HAMDAN MUSA YOUSIF BABIKER

MOBILE ROBOT PATH FOLLOWING AND OBSTACLES AVOIDANCE USING FUZZY LOGIC CONTROL

NEU 2019

MOBILE ROBOT PATH FOLLOWING AND OBSTACLES AVOIDANCE USING FUZZY LOGIC CONTROL

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
HAMDAN MUSA YOUSIF BABIKER

In Partial Fulfillment of the Requirements for the Degree of Master of Sciences in Electrical and Electronic Engineering

NICOSIA, 2019

MOBILE ROBOT FOLLOWING AND ABSTCLES AVOIDANCE USING FUZZY LOGIC CONTROL

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES
OF
NEAR EAST UNIVERSITY

By
HAMDAN MUSA YOSIF BABIKER

In Partial Fulfillment of the Requirements for the Degree of Master of Sciences
in
Electrical and Electronic Engineering

NICOSIA, 2019

Hamdan Musa Yousif BABIKER: MOBILE ROBOT PATH FOLLOWING AND ABSTCLES AVOIDANCE USING FUZZY LOGIC CONTROL

Approval of Director of Graduate School of Applied Sciences

Prof.Dr.Nadire CAVUS

We certify this thesis is satisfactory for the award of the degree of Master of Sciences in Electrical and Electronic Engineering

Examine committee in charge:

Prof. Dr. Ahmet DENKER	Supervisor, Department of Mechatronics Engineering, NEU
Prof. Dr. Bülent BİLGEHAN	Department of Electrical and Electronic Engineering, NEU
Assist. Prof. Dr. Parvaneh ESMAİLİ	Department of Electrical and Electronic Engineering, NEU

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original of this work.

Name, Last name: Hamdan Musa Yousif Babiker

Signature:

Date:

ACKNOWLEDGMENTS

I would like to express my special appreciation and thanks to my advisor Asst.Prof. Dr.Ahmet Denker, you have been a tremendous mentor for me. I would like to thank you for supporting me in my research and for allowing me to finish my master study. Your advice always has been invaluable.

A special thanks to my family. Words cannot express how grateful I am to them especially to My Mother, My Father, Hago, Mohamed and Sara for all of the sacrifices that you've made on my

To my Beloved Mother SEETALBNAT…

ABSTRACT

The thesis presented deals with fuzzy logic control obstacles avoidance and path following of a mobile robot in the environment with concerning the design of a fuzzy controller and path following of a mobile robot to converge towards a target. In the tests, the behavior convergence towards the target was simulated by considering an environment with obstacles. The results obtained are quite satisfactory. Indeed, to simulate behaviors of the robot in an environment close to the real, a second test was carried out. In the latter, the environment considered has more obstacles and different shapes. The detection of these obstacles is done by range sensors. It transmits the states of the sensors by getting the angles of the obstacle and compare it with angles of the goal by using a fuzzy controller to avoidance obstacles. The operation generates outputs (left and right deviations) to divert the robot obstacles encountered.

***Keywords*:** fuzzy logic; obstacles avoidance; path following; kinematics

ÖZET

Bu kitapta sunulan çalışma, ortamdaki mobil bir robotu izleyen bulanık mantık kontrolü engellerinden kaçınma ve yol ile ilgilidir. Bulanık bir kontrol cihazının tasarımı ve mobil robotun bir hedefe yakınlaşmak için izleyeceği yol hakkındadır. Testlerde, hedefe yönelik davranış yakınsamaları engelli bir ortam göz önünde bulundurularak simüle edildi. Elde edilen sonuçlar oldukça tatmin edici. Gerçek dünyadan ikinci bir test yapıldı. İkincisi, çevre daha fazla engel ve farklı şekillere sahiptir. Bu engellerin tespiti menzil sensörleri ile yapılır. Engellerin açılarını engelin açılarını alarak iletir ve engellerden kaçınmak için bulanık denetleyici kullanarak hedefin açılarını karşılaştırır. Operasyon genel çıktısı (sol ve sağ sapma) karşılaşılan robot engellerini yönlendirmek için çıktı.
***Anahtar Kelimeler*:** bulanık mantık; engellerden kaçınma; izleyen yol; kinematik

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FLC:	Fuzzy Logic Control
ANN :	Artificial Neural Network
GA:	Genetic Algorithm
ANN :	Artificial Neural Network
ANFIS:	Adaptive Neuro-Fuzzy Inference System
NN:	Neural Network
MOGA	Multi Objectives Genetic Algorithm
FIS:	Fuzzy Inference System

CHAPTER 1
INTRODUCTION

1.1. Background

Human has always sought to design entities in the image of the human being on which he could exercise his authority, thus fulfilling his least desires. The first approaches date back to medieval times when automata were placed in museums to impress the peasants, fascinated by these "supreme beings". Then Automata become popular in the 18th century in miniature form. The first work in mobile robotics in an industrial environment did not confer on robots that travel autonomy was very limited and necessitated important and expensive infrastructure works. This was the case, in particular, of the trolleys wire-guided, limiting the movement of the robot to lanes reserved for it.

Nowadays, there has been developing an interest in the robotics community, the development of intelligent and autonomous systems in the context of mobile robotics. Such interest can be seen as a consequence of the appearance of applications potential (cleaning, mobility assistance for disabled people ...) and the desire to robots on new spots such as exploring sites inaccessible to humans (planets, ocean floor) or operate in environments that are hostile to it (media radioactive).

In such contexts, providing robotic systems with a decision-making capability and, more specifically, planning their path or navigating autonomously remains one of the key elements in the implementation of their autonomy. Indeed, the autonomous robot must be able to make decisions, as to the movements to be made, according to the information available on the environment or from its sensors. For this, there are a variety of sensors (video cameras, rangefinder, ultrasonic or infrared sensors, etc.) and a navigation system capable of handling the different situations already cited.

To this end, the major concern is to develop effective navigation techniques, such that safety is a priority over optimality. The navigation strategy can integrate information, a priori, available on the environment, in the form of a model of the latter. However, when the environment becomes more complex (partially known, dynamic…), it seems indispensable that the robot is endowed with decision-making capabilities able to react to the hazards that can hinder its movements.

There are Several solutions have been planned for the autonomous navigation of a robot mobile, However, this navigation is in any case quite limited, encountering difficulties that make it specific to a specific situation. So here path following is an important mission for a mobile robot. It consists of calculating the commands sent to the actuators to perform the planned movement, taking into account the location of the mobile robot for good control of our mobile robot.

1.1.1. Path Following and Obstacle Avoidance With Fuzzy Logic

The problem to be treated in this memory is the reactive navigation for a robot mobile in an environment using the principle of fuzzy logic. The task entrusted to the robot is the achievement of targets without any external intervention, and this, in an environment on which little prior information is provided. The robot is brought to exploit the information from its sensors to be able to carry out these tasks and to guide oneself so as to avoid the obstacles that may arise in its path. In this context, several applications have been developed to ensure reactive navigation in a totally unknown environment such as a potential field. In work, we opted for fuzzy logic.

For twenty years, the implement of fuzzy logic in the modeling and control of complex systems has become a tool in its own right. The concept of fuzzy sets was introduced by(Zadeh, 1965). It constitutes a command interface for the modeling of natural language, in particular, linguistic concepts used by the expert of a process.

The principle of fuzzy control was tested for the first time, on a steam turbine by Mamdani and Assilian in 1974. The first large-scale industrial application, dating from 1987 in Japan, in the rail (metro) transport of Sendai (Tokyo), whose performance rivals that of a conventional control system.

Fuzzy logic control includes segments as inference mechanism and many semantic IF-THEN rules for encoding the behaviors of the mobile robot while the fundamental problem in designing an FLS is an effective definition of the rules of the fuzzy. It is challenging to deliver the resulting parts without master learning. Three stages of fuzzy decision making are shown in figure (Choomuang & Afzulpurkar, 2005)

1) Fuzzification: Its step of converting the input data into fuzzy sets that make it easy for inference mechanism to use it while applying the rules suction will be activated

2) Rule base: it’s a set of input data that contains IF-Then rules that rules is have a quantification of the possibility of the best control.

3) Defuzzification: This is the operation of converting fuzzy sets output of the interface mechanism to a crisp input that translated to actions.

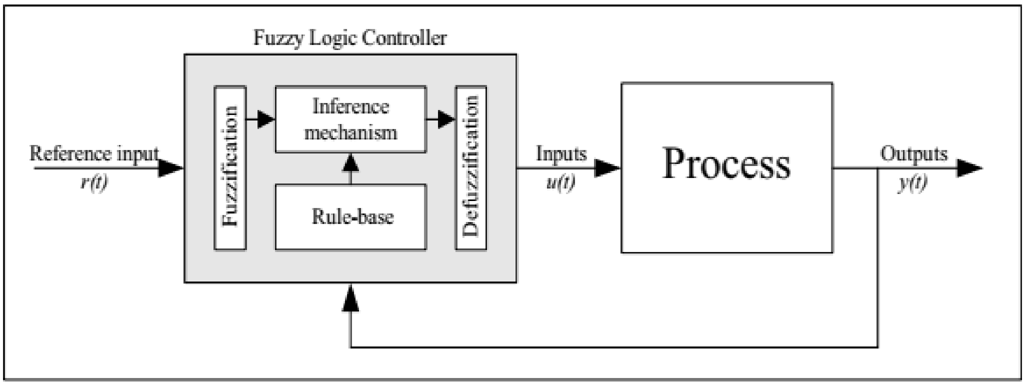


Figure 1.1: Fuzzy decision making

Nowadays, the fields of application of fuzzy control are becoming more and more important (industry, automobile, robotics ...), and can be classified into two categories:-

-The design of controllers for processes that are difficult to model.

- Controller design for nonlinear modeling processes.

1.2. Statement of the Problem

The solution of the path following and obstacles avoidance problem of a mobile robot is challenging, in many studied, In this research, one of the primary options that must integrate mobile robot is the ability to move and retch to targets that are indicated to it at prior without any external intervention. In addition, in a partially known environment, annoying objects may arise. The robot must have a second behavior that allows it to avoid these obstacles. In our case, these methods that used is fuzzy logic.

1.3. Thesis Objective

To develop a path following and obstacles avoidance controller using fuzzy logic control that gives the mobile robot the ability to navigate safely in the environment while going to the desired position.

CHAPTER 2
LITERATURE REVIEW

In a research made (Yousfi, et al., 2010) the researcher proposed a fuzzy logic controller for controlling the wheeled of a mobile robot in an unknown environment. The researcher used a gradient method to optimize the consequences of a Sugeno fuzzy logic optimal controller, in this paper the researcher presented the problem formulation. The researcher study based on the widespread robot (Khepera II, the robot contains two wheels the able of oriented and commanded by velocity in each wheel, the paper gives the kinematic model.

Furthermore, the researcher described the fuzzy controller design, additionally the FLC comprised membership functions, fuzzifier, an inference engine, control rule base, and defuzzifier. the proposed controller allowed the robot to reach the desired position starting a given orientation, an odometer module used to calculate the position and orientation of the robot in real-time, the input of the controller is distance which is the distance between the robot center and the target point, and the angle which is the difference between the robot direction and the argument of the target .fuzzy partition of input variables. Gaussian membership function used for fuzzy quantitation of the variables, the input is d: VL: Very Large; L: Large; M: Medium; S: Small and V S: Very Small .while the value of the angle is EZ: Equal Zero; PS: Positive Small; PM: Positive Medium; PL: Positive Large**.** The output of the controller are speeding VR and VL are quantified into five fuzzy subsystems = (Zero), (Small), (Medium), and (Big) and (VB: Very Big).

The researcher also depicted the (FLC) optimization using a gradient method, it’s an algorithm that expresses the distance between the desired position and the actual position and also used to optimize the rule consequence. the paper contains the obstacle avoidance problem after and before optimization, the robot has three sensors, two in the sides of the robot one in front of the robot to detect obstacles in three direction, therefore the proposed control technique has two fuzzy logic controllers one for reaching the target and the other on for avoiding the obstacles **.**finally, simulations carried out for small values of speeds the target in the environment delivered with the obstacle avoidance., and the results show the Optimized trajectories with obstacle avoidance

Also, in a research (El-teleity, 2011) the researcher approached a simple and powerful fuzzy logic technique to solve the problem for the same robot in the (Yousfi et al., 2010) , therefore he devised the hierarchical behavior-based control into 4 reactive behaviors to combined with a fuzzy supervisor, in this paper the hierarchical FC system proposed with target reaching behavior and obstacles avoidance behavior that carried out using two FLCs , the first behavior is dealing with position between the target (FLC1)and the robot and the second behavior used IR sensor signals(FLC2) , third behavior had been added it to avoid trap therefore other two fuzzy controller deal with "left wall follow"(FLC3)& right wall follow(FLC4) to achieve and avoid trap by leading the robot to follow either right or left wall to exit from that situation while the previse two controller is still working to get to the desired position .

According to this paper (El-teleity, 2011), the FLC2 received the input signals from 8 IR sensor to control the speed of the wheels to avoid the obstacles or go directly to the desired position in the case of clear way. In this case of membership function, there was 3 input for the behavior, 5 fuzzy sets included so the total number of rules are 125.

In short, the results show that the robot is capable of reaching the target and avoid the obstacles but without having the environment will lead to a non-optimized path.

