



UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FACULTY OF ELECTRICAL ENGINEERING

FINAL YEAR PROJECT REPORT

DEVELOPMENT OF OBSTACLE AVOIDANCE ALGORITHM
FOR ROBOTIC WHEELCHAIR

PREPARED BY:

AMAR AWAD MOHAMED AHMED ISMAIL

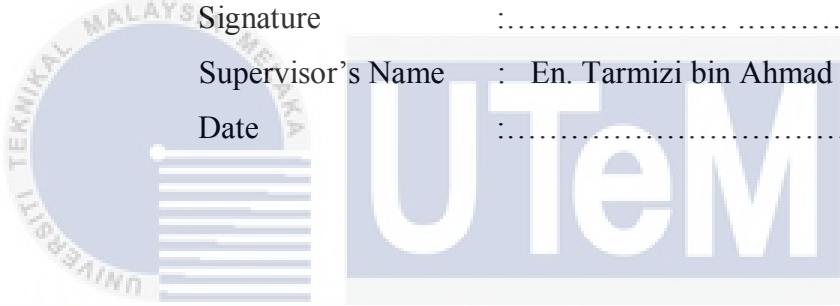
B011310371

BACHELOR OF ELECTRICAL ENGINEERING
(CONTROL, INSTRUMENTATION, AND AUTOMATION)

SUPERVISOR'S ENDORSEMENT

I hereby declare that I have read this report entitled “Development of Obstacle Avoidance Algorithm for Robotic Wheelchair” and found that it complies with the partial fulfillment for the awarding the degree of Bachelor of Electrical Engineering (Control, Instrumentation, and Automation).

Signature	:
Supervisor's Name	:	En. Tarmizi bin Ahmad Izzuddin
Date	:

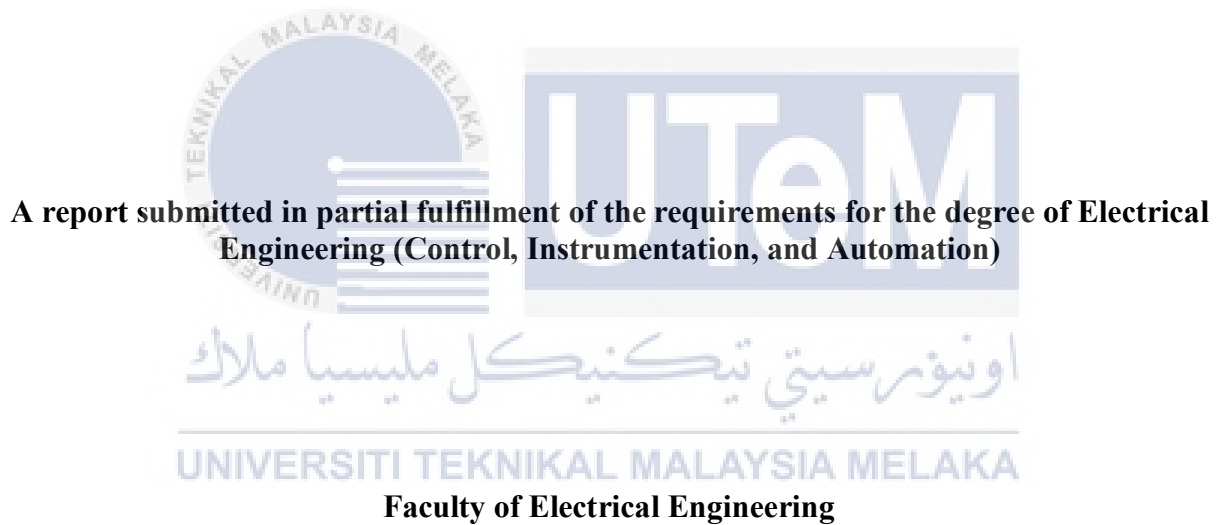


اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**DEVELOPMENT OF OBSTACLE AVOIDANCE ALGORITHM FOR ROBOTIC
WHEELCHAIR**

AMAR AWAD MOHAMED AHMED ISMAIL



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2017

STUDENT'S DECLARATION

I hereby declare that this report entitled “Development of Obstacle Avoidance Algorithm for Robotic Wheelchair” is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in the candidature of any other degree.



Signature

.....

Name

: Amar Awad Mohamed Ahmed Ismail

Date

.....

اونيورسيتي تیکنیکل ملیسیا ملاک

UNIVERSITI TEKNIKAL MALAYSIA MELAKA



To my beloved mother and father and to my dearest brother and sisters.

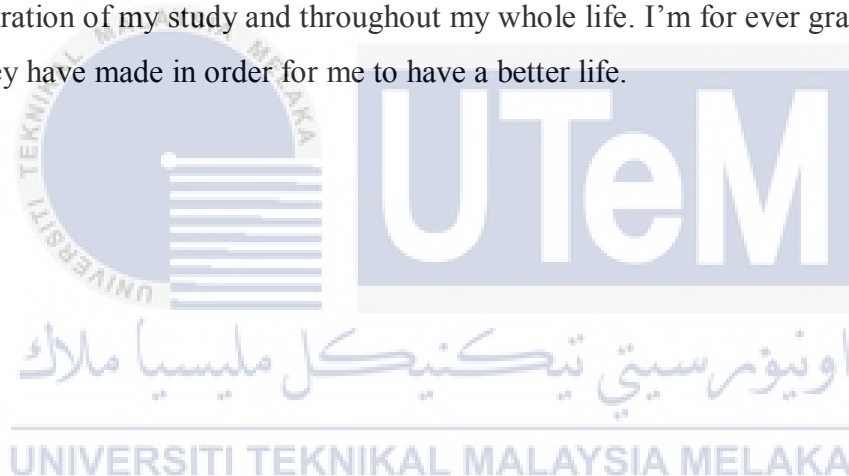
اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ACKNOWLEDGEMENT

Firstly, I would like to thank **En. Tarmizi bin Ahmad Izzuddin** for accepting to be my supervisor for my Final Year Project. He has been guiding me throughout the period of the final year project. I would like to thank him for his patience, motivation, and knowledge while guiding me to a complete this project.

I would also like to thank my family who has given me their full support throughout the whole duration of my study and throughout my whole life. I'm for ever grateful for all the sacrifices they have made in order for me to have a better life.



ABSTRACT

With the rising number of disabled people around the world and the increasing number of wheelchair users, research and developments have increased rapidly to produce autonomous and robotic wheelchairs that can avoid obstacles. This project proposes a behavior-based obstacle avoidance algorithm implemented using a fuzzy logic controller to ensure the safety of the wheelchair user. Two behaviors; Go-to-Goal and Avoid Obstacles are created using 11 fuzzy rules and are combined using rule weights. The proposed fuzzy logic controller has two inputs which are the target direction and the readings of three (IR) sensors attached to the right, left and front of the wheelchair and two outputs which are the linear velocities of the right and left motors. The testing and analysis of the controller are done using software simulation in MATLAB and Simulink environments. The results of the testing show that the proposed controller is effective in avoiding obstacles in low congested environments where the number of obstacles is low.

ABSTRAK

Dengan peningkatan bilangan orang kurang upaya di seluruh dunia dan peningkatan jumlah pengguna kerusi roda, penyelidikan dan pembangunan telah meningkat dengan pesat untuk menghasilkan kerusi roda autonomi dan robot yang boleh mengelak halangan. Projek ini mencadangkan algoritma mengelak halangan berasaskan tingkah laku dilaksanakan menggunakan kawalan logik kabur untuk memastikan keselamatan pengguna kerusi roda. Dua tingkah laku; Go-to-gol dan Elakkan Halangan adalah dicipta menggunakan 11 peraturan kabur dan digabungkan menggunakan berat peraturan. dicadangkan pengawal logik fuzzy mempunyai dua input yang menyasar dan bacaan tiga (IR) sensor dilampirkan ke kanan, kiri dan hadapan kerusi roda untuk mengesan dan mengukur jarak ke halangan. Ujian dan analisis pengawal dilakukan dengan menggunakan simulasi perisian dalam persekitaran MATLAB dan Simulink. Keputusan ujian menunjukkan bahawa pengawal yang dicadangkan berkesan dalam mengelak halangan dalam persekitaran yang sesak di mana bilangan halangan adalah rendah.

Table of Contents

ABSTRACT		I
ABSTRAK		II
TABLE OF CONTENTS		III
LIST OF TABLES		V
LIST OF FIGURES		VI
1 INTRODUCTION		1
1.1	MOTIVATION	1
1.2	PROBLEM STATEMENT	2
1.3	OBJECTIVES	2
1.4	SCOPE	2
2 LITERATURE REVIEW		4
2.1	INTRODUCTION	4
2.2	WHEELCHAIR KINEMATIC MODEL	4
2.3	BEHAVIOR-BASED ROBOTICS	6
2.4	FUZZY LOGIC CONTROL	9
2.5	FUZZY BEHAVIOR-BASED ALGORITHM IN MOBILE ROBOTICS	15
2.6	SUMMARY OF THE REVIEW	19
3 METHODOLOGY		20
3.1	PROJECT METHODOLOGY	20
3.2	WHEELCHAIR MODEL	22
3.3	DISTANCE MEASURING SENSORS	23
3.4	SIMULATOR SETUP	24
3.4.1	Robot Workspace Configuration	25
3.4.2	Robot Configuration	26
3.4.3	Simulink Model Configuration	28
3.5	THE PROPOSED OBSTACLE AVOIDANCE ALGORITHM	33
3.5.1	The Developed Fuzzy Controller	33
3.5.2	The Fuzzification Procedure and Membership Function Design	34
3.5.3	The Inference Mechanism and The Rule Base Design	36
3.5.4	The Defuzzification Method	37
4 RESULTS		39

5	CONCLUSION	49
6	REFERENCES	51



List of Tables

Table 2.1: Reactive versus deliberative architecture	8
Table 2.2: Rule table for ultrasonic sensors (Li and Yang)	16
Table 3.1: Wheelchair dimensions.....	23
Table 3.2: MATLAB commands to create the robot's workspaces	25
Table 3.3: MATLAB commands to configure the robot.....	27
Table 3.4: MATLAB commands for ExtractRangeData and quat2eulBlk function blocks	30
Table 3.5: extractGoal function block.....	31
Table 3.6: The rule base for the fuzzy controller;	37



List of Figures

Figure 2.1: Differential drive mobile robot.....	4
Figure 2.2: Behavior-based robot.....	7
Figure 2.3: Behavior-based architecture.....	8
Figure 2.4: Fuzzy logic controller.....	9
Figure 2.5: Comparison between crisp and fuzzy sets.....	10
Figure 2.6: Driving control system. (H. Murakami and H. Seki).....	11
Figure 2.7: Fuzzification of the variables T_j , d , θ_g , and v . (H. Murakami and H. Seki).....	11
Figure 2.8: Sensors and driving direction on the wheelchair. (H. Murakami and H. Seki).....	12
Figure 2.9: Fuzzy IF-THEN rules for r . (H. Murakami and H. Seki).....	12
Figure 2.10: Fuzzy IF-THEN control rules of θ_d . (H. Murakami and H. Seki).....	13
Figure 2.11: Singleton-type fuzzy reasoning. (H. Murakami and H. Seki).....	14
Figure 2.12: Driving trajectory of experimental result. (H. Murakami and H. Seki).....	14
Figure 2.13: Layout of the ultrasonic and bumper sensors (Li and Yang).....	15
Figure 2.14: Input membership functions (Li and Yang).....	16
Figure 2.15: Subsumption architecture of the robot. (Li and Yang).....	17
Figure 2.16: Obstacle-avoidance behavior (Li and Yang).....	18
Figure 3.1: Overall project flowchart.....	21
Figure 3.2: Differential drive mobile robot.....	22
Figure 3.3: Wheelchair dimensions.....	23
Figure 3.4: IR sensors placing in the wheelchair.....	24
Figure 3.5: Robot workspaces.....	26
Figure 3.6: The robot in workspace 1.....	27
Figure 3.7: The robot in Workspace 2.....	28
Figure 3.8: Complete Simulink model.....	29
Figure 3.9: Inputs subsystem block.....	29
Figure 3.10: Path Extraction subsystem.....	31
Figure 3.11: Controller subsystem.....	32
Figure 3.12: Robot simulator sensor diagram.....	32

Figure 3.13: Outputs subsystem	33
Figure 3.14: Fuzzy controller block diagram	34
Figure 3.15: Membership functions.....	35
Figure 4.1: First simulation test; no obstacles	39
Figure 4.2: Target direction robot response for first simulation test; no obstacles	40
Figure 4.3: Further examples of simulation testing with no obstacles	40
Figure 4.4: Simulation result with obstacles	41
Figure 4.5: Target direction robot response for simulation test; with obstacles	42
Figure 4.6: Obstacle distance read by the sensors	42
Figure 4.7: Further examples of simulation results; with obstacles	43
Figure 4.8: Simulation result for the third map	44
Figure 4.9: Target direction vs the robot angular velocity; third map	44
Figure 4.10: Obstacle distance read by the sensors; third map	45
Figure 4.11: Further examples of simulation results; third map	45
Figure 4.12: Simulation result for the third map (failed to avoid obstacles)	46
Figure 4.13: Controller subsystem; modified for sensor fusion.....	47
Figure 4.14: Robot sensor diagram; sensor fusion	47
Figure 4.15: Simulation result for the third map; with sensor fusion.....	48

CHAPTER 1

INTRODUCTION

1.1 Motivation

The World Health Organization (W.H.O.) has estimated that 10% of the world population have disabilities, i.e. around 650 million people. Researchers also show that almost 10% of these need a wheelchair. An appropriate wheelchair has been defined as a wheelchair that meets the individual's needs and environmental conditions, provides proper fit and postural support based on sound biomechanical principles, is safe and durable, is available and can be accessed, maintained and sustained in the country at the most economical and affordable price [1]. There are many types of wheelchairs to meet the different needs of users and the most used wheelchair type is the traditional manual wheelchair. A manual wheelchair is adequate for most users with physical disabilities but it is not suitable for individuals with a mixture of physical and cognitive disabilities. To accommodate users who find the manual wheelchair unsuitable, researchers have been developing smart/robotic wheelchairs. A robotic wheelchair is a standard powered wheelchair with a computer and a collection of sensors added [2]. One important feature of the robotic wheelchair is obstacle avoidance. A robotic wheelchair that provides obstacle avoidance but does not provide any path-planning assistance gives greater control to the user. Smart wheelchairs in this category would potentially be useful for wheelchair users with—

- Visual impairments who might not see obstacles but can navigate without visual cues.
- Physical impairments that can cause them to temporarily lose control of the chair.
- Cognitive impairments that make driving unsafe (e.g., poor impulse control).

One category of patients who will benefit from a robotic wheelchair with obstacle avoidance capability is quadriplegic patients [2]. Quadriplegia is a paralysis caused by disease or accidental injury that leads to the partial or full loss of use of all limbs and torso. A robotic

wheelchair designed specifically for the aforementioned category would give those patients a form of independence and great mobility.

1.2 Problem Statement

Wheelchairs provide independent mobility for many of its users. However, this mobility is hindered by the many obstacles that exist in the environment. Many wheelchair users suffer from symptoms that make the task of safely avoiding obstacles along their path difficult or impossible to achieve independently. A robotic wheelchair that has obstacle avoidance capability could potentially benefit many wheelchair users suffering from various physical, cognitive, or perceptual symptoms associated with diseases such as spinal cord injury, multiple sclerosis, and cerebral palsy. As such, this project aims to develop an obstacle avoidance algorithm for a robotic wheelchair. However, developing such an algorithm can pose many challenges to the designer, some of these challenges can be attributed to:

- The fact that not all obstacles are the same, e.g., walls, objects, moving pedestrians, etc.
- The limitations of sensors, e.g., a single type of sensor might not be enough to detect an obstacle.
- The limitations of processing power.

1.3 Objectives

The objectives of this project are:

- To develop an obstacle avoidance algorithm with fuzzy behavior-based controller for the modeled wheelchair using MATLAB/Simulink simulation.
- To simulate and analyze the developed algorithm in MATLAB/Simulink environment.

1.4 Scope

The scope of this project is:

- The fuzzy behaviors to be designed includes:

- a. Go-to-Goal
 - b. Avoid Obstacles
- ii. Only static obstacles will be considered for the design.
 - iii. Only three IR sensors will be used.



CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Researchers have been developing robotic wheelchairs for a long time, as such, many theories exist about controlling the robotic wheelchair. The robotic wheelchair can be classified as a differential drive mobile robot, making the theories implemented in controlling mobile robots applicable to the robotic wheelchair. Although the literature on controlling mobile robots covers a wide range of strategies, this review will focus on behavior-based mobile robots and the use of fuzzy logic. Given that the main objective of this project is the development of obstacle avoidance algorithm, this review will examine the literature on obstacle avoidance algorithms.

2.2 Wheelchair Kinematic Model

A wheelchair can be modeled as a differential drive mobile robot (DDMR) with two driving wheels and two free caster wheels (2DW/2FW). The caster wheels are ignored in getting the kinematic model of the system [3]. A kinematic model for differential drive mobile robot is presented in [4].

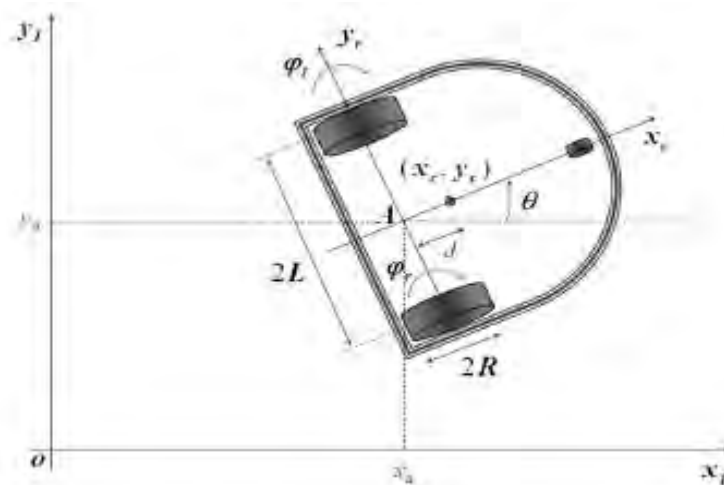


Figure 2.1: Differential drive mobile robot [4]

To determine the kinematic model, first, the coordinate system needs to be defined. Figure 2.1 shows the (DDMR) in the xy -plane with the parameters that define the model. The coordinate systems (frames) are:

- i. Inertial Coordinate System: This coordinate system is a global frame which is fixed in the environment or plane in which the robot moves in. Moreover, this frame is considered as the reference frame and is denoted as $\{X_I, Y_I\}$.
- ii. Robot Coordinate System: This coordinate system is a local frame attached to the robot, and thus, moving with it. This frame is denoted as $\{X_r, Y_r\}$.

