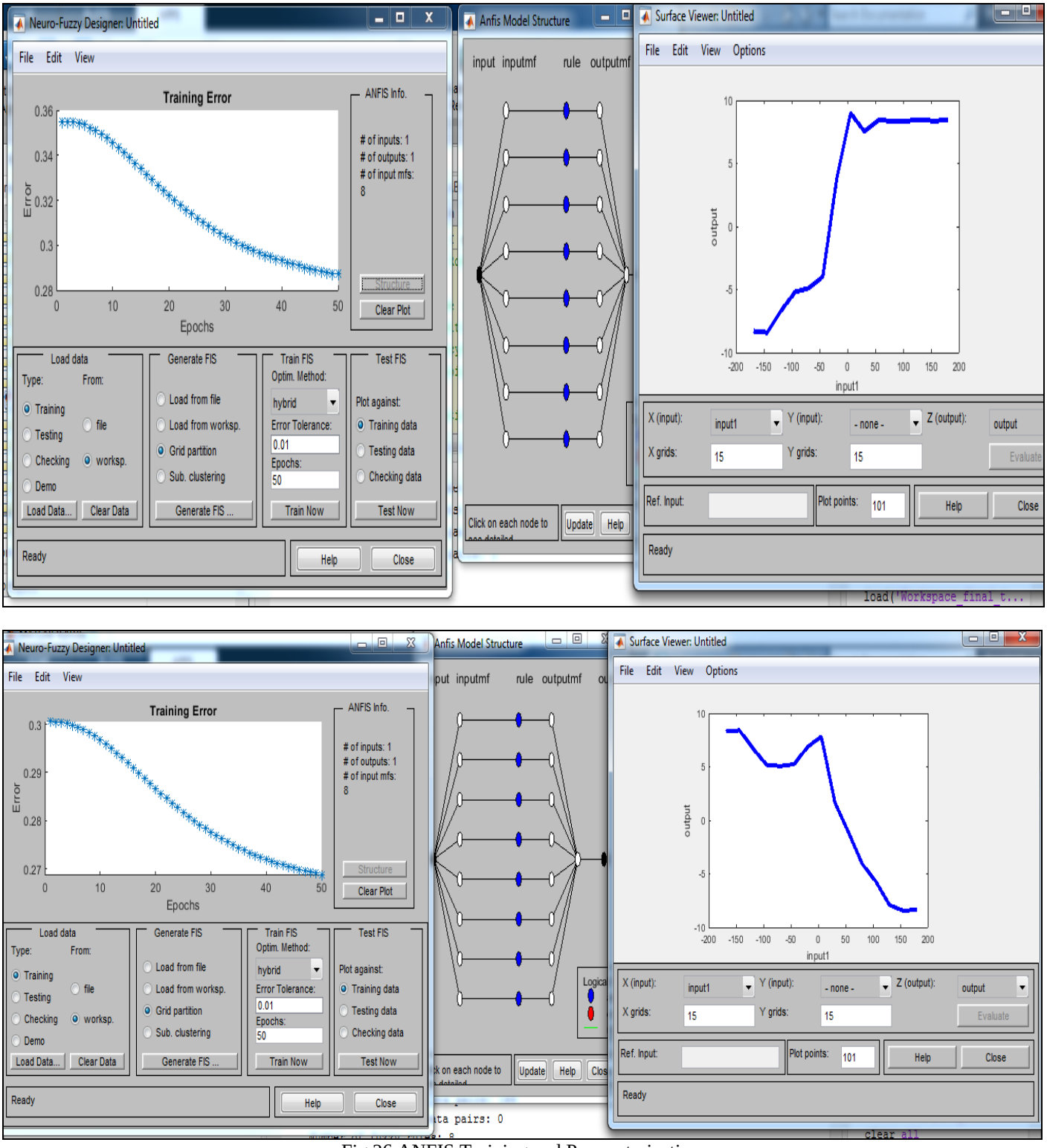


Fig 26 ANFIS Training and Parameterization

IX. NEURO-FUZZY CONTROLLER DEPLOYMENT

To evaluate the performance of the trained neuro-fuzzy controllers, the robot was redeployed under five different conditions. In the first experiment, only the obstacle-avoidance neuro-fuzzy controller was active, and the robot drove until battery exhaustion, avoiding every obstacle in its path. The following five figures correspond to the same navigation scenarios previously simulated with the fuzzy logic controller, now executed with the neuro-fuzzy models. Observation. In all four scenarios, the ANFIS neuro-fuzzy controller consistently reduces the time to reach the target compared to the pure fuzzy controller. The greatest improvement occurs in scenario 4 (approximately 2.25s less), demonstrating better adaptation to complex avoidance trajectories.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