Indeed, autonomous mobile robots are attractive many researchers and the main problem was to design an efficient and reliable collision and obstacle avoidance method for the purpose of implementing an effective navigation system. In (SHITSUKANE, 2018) the proposed system constitutes a fusion model based on fuzzy logic for collision and obstacle avoidance. The proposed model uses 8 distance sensors, 27 fuzzy roles, and 2 output variables for the robot navigation process. The researchers modeled two-wheeled mobile robots in order to navigate the environment, and the wheels are controlled by pulse with a modular controller which are DC motors and also a castor stability mechanism that will help the robots to be at stable state as much as possible. The 8 sensors are responsible for quantifying the distance around the robot from the right, left and from obstacles. Therefore, based on the distance information gathered from the 8 sensors the fuzzy rules are applied and activated.

The output of the applied fuzzy rules is combined using the fuzzy logic operation to determine and control the velocities, steering angles the robot wheels. The flow of the navigation process presented by the proposed system is, first we initialize the position of the robots then, the mobile robots will be starting navigating to the given position. While moving to the given destination position if the robot encounters an obstacle the robot will accept distance information about the obstacle from the 8 sensors with respect to the current position of the robot. After getting distance information the robot will get the velocity common after activating the fuzzy rules then finally the robot will adjust its wheels based on the given velocity commands. The researches also performed a simulation for the purpose of validating the feasibility and efficiency of the proposed system using V_REP and MATLAB.

The researchers compared the simulation results of the proposed system with other model having three sensors the result was easily differentiable. Based on the comparison results of the researchers both models are successful in avoiding collision and obstacles however, the proposed system achieved it with much smaller time. Their results show us that the robot path planning and navigation with obstacle and collision avoidance can be achieved with simple fuzzy logic rules without applying a complicated mathematical system.

Additionally, navigation task for in the dynamical Technique for a mobile robot is very difficult exactly when it’s come to path planning in an unknown environment, and this can be solving in a different way and challenging in the same time to carry out with accurate equations. An artificial neural network to expressed and solve the navigation problem proposed in a research (Xiao, 2007) for a mobile robot, artificial neural network is a type of information processing method of the verve simulation of the biology that can deal with systems and processes which are really difficult to be defined by rules or models.

This research (Xiao, 2007) had an input layer, an output layer, and a hidden layer, it assumed of a multi-layer feed-forward artificial neural network, toward the multi-layer, will solve the nonlinear classified matters.

The mobile robot (pioneer 3) has 8 ultrasonic sensors reader as inputs of ANN and the five actions of the motors are the outputs of the actions so that artificial neural network need 5 output neurons a result of 10 input neurons.

The researcher used Q- learning to focus on the behavior of the path planning, Q- learning is one best learning method that is broadly used because it’s simple and gives an optimal result.

To find the series state action the researcher used Q-learning to make Q(S, M) maximum

The ANN control method it caring the oath training of the data by using Q-learning and also weights connection, the result shows simulations demonstration the method shows a good resolving means of this control problem.

In a research made(Patiln & Carelli, 2004) the researchers proposed an optimal autonomous mobile vehicle controller using a neural network-based approach. The main objective of the proposed system lies with developing an optimal and optimized autonomous controller that guides the center of the vehicle based on a given number of path points or nodes in a particular order and with the use of the required minimum amount of time.

Basically, according to the researchers, the guidance of a robotic based vehicle is highly dependent on the interaction that occurs between the robot and the environment/medium that the robot will operate. Also, the proposed system is responsible for planning the coming encounters through given points. Hence, the movement of the vehicle through the gates highly changes the process of navigation in the upcoming points.

Since the proposed system is ANN it learns from a set of points anonymously and can guide the vehicle accordingly as explained by the researchers. Basically, the motion controller of the proposed vehicle control system is classified into three different classifications which are positioning the vehicle without any prescribing orientation, positioning the vehicle a specific prescribing orientation, and path following.

In the first classification group, the final destination point is specified for the vehicle controller and in the second case the destination point should be obtained with the desired orientation and in the third case, a path should be defined as a sequence of points. The researches also present the simulation results of the proposed model, the used a feedforward artificial neural network and their simulation results show that we can design an autonomous vehicle controlling system using an Artificial Neural Network (ANN) model which is an intelligent mode and it can learn and guide the vehicle optimally.

Furthermore, in another research (Kim & Chwa, 2015) the researchers conducted an obstacle avoidance approach for wheeled mobile robots based on interval type-2 fuzzy neural network (IT2FNN). Previously the researchers conducted an obstacle avoidance technique for wheeled robots using the type-1 fuzzy neural network (T1FNN). Even though the previous model which is based on T1FNN was successful in performing the required tasks, its performance was not that much appreciable and efficient. Therefore, the main aim of the proposed model IT2FNN is to improve the problems of the previous T1FNN model.

The T1FNN approach has two main problems which are it can't effectively reduce the influences of the external and internal uncertainties as a result of the crisp set member function and sometimes it might result with a large amount of oscillation behavior while it is performing obstacle avoidance. Therefore, the proposed system is designed under the considerations of improving those performance problems of T1FNN with the help of smoother behavior and improved obstacle avoidance process.

The major problem of the type-1 fuzzy neural network (T1FNN) model was it uses a crisp set. However, the proposed system uses a fuzzy set which will results in robustness against uncertainties. The process of obstacle avoidance can be incredibly improved particularly at the existence of obstacles. The researchers have presented comparison results of both simulation and experimental results performed on real-time wheeled mobile robots. The researchers compared both the simulation and experimentation results of the T1FNN and IT1FNN based on their final distance error and moved distance

The simulation results of the final distance error and moved distance for the T1FNN are 0.0677 and 1.4149 respectively. Results of the final distance error and moved distance for the IT1FNN are 0.0523 and 1.4101 respectively. Furthermore, the experimentation results of the final distance error and moved distance for T1FNN are 0.0438 and 3.5625 respectively. Furthermore, the experimentation results of the final distance error and moved distance for IT2FNN are 0.0400 and 2.7984 respectively.

As we can clearly see the results of the simulation and experimentation, we can conclude that the robot with interval type-2 fuzzy neural network (IT2FNN) can achieve a better position stabilization and obstacle avoidance than the previous T1FNN. The proposed model can guarantee us a smaller final distance error and that the T1FNN model and also a smaller moved distance. These results tell us that the robot with IT2FNN is able to obtain a smoother obstacle avoidance movement while following the shorter path.

Additionally, in another research paper (Ghorbani & Nodehi, 2009)the research's proposed a genetic algorithm (GA) based approach for mobile robot path planning purpose. Path planning in a static environment for a mobile robot is an important and most challenging problem therefore, the paper focuses on presenting an optimal and global genetic algorithm-based path planning technique for mobile robots. For the purpose of minimizing unwanted complexity and required resources, the researches converted the 2-dimensional coding for the path into 2-dimensional coding. Also, the fitness of both the shortest path and collision avoidance path are integrated into a fitness function which helps us in determining the current condition of the genetic algorithm.

Genetic algorithm is the most robust and efficient searching and optimization algorithm, especially for complex problems. In the genetic searching algorithm, the most important thing is the coding technique so, the points of the path are converted from 2d - to 1d in order to boost and reduce the complexity of the genetic algorithm. The main aim of the designed genetic algorithm is to find out the points of the path constitute an optimal path from the beginning to the global position. Moreover, the fitness function is an important attribute for the stability and convergence of the proposed genetic algorithm.

The simulation results of the proposed system are also presented by the researches in the paper for the purpose of proving the efficiency of the proposed system. The simulation results are presented in three different groups of the simulation environment which are simple, middle, and hard. In the simple simulation environment, the result of the best and mean fitness is 16.71 and 16.71 respectively. In the middle simulation environment, the result of the best and mean fitness is 14.44 and 14.48 respectively. In the hard simulation environment, the result of the best and mean fitness is 16.47 and 16.47 respectively.

Their experimentation results show us that the proposed system is successful and efficient in global path planning. Therefore, this proposed system indicates the advantages of intelligent path planning algorithms.

In another research (Castillo, et al., 2007) the researches used a similar GA approach with the research made (Ghorbani & Nodehi, 2009) for path planning of offline point-point autonomous mobile robots. The problem which arises in the path planning of these autonomous mobile robots is from the genetic paths or trajectories for a Holonomic Robot while moving from the starting point to the destination point. The path involves a 2dimentional grid with obstacles and dangerous ground that the autonomous robots should follow. The main goal of the proposed genetic algorithm model is to efficiently optimize based on two different criterions.

These two different criteria are the length or distance of the path and the difficulty or the condition and existence of the obstacles. The researchers first try to solve the pathing and optimization problem using only a single criterion. Then the researchers conducted the same process with the two criteria's this makes the genetic algorithm a multi objectives genetic algorithm (MOGA). As we know it a real-world problem involves the optimization of more the two objectives at once so the proposed system focus in the optimization of multiple objectives at once. However, with multiple objectives there comes to another difficult problem which is when we have more objective to optimize it sometimes becomes impossible to obtain an optimal and efficient solution always.

Basically, there are different ways of solving multi-objective problems one example of this is a linear combination of the objectives by creating a single objective optimization function. However, the proposed system uses the concept of Pareto optimality and Pareto optimal set. These concepts are mainly useful in allocating resources in which it is impossible to reallocate so as to make anyone individual criterion better without making others criterion worse. The researchers also presented the simulation and experimentation result of the proposed system in two different sections one for each proposed mode for the single objective and for two optimization objectives

The experimentation results for the single objective optimization shows us that the system has a 3 to 5 percent performance improvement, also as while the number of cells in given path increase the valid solution is the use of a generational replacement scheme for the Genetic algorithm with dramatically increase the performance improvement percent into 44 percent. Additionally, the experimental results for the two optimization objectives are satisfying, the use of an elitist replacement strategy along with a Pareto based approach for multi or two objective problems gives many satisfying results. The researches present the result of the proposed system with two different performance measurements.

The results of the MOGA generational and MOGA elitist for the first performance measurement is 32 and 34 respectively. And the results of the MOGA (generational) and MOGA (elitist) for the second performance measurement is 48 and 100 respectively. As the results clearly describe for both measurements the MOGA elitist shows better and improved results.

Furthermore, in another research (Arora, 2014) the researches presented a genetic algorithm (GA) based mobile robot path planning for a point in the space rather than the entire space. In general robot path planning focuses in the process providing an optimal path in terms of path length and execution time for the process of reaching end position from the start position while dealing and avoiding obstacles and in any environment containing obstacles. And this optimization problem can be solved and achieved using the most known and global optimization technique especially for larger problems which is a genetic algorithm. The main difference between the proposed system and the systems proposed by the above papers is that the proposed system focusses in the optimization of a single point from the given space than the entire space points.

The proposed system works in such a way, in which at first the robot will start moving from the initial point or source and follows the diagonal path to the final or destination point, in which the diagonal is the minimum path to the destination point in a rectangle space. The direction is calculated as $\theta = \tan\left(\frac{x1}{y1}\right)$ in which x1 and y1 are coordinated and they are calculated as $x1 = x1 + cos\theta 1$ and $y1 = y1 + sin\theta 1$. While the robot is moving diagonally or in the minimum path if the robot faces an obstacle between the robot and the destination point the robot takes 3 steps back and will reach the new position coordinate (x, y) and it uses the genetic algorithm to determine and optimize the next obstacle to free the path. The researches implemented the proposed genetic algorithm using MATLAB and they presented the CPU time used to execute the algorithm based on a number of iterations.

For iteration of 10, 20, 30, and 40 the CPU time taken by the genetic algorithm are 26,102,122, and 139 respectively. Finally, according to the researcher's conclusion and comparison results the simulation results are much efficient and better than results achieved by other robot path planning and collusion and obstacle avoidance systems.

Additionally, in another researcher(Pandey, et al., 2014) conducted path planning with collision and obstacle avoidance using a fuzzy logic controller for autonomous mobile robots. Similarly, with the other fuzzy-based path planning and collision avoidance models, the researchers used the fuzzy model to improve the motion of the mobile robots under the consideration of the positions of obstacles by defining input and output variables, fuzzy member functions, fuzzy inference rules, and Defuzzification. Then it will plan to the destination position but, the navigation should identify all the obstacles to create an obstacle-free path to reach the final destination. This obstacle and collision avoidance can be achieved by changing the angle and direction of autonomous mobile robots. The proposed fuzzy model has three inputs which are obtained from the sensors found at the left side, right side, and inform of the robot. Then it uses the three inputs to determine the speed and directions of the right and left wheels.

The main aim of the fuzzy model is to guide the robot safely in known or partially known space starting from the start and destination points which mean we need to have a full or partial map of the environment. The three inputs which are the distance between the robot and the obstacle are obtained from the left obstacle distance, right obstacle distance, and front obstacle distance respectively and they are divided into a three triangular membership fuzzy functions which are "close" to indicate the obstacle is too near, "Medium" for indicating that the obstacle is at medium distance from the robot, and "far" for indicating that the obstacle too far from the robot. Also, the model has two output variables which are right motor speed(RMS) and left motor speed(LMS and they are divided into three triangular fuzzy membership functions which are "fast" for indicating that the motor should fast, "medium" for indicating the motor speed should be medium, and "low" for indicating the motor speed should below.