The variables that define the model are as follows:

- A: midpoint on the axis between the two wheels.
- (x_c, y_c) : center of mass of the robot, assumed to be at the axis of symmetry, at a distance d from A.
- R: radius of the two wheels.
- $2L$: Length between the two wheels.

The kinematic model is used to study the motion of the system without considering the forces that affect the motion. The goal of this model is to represent the robot velocities as a function of the driving wheels' velocity as well as the other parameters of the robot. The kinematic model of the differential drive mobile robot can be defined as follows:

$$v = \frac{v_R + v_L}{2} = R \frac{(\dot{\phi}_R + \dot{\phi}_L)}{2} \quad (2.1)$$

$$\omega = \frac{v_R - v_L}{2L} = R \frac{(\dot{\phi}_R - \dot{\phi}_L)}{2} \quad (2.2)$$

Where v and ω are the linear and angular velocities of the DDMR respectively. The velocities are then written using point A velocities located in the center of the robot:

$$\begin{cases} \dot{x}_a^r = R \frac{(\dot{\phi}_R + \dot{\phi}_L)}{2} \\ \dot{y}_a^r = 0 \\ \dot{\theta} = \omega = R \frac{(\dot{\phi}_R - \dot{\phi}_L)}{2L} \end{cases} \quad (2.3)$$

Thus,

$$\begin{bmatrix} \dot{x}_a^r \\ \dot{y}_a^r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ 0 & 0 \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\phi}_R \\ \dot{\phi}_L \end{bmatrix} \quad (2.4)$$

In the inertial frame, the velocities can be represented as follows:

$$\dot{q}^I = \begin{bmatrix} \frac{R}{2} \cos \theta & \frac{R}{2} \cos \theta \\ \frac{R}{2} \sin \theta & \frac{R}{2} \sin \theta \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\phi}_R \\ \dot{\phi}_L \end{bmatrix} \quad (2.5)$$

Alternatively, the kinematic model can be obtained by representing the velocities of the differential drive mobile robot using the linear and angular velocities of the robot.

$$\dot{q}^I = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (2.6)$$

In order to have a smooth drive, S. M. Lavalle proposed a second order differential drive model in [5]. This is done by setting the inputs u_l and u_r that accelerate the motors, instead of setting the velocities. Letting ω_r and ω_l represent the angular velocities of the right and left wheels respectively, the state transition equation is

$$\begin{aligned} \dot{x} &= \frac{r}{2} (\omega_l + \omega_r) \cos \theta & \dot{\omega}_l &= u_l \\ \dot{y} &= \frac{r}{2} (\omega_l + \omega_r) \sin \theta & \dot{\omega}_r &= u_r \\ \dot{\theta} &= \frac{r}{L} (\omega_r - \omega_l) \end{aligned} \quad (2.7)$$

2.3 Behavior-Based Robotics

Behavior-based robotics (BBR) is a robotic control strategy that tries to mimic the behaviors of living creatures. The combination and interaction of different behaviors give the desired result. Unlike classical Artificial Intelligence, BBR (shown in Figure 2.2), builds intelligent behaviors using a bottom-up approach [6]. Behavior-based robotics first emerged in

the 1950s when Grey Walter invented the electronic tortoise, which was the first robot to have reactive behavior. The robot had the ability to react to different forms of light intensities in various ways without the robot having any model of its environment [7].

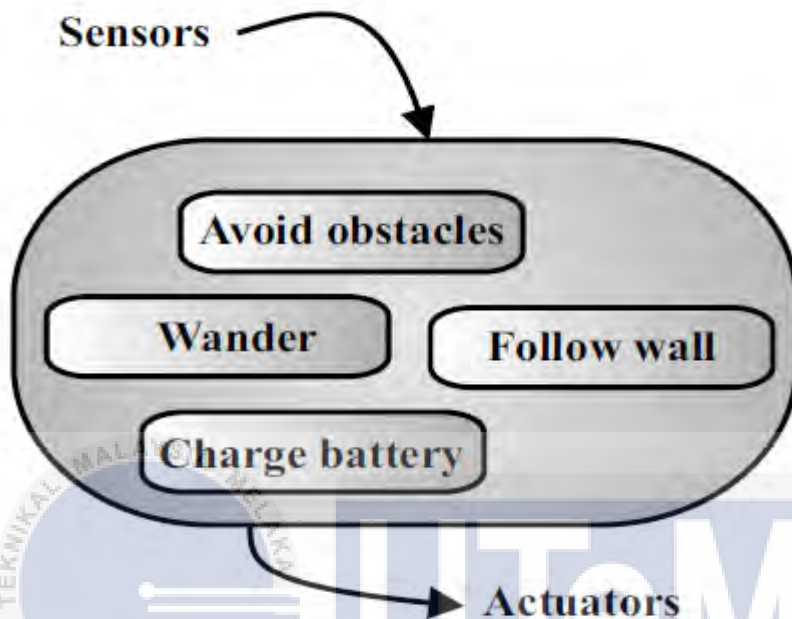


Figure 2.2: Behavior-based robot [6]

Walter's invention and others paved the way for the development of the more organized hierarchal paradigm. In the hierarchal paradigm, the robot systems are designed to follow a rigid order of states defined as "SENSE", "PLAN" and "ACT". In the "SENSE" stage, the robot senses its environment and creates a world model. Then, the "PLAN" stage develops an action strategy based on the received world model. Finally, the "ACT" stage performs the actuator commands set in the "PLAN" stage. This cycle of "SENSE", "PLAN", and "ACT", is repeated until the robot reaches its goal. However, this hierarchal paradigm faced two major challenges. The first issue is that, in the hierarchal paradigm the world is assumed to be closed, i.e., the robot has all the information about the environment. The second issue is called the frame problem, which is the incapability of modeling all environment information required by the robot in an efficient way [7].

The drawbacks of the hierarchical paradigm led to researchers to look for new robotic control paradigm. The technological leaps achieved in cognitive psychology and ethology paved the way for researchers to study animal behavior and implement it in robotics control which led to the development of behavior-based architecture (Figure 2.3).

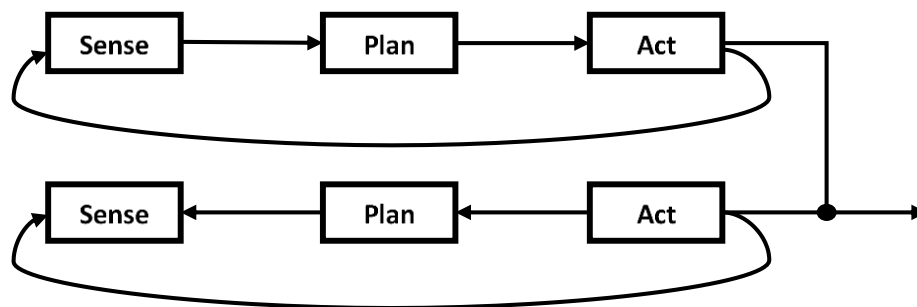


Figure 2.3: Behavior-based architecture [7]

The building of the behavior-based architecture begins with the definition of a behavior. The simplest behavior is basically a mapping of sensor input to actions that achieve a given task. Researchers have categorized animal behaviors to three categories. Namely, reflexive behaviors, reactive behaviors, and conscious behaviors. Reflexive behaviors produce an action that is a direct response to a stimulus without the involvement of a planning stage, for example, tapping of the knee produces a reflexive response. Reactive behaviors are behaviors that are learned over time such as learning to ride a bicycle. Conscious behaviors, on the other hand, require conscious thinking to perform actions. In mobile robotics, reactive behaviors are used more often. Table 2.1 shows the difference between reactive and deliberative architectures.

Table 1.1: Reactive versus deliberative architecture [7]

Deliberative (symbolic)	Reactive (reflexive)
Speed of response	
Predictive capabilities, completeness of world model	
Needs internal representation slow response High-level intelligence (cognition)	No internal representation Real-time response Low-level intelligence Simple (analogue) computation

2.4 Fuzzy Logic Control

Fuzzy control is based on fuzzy logic, which is a logical system that is much closer to human thinking and natural language than traditional logical systems. The fuzzy logic controller based on fuzzy logic provides a means of converting a linguistic control strategy based on expert knowledge into an automatic control strategy [8]. Fuzzy Controllers have been proposed for physical systems that are difficult to model mathematically, and hence, cannot be controlled by traditional control design techniques. Instead, control variables are represented by fuzzy variables which let the level of uncertainty of the variables be modeled in a systematic way [9].

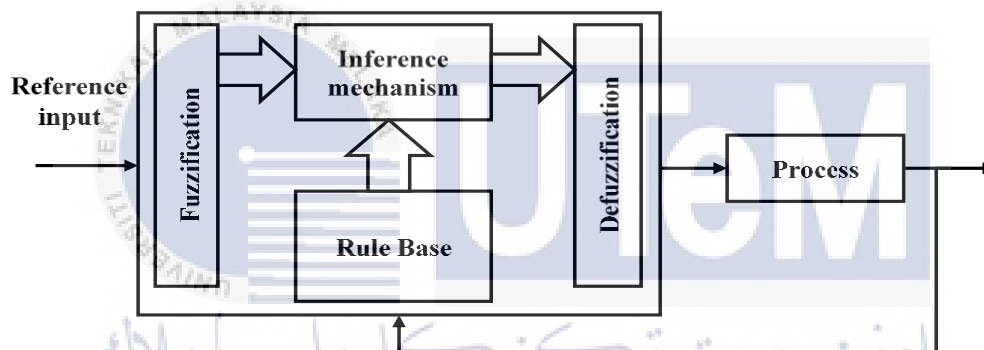


Figure 2.4: Fuzzy logic controller [10]

The fuzzy logic control is a method of artificial intelligence designed to control something usually mechanical. The fuzzy logic controller consists of several components, namely, the rule-base, fuzzification, inference mechanism and defuzzification as shown Figure 2.4 [10]. In traditional set theory built on Boolean or crisp variables, the value can only be either 1 or zero. However, in fuzzy set theory, a variable can have a membership or grade of zero to one, and this makes it different from the crisp set [9]. In Figure 2.5, a comparison between crisp and fuzzy set for a membership function is shown of the fuzzy variable "No. of individuals".

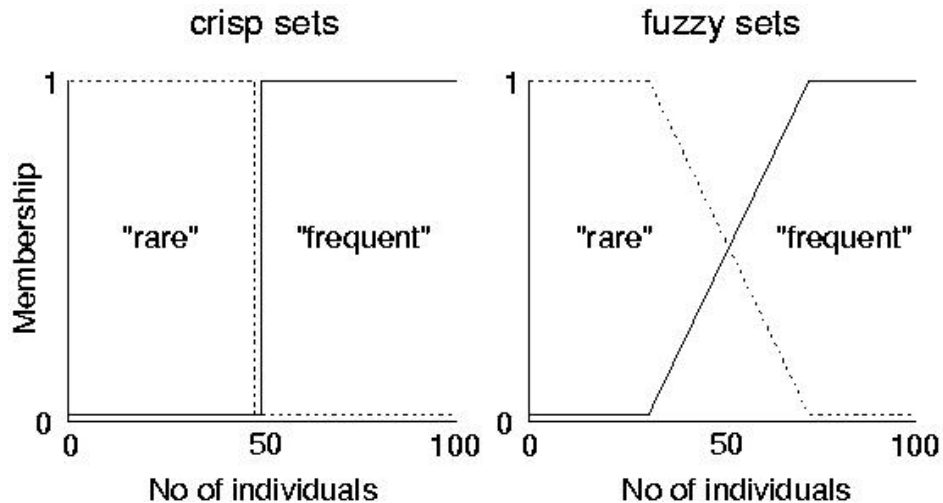


Figure 2.5: Comparison between crisp and fuzzy sets [9]

Classical control theory relies on the availability of a mathematical model of the plant (process) to be controlled. However, when the physical system is time varying and nonlinear - which is the case in most physical systems, techniques such as adaptive and robust methods are used to for corrective measures. Fuzzy set theory is used to model the nonlinearities and uncertainties of plant or control variables. In fuzzy control algorithm, the control law is described by a set of IF...THEN rules similar to an expert system based control. A typical rule has the following format:

IF x is A and y is B THEN z is C

Where x , y , and z are fuzzy control variables, and A , B , and C are the fuzzy subsets in the universe of discourses (all the possible values that a variable can assume) X , Y , and Z , respectively [9].

H. Murakami and H. Seki developed in [11] a fuzzy logic based obstacle avoidance algorithm to control an electric powered wheelchair equipped with ultrasonic sensors. Their control system was constructed based on four signals, the joystick command T_j , the distance to the obstacle d , the difference angle θ_g , between the joystick command T_j and the driving direction θ_d , and the velocity v of the wheelchair. Another variable introduced by the authors is the driving risk r , which is determined based on fuzzy control. The driving control system flowchart is shown in Figure 2.6. The fuzzification of the variables T_j , d , θ_g , and v are shown in Figure 2.7. For the fuzzy variable T_j , the symbols “LB”, “LM”, “ZO”, “RM”, and “RB” stands for Left-Big, Left-Middle, Zero, Right-Middle, and Right-Big respectively. For the

fuzzy variable d , the symbols “S”, “M”, and “L” stands for Short, Middle, and Long respectively. For the fuzzy variables θ_g and θ_d , the symbols “SS”, “S”, “M”, “B” and “BB” stands for Small-Small, Small, Middle, Big, and Big-Big respectively.

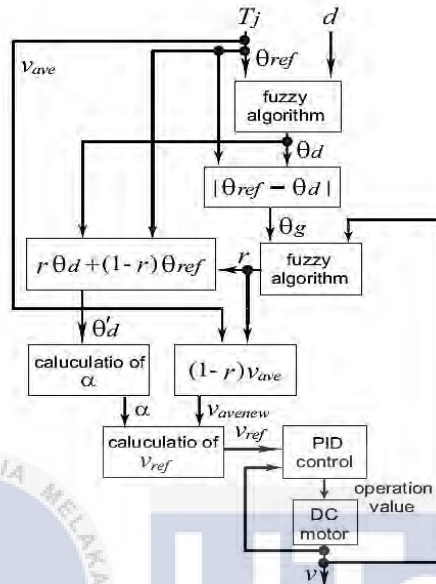


Figure 2.6: Driving control system [11]

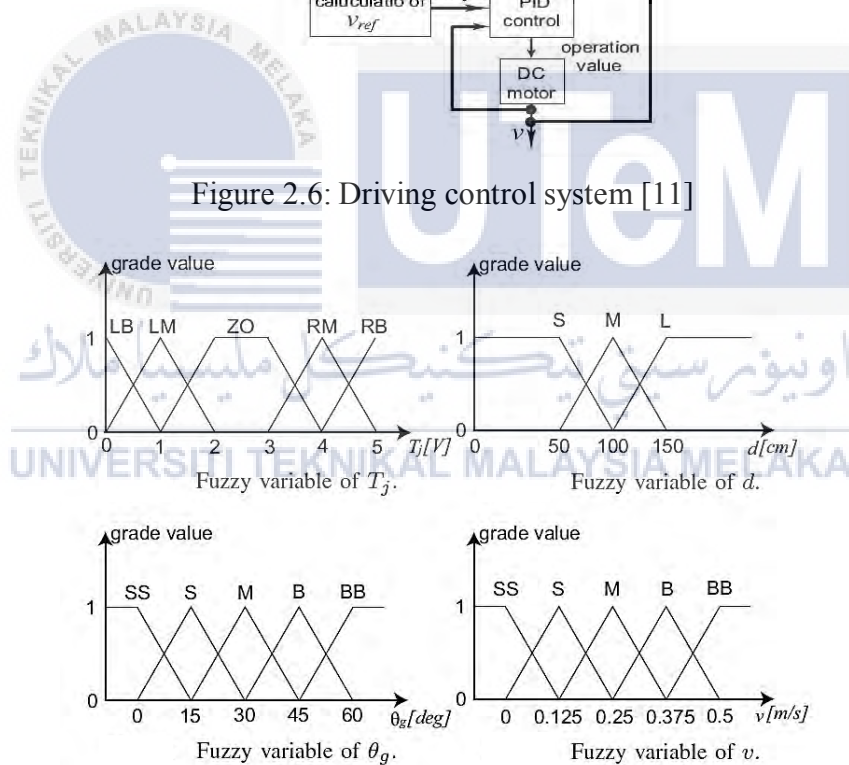


Figure 2.7: Fuzzification of the variables T_j , d , θ_g , and v [11]

The authors used four ultrasonic sensors to detect obstacles. Figure 2.8 shows the sensor placement on the wheelchair as well as the driving direction of the wheelchair.

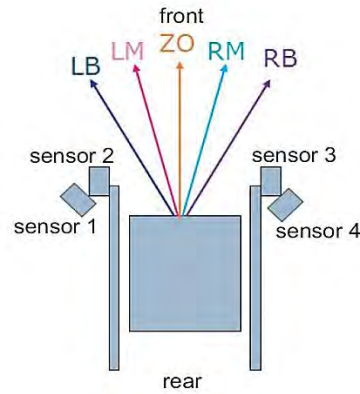


Figure 2.8: Sensors and driving direction on the wheelchair [11]

The IF-THEN fuzzy rules of the control system to calculate the driving direction θ_d and the driving risk r are shown in Figure 2.9 and 2.10 respectively. The authors designed the control system so that when the sensors detect no obstacles, the wheelchair will use the driving direction of the joystick. When an obstacle is detected, the fuzzy control system determines the new driving direction based on the fuzzy rule table.