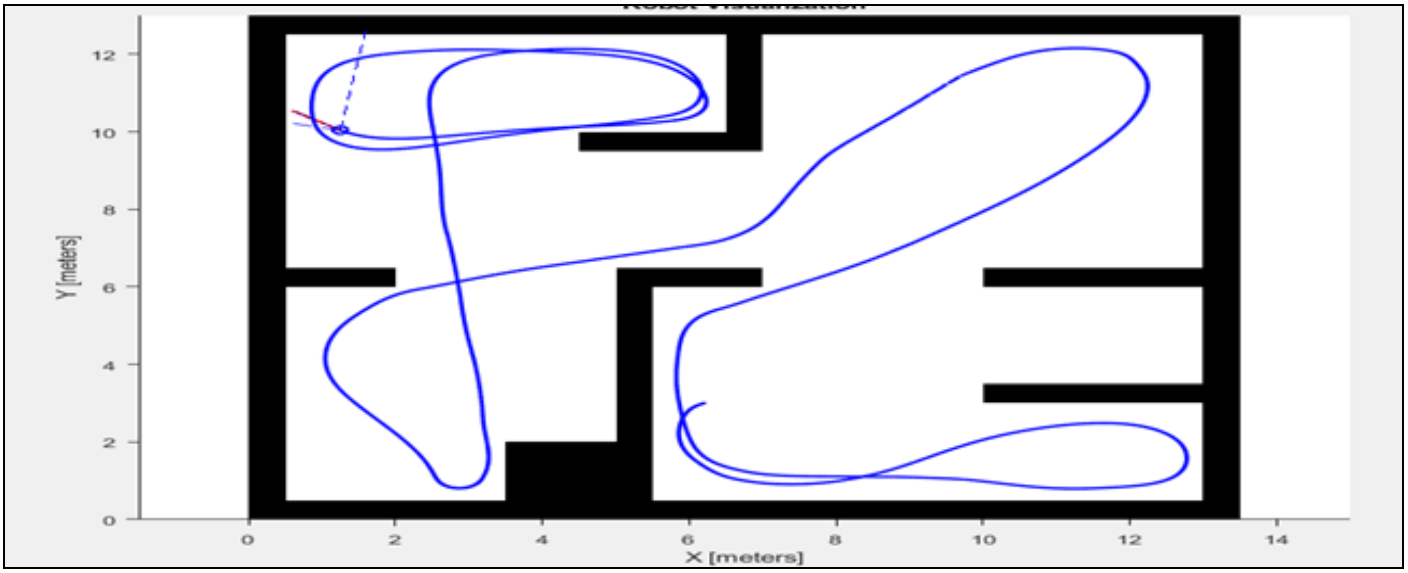


Fig 27 Deployment with Obstacle Avoidance Neuro-Fuzzy Controller Only

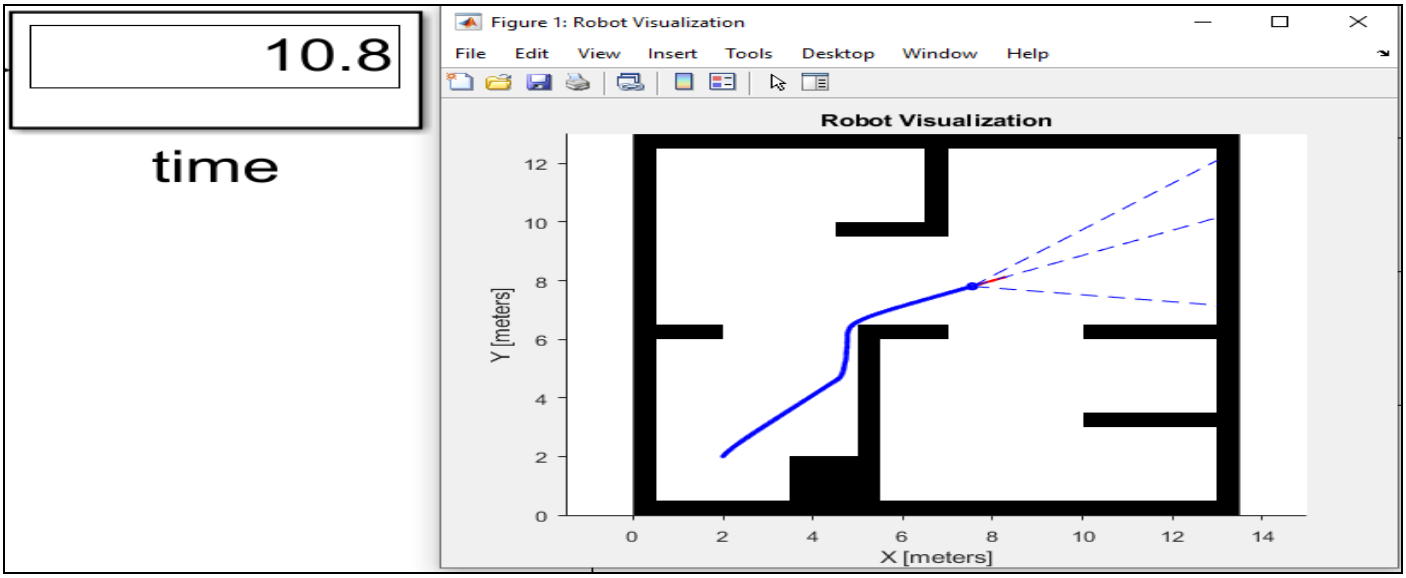