The researches designed a knowledge base using the fuzzy membership functions and they used the knowledge base to develop the fuzzy controller rules. The fuzzy inference system

(FIS) collects the information of the input sensors and gives out the output data until it reaches the final destination. The researches performed a simulation using MATLAB and the results are were satisfying according to the researchers report. The robots are able to reach the destination and also avoid the collusion and obstacles successfully.

In another related research made(Authors, 2016) the researchers proposed a new neural network architecture-based algorithm for the mobile robot navigation system in a cluttered and unknown environment. The general purpose of the proposed system is an online path planning approach for wheeled mobile robots which are based on behavior in an unknown environment using a feedforward back-propagation neural network control technique. The main aim of the proposed model is to find out a path that doesn't involve collusion or obstacle to the destination in a cluttered environment and follow that pass. The input of the feedforward neural network architecture is the different set of distance information which is obtained from the sensors and the output of the neural network architecture is the turning angle of the control motor of the robots' wheels. In order to make the robot able to move in the environment autonomously, the feedforward neural network is trained using the backpropagation algorithm.

Similarly, with any feedforward neural network architecture, the proposed neural network mode has three components which are input, hidden and output layers. The inelegancy of any neural network is highly affected by the training data quality, and the main of the neural network archit4eccture in the proposed model is to for the purpose of path planning and obstacle/collision avoidance. The input layer of the proposed NN architecture has three nodes/neurons in which their purpose us to receive distance information of the hurdle form the left, right and front sensors. The input layer of the neural network is completely connected with the hidden layer and the hidden layer is connected with the output layer having only single neuron which will tell us the turning angle of the robot wheels. The proposed neural network architecture is trained with a list of training data constituting 5 different inputs for each training recorded.

A single training record has the distance of the hurdle from the front sensor, right sensor, left sensor, the turning angle, and turning direction which is either left or right. The general flow of the proposed system is first the robot calculates the distances between the robot and the hurdles and based on the information provided by the sensors the robot will check if the hurdles are close and if they are close the robot will calculate the turning angle and target distance. However, if there are no hurdles near the robot the robot will continue the default navigation process until it reaches the destination or reaches hurdles. Also, the researchers carried out simulation and experimentation for proving the efficiency of the proposed neural network-based path planning and hurdle avoidance system. The researches the results of both simulation and experimentation using time taken to reach the destination, and the total path length in pixels.

For the purpose of only pathfinding behaviors for a mobile robot, the robot took 15.26 seconds to move through 560.43 pixels in simulation and it took 16.10 seconds to move through 578.41 pixels in experimental results. The error difference between the experimentation and simulation result for pathfinding is 5.50 percent. Furthermore, for hurdle avoidance behavior for mobile robots, the robot took 20.41 seconds for covering 655.21 pixels path in simulation with avoiding hurdles and it took 21.70 seconds to cover 669.55-pixel distance while avoiding hurdles too. And the error difference between the experimental and simulation results for hurdle avoidance behavior mobile robots is 6.32 percent. As a conclusion, the researches stated that the feedforward neural network algorithm obtains a satisfactory result in both simulation experimentation.

Additionally, in another research (Al-Mayyahi, et al., 2014)the researchers come up with a solution for navigation problem of an autonomous ground vehicle using an adaptive neuro-fuzzy inference system (ANFIS). The proposed system is composed of four ANFIS controller, out of the four controllers two of them are used for controlling both the left and right velocity of the AGV in order to reach the destination position. And the rest two controllers are used for obstacle avoidance using optimal heading adjustment. The first 2 controllers will receive input distance information from three different sensors which are found at the left, front and right of the robot. The proposed adaptive fuzzy neuron architecture design constitutes fuzzy inference engine and neural network including a number of input and output information. This adaptive neural network model will give the fuzzy logic the ability to easily implement the fuzzy membership functions that can beastly allow the related fuzzy inference system to track the provided input and output data for the adaptive-neuro inference system model.

For the purpose of processing and implementing the fuzzy rules by neural networks, the default and traditional neural networks architecture must be updated. The proposed neural network architecture is consisting of five layers in which in the first layer every node is an adaptive neural node. The second layer constitutes of fixed nodes whose output is the products of all the previous or input signals. And every node in the third layer is a fixed node which calculates the ratio of the rule to the sum of all the rules. In the fourth layer, every node is an adaptive node with a node function and finally, the fifth layer contains a single fixed node which is responsible for computing the total output by calculating the summation of all the input incoming input signals. As we know it the quality and efficiency of any neural network architecture can be highly affected by the quantity and quality of the training data, the proposed neural network is trained with the prepared dataset.

The training data for navigation contains a list of information that will be trained to the neural network and this data contains angle difference, right angular velocity and left angular velocity. The neural network is trained with 200 epoch values also when the number of epoch values increases the trained error decreases. Similarly, the proposed model should avoid obstacles also the neural network should be trained to avoid obstacles. A list of information is prepared to train the proposed neural network to avoid obstacle and the data contains information such as front distance form front sensor, the right distance from the right sensor, left distance from the left sensor, right angular velocity, and left angular velocity. Similarly, with navigation training, the neural network for obstacle avoidance is trained with 200 epoch values whenever the epoch value increases the error rate decreases. The researchers performed a simulation process for proving the efficiency of the proposed model using a MATLAB-SIMULINK environment. According to the researchers report the results were satisfactory and efficient in both navigating the robot and avoiding obstacles.

CHAPTER 3
PROPOSED METHODOLOGY

3.1. Kinematics

Kinematics is a representation of mechanical systems that has an understanding of the system behave in order to design and creating a control system of a mobile robot. The workspace of the robot is important because of the position and range poses definitions for the mobiles robot to realize in the environment, The goal of kinematic robot modeling is to find the speed of the robot in the inertial frame according to the wheel speeds and the geometric parameters of the robot (configuration coordinates).

This chapter will show the kinematics of the robot and the design model. The understanding robot's motions start with the description of each wheels contributions that the wheels deliver for motion, in the first section, the expression of the robot motion with respect to the moving frame and the reference frame of the robot.

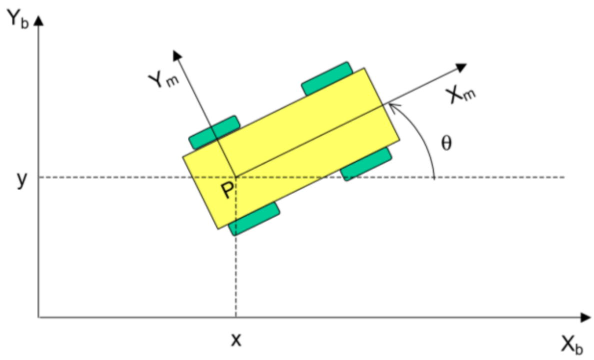


Figure 3.1: Robot in word frame

Xm and Ym are the moving frame while Xb and Yb are base frame The position of the robot in the word frame and the robot can be defined as following while P is the position in word frame

$$P = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix} \tag{3.1}$$

The Rotation matrix expressing the orientation of the base frame with respect to the moving frame, To know the orientation in the base frame as for the moving frame , the rotation matrix help to express that (Aouf, et al., 2019):

$$R(\theta) = \begin{bmatrix} \sin\theta & \cos\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Using the equations above, we have the relationship between robot speeds in the local coordinate system and the word frame which is very important in the kinematics of the robot.

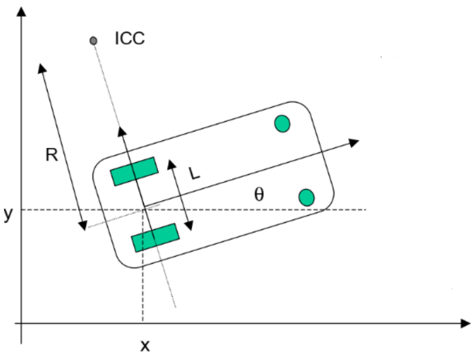


Figure 3.2: Instantaneous Center of Curvature

$$ICC = (X - R\sin\theta\,, Y + R\cos\theta) \tag{3.3}$$

(Carlos, et al., 2017) Our controlling variables are$Vr(t)$ the linear velocity of the right wheel and $Vl(t)$ the linear velocity of left the heel while r is the radius value of the wheels and R is the instantaneous curvature radius of the robot trajectory, relative to the mid-point axis. Left and right wheels curvature radius of the trajectory are $R - \frac{1}{L}$ for the left and $R + \frac{1}{L}$ for the right wheel.

Wheel parameters of Wheel are r which is wheel radius, v is wheel linear velocity and ω is wheel angular velocity.

The angular velocity the right and the left wheel as follow:

$$\omega r(t) = \frac{Vr(t)}{R+\frac{L}{2}} \tag{3.4}$$

$$\omega l(t) = \frac{Vl(t)}{R - \frac{L}{2}} \tag{3.5}$$

$$\omega(t) = \frac{Vr(t) - Vl(t)}{L} \tag{3.6}$$

$$R = \frac{L}{2}\frac{Vl(t) + Vr(t)}{Vl(t) - Vr(t)} \tag{3.7}$$

Therefore, the robot linear velocity is:

$$V(t) = \omega(t)R = \tfrac{1}{2}(Vr(t) + Vl(t)) \tag{3.8}$$

The forward and inverse kinematics equations in the world frame can be represented as follows.

$$\dot{X} = V(t)\cos\theta(t) \quad \approx \quad X = \int_0^t V(t)\cos\theta(t)\,dt \tag{3.9}$$

$$\dot{y} = V(t)\sin\theta(t) \quad \approx \quad y = \int_0^t V(t)\sin\theta(t)dt \tag{3.10}$$

$$\dot{\theta} = \omega(t) \quad \approx \quad \theta = \int_0^t \omega(t)dt \tag{3.11}$$

The shown equation can be written in the matrix for as follow:

$$\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}V(t)cos\theta & 0\\ V(t)sin\theta & 0\\ 0 & 1\end{bmatrix} \begin{bmatrix}V(t)\\ \omega(t)\end{bmatrix} = \begin{bmatrix}V(t)cos\theta\\ V(t)sin\theta\\ \omega(t)\end{bmatrix} \tag{3.12}$$

Since the control variables are Vr and Vl $\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{\theta}\end{bmatrix}$ will be

$$= \begin{bmatrix}(Vr + Vl)cos\theta/2\\ (Vr + Vl)sin\theta/2\\ (Vr - Vl)/2\end{bmatrix} \tag{3.12}$$

3.2. Modeling and design

In order to design behaviors or controller for a robot, It is important to have models of how the robot behave for this research the differential drive wheeled mobile robot has two wheels and two freewheels , the two wheels can turn at difference rates , to make the robot move around , in the Figure 3.2 , the dimensions of the robot should be known in the design , somehow for many designs the controller does not need the parameters because of not knowing the friction coefficient, in these design the wheelbase (L)should be known and the radius of the wheel (R) , the two parameters are important to design the controller of the robot .

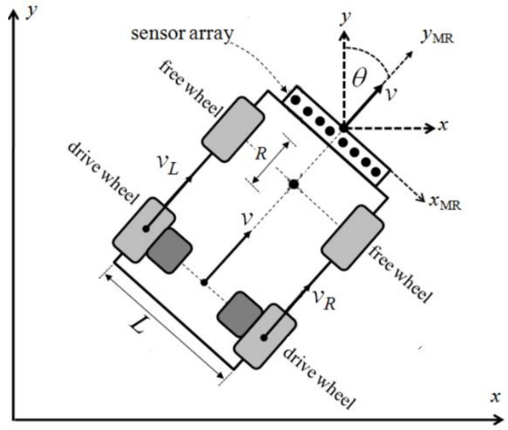


Figure 3.3: The Differential Drive Wheeled Mobile Robot

The control signals that we what to control are VL and VR to the system, and the state of our the location(X, Y) of it and the orientation (θ) of the robot, the robot model needs to connect the control input Vl &Vr to the state to do the transition, the kinematics of the differential drive model as follow (Gyawali & Agarwal, 2019).

$$\begin{pmatrix} \dot{X} = \frac{R}{2}(VR + VL)\cos\theta \\ \dot{Y} = \frac{R}{2}(VR + VL)\sin\theta \\ \dot{\theta} = \frac{R}{L}(VR - VL) \end{pmatrix} \tag{3.14}$$

This the model need it in term of mapping control inputs onto states, while the unicycle model of the robot and the dynamic of it represented as follow and the input V and ω

$$\begin{pmatrix} \dot{X} = V\cos\theta \\ \dot{Y} = V\sin\theta \\ \dot{\theta} = \omega \end{pmatrix} \quad (3.15)$$

Implementing this model to the differential drive model, by identifying $\dot{X}$,$\dot{Y}$ and ω by solving the equation we get an equation that connects the translational velocity V to wheels velocity.

$$\frac{2V}{R} = VR + VL \; and \; \frac{\omega L}{R} = VR - VL \quad (3.16)$$

Solving this two-equation for (VL ,VR) we see that V and R are designed perimeters and L and R are measured parameters for this robot.

$$VR = \frac{2V+\omega L}{2R} and \; VL = \frac{2V-\omega L}{2R} \quad (3.17)$$

And by a having this values it is easy now to map the designed inputs(V,ω) the actual input (VL ,VR) that running in the robot.