θ_g^v	SS	S	M	B	BB
SS	SS	SS	SS	SS	SS
S	SS	S	S	S	S
M	S	S	M	M	M
B	S	M	M	B	B
BB	M	B	B	BB	BB

Figure 2.9: Fuzzy IF-THEN rules for r [11]

					joystick operation				
					LB	LM	ZO	RM	RB
sensor output	sensor 1	S	sensor 2	S	RB	RB	RB	RB	
				M	RM	RM	RM		
				L	ZO	ZO			
		S		RB	RB	RB	RB		
		M		RM	RM	RM			
		L							
	sensor 4	S	sensor 3	S		LB	LB	LB	LB
				M			LM	LM	LM
				L				ZO	ZO
		S			LB	LB	LB	LB	
		M				LM	LM	LM	
		L							
sensor 4	S	L	S		LB	LB	LB	LB	
			M			LM	LM	LM	
			L						

Blank spaces show the cases which fuzzy control isn't applied.

Figure 2.10: Fuzzy IF-THEN control rules of θ_d [11]

The fuzzy control system developed by the authors applies “Min-Max” method. This method selects the rule based on the grade. The output of the fuzzy control system is shown in Figure 2.11. The system uses singleton type fuzzy reasoning as the defuzzification method. The driving direction θ_d and the driving risk r , are determined by calculating the center of mass by using equations 2.8 and 2.9.

$$\theta_d = \frac{60 \times LB + 75 \times LM + 90 \times ZO + 105 \times RM + 120 \times RB}{LB + LM + ZO + RM + RB}$$

(2.8)

$$r = \frac{0 \times DSS + 0.25 \times DS + 0.5 \times DM + 0.75 \times DB + 1 \times DBB}{DSS + DS + DM + DB + DBB} \quad (2.9)$$

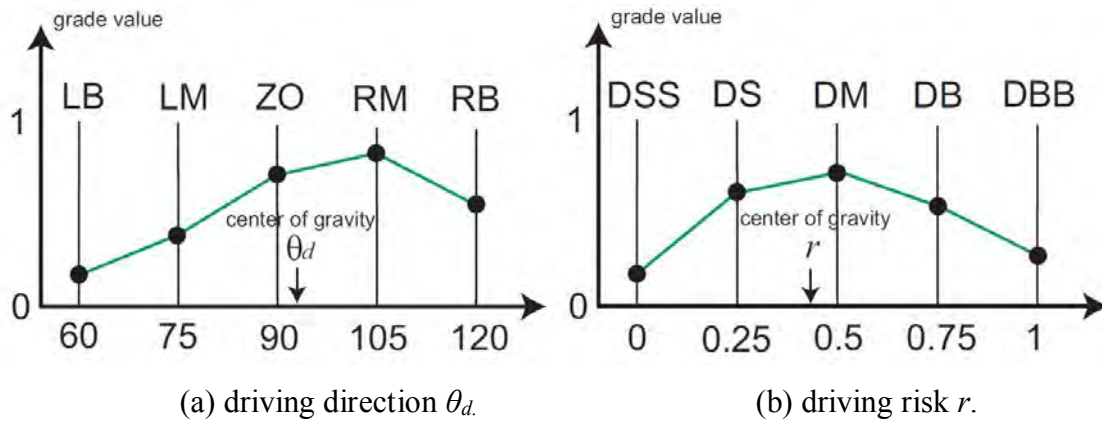


Figure 2.11: Singleton-type fuzzy reasoning [11]

The fuzzy control system designed by the authors managed to avoid obstacles as shown in Figure 2.12. The driving risk r , is a new variable proposed by the authors that reduce the velocity of the wheelchair depending on the difference angle θ_g and the velocity v . This proposed variable reduce the velocity of the wheelchair to ensure safe navigation.

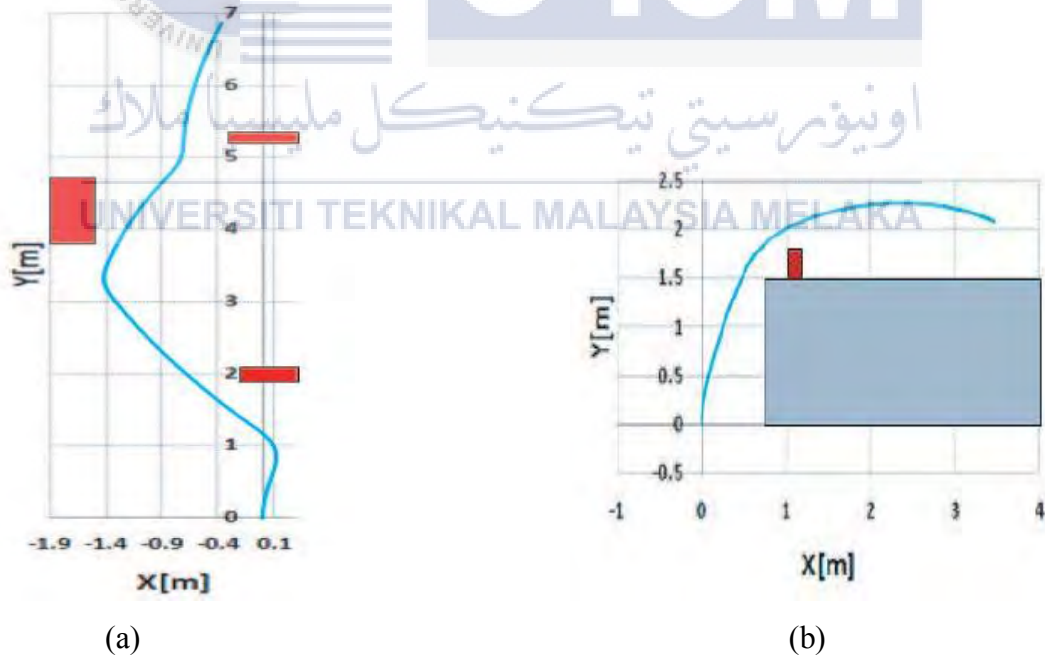


Figure 2.12: Driving trajectory of experimental result [11]

2.5 Fuzzy Behavior-Based Algorithm in Mobile Robotics

Nowadays, the use of fuzzy logic in the design of navigation behaviors for mobile robots is becoming the preferred method for mobile robot designers. Behaviors used to control the navigation of the mobile robot can include, wall following, obstacle avoidance, or navigating through corridors. However, in building these behaviors there is not a concrete way of designing their rule bases. A lot of approaches rely on expert knowledge to design the behavior response based on the objective of the behavior without the objective being explicitly defined [12].

Due to the capability of fuzzy systems to handle uncertainties and imprecise information using linguistic rules, many researchers have investigated fuzzy logic approaches to defining behaviors in behavior based architecture.

Li and Yang proposed an obstacle avoidance approach using fuzzy logic in [13]. The algorithm was based on behavior-based artificial intelligence and built for a fully autonomous mobile robot. A set of eight ultrasonic sensors was built to implement obstacle avoidance using a set of fuzzy rules. The eight ultrasonic sensors were installed on the robot with four in the front, two on the back, and one on each side of the robot (Figure 2.13).

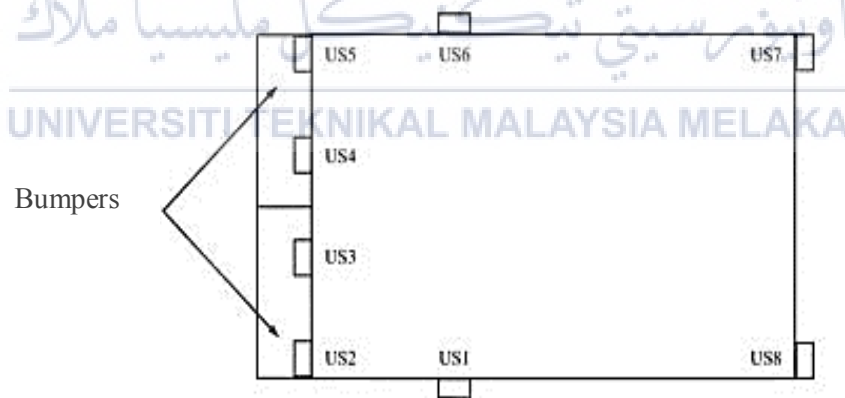


Figure 2.13: Layout of the ultrasonic and bumper sensors [13]

The obstacle avoidance system consists of eight separate fuzzy logic controllers with the same structure but different rule bases. The fuzzification of the input uses three membership functions for each ultrasonic sensor which are “too close”, “close”, and “far” as shown in Figure 2.14.

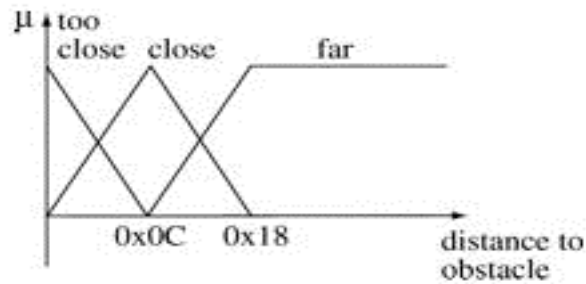


Figure 2.14: Input membership functions [13]

The inference mechanism used by Li and Yang in their proposed design is shown in Table 2.1. The outputs of the system are the linear velocity, v , and rotation angle, Φ . “positive fast”, “positive”, “zero”, “negative”, “negative fast” are the fuzzy values for the linear velocity. “more left”, “left”, “zero”, “right”, “more right” are the fuzzy values for the rotation angle.

Table 2.2: Rule table for ultrasonic sensors [13]

	Distance to the obstacle		
	far	close	Too close
v_1	zero	positive	positive fast
Φ_1	zero	right	more right
v_2	zero	negative	negative fast
Φ_2	zero	left	more left
v_3	zero	negative	negative fast
Φ_3	zero	left	more left
v_4	zero	negative	negative fast
Φ_4	zero	right	more right
v_5	zero	negative	negative fast
Φ_5	zero	right	more right
v_6	zero	positive	positive fast
Φ_6	zero	left	more left
v_7	zero	positive	positive fast

Φ_7	zero	right	more right
v_8	zero	positive	positive fast
Φ_8	zero	left	more left

The defuzzification procedure gives the crisp output signal corresponding to the fuzzy output. Li and Yang used “center of gravity” (COG) defuzzification method to integrate the outputs from all the rules. The robot behavior to the obstacle detected by the ultrasonic sensors relies on the output of the fuzzy controllers that represent the obstacle location. If the sensors detect a stimulus, it is treated as a force vector pushing the robot. A resultant direction is calculated by:

$$V = \frac{1}{8} \sum_{i=1}^8 v(i) \quad (2.10)$$

$$\Phi = \frac{1}{8} \sum_{i=1}^8 \Phi(i) \quad (2.11)$$

The authors’ design of the behaviors implemented a subsumption architecture as shown in Figure 2.15. The problem of robot navigation was vertically decomposed into smaller sub-problems, i.e., sensing, mapping sensor data into a world representation, path planning, task execution, and motor control.

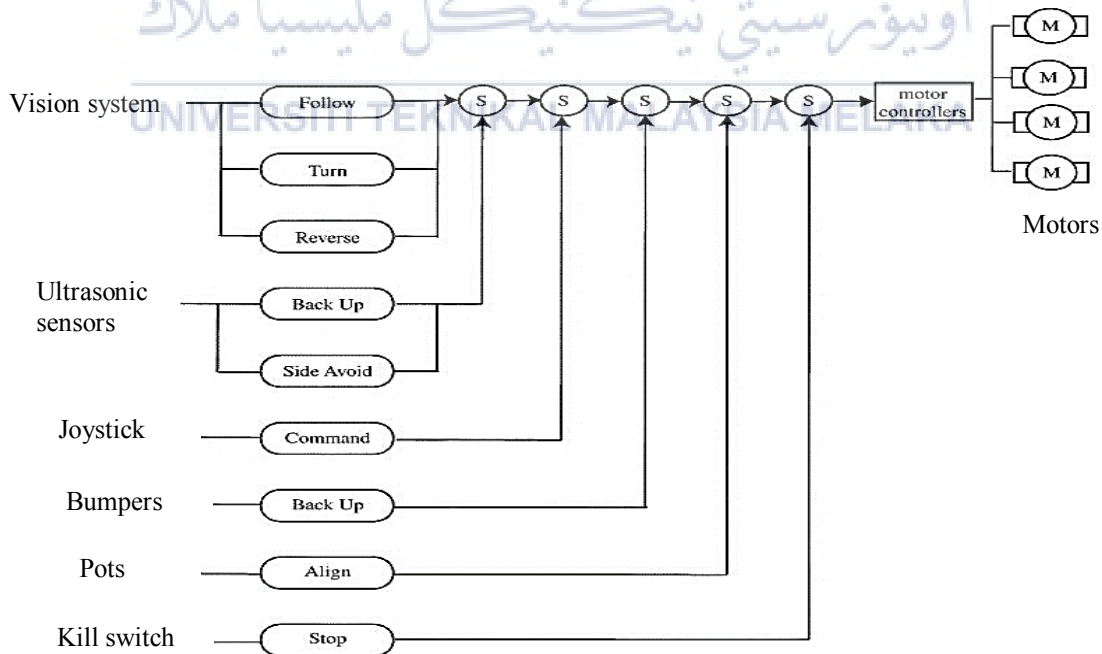
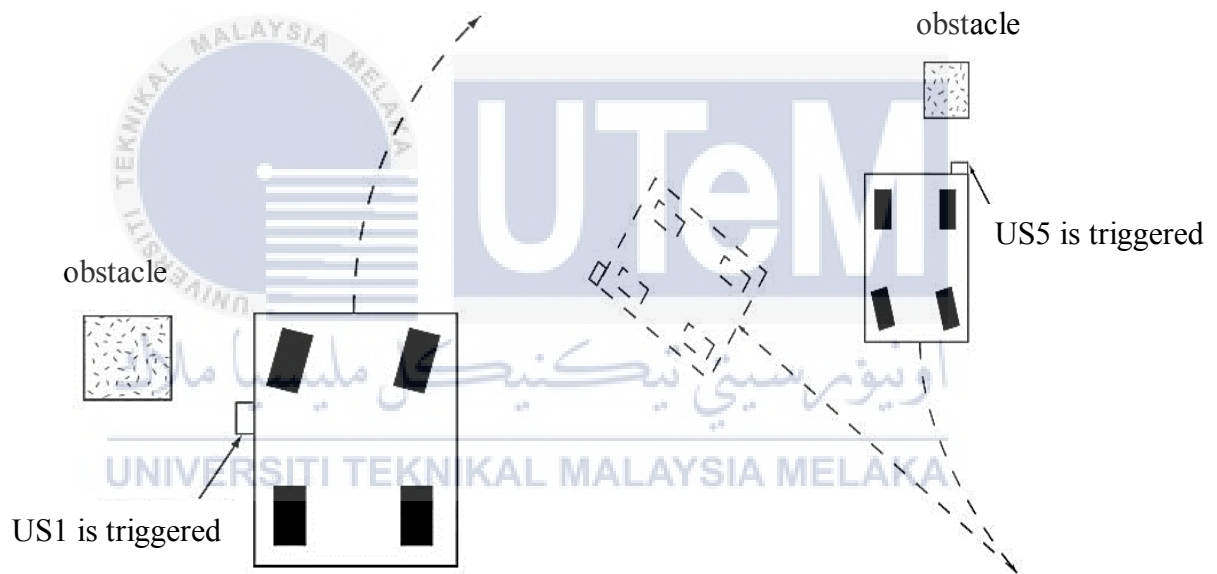


Figure 2.15: Subsumption architecture of the robot [13]

The design of behaviors implemented five layers of competencies which are:

- 0) stop immediately at emergency via the kill switch,
- 1) move along the predefined trajectory,
- 2) stop automatically when hit an obstacle,
- 3) move under human operation via the joystick,
- 4) avoid obstacles via fuzzy control,
- 5) reason about the world in terms of identifiable objects, head for the target and perform tasks related to certain objects.

Figure 2.16 (a) and (b) show the robot behavior when an obstacle is on the left side of the robot and when the obstacle is in the right front of the robot respectively.



(a) obstacle on the left side of the robot. (b) Obstacle in the right front of the robot.

Figure 0.16: Obstacle-avoidance behavior [13]

The proposed algorithm by Li and Yang in [13] manages to avoid obstacles. However, the input sensors are separately inferred and only a few simple cases are shown in the paper [12].

2.6 Summary of the Review

The wheelchair can be modeled as a differential drive mobile robot. The caster wheels on the wheelchair are ignored in the determination of the kinematic model as they are free wheels and have negligible effect. The kinematic model of the wheelchair is shown in equations (2.1 to 2.6). In order to have a smooth drive, S. M. Lavelle proposed a second order differential drive model in [5]. This is done by setting the inputs u_l and u_r that accelerate the motors, instead of setting the velocities. Letting ω_r and ω_l represent the angular velocities of the right and left wheels respectively, the state transition is shown in equation (2.7).

The use of behavior-based architecture in mobile robotics application has many advantages as it simplifies the implementation of complex systems by dividing the navigation tasks into simpler subtasks such as target seeking, wall following, and obstacle avoidance.

The fuzzy logic obstacle avoidance algorithm developed by H. Murakami and H. Seki managed to avoid obstacles. The driving risk r , is a new variable proposed by the authors that reduce the velocity of the wheelchair depending on the difference angle θ_g and the velocity v . Li and Yang proposed an obstacle avoidance algorithm based on fuzzy behavior-based architecture and built for a fully autonomous mobile robot. The algorithm could avoid obstacles using five levels of competencies. However, the input sensors are separately inferred.