Fig 28 Scenario 1 (Combined Navigation) from (2, 2) to (8, 8)

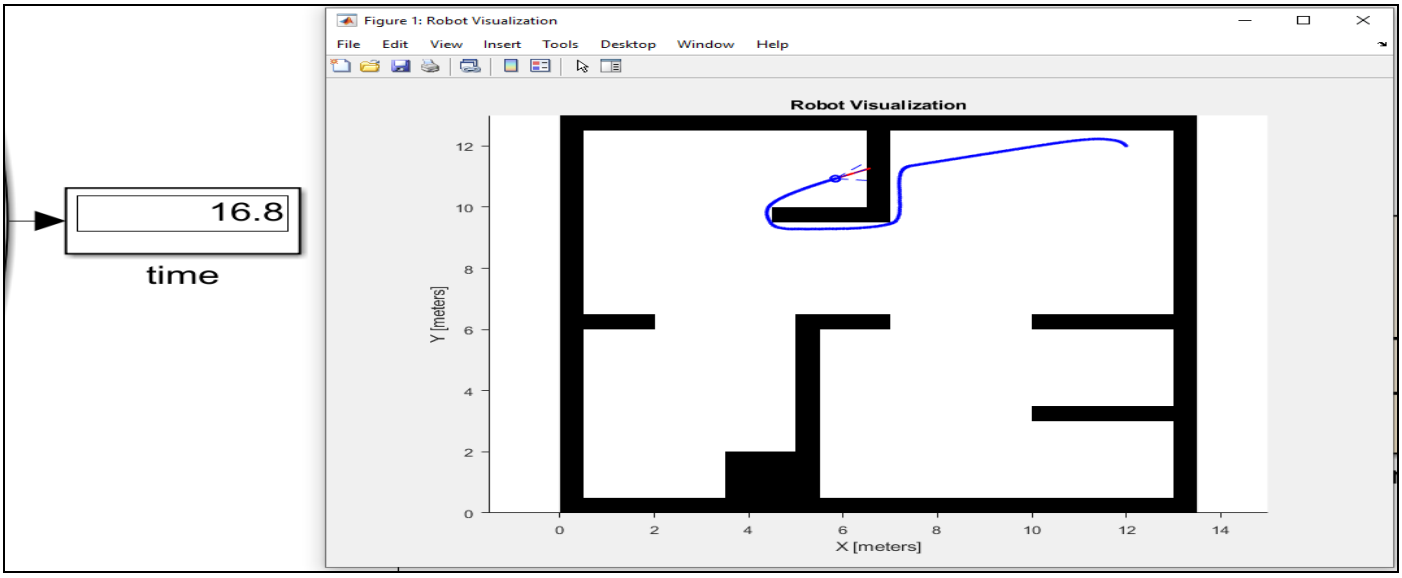


Fig 29 Scenario 2 (Anfis Controller) from (12, 12) to (6, 11)