3.2.1. Developing Behaviors of the Robot

The environment around the robot is not known for the robot that makes the design to not care about how to act optimally by having their assumption about the environment that might be important manufacturing plant industrial robot when the robot is repeating the same motion many times. In this study, the robot is exploring and navigate in the environment without acting optimally to achieve that two behaviors have been taken into account path following behavior and obstacles avoidance behavior using fuzzy logic control.

3.3.2. Sensors of the Robot

Additionally, after having a model of the robot and known the Odomerty of it, the robot needs to know what around it in this case range sensors are used to measure the distance between the obstacles and the robot in the environment, there are many different sensing modalities and they all have the same king of abstraction, What really important is the abstraction of these sensors for this research the disk abstraction, the robot has its own coordinate system in the environment with having the information of the obstacles $(d_x, \emptyset_x)$, while $\emptyset_x$ is mustered relative to the robot orientation in the environment, and this will make it clear to know where the obstacles are global.

By known the robot orientation and position($X, Y\ , \theta$)and knowing the obstacles around the robot in the environment(d_x, θ_x), this abstraction that the controller designed around. The following equation will show where the global position of the obstacles is:

$$X_x = X + d_x \cos(\theta_x + \theta) \tag{3.18}$$

$$Y_x = Y + d_x \cos(\theta_x + \theta) \tag{3.19}$$

3.3.3. Path Following Behavior

In order to go between to point in the in-world frame of the robot, its need go to goal behavior and avoid obstacles behavior since we mode a differential drive robot as unicycle, by feed a constant velocity V_0 it's easy to control the angler velocity using the fuzzy controller for avoiding obstacles so if the robot finds a clear path it will use its maximum velocity.

$$\begin{aligned} \dot{X} &= V_0\ \cos\theta \\ \dot{Y} &= V_0\ \sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{3.20}$$

The controlling part will be $\dot{\emptyset} = \omega$ to define error here $e = \emptyset_d - \emptyset$

θ_d= desires angle

$$\theta_d = arctan\left(\frac{y_g - y}{x_g - x}\right) \tag{3.21}$$

While the distance to the desires position is measured with following equation (Singh & Thongam, 2018)

$$D_d = \sqrt{\left(x - x_g\right)^2 + (y - y_g)^2} \tag{3.22}$$

The angles issue is real take to account here by ensuring that the error is always between $(-\pi, \pi\ )$,

To achieve that that in MATLAB is using function $atan2\ \ e \in \{-\pi, \pi\}$.

3.4. Obstacle Avoidance Behavior Using Fuzzy Logic Control

This part is dealing with obstacle avoidance behaviors after the robot has the ability to go to the target position it more challenging if we are designing a controller to do an elaborate job like combining both behaviors together. Fuzzy logic control has chosen to avoid obstacles while the robot is heading to goal the implementation of the fuzzy controller used toolbox in Matlab with a graphical interface. Mamdani method is computationally effective to use according to (Farooq & Hasan, 2011)

3.4.1. Mamdani Fuzzy Controller Design and Modeling

Mamdani fuzzy inference is the most commonly seen fuzzy methodology, in Mamdani's model the fuzzy implication is modeled by Mamdani's minimum operator, the conjunction operator is min, the t-norm from compositional rule is min and for the aggregation of the rules the max operator is used. In order to explain the working with this model of FLC will be considered the example where a simple two-input two-output problem that includes

Rule: IF x is A1 AND y is B1 THEN z is C1 AND w is D1.

In order to blend two behaviors and to get a smoother ride when it switching between behaviors, the designed fuzzy logic controller is controlling the motion of the robot .since the robot knows the angle of the obstacles by using the data of the range sensors, there are 25 sensor are set between $45°$and $-45°$. Since the robot knows the angles to the goal position. Therefore, using the comparison of the two angles to avoid the obstacle by controlling the left and right wheels output while keep trying to go to the target position.

The fuzzy logic controller that has implemented contains two inputs which the angles of the goal and the angle of obstacles ($Teta_G, teta\_Obstacle$) and two output the deviation value of the left and the right wheels ($LeftDeviation, RightDevitation$) since velocity in the case has the max velocity has a constant value

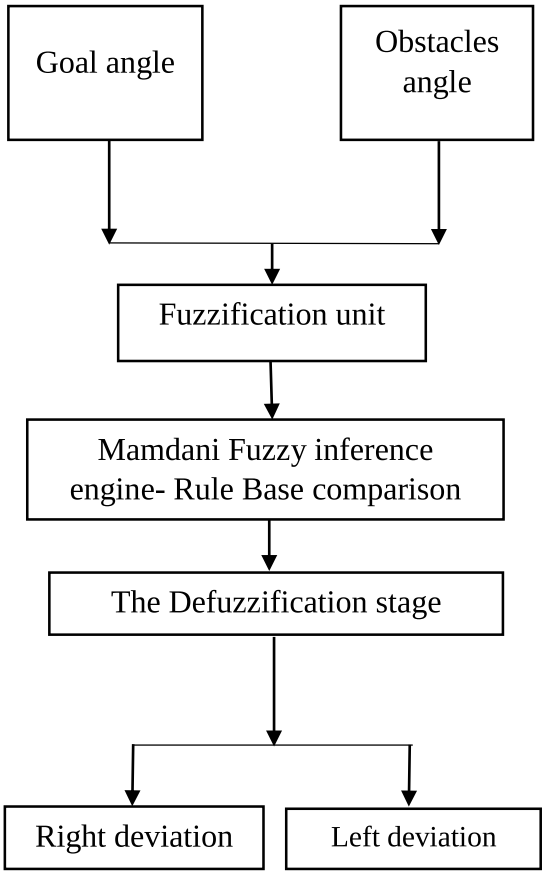


Figure 3.4: Block diagram shows the FIC the architecture of the obstacles avoidance behavior

The chart explains that the controller get the input of the angles as crisp values to the Fuzzification process that changes the scale of the input to fuzzy value which had been done with Mamdani fuzzifier, it used a range set and chosen membership function, it takes the crisp inputs to determine a degree that shows where it belongs in the fuzzy set then it evaluate the rule to apply the rule to the antecedents of the given fuzzy rules in the inference engine that output will not be used until its change from a fuzzy sets to crisp values again in the Defuzzification then the output will be decided action .

3.4.2Linguistic Variables of the Membership Function

Firstly, the identification of the fuzzy sets known as initialization and the identification of the linguistic output and input variables.

The linguistic variables of the membership function of the angle of the goal are N (Negative), NM (Negative Medium) Z (Zero), PM (Positive Medium) and P (Positive) with a given range of $(-1, 1)$ with triangular membership faction that gives is ringing as follow:

Table 3.1: Goal angle classification

terms	Range of Membership Function	Membership Function
N (Negative)	[-1.5 -1 -0.5]	Triangular
NM (Negative Medium)	[-1 -0.5 0]	Triangular
Z (Zero)	[-0.5 0 0.5]	Triangular
PM (Positive Medium)	[0 0.5 1]	Triangular
P (Positive)	[0.5 1 1.5]	Triangular

The linguistic variables of the membership function of the angle of the obstacles are LN, LF, NF, RF, and RN with a given range of $(-1, 1)$ with triangular membership faction that gives is rang as follow:

Table 3.2: Obstacle angle classification

terms	Range of Membership Function	Membership Function
LN(Left Near)	[-1.5 -1 -0.5]	Triangular
LF(Left Far)	[-1 -0.5 0]	Triangular
NF(Near Front)	[-0.5 0 0.5]	Triangular
RF(Right Far)	[0 0.5 1]	Triangular
RN(Right Near)	[0.5 1 1.5]	Triangular

The triangular membership function shape is selected as the inputs of the fuzzy controller the Fuzzification will convert the data coming from the crisp set to variables as follow:

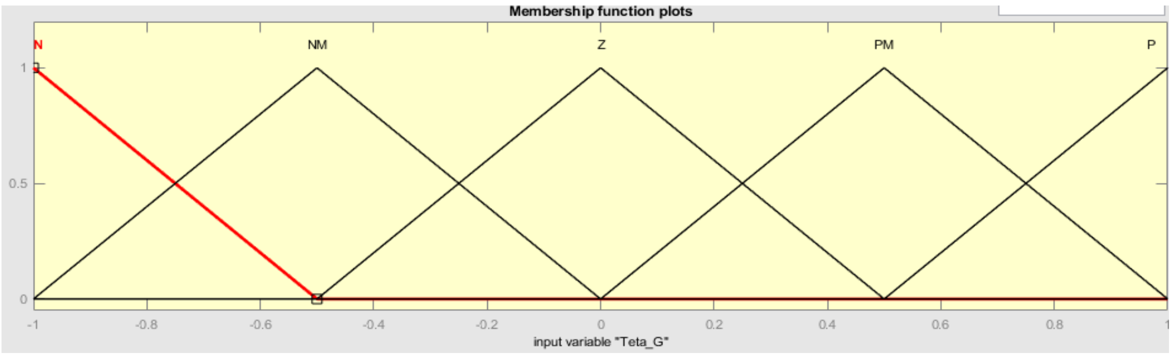


Figure 3.5: Membership function of the fuzzy logic of the input variable $Teta\_G$

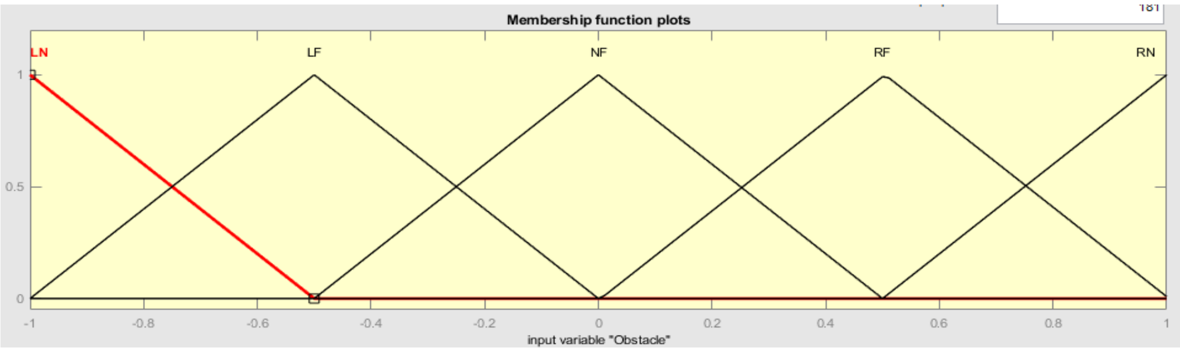


Figure 3.6: Membership function of the fuzzy logic of input variable $Teta\_G$

The output of the fuzzy controller is the deviation of the right and the left velocity , The linguistic membership function are VVN(Very-very Negative), VN (Very Negative), N(negative), NM(Medium Negative) and Z(Zero) for the left velocity (Vl) and similar to the right velocity (Vr) .

Table 3.3: Parameters of output variable for $Left\ Deviation$

terms	Range of Membership Function	Membership Function
(Very-very Negative) VVN	[-2.5 -2 -1.5]	triangular
(Very Negative) VN	[- 2 -1.5 -1]	Triangular
(Negative) N	[-1.5 - 1 -0.5]	Triangular
(Medium Negative) MN	[- 1 - 0.5 0]	Triangular
(Zero)Z	[-0.5 0 0.5]	Triangular

Table 3.4: Parameters of output variable for $Right\ Deviation$

terms	Range of Membership Function	Membership Function
VVN	[-2.5 -2 -1.5]	triangular
VN	[- 2 -1.5 -1]	Triangular
N	[-1.5 - 1 -0.5]	Triangular
MN	[- 1 - 0.5 0]	Triangular
Z	[-0.5 0 0.5]	Triangular

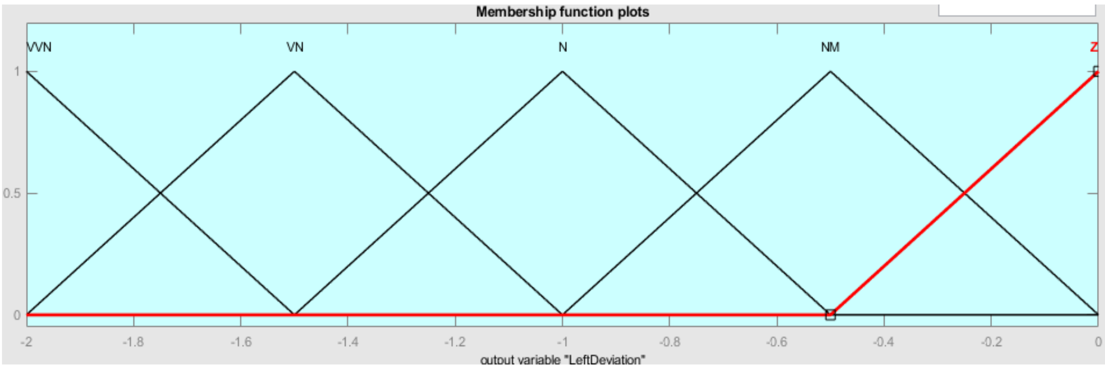


Figure 3.7: Membership function of the fuzzy logic of input variable $Left\ Deviation$

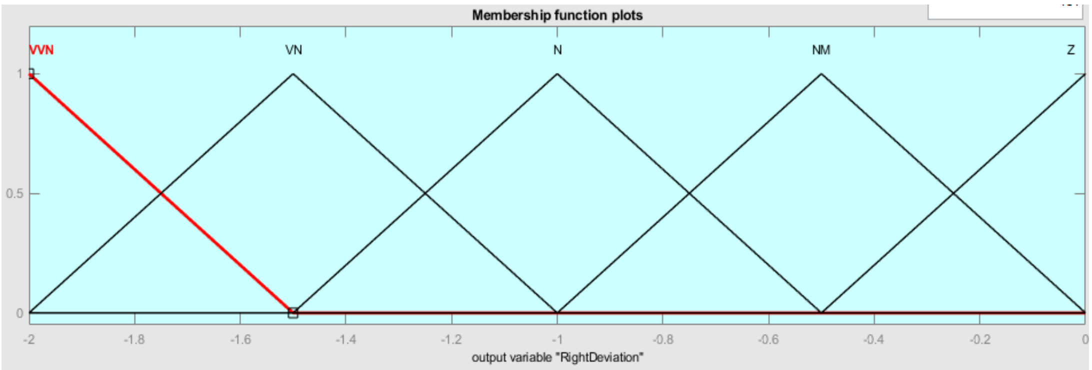


Figure 3.8: Membership function of the Fuzzy logic of input variable $Right\ Deviation$

3.4.3. Fuzzy Rules

In order to obtain the output of the controller as the Vl and Vr (rotational velocity), the sets of the membership functions of (FLC) has this two output, and there are 25 fuzzy rules bases had been generated for the 5-sets partitioning of the goal angle and obstacles angles . The operating rule for the membership function is stated in the Table (3.5).