There are many methods to develop obstacle avoidance algorithms that utilize behavior based architecture. The work done by H. Murakami, et al. and Li and Yang will be used as a guideline to develop the obstacle avoidance algorithm in this project.

CHAPTER 3

METHODOLOGY

This chapter describes the project methodology that is used to implement the project. The chapter includes the project workflow to achieve the objectives, the proposed wheelchair system design, the software setup for the simulation, and the method used to design the fuzzy behavior-based obstacle avoidance algorithm.

3.1 Project Methodology

This project starts by studying the wheelchair kinematic model and obstacle avoidance algorithms. After completing the research and literature review of the project, the software setup for the simulation will be built to test the developed algorithm. The wheelchair dimensions and will be taken from an actual wheelchair that belongs to the Rehabilitation Engineering and Assistive Technology (REAT) lab in UTeM at the Faculty of Electrical Engineering. Figure 3.1 shows the flowchart of the project methodology.

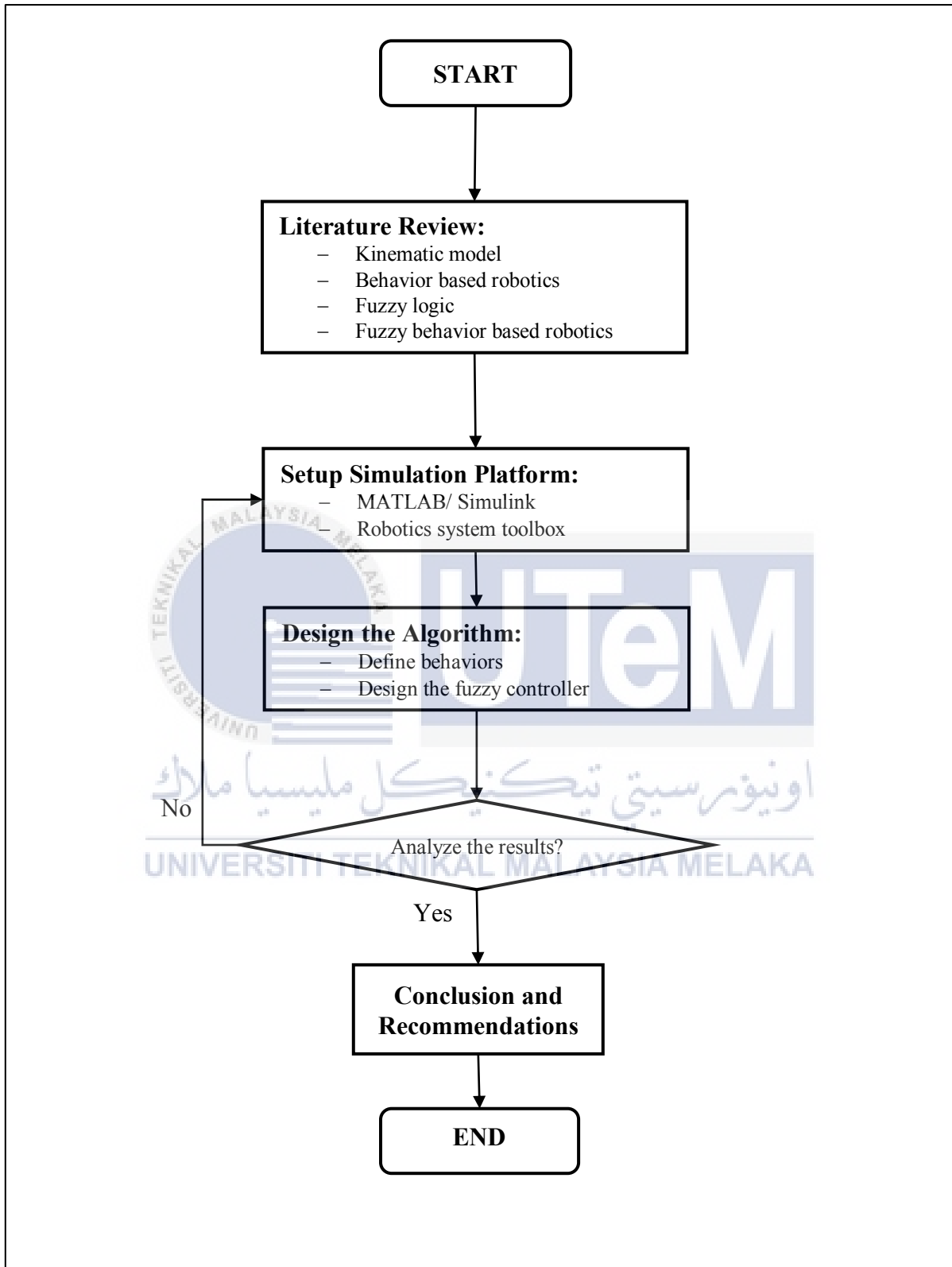


Figure 3.1: Overall project flowchart.

3.2 Wheelchair Model

The wheelchair model used in this project is formulated based on an actual wheelchair in the Rehabilitation Engineering and Assistive Technology Research (REAT) Laboratory at the Faculty of Electrical Engineering in UTeM. Figure 3.2 and equations (3.1) and (3.2) represent the kinematic model of the wheelchair [4]:

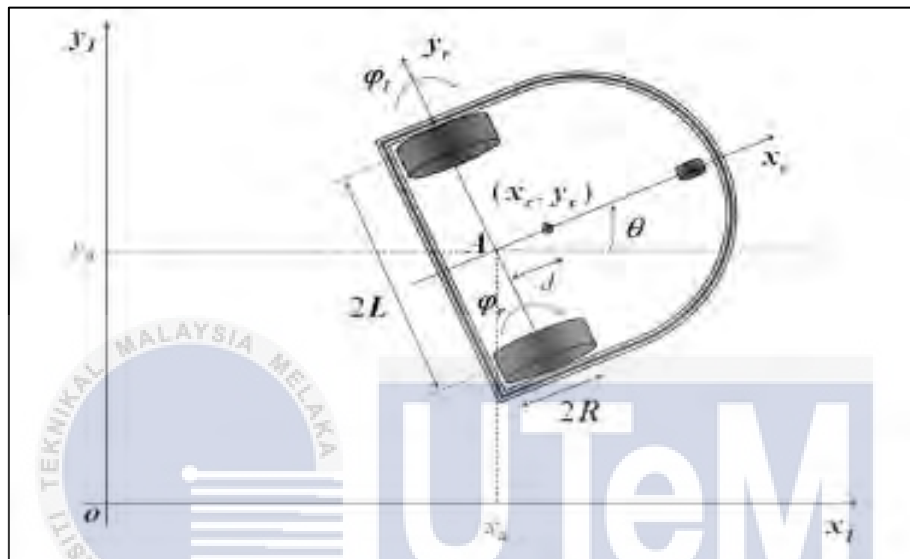


Figure 3.2: Differential drive mobile robot [4]

$$v = \frac{v_R + v_L}{2} \quad (3.1)$$

$$\omega = \frac{v_R - v_L}{2L} \quad (3.2)$$

Where;

- v = the linear velocity of the wheelchair,
- ω = the angular velocity of the wheelchair,
- v_R = the linear velocity of the right wheel,
- v_L = the linear velocity of the left wheel, and
- $2L$ = the distance between the two wheels.

The dimensions of the wheelchair are shown in Figure 3.3 and Table 3.

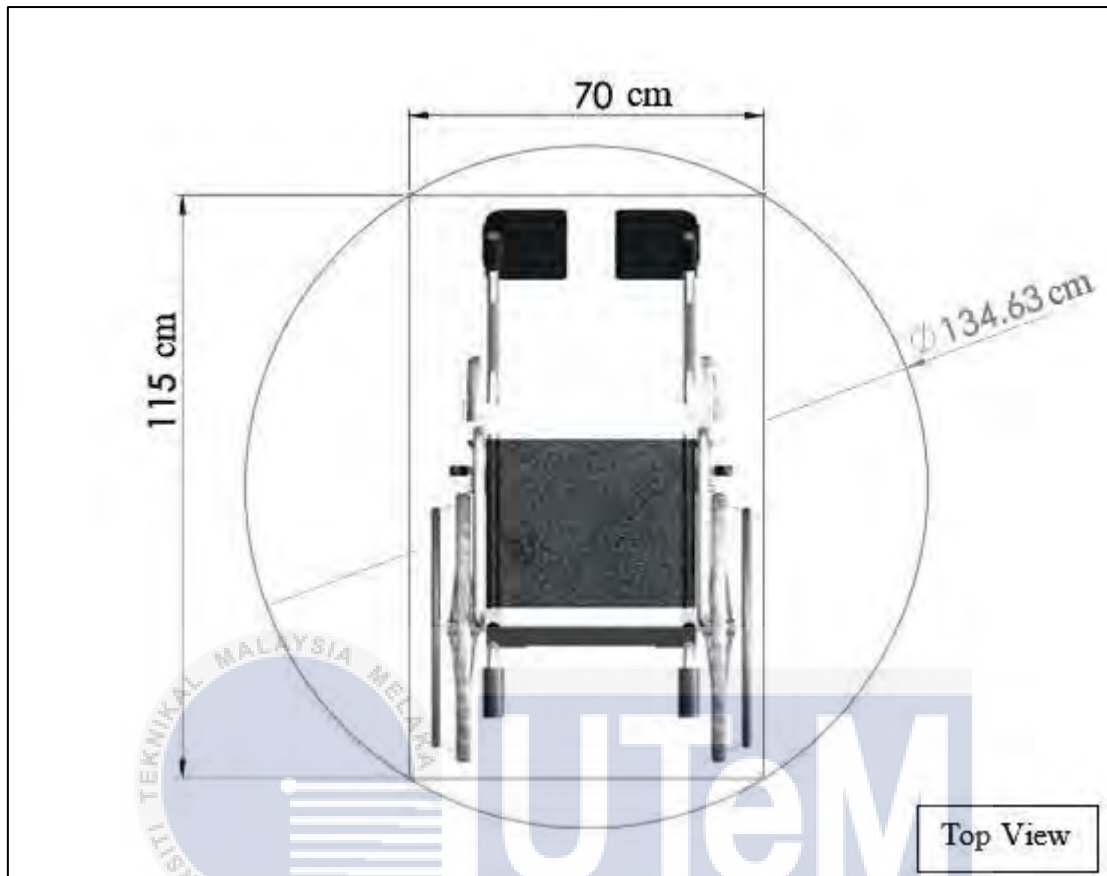


Figure 3.3: Wheelchair dimensions

Table 3.1: Wheelchair dimensions

Wheel radius, R	Distance between the two wheels, 2L	Wheelchair size, (radius)
26 cm	70 cm	67 cm

3.3 Distance Measuring Sensors

The distance measuring sensors that will be utilized in the simulation of the obstacle avoidance algorithm are three GP2Y0A02YK0F IR sensors. GP2Y0A02YK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector), IRED (infrared emitting diode) and signal processing circuit [14]. The sensor has a measuring distance range of 20 to 150 cm. The three sensors are placed to the right, left and center of the wheelchair at the front side. The exact angles of the sensors are shown in Figure 3.4. The differential drive robot simulator that is used in this project employs laser sensors to

measure the distance to obstacles, hence, a few modifications need to be made in order to correctly model the GP2Y0A02YK0F sensor. The laser sensors in the robot simulator have a range of 0 to 5 m and output a “nan” value (*nan* is the IEEE arithmetic representation for Not-a-Number) when the distance is more than 5 m. To model the GP2Y0A02YK0F correctly, the *nan* values need to be changed to the maximum value (5 m). Then, the measuring range of the GP2Y0A02YK0F sensor is accounted for in the fuzzy logic controller and the design of the membership functions.

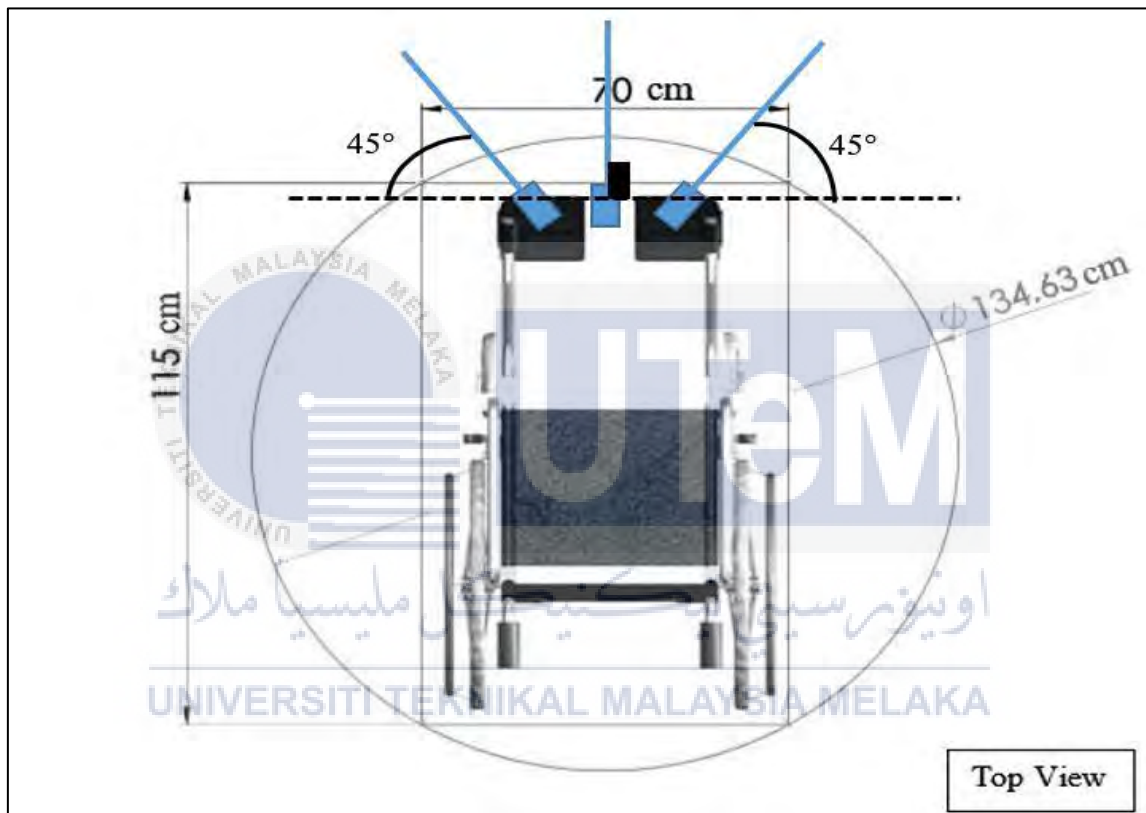


Figure 3.4: IR sensors placing in the wheelchair

3.4 Simulator Setup

To design and test the algorithm, MATLAB and Simulink will be used. The Robotics System Toolbox in MATLAB and Simulink includes a differential drive mobile robot simulator. The Simulink model uses Robot Operating System (ROS) to interface with the robot simulator [15]. An example model available in MATLAB version 2016b was used as a

reference in developing the simulation model for this project [16]. The model is set up in the following steps:

1. Robot workspace configuration,
2. robot configuration, and
3. Simulink model configuration.

3.4.1 Robot Workspace Configuration

To analyze the obstacle avoidance algorithm, three workspaces (or maps) are created. The first map will not include obstacles, while the second map contains three obstacles. The third map is built into MATLAB's Robotic Systems Toolbox. Table 4 shows the MATLAB commands used to create each map and Figures 3.5 shows the created maps.

Table 3.2: MATLAB commands to create the robot's workspaces

Map 1	Map 2
<pre>map = robotics.BinaryOccupancyGrid(14,14); figure show(map)</pre>	<pre>map = robotics.BinaryOccupancyGrid(14,14); x = [2;7;11]; y = [5;7;10]; setOccupancy(map, [x y], ones(3,1)) figure show(map)</pre>
Map 3	
<pre>ExampleHelperSimulinkRobotROS('ObstacleAvoidance')</pre>	

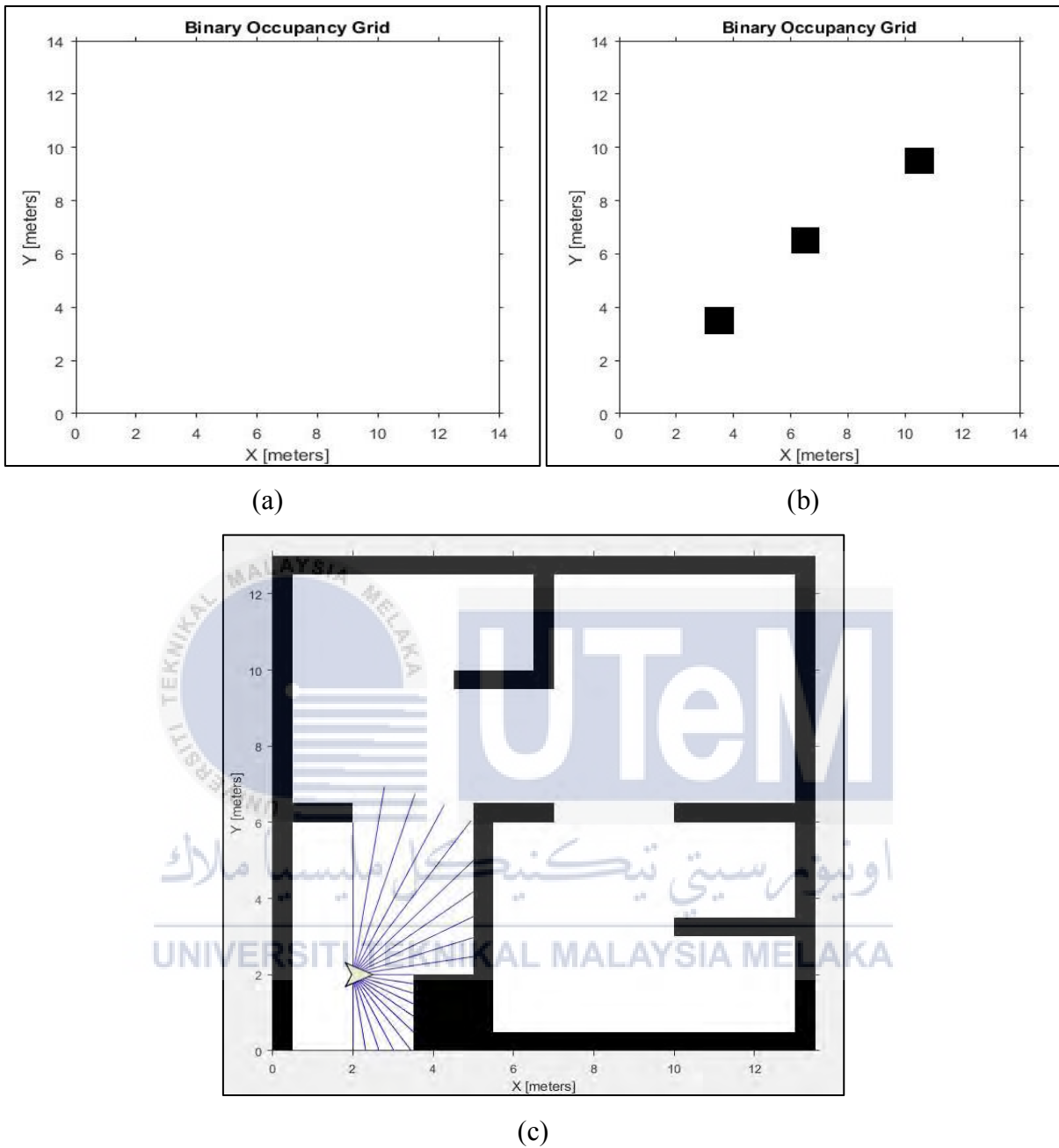


Figure 3.5: Robot workspaces; (a) no obstacles, (b) with three obstacles and (c) with walls

3.4.2 Robot Configuration

A differential drive mobile robot simulator is included in the Robotics System Toolbox in MATLAB. To setup the simulator for the first and second, certain parameters are configured based on the wheelchair model developed earlier. Table 5 shows the MATLAB commands used to configure the differential drive mobile robot. Figure 3.7 and 3.8 shows the

robot placed in the two workspaces developed for this project. The robot configuration for map 3 is fixed and cannot be changed.