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

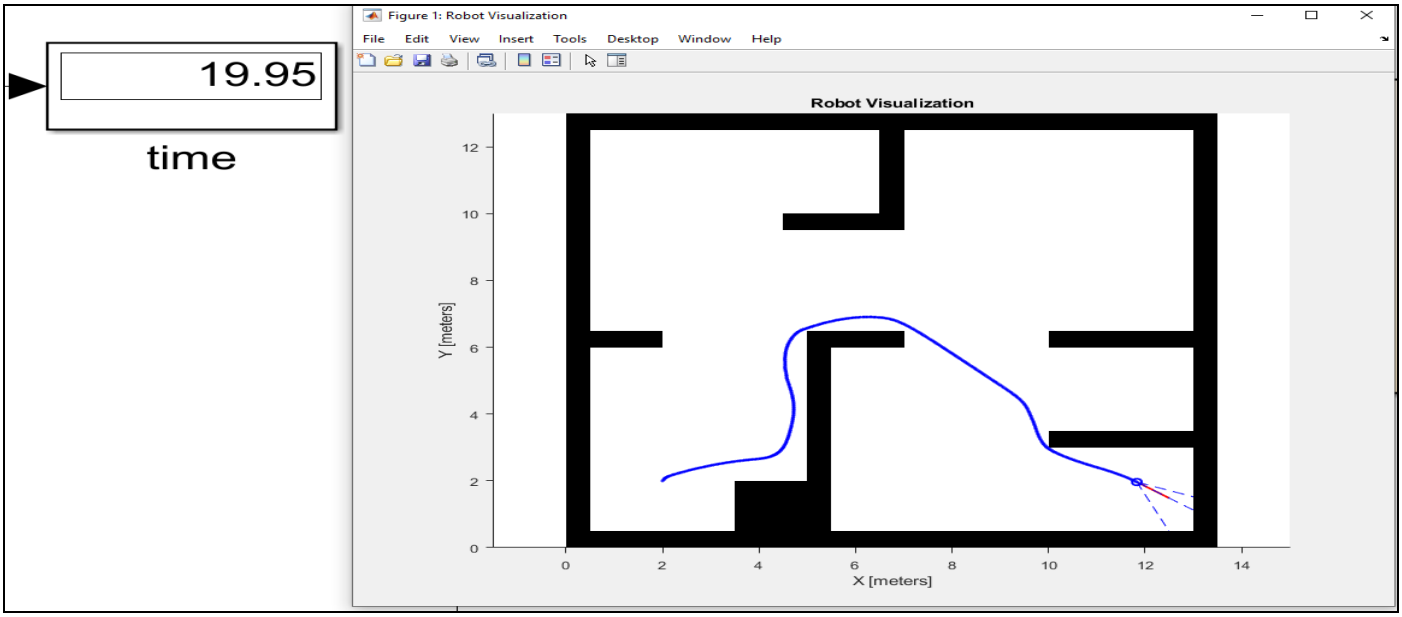


Fig 30 Scenario 3 (Anfis Controller) (2, 2) to (12, 2)