Table 3.5: The list of the fuzzy rule table

Rules No.	Teta_G	Teta_O	Left deviation	Right deviation
1.	P	NF	Z	N
2.	Z	NF	Z	Z
3.	PM	NF	Z	NM
4.	NM	NF	NM	Z
5.	N	NF	N	Z
6.	Z	LF	Z	NM
7.	Z	LN	Z	VVN
8.	Z	RF	NM	Z
9.	Z	RN	VVN	Z

10.	P	LF	Z	N
11.	P	LN	Z	VVN
12.	P	RF	Z	NM
13.	P	RN	VVN	Z
14.	PM	LF	Z	NM
15.	PM	LN	Z	VVN
16.	PM	RF	Z	Z
18.	N	LF	NM	Z
19.	N	LN	Z	VVN
20.	N	RF	N	Z
21.	N	RN	VVN	Z
22.	NM	LF	Z	Z
23.	NM	LN	Z	VVN
24.	NM	RF	NM	Z
25	NM	RN	VVN	Z
	If		Then	

1. If (Teta_G is PM) and (Obstacle is LF) then (left Deviation is Z) (right Deviation is NM)

2. If (Teta_G is NM) and (Obstacle is NF) then (left Deviation is NM) (right Deviation is Z)

3. If (Teta_G is Z) and (Obstacle is LF) then (left Deviation is Z) (right Deviation is NM)

4. If (Teta_G is N) and (Obstacle is LN) then (left Deviation is Z) (right Deviation is VVN)

5. If (Teta_G is PM) and (Obstacle is NF) then (left Deviation is Z) (right Deviation is NM)

6. If (Teta_G is P) and (Obstacle is LF) then (left Deviation is Z) (right Deviation is N) (1)

7. If (Teta_G is Z) and (Obstacle is RF) then (left Deviation is NM) (right Deviation is Z)

8. If (Teta_G is N) and (Obstacle is RN) then (left Deviation is VVN) (right Deviation is Z)

9. If (Teta_G is P) and (Obstacle is RF) then (Left Deviation is Z) (right Deviation is NM)

10. If (Teta_G is PM) and (Obstacle is RF) then (Left Deviation is Z) (right Deviation is Z)

11. If (Teta_G is NM) and (Obstacle is LF) then (Left Deviation is Z) (Right Deviation is Z)

12. If (Teta_G is N) and (Obstacle is LF) then (Left Deviation is NM) (Right Deviation is Z)

13. If (Teta_G is NM) and (Obstacle is RN) then (Left Deviation is VVN) (Right Deviation is Z)

14. If (Teta_G is PM) and (Obstacle is LN) then (Left Deviation is Z) (Right Deviation is VVN)

15. If (Teta_G is N) and (Obstacle is RF) then (Left Deviation is N) (Right Deviation is Z)

16. If (Teta_G is P) and (Obstacle is NF) then (Left Deviation is Z) (Right Deviation is N)

17. If (Teta_G is N) and (Obstacle is NF) then (Left Deviation is N) (Right Deviation is Z)

18. If (Teta_G is Z) and (Obstacle is NF) then (Left Deviation is Z) (Right Deviation is Z)

19. If (Teta_G is P) and (Obstacle is LN) then (Left Deviation is Z) (Right Deviation is VVN)

20. If (Teta_G is Z) and (Obstacle is LN) then (Left Deviation is Z) (Right Deviation is VVN)

21. If (Teta_G is P) and (Obstacle is RN) then (Left Deviation is VVN) (Right Deviation is Z)

22. If (Teta_G is Z) and (Obstacle is RN) then (Left Deviation is VVN) (Right Deviation is Z)

23. If (Teta_G is PM) and (Obstacle is RN) then (Left Deviation is VVN) (Right Deviation is Z)

24. If (Teta_G is N) and (Obstacle is LN) then (Left Deviation is Z) (Right Deviation is VVN)

25. If (Teta_G is NM) and (Obstacle is LN) then (Left Deviation is Z) (Right Deviation is VVN).

3.4.4. Defuzzification

The last stage in the fuzzy inference process is Defuzzification. Fuzziness helps to evaluate the rules, but the final output of a fuzzy system has to be a crisp number. The input for the Defuzzification process is the aggregate output fuzzy set and the output is a single number. There are several Defuzzification methods, but probably the most popular one is the centroid technique. It finds the point where a vertical line would slice the aggregate set into two equal masses. Centroid of area method, mean-max method, first of maxima method, last of maxima method and weighted average method. Out of these, centroid of area method is adopted and is discussed below

Centroid of area method,

$$SA = \frac{\int \mu\alpha(SA_i) \times SA \times d(SA)}{\int \mu\alpha(SA) \times d(SA)} \tag{3.22}$$

Where $\mu\alpha\ (SA)$ is the aggregated output of the membership functions and SA is the desired output i.e. steering angle. The above mentioned equation can be represented in a discrete form as,

$$SA = \frac{\sum_{i=1}^{n} \mu\alpha(SA_i) \times SA \times d(SA)}{\sum_{i=1}^{n} \mu\alpha(SA) \times d(SA)} \tag{3.23}$$

Where $\mu a\ (SAi)$ is defined as the sampled value of the aggregated output membership functions. Additionally that crisp values are the output of the controller that translated to actions as deviations in the left and the right to avoid obstacles based on the situations.

CHAPTER 4
DISCUSSION AND SIMULATION RESULTS

In this chapter, the control system for path following and obstacles has been built with fuzzy logic control, a simulation test of the fuzzy controller to navigate in the environment is carried out with Matlab 2019a feature, with use of the mathematical modeling that was studied in the earlier chapter, the design controller's robustness is has been tested base on the powerful implemented fuzzy controller.

In this section, the presented results of the simulation system used MATLAB and VARARGIN simulator to view the simulation, the robot in the starting position (x_0, y_0) in Figure (4.1) its start moving to reach the target position the output of the fuzzy is (-0.16 ,-0.24 for the left and right deviations) that make turn to the right it's clear that robot turning to the right side, while in Figure (4.2) the design of the fuzzy is to make hard turn left to avoid the obstacle with fuzzy output of (-0.20for the left and -0.50for the right deviation) to turn left . Therefore, in Figure (4.3) the controller gets the value of the angles which are (-0.58,-0.14) then generated an output of (-0.57for the left and -0.16for the right deviation) to turn right. Finally, the robot reaches its desired position in Figure (4.4).

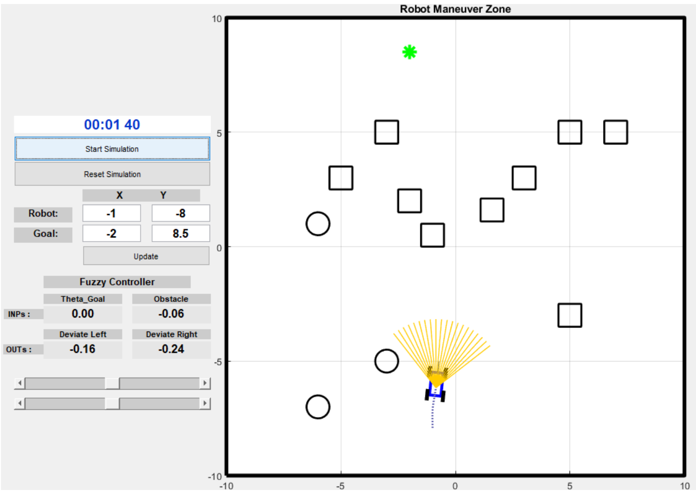


Figure 4.1: Heading to goal

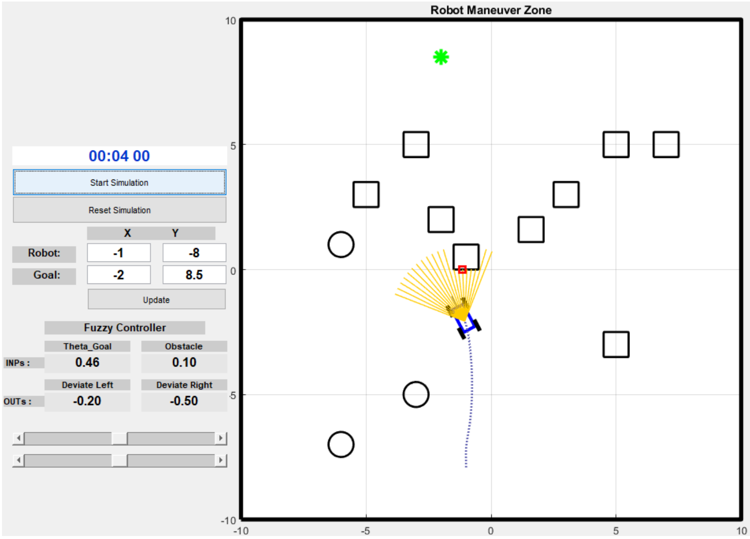


Figure 4.2: Avoiding obstacles to the left side in the first test

The action of the robot descript that the inputs of angles to the inference engine that the angle of the goal is larger than angles of obstacles output from the fuzzy inference engine evaluate rule that has output of large deviation for the left side with output of -0.20, the output feed into the Defuzzification return the output to crisp values again

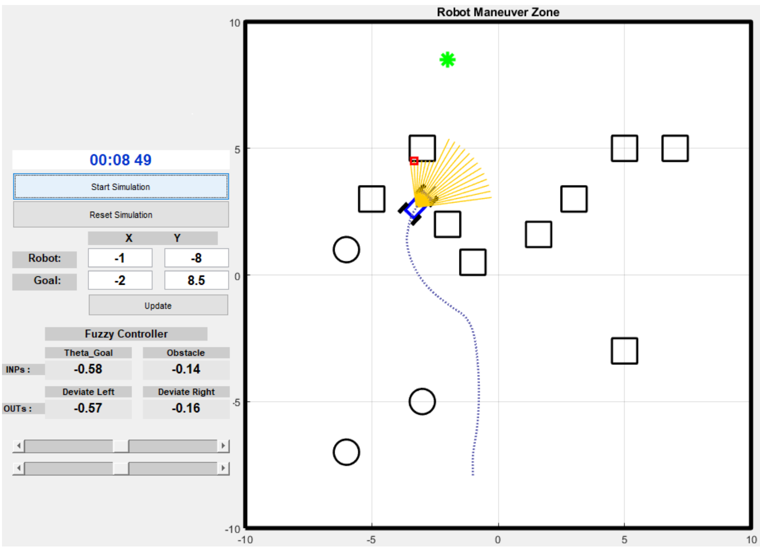


Figure 4.3: Avoiding obstacles to the right side in the first test

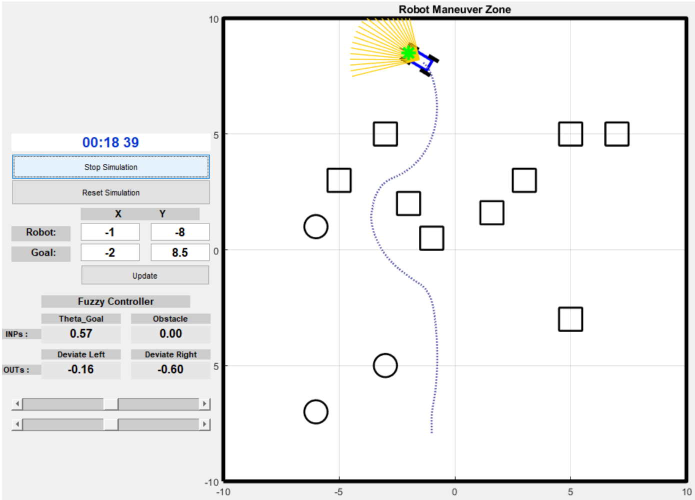
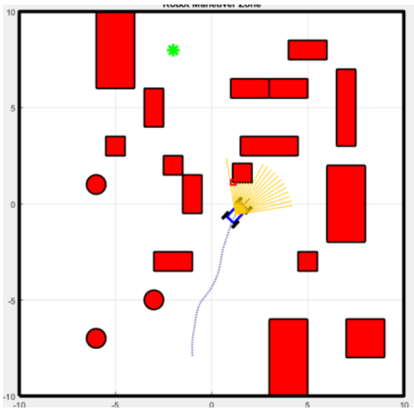
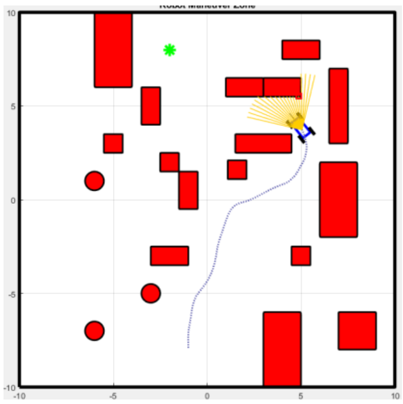