Table 3.3: MATLAB commands to configure the robot

```
obj.Simulator= ExampleHelperRobotSimulator(map);
% Set robot's maximum speed
obj.Simulator.Robot.MaxLinearVelocity = 2;
obj.Simulator.Robot.MaxAngularVelocity = 1;
% Set robot's initial pose
obj.Simulator.setRobotPose([2.5,1,(pi/2)]);
% Set robot's size (bounding radius)
obj.Simulator.setRobotSize(0.67);
% Enable ROS interface for simulator
obj.Simulator.enableROSInterface(true);
% Disable Laser sensor
obj.Simulator.enableLaser(true);
% Disable velocity command timeout, i.e. the robot will follow the latest
% velocity command indefinitely.
obj.Simulator.Robot.enableLimitedCommandTime(true);
% Enable trajectory plot
obj.Simulator.showTrajectory(true);
```

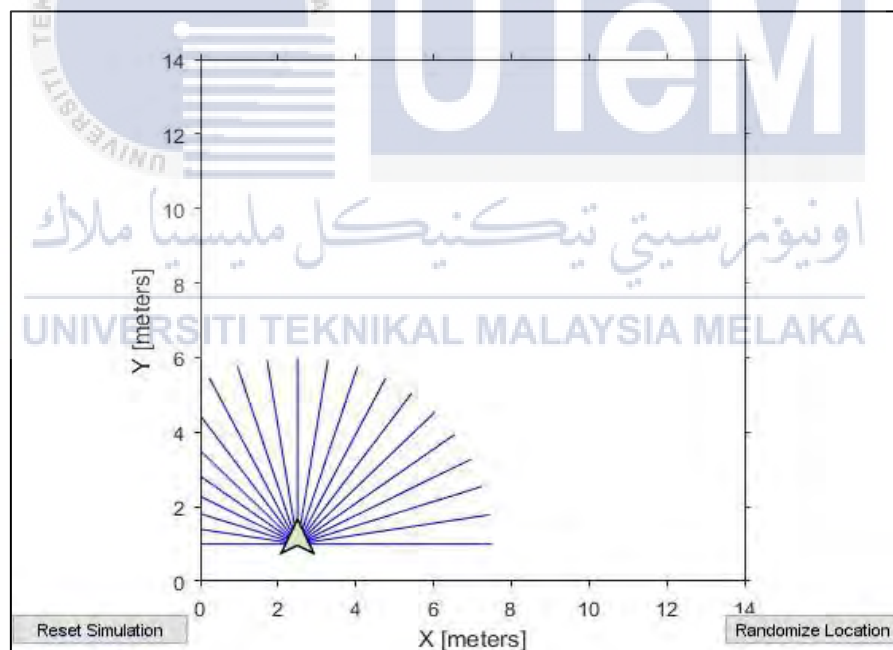


Figure 3.6: The robot in workspace 1

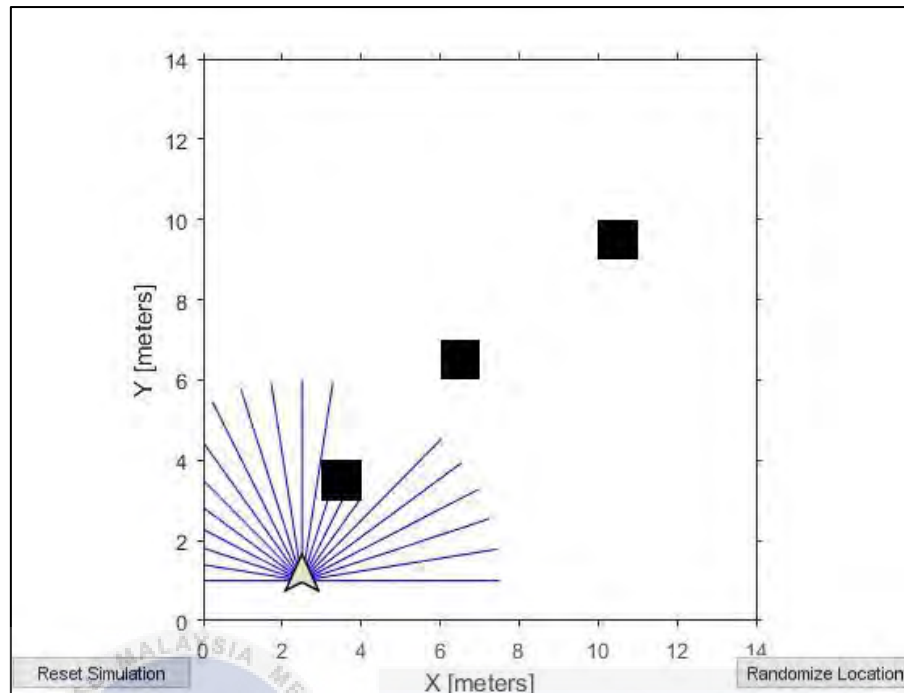


Figure 3.7: The robot in Workspace 2

The robot simulator uses laser sensors with a range of 5 meters. To adapt the laser sensors to the GP2Y0A02YK0F sensor, certain manipulations are created in the Simulink model explained in the following section.

3.4.3 Simulink Model Configuration

The last step in the Simulator setup is the Simulink model configuration. Figure 3.9 shows the complete Simulink model. The model consists of five subsystems which are:

1. Inputs subsystem,
2. Path Extraction subsystem,
3. Controller subsystem, and
4. Outputs subsystem.

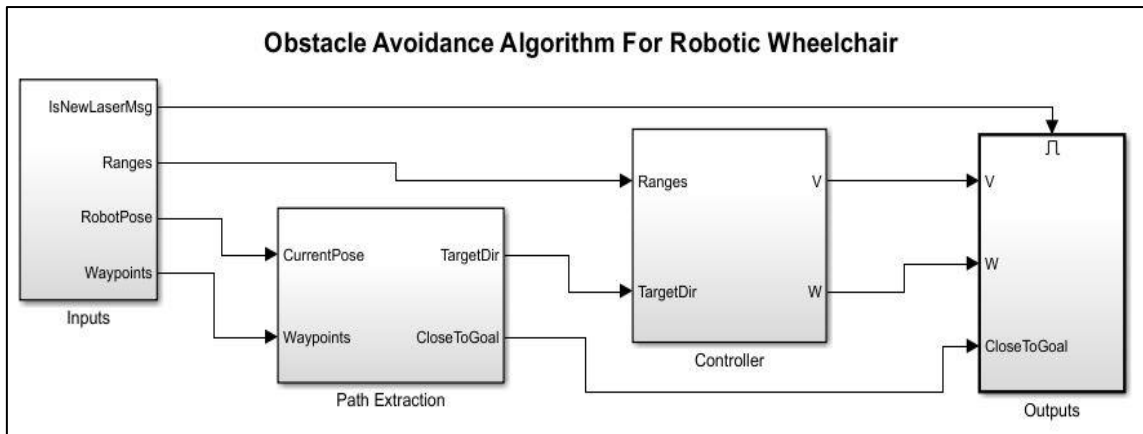


Figure 3.8: Complete Simulink model

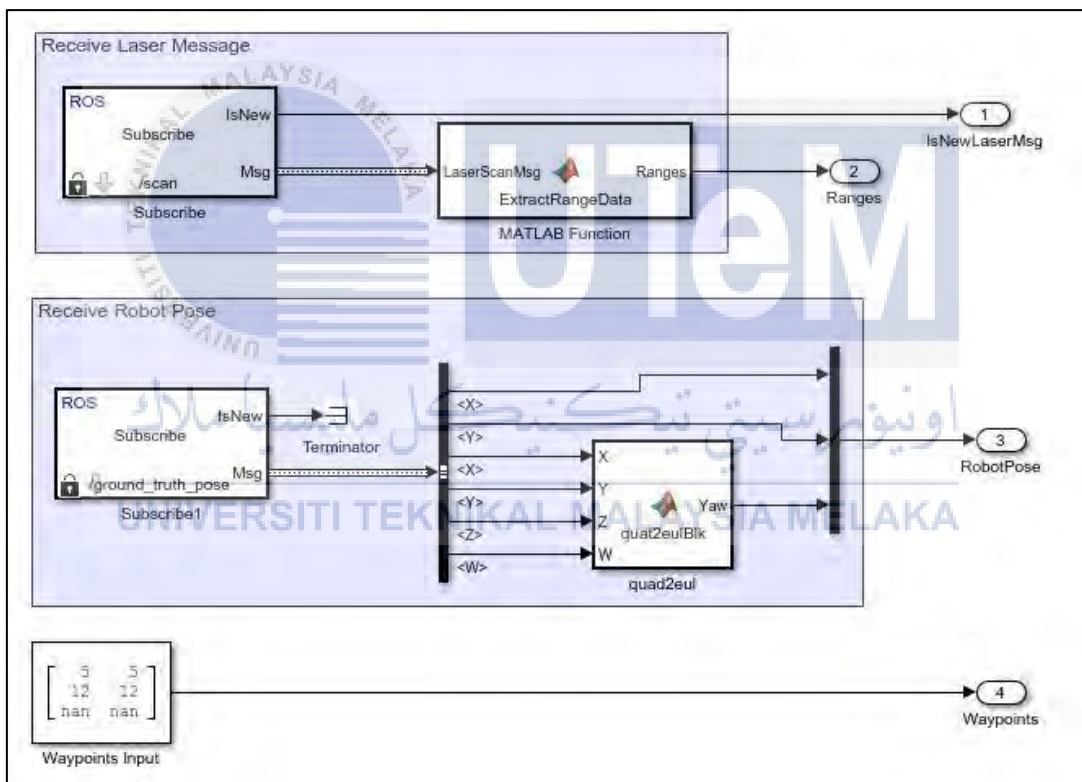


Figure 3.9: Inputs subsystem block

The Simulink model interfaces with the robot and maps that were developed in the previous sections using ROS subscribers and publishers. The Inputs subsystem includes blocks and functions to get the robot pose, sensor ranges, and goal. Figure 3.10 shows the Inputs subsystem. The subscribers are used to get the current robot laser ranges readings (distance to obstacles) and pose from the robot simulator. The *ExtractRangeData* MATLAB function

block includes commands to extract the laser ranges readings. While the *quat2eulBlk* MATLAB function block converts the robot's pose angles from quaternion to Euler angles. Table 4 shows the code inside the two blocks. The final target or goal to which the robot should go to is set using the *constant* block as shown in the bottom left corner of Figure 3.10. The value of the constant blocks represents the waypoints for the robot to follow. In this project, the target is set to point (12,12) with the point (5,5) as a waypoint. The *(nan,nan)* is used as a terminator to indicate that the robot has reached its target.

Table 3.4: MATLAB commands for ExtractRangeData and quat2eulBlk function blocks

ExtractRangeData	quat2eulBlk
<pre>function Ranges = ExtractRangeData(LaserScanMsg) % Extract Range information Ranges = double(LaserScanMsg.Ranges); %if the value of the extracted range is nan change the value to 5m (Max laser sensor range) Ranges(isnan(Ranges))=5;</pre>	<pre>function Yaw = quat2eulBlk(X, Y, Z, W) eularAngles = quat2eul([W, X, Y, Z]); Yaw = eularAngles(1);</pre>

The second subsystem shown in Figure 3.11 is the Path Extraction subsystem. In this subsystem, a path for the robot to follow is generated using the *Pure Pursuit* function block, which is included in the Robotics System Toolbox. The *Pure Pursuit* block computes the linear and angular velocity control commands for path following using waypoints and the current pose of the robot. It also computes the target direction which is used in this project as the reference angle for the fuzzy logic controller. The linear and angular velocity outputs of the *Pure Pursuit* block are ignored and left unconnected. Another function of the Path Following subsystem is to determine if the robot has reached its goal. This is done by first extracting the goal using the *extractGoal* function block (Table 5) then subtracting the goal value from the current pose of the robot. The output of the subtraction is then multiplied by itself using the dot product, then the square root is taken. The result of the square root represents the distance to the goal. Next, a comparison is made to determine whether the robot has reached its goal by comparing the distance to goal with the goal radius. If the distance to

the goal is less than or equal to the goal radius then a true signal is sent to the Outputs subsystem to stop the robot.

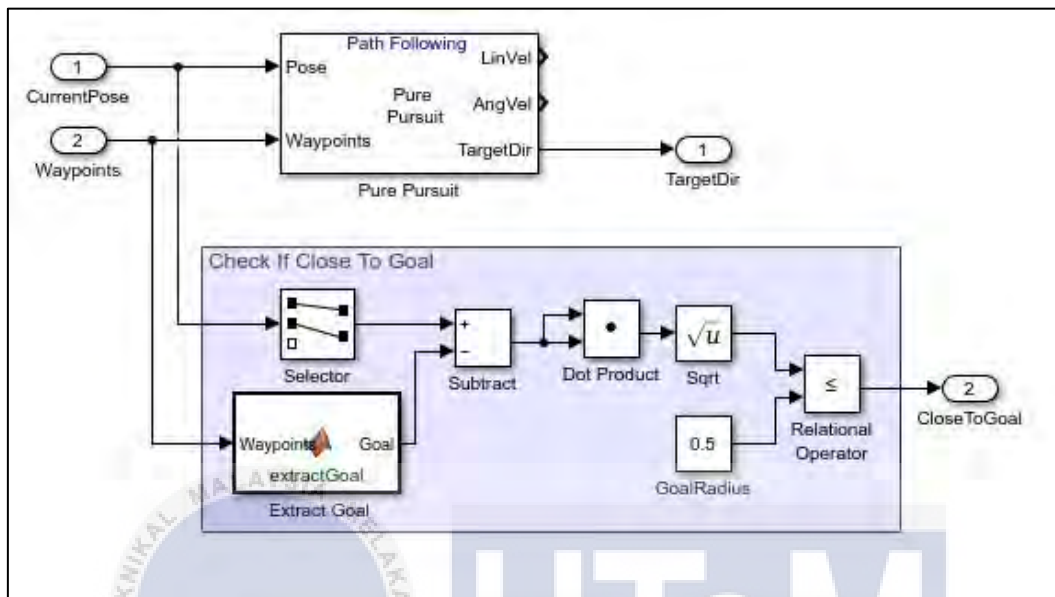


Figure 3.10: Path Extraction subsystem

Table 3.5: extractGoal function block

```
function Goal = extractGoal(Waypoints)
b = isnan(Waypoints);
[rows,~] = find(b);
ridx = setdiff(1:size(Waypoints,1), sort(rows));
Goal = [Waypoints(ridx(end), 1) Waypoints(ridx(end), 2)];
```

The third subsystem is the Controller subsystem shown in Figure 3.12. This subsystem can be divided into three sections. The first section is where the sensor readings are passed to the fuzzy logic controller together with the target direction (reference angle). However, since only three sensors are needed in this project, other sensors are ignored as shown in Figure 3.13. The three sensor readings and the target direction are multiplexed into a single signal and fed into the second section which is the fuzzy logic controller. The third section of the subsystem is where the kinematic model of the wheelchair represented by equations (3.1) and (3.2) is implemented to get the linear and angular velocities of the wheelchair. These velocities are then passed to the Outputs subsystem.