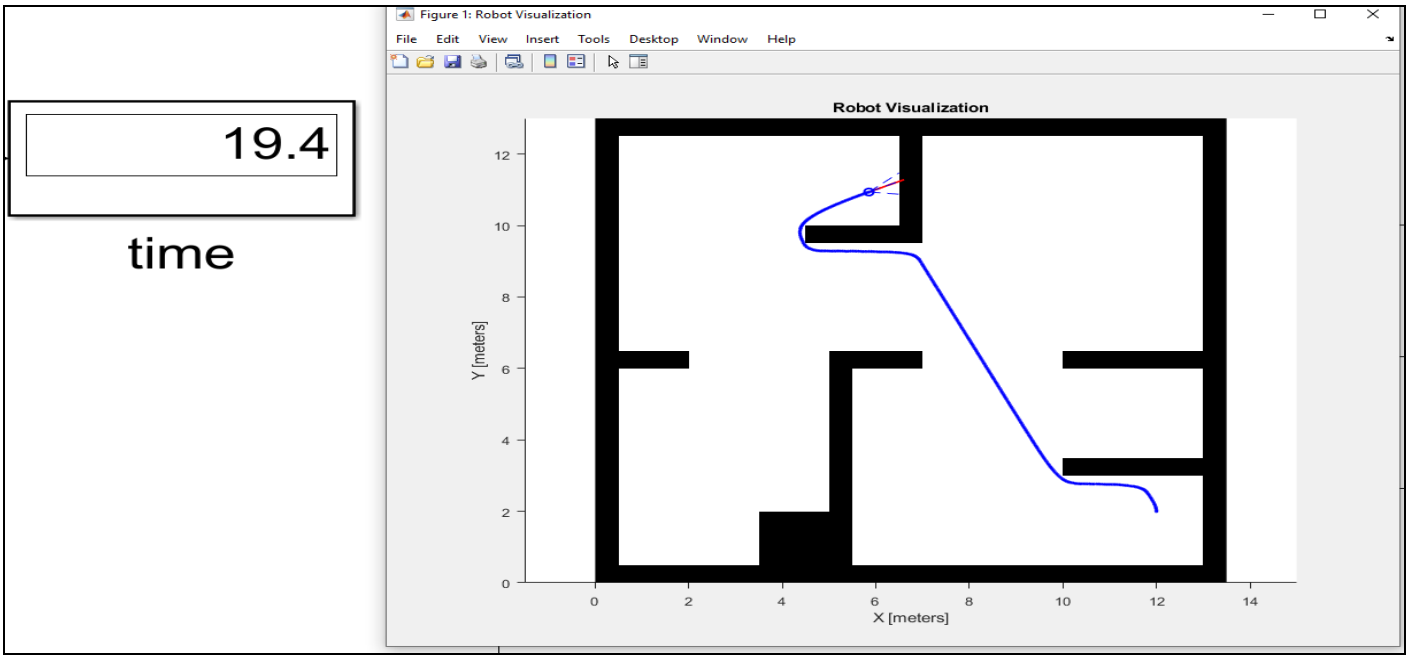


Fig 31 Scenario 4 (Anfis Controller) from (12, 12) to (6, 11)

Table 1 Time to Target for Fuzzy and ANFIS Controller

Scenario	Start	Goal	Fuzzy	ANFIS
1	(2, 2)	(8, 8)	10.95	10.80
2	(12, 12)	(6, 11)	15.95	16.8
3	(2, 2)	(12, 2)	20.2	19.95
4	(12, 2)	(6, 11)	19.45	19.4

X. GENERAL CONCLUSION AND PERSPECTIVES

Our investigation demonstrates that conventional fuzzy control offers notable advantages, including resilience to uncertainties and straightforward deployment on differential-drive mobile platforms. By preprocessing trajectory data from the four scenarios and training an ANFIS model, we achieved significant reductions in target-reaching time while preserving the flexibility of fuzzy logic. The successful deployment on the robot confirms the suitability of ANFIS for navigation and obstacle avoidance tasks. Future work could explore different fuzzy controller variants (Mamdani vs. Sugeno), alternative ANFIS architectures, and training parameter tuning (number of rules, membership functions, optimization algorithms). A comparison with traditional PID controllers would quantify relative benefits in performance and computational cost. Moreover, integrating machine

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25sep1549

learning and deep learning models such as transformers for trajectory planning opens avenues for hybrid controllers capable of online strategy adaptation. These findings pave the way for high-impact applications, including robotic surgery, assistive devices for individuals with reduced mobility, and autonomous vehicles, where decision-making precision and responsiveness are critical.

REFERENCES

[1]. S. L. Chiu. Fuzzy model identification based on cluster estimation. Journal of Intelligent and Fuzzy Systems, 2(3):267–278, 1994.

[2]. Jyh-Shing R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics, 23(3):665–685, 1993.

[3]. Alonzo Kelly. Mobile Robotics: Mathematics, Models, and Methods. Cambridge University Press, Cambridge, UK, 2013.

[4]. George J. Klir and Bo Yuan. Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice Hall, Upper Saddle River, NJ, 1995.

[5]. Jerry M. Mendel. Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Prentice Hall, Upper Saddle River, NJ, 2001.

[6]. Timothy J. Ross. Fuzzy Logic with Engineering Applications. Wiley, Chichester, UK, 2010.

[7]. Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. Introduction to Autonomous Mobile Robots. MIT Press, Cambridge, MA, 2011.

[8]. Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. Robot Modeling and Control. Wiley, Hoboken, NJ, 2006.

[9]. M. Subbash and S. Chong. Data-driven anfis design for mobile robot navigation. IEEE Transactions on Fuzzy Systems, 27(8):1588–1599, 2019.

[10]. The MathWorks, Inc. Fuzzy Logic Toolbox and ANFIS User's Guide. MathWorks, 2024. Version R2024a.

[11]. The MathWorks, Inc. Matlab fuzzy logic toolbox user's guide. https://www.mathworks.com/help/fuzzy/, 2024.

[12]. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics. MIT Press, Cambridge, MA, 2005.

[13]. Lotfi A. Zadeh. Fuzzy sets. Information and Control, 8(3):338–353, 1965.

[14]. Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man and Cybernetics, 3(1):28–44, 1973.