Figure 4.4: Reaching the goal position in the first test

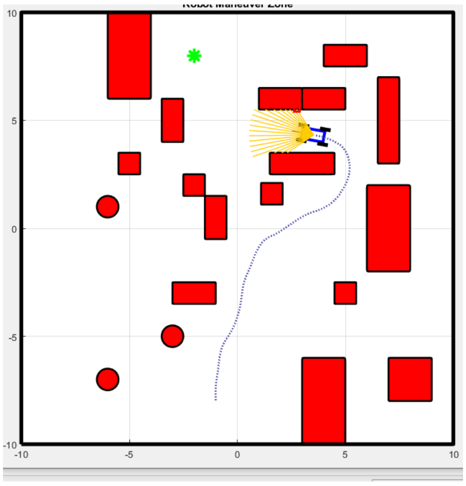
The second test has been done to shows the robustness of the system in different environment with more obstacles as shown in Figure (4.7), the robot defined it path without colliding ,the robot successfully reach the target position on Figure (4.9), the results show the ability of the fuzzy rules for dealing with difficult switching between behaviors. The second test proves the generated rules have the ability to generate output values which are robustness to deal with any condition.



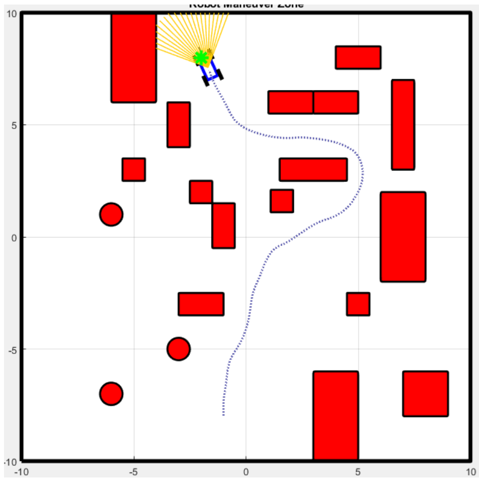
(a) Truing to the right side in the 2nd test



(b) Hard Turing to the left in 2nd



(c) Difficult decision condition



(d) Reaching goal position in 2nd test

Figure 4.5: Second test with deferent environment

Additionally, the control surface is plotted from the signals of the controller's inputs and outputs which are the (angles of goal and obstacles) with respect to the left and right deviations using Matlab toolbox for this conduction Figure 3.10 and Figure 3.1 shows the fuzzy logic controller's surfaces, the surfaces show the highest and the lowers values of the one output with two inputs.

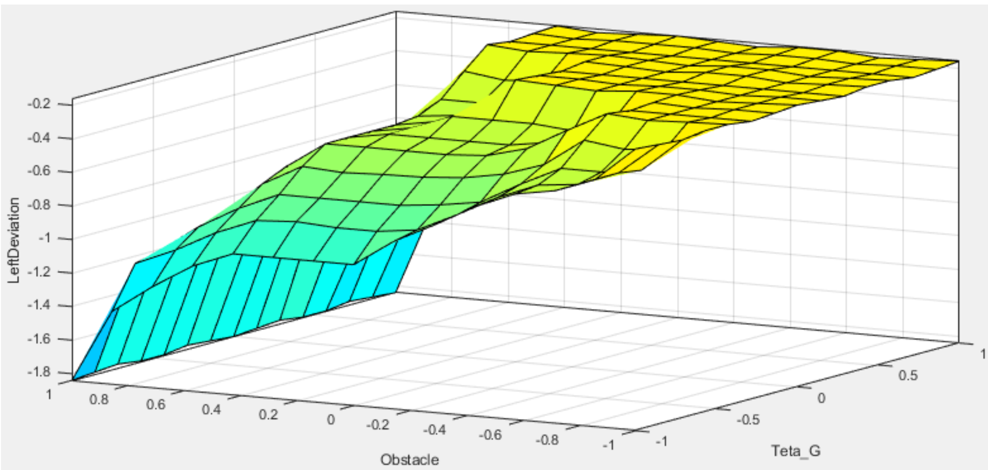


Figure 4.6: Control surface of left deviation

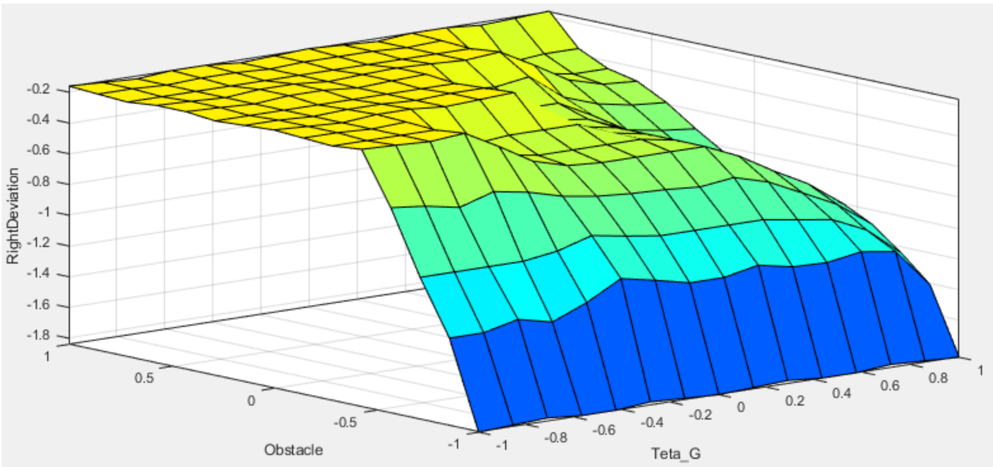


Figure 4.7: Control surface of right deviation

The surfaces show the higher and the lower values of the one output with two inputs.

Indeed the controller that been implanted is capable of dealing with a different environment, its show smooth path to the target even with increasing the maximum velocity the controller has the ability to give good output to avoid obstacles with help of the rules that cover all of the possible conditions that the robot may face.

CHAPTER 5
CONCLUSION

In this thesis, the presented path following and obstacles avoidance of mobile robot used fuzzy logic control (FLC) for navigating in the environment which also studied the kinematic model of the robot and using it in Matlab, simulation work had been done with getting to the target position and avoiding obstacles, the analysis of different works exposed that (FLC) is one of the best methods coordination and developing behaviors for mobiles robot. The work shows how the fuzzy logic applied and designed to solve complex tasks and behaviors, also its address the use of linguistic rules and the benefit the mechanism to design them. It additionally gave a method with powerful robustness for arbitration and a combination of behavior.

The designed fuzzy controller built to prove the impact and robustness of it in the navigation system it was clear in the previse results that the operation was successful and the efficiency of the (FLC) in the environment to generate a collision-avoidance system with clear motion.

In conclusion, the presented control method has many advantages like the ability to deal with difficult condition in the environments, it also shows the ability to develop an understand actions based on the designs controller.

REFERENCES

Al-Mayyahi, A., Wang, W., & Birch, P. (2014). Adaptive Neuro-Fuzzy Technique for Autonomous Ground Vehicle Navigation. *Robotics*, *3*(4), 349–370. https://doi.org/10.3390/robotics3040349

Aouf, A., Boussaid, L., & Sakly, A. (2019). Same Fuzzy Logic Controller for Two-Wheeled Mobile Robot Navigation in Strange Environments. *Journal of Robotics*, *2019*. https://doi.org/10.1155/2019/2465219

Arora, T. (2014). *Robotic Path Planning using Genetic Algorithm in Dynamic Environment*. *89*(11), 8–12.

Authors, F. (2016). *New algorithm for behaviour-based mobile robot navigation in cluttered environment*. https://doi.org/10.1108/WJE-04-2016-018

Carlos, J., Oliver, V., Freddy, P., & Navarrete, H. (2017). *Behaviors in a Differential-Drive Mobile Robot*. (4), 2–5.

Castillo, F. O., Trujillo, L., & Melin, P. (2007). *Multiple objective genetic algorithms for path-planning optimization*. 269–279. https://doi.org/10.1007/s00500-006-0068-4

Choomuang, R., & Afzulpurkar, N. (2005). *Hybrid Kalman Filter / Fuzzy Logic based Position Control of Autonomous Mobile Robot*. *2*(3), 197–208.

El-teleity, S. A. (2011). *Fuzzy Logic Control of an Autonomous Mobile Robot*. 188–193.

Ghorbani, A., & Nodehi, A. (2009). Using Genetic Algorithm for a Mobile Robot Path Planning. *2009 International Conference on Future Computer and Communication*, 164–166. https://doi.org/10.1109/ICFCC.2009.28

Gyawali, P., & Agarwal, P. K. (2019). Fuzzy behaviour based mobile robot navigation in static environment. *2018 IEEE Recent Advances in Intelligent Computational Systems, RAICS 2018*, (1), 190–194. https://doi.org/10.1109/RAICS.2018.8635074

Kim, C., & Chwa, D. (2015). Obstacle Avoidance Method for Wheeled Mobile Robots Using Interval Type-2 Fuzzy Neural Network. *IEEE Transactions on Fuzzy Systems*, *23*(3), 677–687. https://doi.org/10.1109/TFUZZ.2014.2321771

Pandey, A., Sonkar, R. K., Pandey, K. K., & Parhi, D. R. (2014). *Path Planning Navigation of Mobile Robot With Obstacles Avoidance Using Fuzzy Logic Controller*.

Patiln, H. D., & Carelli, R. (2004). *Imtituto de*.

Singh, N. H., & Thongam, K. (2018). Mobile Robot Navigation Using Fuzzy Logic in Static Environments. *Procedia Computer Science*, *125*, 11–17. https://doi.org/10.1016/j.procs.2017.12.004

Xiao, H. (2007). *Mobile Robot Path Planning Based on Q-ANN*. 2650–2654.

Yousfi, N., Rekik, C., Jallouli, M., & Derbel, N. (2010). Optimized fuzzy controller for mobile robot navigation in a cluttered environment. *2010 7th International Multi-Conference on Systems, Signals and Devices*, 1–7. https://doi.org/10.1109/SSD.2010.5585508

Zadeh, L. A. (1965). Fuzzy Sets.pdf. *Information and Control*, *8*, 338–353. https://doi.org/10.1109/2.53

Farooq, U., Hasan, K. M., Abbas, G., & Asad, M. U. (2011, October). Comparative analysis of zero order Sugeno and Mamdani fuzzy logic controllers for obstacle avoidance behavior in mobile robot navigation. In *The 2011 International Conference and Workshop on Current Trends in Information Technology (CTIT 11)* (pp. 113-119). IEEE.

Shitsukane, A., Cheruiyot, W., Otieno, C., & Mvurya, M. (2018, May). Fuzzy Logic Sensor Fusion for Obstacle Avoidance Mobile Robot. In *2018 IST-Africa Week Conference (IST-Africa)* (pp. Page-1). IEEE.