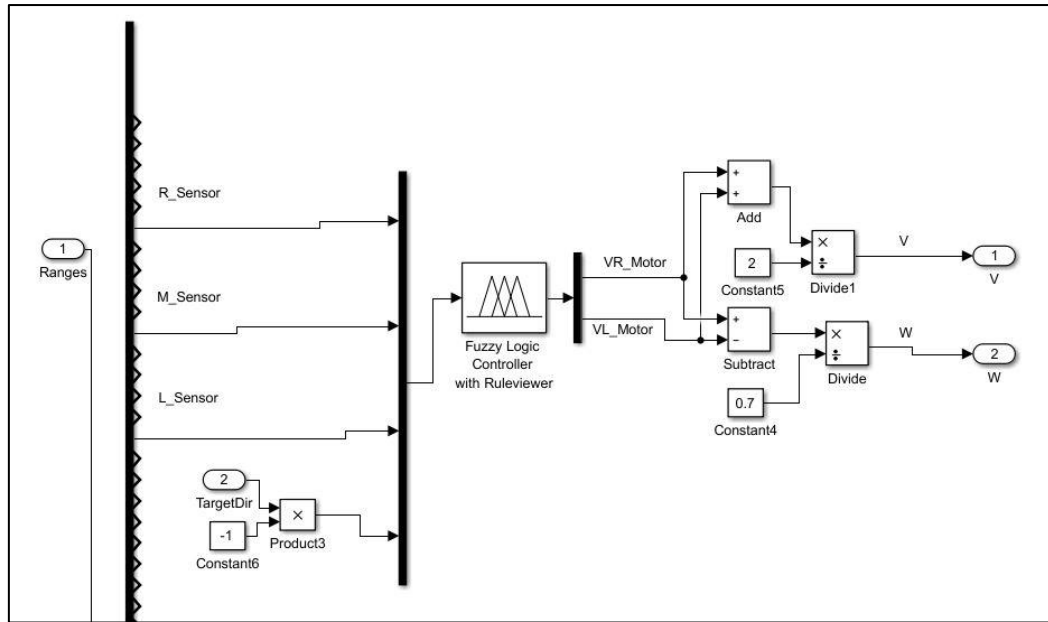


Figure 3.11: Controller subsystem

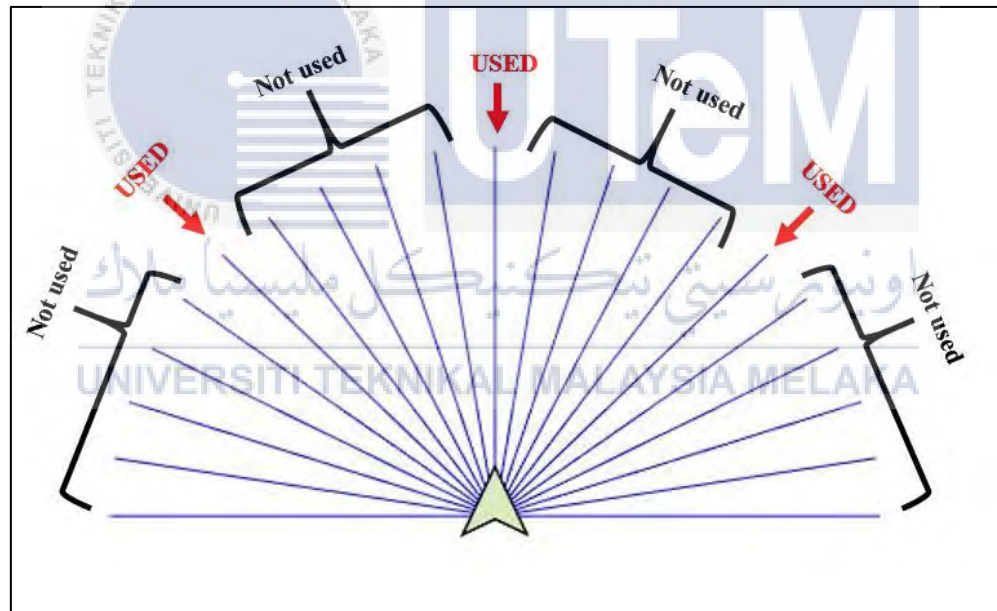


Figure 3.12: Robot simulator sensor diagram

The fourth subsystem shown in Figure 3.14 is the Outputs subsystem. In this subsystem, the linear and angular velocities are sent to the robot simulator using an ROS publisher block. The linear and angular velocities are multiplied by the logical variable *CloseToGoal* that was computed in the Path Extraction subsystem. If the robot has reached its goal, the multiplication output will be equal to zero and the robot will stop. The velocity commands are of the type *geometry_msgs/Twist* which is data type in the robot simulator. The

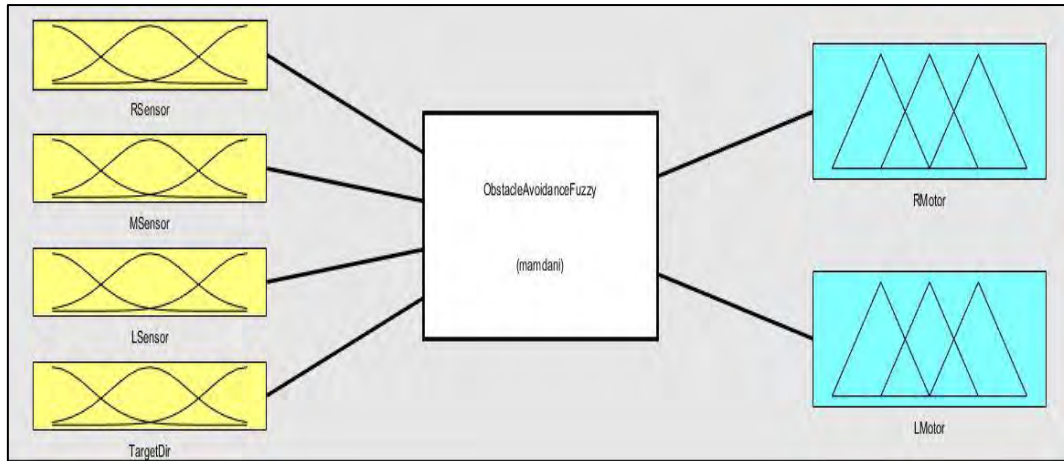


Figure 3.14: Fuzzy controller block diagram

3.5.2 The Fuzzification Procedure and Membership Function Design

The fuzzification procedure converts the crisp values to linguistic fuzzy terms. The membership functions used in this project are triangular, S-type and Z-type functions. Figure 3.16 shows the membership functions for the inputs and outputs of the fuzzy controller. The input variables “*RSensor*”, “*MSensor*” and “*LSensor*” have the same membership functions shown in Figure 3.16(a), these variables are expressed using two linguistic terms; “*NEAR*” and “*FAR*”. The “*TargetDir*” input variable is expressed using four linguistic terms “*big-left*”, “*left*”, “*middle*”, “*right*” and “*big-right*” to increase the accuracy as shown in Figure 3.16(b). The output variables “*RMotor*” and “*LMotor*” have the same membership functions and are expressed using three linguistic terms; “*slow*”, “*medium*” and “*fast*” as shown in Figure 3.16(c).

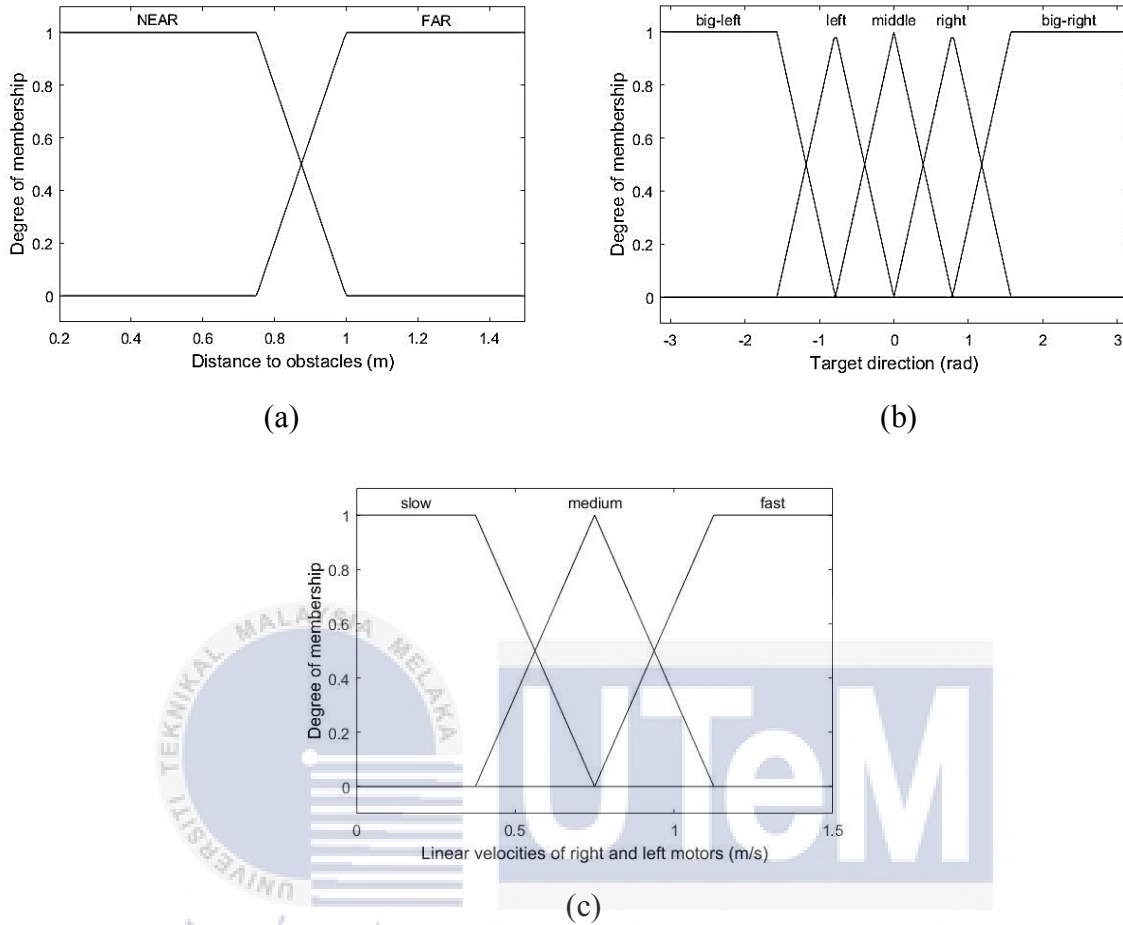


Figure 3.15: Membership functions for (a) the values for the right, left and center sensors, (b) the target direction, and (c) the linear velocities of the right and left motors

The fuzzification output for the proposed membership functions are as follows;

For triangular functions,

$$y_{ij} = \begin{cases} 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{if } m_{ij} - \frac{\sigma_{ij}}{2} < u_i < m_{ij} + \frac{\sigma_{ij}}{2}, \\ 0, & \text{otherwise;} \end{cases} \quad (3.3)$$

For an S-type function,

$$y_{ij} = \begin{cases} 0, & \text{if } u_i < m_{ij} - \frac{\sigma_{ij}}{2}, \\ 1, & \text{if } u_i > m_{ij}, \\ 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{otherwise;} \end{cases} \quad (3.4)$$

For Z-type functions,

$$y_{ij} = \begin{cases} 0, & \text{if } u_i > m_{ij} + \frac{\sigma_{ij}}{2}, \\ 1, & \text{if } u_i < m_{ij}, \\ 1 - \frac{2|u_i - m_{ij}|}{\sigma_{ij}}, & \text{otherwise;} \end{cases} \quad (3.5)$$

Where $i = 1, 2, \dots, 4$, is the number of the input signal; and $j = 1, 2, \dots, 4$, is the number of terms of the input variables; y , is the degree of membership for the i th input corresponding to the j th term of the input variable; u , is the i th input signal to the fuzzy controller, $\{u_1, u_2, u_3, u_4\} = \{RSensor, MSensor, LSensor, TargetDir\}$; m_{ij} is the center of the membership function corresponding to the i th input and the j th term of the input variable; σ_{ij} is the width of the membership function corresponding to the i th input and the j th term of the input variable. For example, $u_2 = MSensor$, represents the input value for the distance to the obstacle measured by the middle sensor; $m_{43} = 0$, means the value of the membership function center of the 3rd term (“middle”) of the 4th input variable (u_4 or $TargetDir$) is 0; and $\sigma_{43} = 1.571$ rad (= 90°) is the width of the membership function [12].

3.5.3 The Inference Mechanism and The Rule Base Design

The inference mechanism determines the output of the fuzzy logic controller using fuzzy rules. Eleven rules are designed for the proposed fuzzy controller as shown in Table 3.6. There are five main rules for the Go-to-Goal behavior and six main rules for the Obstacle Avoidance behavior. Every rule has a weight associated with it and is equal to 0.3 for the Go-to-Goal behavior and 1 for the Obstacle Avoidance behavior. The use of weights in the rule base is proposed as a method to combine the two behaviors. The rule weight can be interpreted as a measure of “influence” or “importance”. In this proposed design, the obstacle avoidance behavior is given priority to the Go-to-Goal behavior by assigning the Go-to-Goal behavior a smaller weight value.

Table 3.6: The rule base for the fuzzy controller;

N: Near, F: Far, R: Right, M: Middle, L: Left, BR: Big-Right, BL: Big-Left, S: slow, ME: Medium, FA: Fast, GG: Go-to-Goal, OA: Obstacle Avoidance

Rule No.	Inputs				Outputs		Weight	Behavior
	RSensor	MSensor	LSensor	TargetDir	RMotor	LMotor		
1	-	-	-	M	S	S	0.3	GG
2	-	-	-	L	ME	S		
3	-	-	-	R	S	ME		
4	-	-	-	BR	S	FA		
5	-	-	-	BL	FA	S		
6	N	F	F	-	FA	S	1	OA
7	F	F	N	-	S	FA		
8	F	N	F	-	S	FA		
9	F	N	N	-	S	FA		
10	N	N	F	-	FA	S		
11	N	F	N	-	S	S		

3.5.4 The Defuzzification Method

The defuzzification procedure converts the fuzzy output value to a crisp value. The defuzzification procedure is needed to actuate the mobile robot. There are many defuzzification methods and in this project, the “*Center of Gravity*” method is chosen for the proposed fuzzy controller. The “*Center of Gravity*” method calculates the crisp output using a similar formula to the one used for calculating the center of gravity in physics. The center of gravity of the area bounded by the membership function curve is the crispest value of the fuzzy output. The values of the output variables “*RMotor*” and “*LMotor*” are given by:

$$RMotor = \frac{\sum_{k=1}^{11} \mu_{k,1} q_k}{\sum_{k=1}^{11} q_k} \quad \text{and} \quad LMotor = \frac{\sum_{k=1}^{11} \mu_{k,2} q_k}{\sum_{k=1}^{11} q_k} \quad (3.6)$$

Where $\mu_{k,1}$ and $\mu_{k,2}$ are the outputs from the k th rule, which are related to the center of membership functions of the output variables.



CHAPTER 4

RESULTS

The first test for the proposed fuzzy logic controller is on a map with no obstacles that was constructed in Figure 3.5. The final goal or target for the mobile robot for this experiment are shown with a red dot. Figure 4.1 shows the simulation result for the first test.

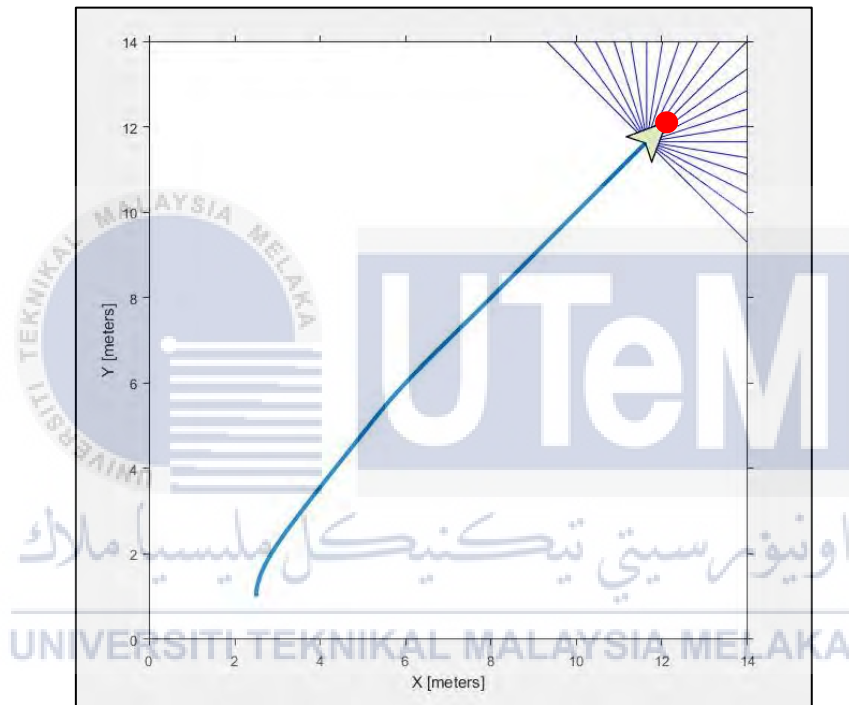


Figure 4.1: First simulation test; no obstacles

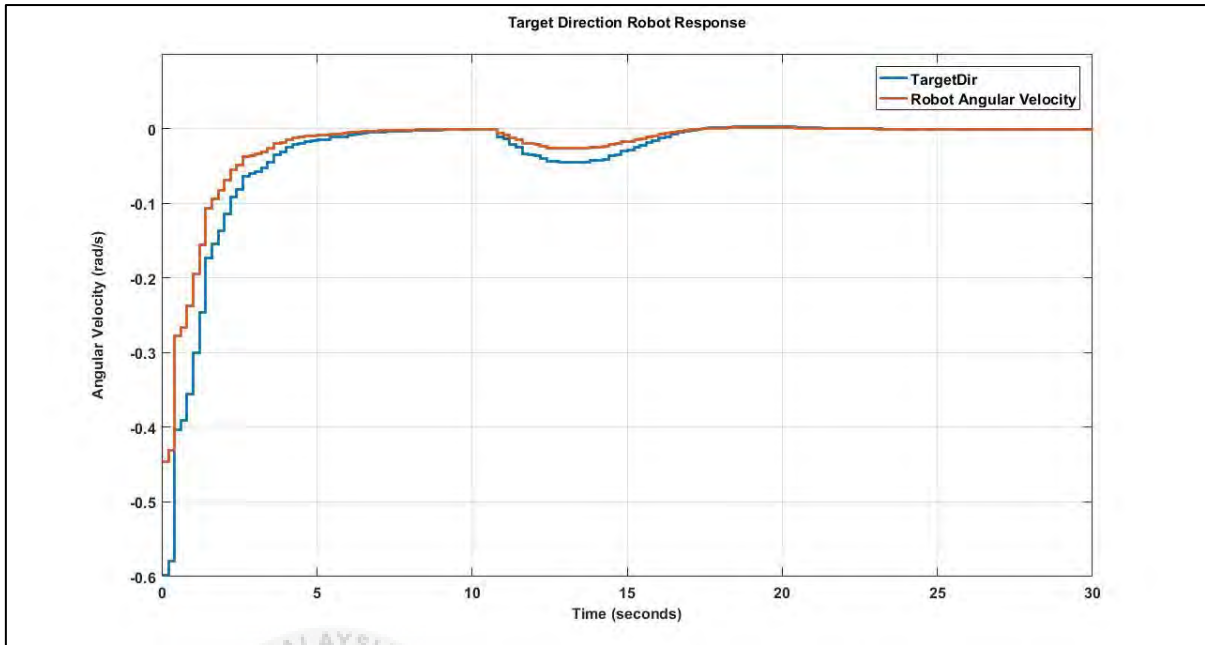


Figure 4.2: Target direction robot response for first simulation test; no obstacles

As can be seen in Figure 4.2, when there are no obstacles in the robot's path the robot follows the target direction until it reaches its goal. Other examples for the testing of the fuzzy controller without obstacles are shown in Figure 4.3. As can be seen from the graphs the proposed fuzzy logic controller has a good response to the target direction commands and manages to reach its final goal.

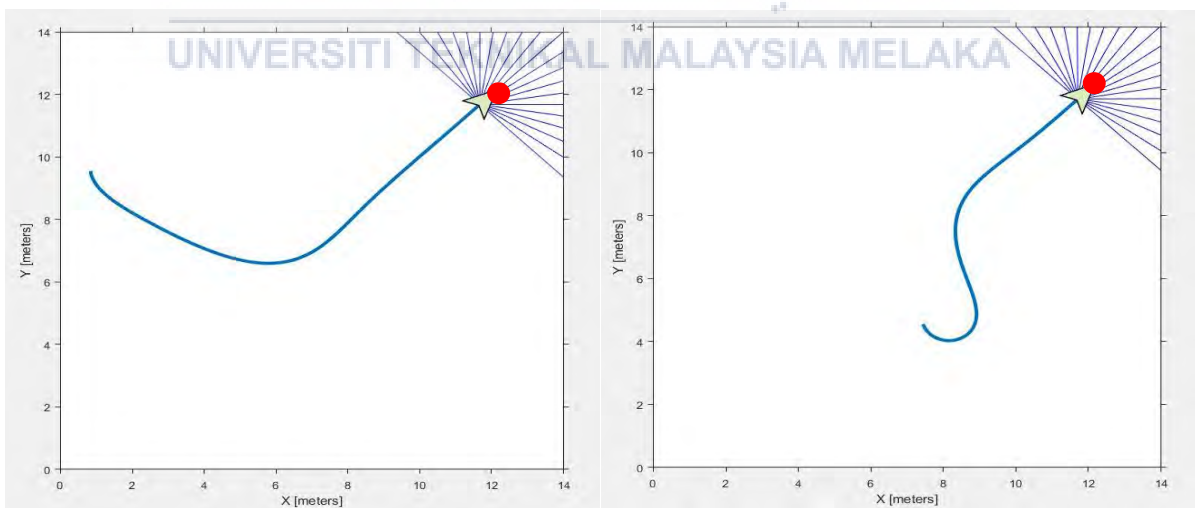


Figure 4.3: Further examples of simulation testing with no obstacles

Since the performance of the proposed fuzzy logic controller has proved to give the desired result of reaching the final goal when there are no obstacles, the next stage of testing is

to run the simulation on the map with obstacles that was constructed as in Figure 3.6. There are three obstacles along the path of the robot and it must avoid them to reach the goal. Figure 4.4 shows the performance of the controller in that environment. It can be seen that the robot reaches its destination successfully.

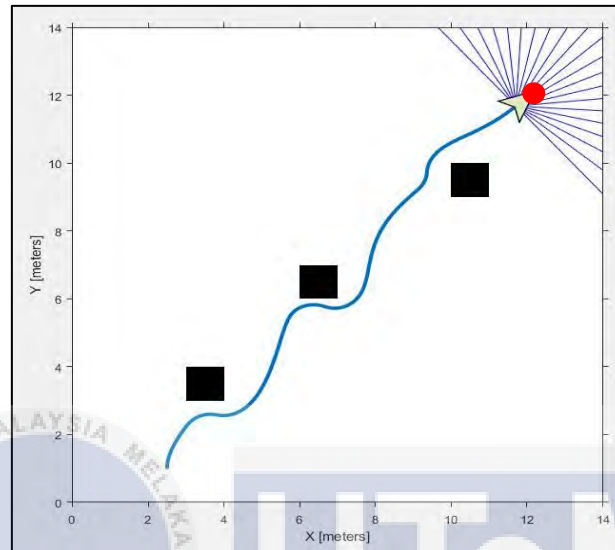


Figure 4.4: Simulation result with obstacles

Figure 4.5 shows the target direction vs the actual robot movement. The positive values represent a left turn while the negative values represent a right turn. Figure 4.6 shows the obstacle distance read by the three sensors. Looking at the two graphs together, when the robot approaches the first obstacle (at time 2s to 4s) the target direction commands the robot to turn left but the middle and left sensors detect an obstacle to the left and front sides of the robot so the controller directs the robot to turn right to avoid the obstacle. Similarly, the fuzzy controller manages to steer the robot away from the other two obstacles and reach its goal at the end.

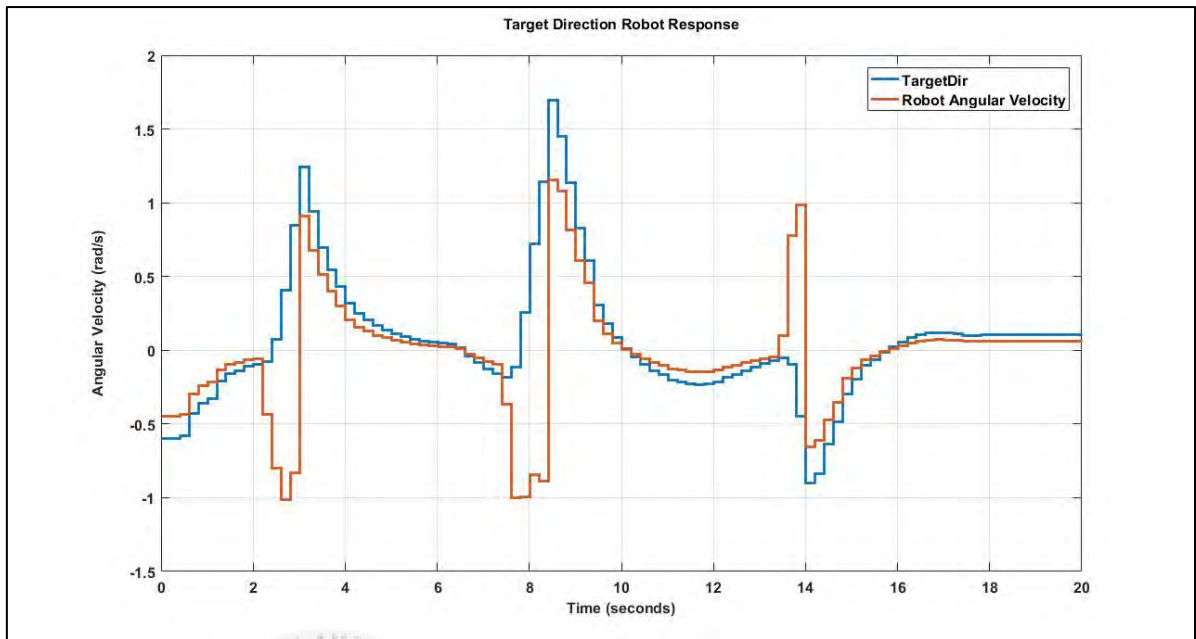


Figure 4.5: Target direction robot response for simulation test; with obstacles

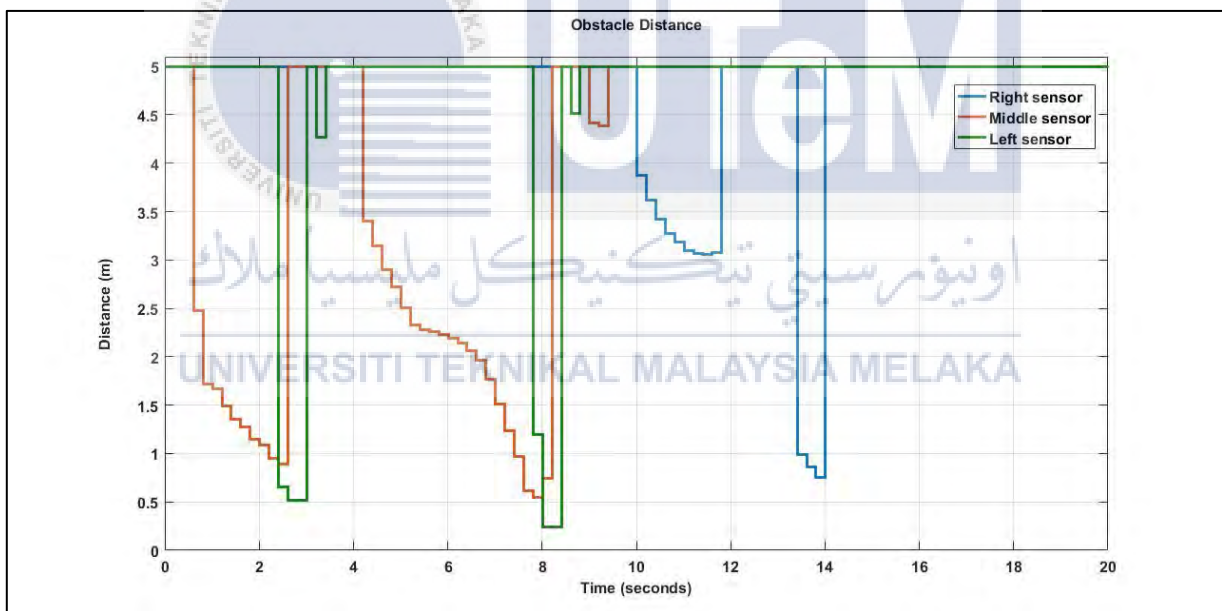


Figure 4.6: Obstacle distance read by the sensors

To further proof the effectiveness of the proposed fuzzy logic controller a few more testing is conducted with different starting position and final goal. Figure 4.7 shows the results of this testing.

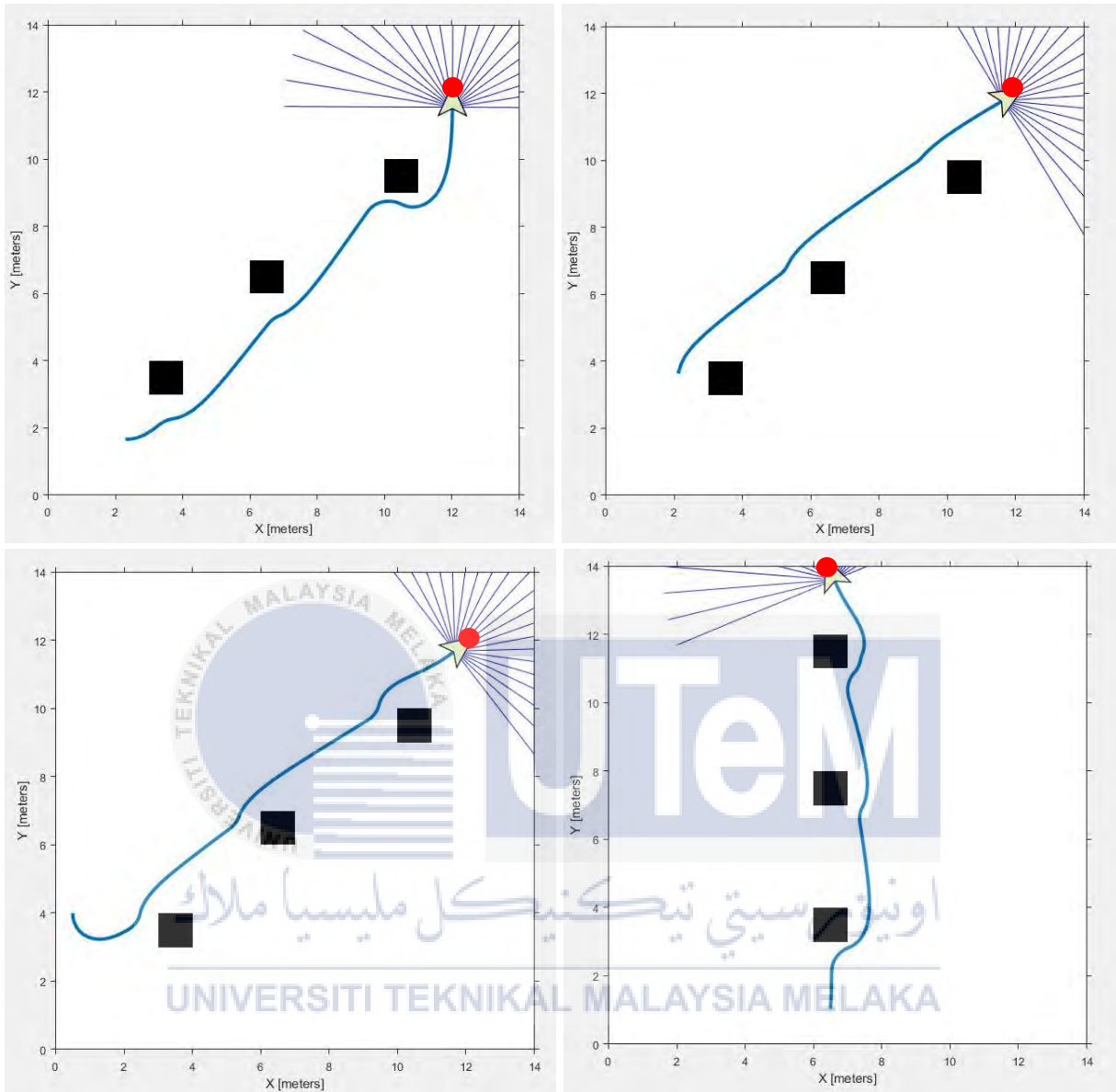


Figure 4.7: Further examples of simulation results; with obstacles

From the previously conducted simulations, it can be said that the proposed fuzzy logic controller can manage to avoid obstacles in different circumstances. To test the proposed controller even further, the map is changed to the third map as shown in Figure 3.5(c). The result of the simulation is shown in Figure 4.8.