APPENDICES

APPENDIX 1
SIMULATOR EDITOR

```
function varargout = Simulate(varargin)
clc
SamplingPeriod = 0.1;
Time = 0;
timer1 = [];
txt_timer_pre = [];
FuzzyLoaded = 0;
Z1 = [];
Z2 = [];
W_Zone =20;
H_Zone = 20;


data = importdata ('fuzzytabledata.mat');
OUT1 = data.OUT1;
OUT2 = data.OUT2;
Z1 = reshape(OUT1(102:end,1)',101,[]);
Z2 = reshape(OUT2(102:end,1)',101,[]);
FuzzyLoaded = 1;

scrsz = get(0,'ScreenSize');

% Creating a figure that will have a checkboxes uitable, axes
figure('Position',[15*scrsz(3)/100 70 scrsz(3)-2*(15*scrsz(3)/100) scrsz(4)-100],...
    'WindowStyle', 'normal',...
    'CloseRequestFcn',{@CloseFcn},...
    'Name', 'hamdanFuzzyNavi-&ObstacleAvoi',...  % Title figure
    'NumberTitle', 'off',...
    'MenuBar', 'none');

% Create an axes set x and y limits
W_Zone = ceil(W_Zone/10)*10;
H_Zone = ceil(H_Zone/10)*10;
haxes = axes('Units', 'normalized',...
        'Position', [.25 .05 0.75 0.9],...
        'XLim', [-W_Zone/2 W_Zone/2],...
        'YLim', [-H_Zone/2 H_Zone/2],...
        'XLimMode', 'manual',...
        'YLimMode', 'manual',...
        'XTick',-W_Zone/2:5:W_Zone/2,...
        'YTick',-H_Zone/2:5:H_Zone/2);
set(haxes,'DataAspectRatio',[1 1 1]);
title(haxes, 'Robot Maneuver Zone')  % ....Describtion data set......
% when new lines or markers are plotted Prevent axes from clearing
```

```
hold(haxes, 'all')
grid on;

uicontrol('Style', 'pushbutton',...
        'Units', 'normalized',...
        'Position', [.02 .86 .2 .05],...
        'String', 'Start Simulation',...
        'Value', 0,...
        'Callback', {@StartSimulation});

uicontrol('Style', 'pushbutton',...
        'Units', 'normalized',...
        'Position', [.02 .81 .2 .05],...
        'String', 'Reset Simulation',...
        'Value', 0,...
        'Callback', {@ResetSimulation});

% % show timerpy Create a text ;
txt_timer = uicontrol('Style', 'text',...
          'Units', 'normalized',...
          'Position', [.02 .915 .2 .035],...
          'FontWeight', 'bold',...
          'ForegroundColor', [0 .2 .8],...
          'fontname', 'Helvetica',...
          'fontsize', 14,...
          'BackgroundColor', 'w', 'String','00:00 00');

% %  monitor variables Creating some texts ;
txt_input1 = uicontrol('Style', 'text', 'Units', 'normalized', 'Position', [.05 .54 .08 .035],...
          'FontWeight', 'bold', 'ForegroundColor', [0 0 0], 'fontname', 'Helvetica', 'fontsize',
12, 'BackgroundColor', [0.9 0.9 0.9]);
txt_input2 = uicontrol('Style', 'text', 'Units', 'normalized', 'Position', [.14 .54 .08 .035],...
          'FontWeight', 'bold', 'ForegroundColor', [0 0 0], 'fontname', 'Helvetica', 'fontsize',
12, 'BackgroundColor', [0.9 0.9 0.9]);
txt_output1 = uicontrol('Style', 'text', 'Units', 'normalized', 'Position', [.05 .47 .08 .035],...
          'FontWeight', 'bold', 'ForegroundColor', [0 0 0], 'fontname', 'Helvetica', 'fontsize',
12, 'BackgroundColor', [0.9 0.9 0.9]);
txt_output2 = uicontrol('Style', 'text', 'Units', 'normalized', 'Position', [.14 .47 .08 .035],...
          'FontWeight', 'bold', 'ForegroundColor', [0 0 0], 'fontname', 'Helvetica', 'fontsize',
12, 'BackgroundColor', [0.9 0.9 0.9]);



uicontrol('Style', 'text','Units', 'normalized', 'Position', [.05 .61 .15 .025],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 10,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'Fuzzy Controller');
```

```
uicontrol('Style', 'text','Units', 'normalized', 'Position', [.00 .55 .05 .02],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 8,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'INPs :');
uicontrol('Style', 'text','Units', 'normalized', 'Position', [.00 .48 .05 .02],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 8,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'OUTs :');
uicontrol('Style', 'text','Units', 'normalized', 'Position', [.05 .58 .08 .02],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 8,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'Theta_Goal');
uicontrol('Style', 'text','Units', 'normalized', 'Position', [.14 .58 .08 .02],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 8,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'Obstacle');
uicontrol('Style', 'text','Units', 'normalized', 'Position', [.05 .51 .08 .02],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 8,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'Deviate Left');
uicontrol('Style', 'text','Units', 'normalized', 'Position', [.14 .51 .08 .02],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 8,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'Deviate Right');


uicontrol('Style', 'text','Units', 'normalized', 'Position', [.09 .78 .12 .025],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 10,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'X            Y');
uicontrol('Style', 'text','Units', 'normalized', 'Position', [.02 .74 .06 .03],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 10,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', 'Robot:');
uicontrol('Style', 'text','Units', 'normalized', 'Position', [.02 .7 .06 .03],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 10,...
          'BackgroundColor', [0.8 0.8 0.8], 'String', '  Goal:');

% %  geting the  Robot X0 Posetion by cteating a text ;
edit_robot_xo = uicontrol('Style', 'edit',...
          'Units', 'normalized',...
          'Position', [.09 .74 .06 .035],... %'String',... %'Use Plot check boxes to graph
columns',...
          'FontWeight', 'bold',...
          'fontname', 'Helvetica',...
          'fontsize', 12);
% % geting the  Robot Y0 Pos;
edit_robot_yo = uicontrol('Style', 'edit',...
          'Units', 'normalized',...
          'Position', [.16 .74 .06 .035],...
          'FontWeight', 'bold',...
          'fontname', 'Helvetica',...
          'fontsize', 12);

% %  geting Goal X Pos;
```

```
edit_goal_x_posetion = uicontrol('Style', 'edit',...
          'Units', 'normalized',...
          'Position', [.09 .7 .06 .035],...
          'FontWeight', 'bold',...
          'fontname', 'Helvetica',...
          'fontsize', 12);
% % geting  get Goal Y Pos;
edit_goal_y_posetion = uicontrol('Style', 'edit',...
          'Units', 'normalized',...
          'Position', [.16 .7 .06 .035],...
          'FontWeight', 'bold',...
          'fontname', 'Helvetica',...
          'fontsize', 12);

uicontrol('Style', 'pushbutton',...
        'Units', 'normalized',...
        'Position', [.09 .655 .13 .04],... %'to Update Values Independently',...
        'String', 'Update',...
        'Value', 0,...
        'Callback', {@UpdateParams});

slider_of_x = uicontrol('Style', 'slider','Units', 'normalized', 'Position', [.02 .41 .2 .025],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 10,...
          'BackgroundColor', [0.8 0.8 0.8],'Min',0,'Value',1,'Max',2,'SliderStep',[0.01 0.1]);

slider_of_y = uicontrol('Style', 'slider','Units', 'normalized', 'Position', [.02 .37 .2 .025],...
          'FontWeight', 'bold', 'fontname', 'Helvetica', 'fontsize', 10,...
          'BackgroundColor', [0.8 0.8 0.8],'Min',0,'Value',1,'Max',2,'SliderStep',[0.01 0.1]);
```

APPENDIX 2
THE DESIGN MATLAB CODES

```
w = 0.5;
l = 1;
D = 0.7;

xo = -1;
yo = -8;
tetao = 0;
Vr0 = 0;
Vl0 = 0;
x = xo;
y = yo;
teta = tetao;
Vr = Vr0;
Vl = Vl0;
V_MAXIMUM = 3;

x_Goal_P = -2;
y_Goal_P = 8;

No_of_obs = 21;


x_obsi0(2) = -1;
y_obsi0(2) = 0.5;
x_obsi(2) = x_obsi0(2);
y_obsi(2) = y_obsi0(2);
w_obsi(2) = 1;
h_obsi(2) = 2;
n_obsi(2) = 50;


x_obsi0(3) = 5;
y_obsi0(3) = -3;
x_obsi(3) = x_obsi0(3);
y_obsi(3) = y_obsi0(3);
w_obsi(3) = 1;
h_obsi(3) = 1;
n_obsi(3) = 50;


x_obsi0(4) = -3;
y_obsi0(4) = 5;
x_obsi(4) = x_obsi0(4);
y_obsi(4) = y_obsi0(4);
w_obsi(4) = 1;
```

```
h_obsi(4) = 1;
n_obsi(4) = 50;


x_obsi0(5) = -3;
y_obsi0(5) = -5;
x_obsi(5) = x_obsi0(5);
y_obsi(5) = y_obsi0(5);
w_obsi(5) = 1;
h_obsi(5) = 1;
n_obsi(5) = 2;


x_obsi0(6) = -6;
y_obsi0(6) = 1;
x_obsi(6) = x_obsi0(6);
y_obsi(6) = y_obsi0(6);
w_obsi(6) = 1;
h_obsi(6) = 1;
n_obsi(6) = 2;

x_obsi0(7) = -6;
y_obsi0(7) = -7;
x_obsi(7) = x_obsi0(7);
y_obsi(7) = y_obsi0(7);
w_obsi(7) = 1;
h_obsi(7) = 1;
n_obsi(7) = 2;


 x_obsi0(8) = -2;
y_obsi0(8) = 2;
x_obsi(8) = x_obsi0(8);
y_obsi(8) = y_obsi0(8);
w_obsi(8) = 1;
h_obsi(8) = 1;
n_obsi(8) = 30;

 x_obsi0(9) = -5;
y_obsi0(9) = 3;
x_obsi(9) = x_obsi0(9);
y_obsi(9) = y_obsi0(9);
w_obsi(9) = 1;
h_obsi(9) = 1;
n_obsi(9) = 50;
```

```
 x_obsi0(10) = 5;
y_obsi0(10) = 8;
x_obsi(10) = x_obsi0(10);
y_obsi(10) = y_obsi0(10);
w_obsi(10) = 2;
h_obsi(10) = 1;
n_obsi(10) = 50;


 x_obsi0(11) = 7;
y_obsi0(11) = 5;
x_obsi(11) = x_obsi0(11);
y_obsi(11) = y_obsi0(11);
w_obsi(11) = 1;
h_obsi(11) = 4;
n_obsi(11) = 50;


 x_obsi0(12) = -3;
y_obsi0(12) = 8;
x_obsi(12) = x_obsi0(4);
y_obsi(12) = y_obsi0(4);
w_obsi(12) = 1;
h_obsi(12) = 2;
n_obsi(12) = 50;


 x_obsi0(13) = 1.6;
y_obsi0(13) = 1.6;
x_obsi(13) = x_obsi0(13);
y_obsi(13) = y_obsi0(13);
w_obsi(13) = 1;
h_obsi(13) = 1;
n_obsi(13) = 50;

x_obsi0(14) = 3;
y_obsi0(14) = 3;
x_obsi(14) = x_obsi0(14);
y_obsi(14) = y_obsi0(14);
w_obsi(14) = 3;
h_obsi(14) = 1;
n_obsi(14) = 50;

 x_obsi0(15) = -2;
y_obsi0(15) = -3;
x_obsi(15) = x_obsi0(15);
y_obsi(15) = y_obsi0(15);
```

```
 w_obsi(15) = 2;
 h_obsi(15) = 1;
 n_obsi(15) = 50;

  x_obsi0(16) = 4;
 y_obsi0(16) = 6;
 x_obsi(16) = x_obsi0(16);
 y_obsi(16) = y_obsi0(16);
 w_obsi(16) = 2;
 h_obsi(16) = 1;
 n_obsi(16) = 50;

 x_obsi0(17) = 2;
 y_obsi0(17) = 6;
 x_obsi(17) = x_obsi0(17);
 y_obsi(17) = y_obsi0(17);
 w_obsi(17) = 2;
 h_obsi(17) = 1;
 n_obsi(17) = 50

 x_obsi0(18) = 7;
 y_obsi0(18) = 0;
 x_obsi(18) = x_obsi0(18);
 y_obsi(18) = y_obsi0(18);
 w_obsi(18) = 2;
 h_obsi(18) = 4;
 n_obsi(18) = 50;

  x_obsi0(19) = -5;
 y_obsi0(19) = 8;
 x_obsi(19) = x_obsi0(19);
 y_obsi(19) = y_obsi0(19);
 w_obsi(19) = 2;
 h_obsi(19) = 4;
 n_obsi(19) = 50;


  x_obsi0(20) = 4;
 y_obsi0(20) = -8;
 x_obsi(20) = x_obsi0(20);
 y_obsi(20) = y_obsi0(20);
 w_obsi(20) = 2;
 h_obsi(20) = 4;
 n_obsi(20) = 50;

x_obsi0(21) = 8;
 y_obsi0(21) = -7;
```

```
    x_obsi(21) = x_obsi0(21);
    y_obsi(21) = y_obsi0(21);
    w_obsi(21) = 2;
    h_obsi(21) = 2;
    n_obsi(21) = 50;


   x_obsi0(22) = 7;
    y_obsi0(22) = -3;
    x_obsi(22) = x_obsi0(22);
    y_obsi(22) = y_obsi0(22);
    w_obsi(22) = 2;
    h_obsi(22) = 2;
    n_obsi(22) = 50;

    Counter = 0;
    trace_x = [];
    trace_y = [];
    trace_i = 1;
    function Command(obj, event, string_arg)
        Time = Time + SamplingPeriod;
        Counter = Counter + 1;
        set(txt_timer,'String',[num2str(floor(Time/60),'%2.2d'),':',
num2str(mod(floor(Time),60),'%2.2d'),' ',num2str(mod(floor(Time*100),100),'%2.2d')]);


        %dx = -3*sin(time); dy = -2*cos(time); dteta = 1;
        dx = 0; dy = 0; dteta = 0;

        ss = (x_obsi0-x).^0+(y_obsi0-y).^0;

        [ss_min,K] = min(ss);
        if(abs(x_obsi(K)-x_obsi0(K)) < get(slider_of_x,'Value'))
            x_obsi(K) = x_obsi(K) + ((sin(Time*3+rand(1)*2)+0.2*(x-
x_obsi(K)))*rand(1)*3)*SamplingPeriod;
        else
            x_obsi(K) = x_obsi(K) + (x_obsi0(K)-x_obsi(K))*0.2*SamplingPeriod;
        end

        if(abs(y_obsi(K)-y_obsi0(K)) < get(slider_of_y,'Value'))
            y_obsi(K) = y_obsi(K) + (cos(Time*4+rand(1)*3)*rand(1)*3)*SamplingPeriod;
        else
            y_obsi(K) = y_obsi(K) + (y_obsi0(K)-y_obsi(K))*0.2*SamplingPeriod;
        end

        ss(K)= max(ss);
```

```
        [ss_min,K] = min(ss);
        if(abs(x_obsi(K)-x_obsi0(K)) < get(slider_of_x,'Value'))
           x_obsi(K) = x_obsi(K) + ((sin(Time*3+rand(1)*2)+0.2*(x-
x_obsi(K)))*rand(1)*3)*SamplingPeriod;
        else
           x_obsi(K) = x_obsi(K) + (x_obsi0(K)-x_obsi(K))*0.2*SamplingPeriod;
        end

        if(abs(y_obsi(K)-y_obsi0(K)) < get(slider_of_y,'Value'))
           y_obsi(K) = y_obsi(K) + (cos(Time*4+rand(1)*3)*rand(1)*3)*SamplingPeriod;
        else
           y_obsi(K) = y_obsi(K) + (y_obsi0(K)-y_obsi(K))*0.2*SamplingPeriod;
        end

        ss(K)= max(ss);

        [ss_min,K] = min(ss);
        if(abs(x_obsi(K)-x_obsi0(K)) < get(slider_of_x,'Value'))
           x_obsi(K) = x_obsi(K) + ((sin(Time*3+rand(1)*2)+0.2*(x-
x_obsi(K)))*rand(1)*3)*SamplingPeriod;
        else
           x_obsi(K) = x_obsi(K) + (x_obsi0(K)-x_obsi(K))*0.2*SamplingPeriod;
        end

        if(abs(y_obsi(K)-y_obsi0(K)) < get(slider_of_y,'Value'))
           y_obsi(K) = y_obsi(K) + (cos(Time*4+rand(1)*3)*rand(1)*3)*SamplingPeriod;
        else
           y_obsi(K) = y_obsi(K) + (y_obsi0(K)-y_obsi(K))*0.2*SamplingPeriod;
        end

        %Checking for Obstacles
        ray_of_i = 1;
        ray_x = []; ray_y = []; ray_s = []; ray_teta = [];
        rayplot_x = []; rayplot_y = []; rayplot_i = 1;
        for dir = linspace(-pi/4,pi/4,20)
           Obstacle_Found = 0;
           s = 0;
           while(Obstacle_Found==0)
              s = s + 0.02;
              ray_pos = [x + s*sin(teta+dir), y + s*cos(teta+dir)];
              for i = 1:No_of_obs
                 if( (2*(ray_pos(1)-x_obsi(i))/w_obsi(i))^n_obsi(i)+(2*(ray_pos(2)-
y_obsi(i))/h_obsi(i))^n_obsi(i) < 1 )
                    Obstacle_Found = 1;
                 end
              end
              if( (2*ray_pos(1)/W_Zone)^100+(2*ray_pos(2)/H_Zone)^100>1 || s>3)
```

```
            Obstacle_Found = 1;
          end
          if(Obstacle_Found==1)
            ray_x(ray_of_i) = ray_pos(1);
            ray_y(ray_of_i) = ray_pos(2);
            ray_s(ray_of_i) = s;
            ray_teta(ray_of_i) = dir;
            rayplot_x(rayplot_i:rayplot_i+2) = [x ray_pos(1) x];
            rayplot_y(rayplot_i:rayplot_i+2) = [y ray_pos(2) y];
          end
        end
        ray_of_i = ray_of_i + 1;
        rayplot_i = rayplot_i + 3;
      end

      j=0;
      [s_min,j]=min(ray_s);
      obstacle_value = 0;
      if(s_min<3)
        obstacle_value = min((3-s_min)/(3-max(l,w)),1);
        if(ray_teta(j)<0)
          obstacle_value = -obstacle_value;
        end
      else
        j=0;
      end
      goal_distance = ((x-x_Goal_P)^2+(y-y_Goal_P)^2)^0.5;
      tetap = rem(teta,2*pi);
      if(tetap<-pi), tetap = tetap + 2*pi; elseif(tetap>pi) tetap = tetap - 2*pi; end;
      tetagoal = (atan2(x_Goal_P-x,y_Goal_P-y)-tetap);
      tetagoal = rem(tetagoal,2*pi);
      if(tetagoal<-pi), tetagoal = tetagoal + 2*pi; elseif(tetagoal>pi) tetagoal = tetagoal -
2*pi; end;
      tetagoal_value = tetagoal/(2*pi);



      set(txt_input1,'String',[num2str(obstacle_value,'%2.2f')]);
      set(txt_input2,'String',[num2str(tetagoal/pi,'%2.2f')]);
      if(FuzzyLoaded)
        obstacle__index = int32(round((obstacle_value+1)*50+1));
        teta_goal__index = int32(round((tetagoal_value+1)*50+1));
        if(obstacle__index<1) obstacle__index=1; elseif(obstacle__index>101)
obstacle__index=101; end
        if(teta_goal__index<1) teta_goal__index=1; elseif(teta_goal__index>101)
teta_goal__index=101; end
        DVl = Z1(teta_goal__index,obstacle__index);
```

```
            DVr = Z2(teta_goal__index,obstacle__index);
            Vrc = V_MAXIMUM;
            if(goal_distance<3)
               Vrc = V_MAXIMUM*((goal_distance-0.5)/(3-0.5));
               if(Vrc<0) Vrc=0; end
            end
            Vlc = Vrc;
            Vrc = (1+DVr)*Vrc;
            Vlc = (1+DVl)*Vlc;
            Vr = Vr + 0.5*(Vrc-Vr);
            Vl = Vl + 0.5*(Vlc-Vl);

            dx = (Vr+Vl)/2*sin(teta);
            dy = (Vr+Vl)/2*cos(teta);
            dteta = (Vr-Vl)/D;
            set(txt_output1,'String',[num2str(DVl,'%2.2f')]);
            set(txt_output2,'String',[num2str(DVr,'%2.2f')]);
         end
         SamplingPeriod2 = SamplingPeriod;
         x = x + dx*SamplingPeriod2;
         y = y + dy*SamplingPeriod2;
         teta = teta + dteta*SamplingPeriod2;

         if(trace_i>1)
            if((x-trace_x(trace_i-1))^2+(y-trace_y(trace_i-1))^2>0.04)
               trace_x(trace_i) = x;
               trace_y(trace_i) = y;
               trace_i = trace_i + 1;
            end
         else
            trace_x(trace_i) = x;
            trace_y(trace_i) = y;
            trace_i = trace_i + 1;
         end

         if(mod(Counter,1)==0)
            cla(haxes);
            robot_extents = [cos(teta) sin(teta) x; -sin(teta) cos(teta) y] * [-w/2,-l/2,1; w/2,-
l/2,1; w/2,l/2,1; -w/2,l/2,1; -w/2,-l/2,1; w/2,-l/2,1]';
            plot(haxes,robot_extents(1,:),robot_extents(2,:), 'LineWidth' ,3,'Color',[0 0 1]);
            robot_tyre = [cos(teta) sin(teta) x; -sin(teta) cos(teta) y] * [-w/2-0.1,-l/2-l/4,1; -w/2-
0.1,-l/2+l/4,1; w/2+0.1,-l/2-l/4,1; w/2+0.1,-l/2+l/4,1; -w/2-0.1,l/2-l/4,1; -w/2-0.1,l/2+l/4,1;
w/2+0.1,l/2-l/4,1; w/2+0.1,l/2+l/4,1; 0,l/2,1; 0,l/2+l/2,1]';
            plot(haxes,robot_tyre(1,1:2),robot_tyre(2,1:2), 'LineWidth' ,4,'Color',[0 0 0]);
plot(haxes,robot_tyre(1,3:4),robot_tyre(2,3:4), 'LineWidth' ,4,'Color',[0 0 0]);
            plot(haxes,robot_tyre(1,5:6),robot_tyre(2,5:6), 'LineWidth' ,4,'Color',[0 0 0]);
plot(haxes,robot_tyre(1,7:8),robot_tyre(2,7:8), 'LineWidth' ,4,'Color',[0 0 0]);
```

```
        plot(haxes,robot_tyre(1,9:10),robot_tyre(2,9:10), 'LineWidth' ,1,'Color',[0.4 0.4
0.4]);

        s = linspace(0,2*pi,360);
        for i = 1:No_of_obs
          x_o = (1./((sin(s)/(h_obsi(i)/2)).^n_obsi(i) +
(cos(s)/(w_obsi(i)/2)).^n_obsi(i)).^(1/n_obsi(i))).*cos(s) + x_obsi(i);
          y_o = (1./((sin(s)/(h_obsi(i)/2)).^n_obsi(i) +
(cos(s)/(w_obsi(i)/2)).^n_obsi(i)).^(1/n_obsi(i))).*sin(s) + y_obsi(i);
          plot(haxes,x_o,y_o, 'LineWidth' ,2,'Color',[0 0 0]);
          %if(gca == haxes), fill(x_o,y_o,[1 0 0],'LineWidth' ,2); end
        end
        plot(haxes,[-W_Zone/2 W_Zone/2 W_Zone/2 -W_Zone/2 -W_Zone/2],[-H_Zone/2
-H_Zone/2 H_Zone/2 H_Zone/2 -H_Zone/2], 'LineWidth' ,4,'Color',[0 0 0]);
        plot(haxes, rayplot_x, rayplot_y, 'LineWidth' ,1,'Color',[1 0.8 0]);
        plot(haxes, trace_x, trace_y, ':', 'LineWidth' ,2,'Color',[0.4 0.4 0.7]);
        plot(haxes, x_Goal_P,
y_Goal_P,'*','MarkerSize',15,'MarkerEdgeColor','g','LineWidth',3);
        if(j>0)
          plot(haxes, ray_x(j),
ray_y(j),'s','MarkerSize',8,'MarkerEdgeColor','r','LineWidth',3);
        end
         %plot(haxes, [x x_Goal], [y y_Goal],'-.', 'LineWidth' ,1,'Color',[0.8 0.95 0.2]);
      end
  end
%% for executing the function Command sequentially we creating and start timer1 t.
  timer1 = timer('TimerFcn', @Command, 'Period', SamplingPeriod, 'ExecutionMode',
'fixedRate');

  function CloseFcn(src,evnt)

     selection = questdlg(' Close Application ?',...
       'Close Warning',...
       'Yes','No','Yes');
     switch selection,
       case 'Yes',
         stop(timer1);
         delete(timer1);
         delete(gcf)
       case 'No'
       return
     end
  end

  % --- Executes on button press in pushbutton StartSimulation.
  function StartSimulation(hObject, eventdata)
      str = get(hObject,'String');
```

```
        if(strcmp(str,'Start Simulation')==1)

            UpdateParams(0,0);
            if(FuzzyLoaded==0)
                try
                    Z1 = evalin('base', 'Z1');
                    Z2 = evalin('base', 'Z2');
                    FuzzyLoaded = 1;
                catch ME
                    FuzzyLoaded = 0;
                    try
                        OUT1 = evalin('base', 'OUT1');
                        OUT2 = evalin('base', 'OUT2');
                        Z1 = reshape(OUT1(102:end,1)',101,[]);
                        Z2 = reshape(OUT2(102:end,1)',101,[]);
                        FuzzyLoaded = 1;
                    catch ME2
                        msgbox('Please Create 2D table Z1 and Z2 or 1D arrays OUT1 and OUT2
as Fuzzy Deviations in Workspace','Warning','warn');
                    end
                end
            end
            start(timer1);
            set(hObject,'String','Stop Simulation');
        else
            stop(timer1);
            set(hObject,'String','Start Simulation');
        end
    end

    function UpdateParams(hObject, eventdata)  % --- Executes on button press to
ResetSimulation.
        x_Goals = str2double(get(edit_goal_x_posetion,'String'));
        y_Goals = str2double(get(edit_goal_y_posetion,'String'));
        x0s = str2double(get(edit_robot_xo,'String'));
        y0s = str2double(get(edit_robot_yo,'String'));
        if(isnan(x_Goals)==0), x_Goal_P = x_Goals; end
        if(isnan(y_Goals)==0), y_Goal_P = y_Goals; end
        if(isnan(x0s)==0), xo = x0s; end
        if(isnan(y0s)==0), yo = y0s; end
    end
    % --- Executeing on button press to Update.
    function ResetSimulation(hObject, eventdata)
        UpdateParams(0,0);
        Time=0;
        x = xo;
        y = yo;
```

```
        teta = tetao;
        Vr = Vr0;
        Vl = Vl0;
        trace_x = [];
        trace_y = [];
        trace_i = 1;
    end

    set(edit_goal_x_posetion,'String',num2str(x_Goal_P));
    set(edit_goal_y_posetion,'String',num2str(y_Goal_P));
    set(edit_robot_xo,'String',num2str(xo));
    set(edit_robot_yo,'String',num2str(yo));
end
```