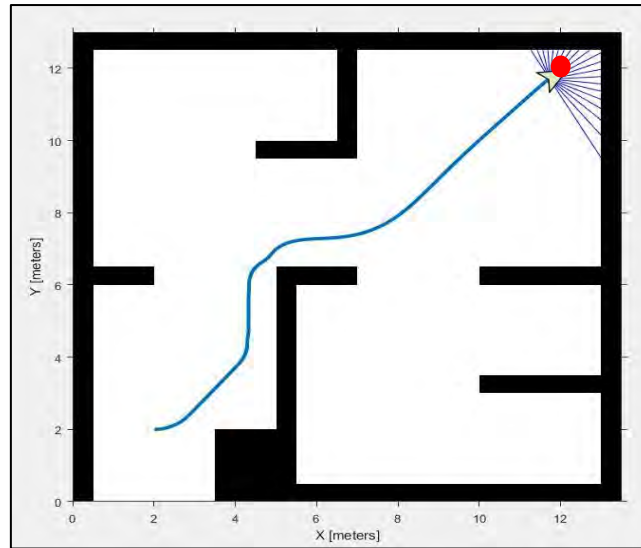


Figure 4.8: Simulation result for the third map

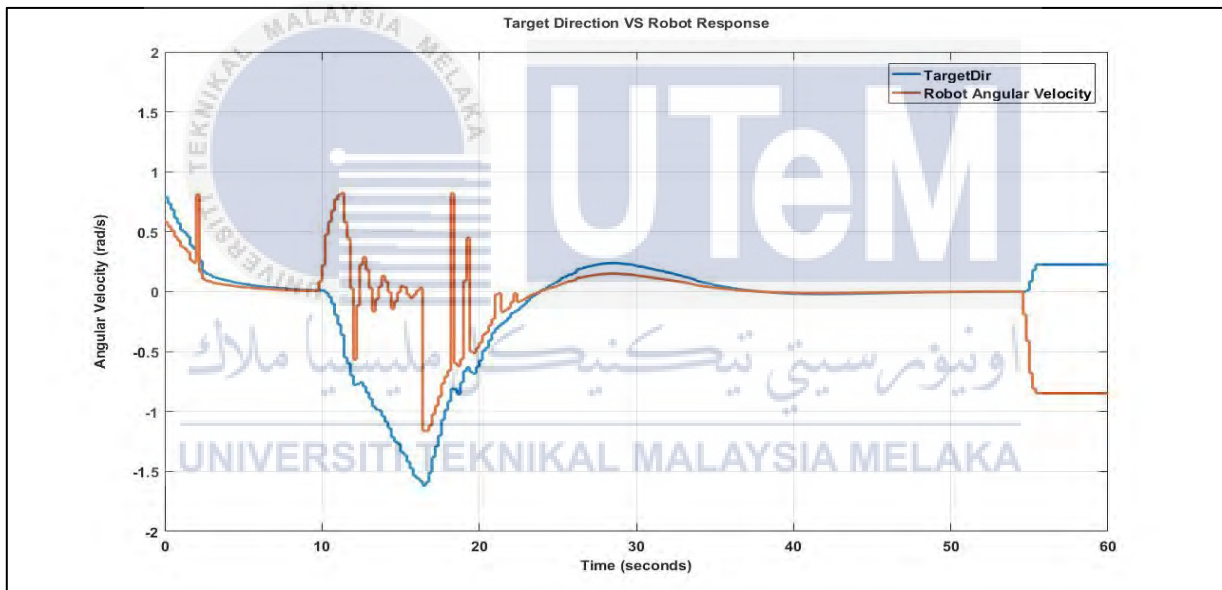


Figure 4.9: Target direction vs the robot angular velocity; third map

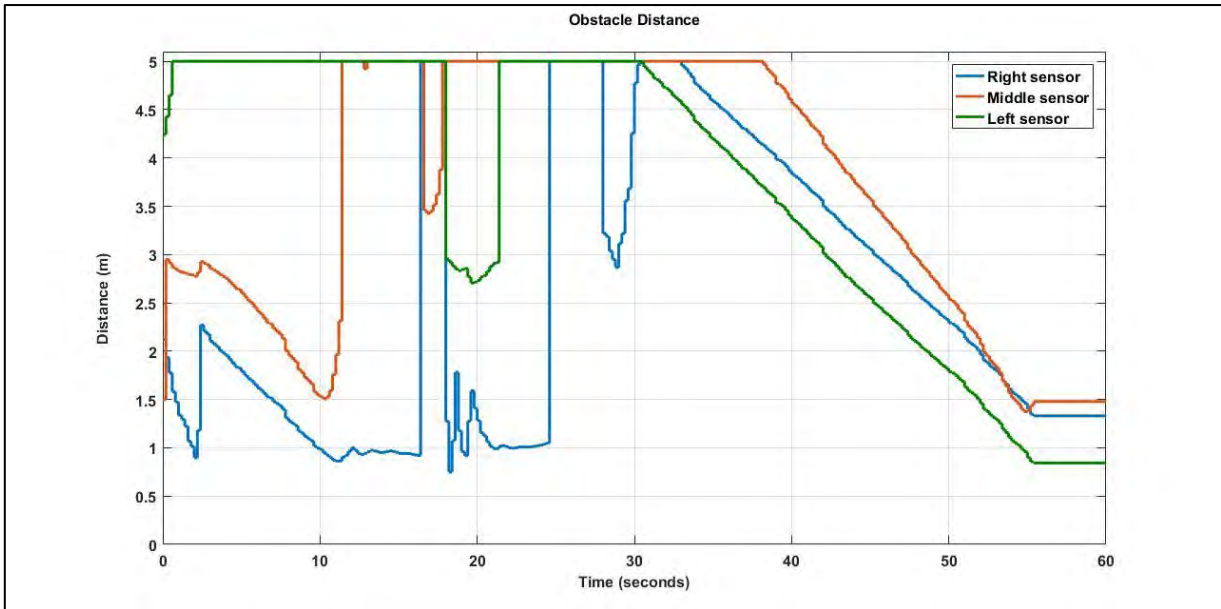


Figure 4.10: Obstacle distance read by the sensors; third map

As can be seen from Figures 4.9 and 4.10, the robot manages to steer away from the obstacles and reach its target. A few more examples are shown in Figure 4.11.

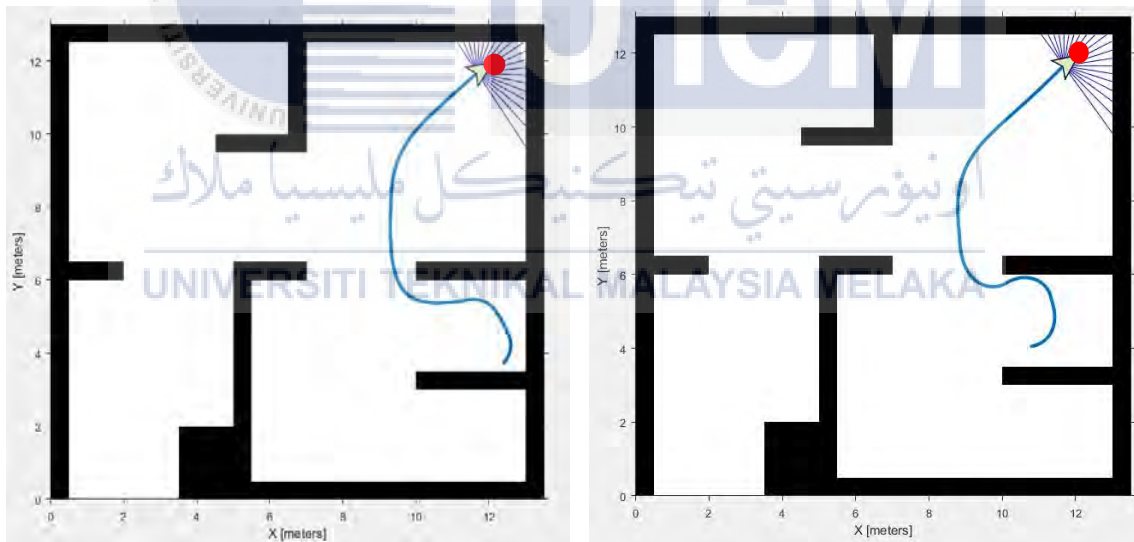


Figure 4.11: Further examples of simulation results; third map

From all the previous results, the robot has managed to reach its goal and avoid obstacles and it can be said that the proposed fuzzy logic controller has achieved the desired objectives. However, there have been some cases where the robot couldn't avoid collisions. Figure 4.12 shows such a case. There are many reasons as to why the robot collided with the obstacle, including:

- i. Fuzzy rules don't include all the possible cases such as: should the robot turn right or left when there is an obstacle in the middle? In this project, the rule is to turn right.
- ii. Low no. of sensors: the obstacle is not detected until it is too late.

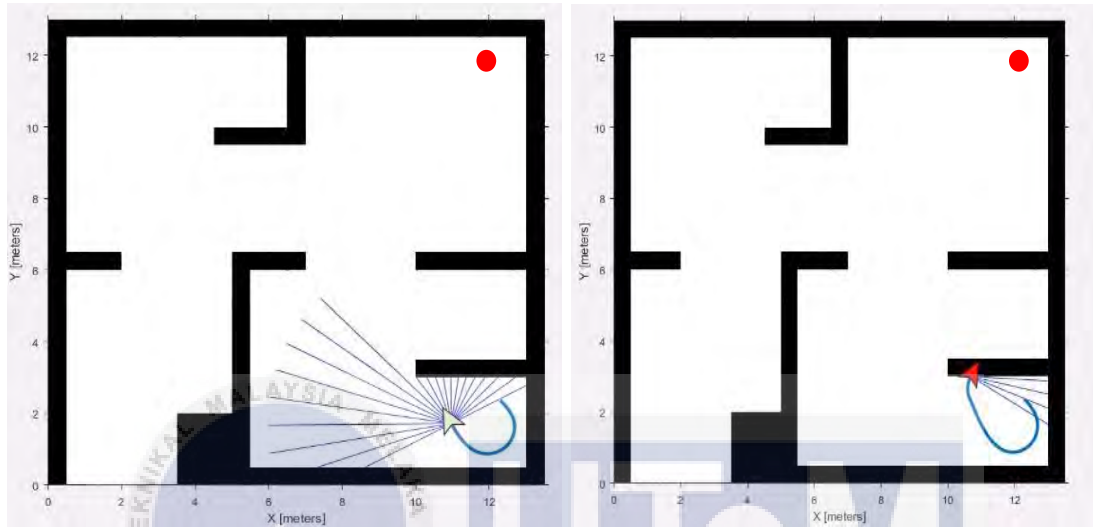


Figure 4.12: Simulation result for the third map (failed to avoid obstacles)

This problem can be solved by increasing the number of sensors for each side and grouping the sensors using sensor fusion. To prove the method of sensor fusion, the Simulink model is edited to include all the 21 sensors of the mobile robot simulator. Figure 4.13 shows the modified Simulink model while Figure 4.14 the robot's sensors diagram. The fusion method works by adding the sensors together then dividing by the number of sensors, then the output, which is the average of all the sensor is sent to the controller.

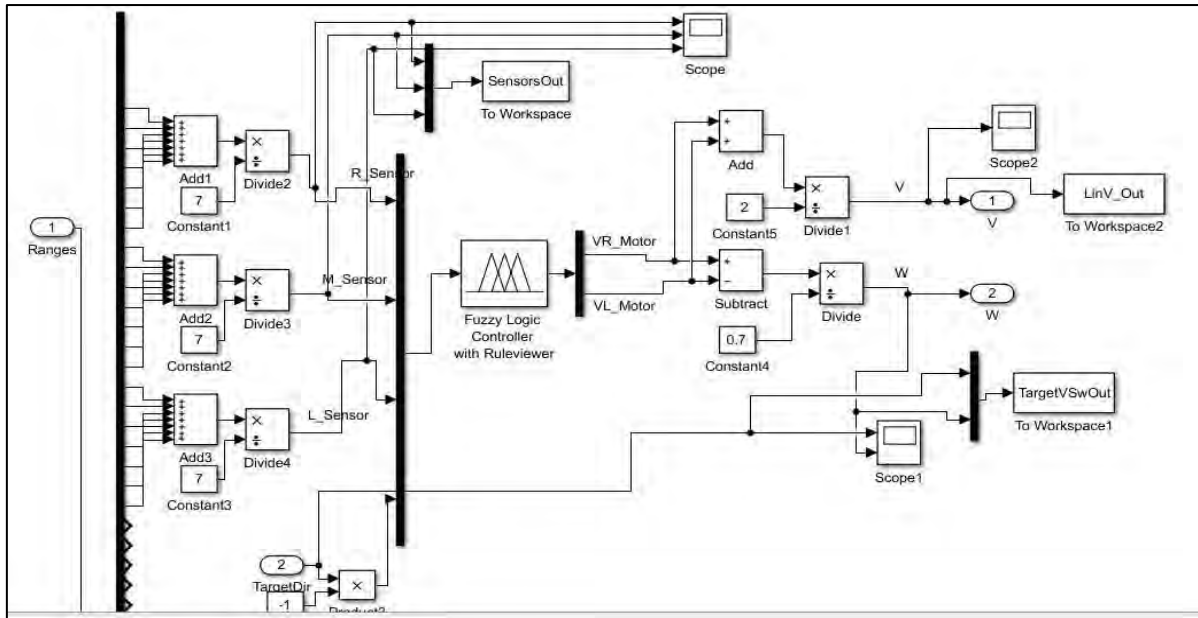


Figure 4.13: Controller subsystem; modified for sensor fusion

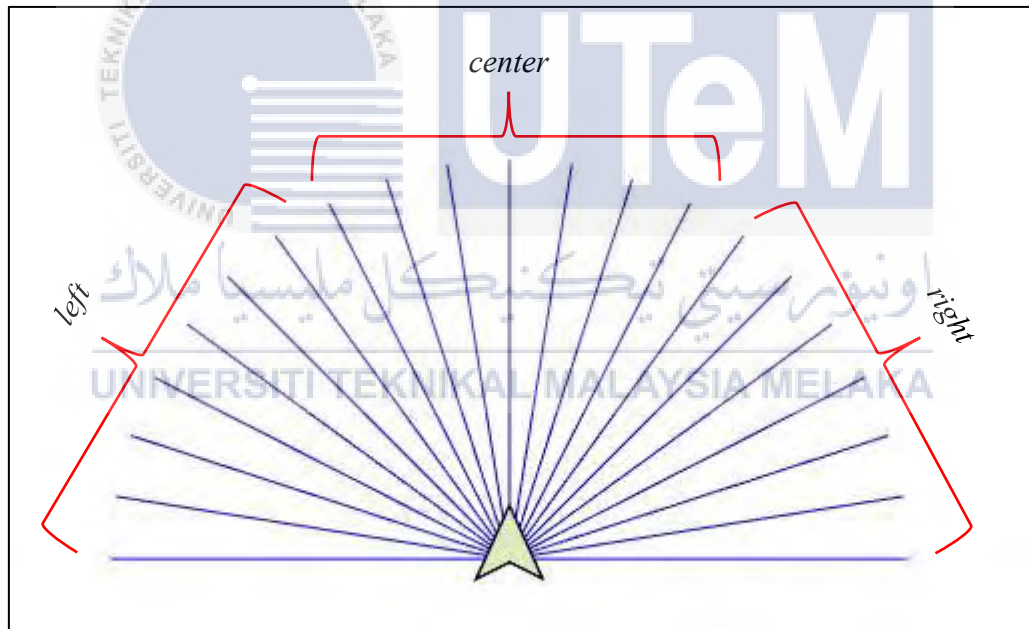


Figure 4.14: Robot sensor diagram; sensor fusion

After modifying the Simulink model to utilize sensor fusion, the same case in Figure 4.12 has been repeated. As shown in Figure 4.15, the new method managed to avoid the obstacle and reach its target.

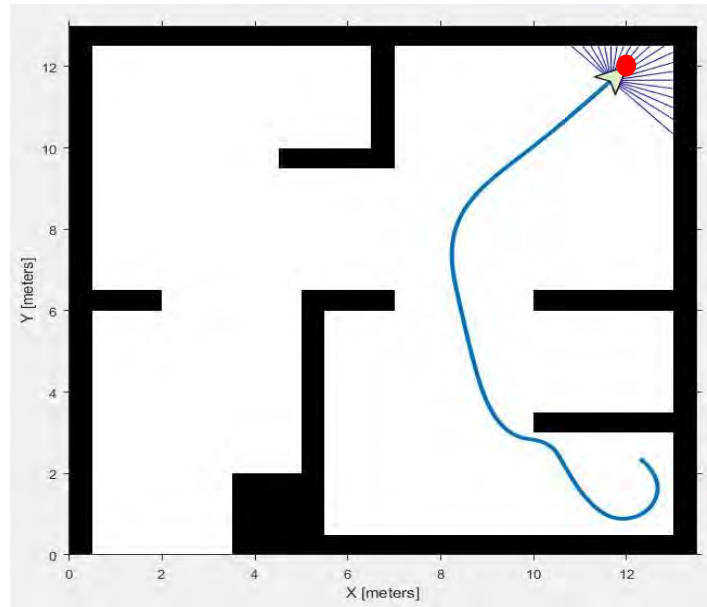


Figure 4.15: Simulation result for the third map; with sensor fusion

From the previous results, it can be concluded that the proposed controller managed to reach its target while also avoiding obstacles in the maps with no or low number of obstacles. The robot did crash in some cases as was shown in Figure 4.12. However, the problem was solved using seven sensors for each side and combining the sensor readings using sensor fusion. This has improved the performance of the controller as the sensors now cover a wider angle. The behaviors in this proposed approach are combined using rule weights. The obstacle avoidance behavior is assigned a larger value to emphasize its priority over the Go-to-Goal behavior. This is seen in the results, where the goal direction is ignored to avoid the obstacle. The values for the rule weights were developed using trial and error method through numerous experiments. There is still more room for improvement and the full effect of the rule weights needs more studying.

CHAPTER 5

CONCLUSION

In this project, an obstacle avoidance algorithm for a robotic wheelchair was developed using fuzzy behavior based architecture. The literature review provided a clear insight on how to proceed with the project implementation. The decision to use behavior-based architecture is based on the fact that it does not need an internal representation of the robot and that it doesn't need a complete world model. This leads to the increased speed of the response. Fuzzy logic is very useful in mobile robotics application and is used extensively in this domain. The power of fuzzy logic lies in its power to deal with the uncertainties and imprecision of the sensors used in mobile robotics.

After setting up the software simulation environment, the fuzzy logic controller was designed to take four inputs and two outputs. The inputs to the controller are the sensor readings and the target direction while the outputs are the right and left motors' linear velocities. The fuzzy logic controller is designed to utilize two main behaviors; Go-to-Goal and obstacle avoidance. A total of 12 rules were designed to implement the two behaviors. The proposed fuzzy logic controller combines the behaviors using rule weights. By assigning a larger rule weight to the obstacle avoidance behavior, the priority of the robot is to avoid obstacles than to go to the goal.

The proposed fuzzy logic controller was tested on three maps. For the first and second maps, the robot managed to reach its target and avoid the obstacles on its way. However, on the third map, the robot did collide with obstacles in some of the cases. The reasons it crashed can be attributed to the incompleteness of the fuzzy rules or the small number of sensor that are being used. The problem was solved by using more sensors for each side and combining them using sensor fusion.

Based on the obtained results, it can be said that the proposed fuzzy logic controller is quite effective in avoiding obstacles in low congested environments such as the one in the second map used in this project. However, implementing this controller on a real wheelchair is not advisable at its current form as it has crashed on the third map. Although the problem was

solved by using sensor fusion, more research is required to test the controller's reliability and robustness in real life situations where it involves complex obstacles including static and dynamic obstacles.

For future research, it is recommended to increase the number of fuzzy rules as well as increase the number of sensors. Also, another input could be added such as a bumper sensor to stop the robotic wheelchair in the case of a collision.

In the end, based on the objectives of this project which are to develop an obstacle avoidance algorithm with fuzzy behavior-based controller for the modeled wheelchair using MATLAB/Simulink simulation and to simulate and analyze the developed algorithm in MATLAB/Simulink environment, it can be said that the objectives have been fulfilled.



REFERENCES

- [1] N. A. Jacobs and S. Sheldon, "Report of a consensus conference on wheelchairs for developing countries," Bengaluru, India, 2006.
- [2] R. C. Simpson, E. F. Lopresti, and R. A. Cooper, "How many people would benefit from a smart wheelchair?," *J. Rehabil. Res. Dev.*, vol. 45, no. 1, pp. 53–71, 2008.
- [3] R. Solea, A. Filipescu, A. F. Jr, and E. Minca, "Wheelchair control and navigation based on kinematic model and iris movement," in *2015 IEEE 7th International Conference on CIS & RAM*, 2015, pp. 78–83.
- [4] R. Dhaouadi and A. A. Hatab, "Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies : a unified framework," *Adv. Robot. Autom.*, vol. 2, no. 2, p. 107, 2013.
- [5] S. M. Lavalle, "Chapter 13 differential models," in *Planning Algorithms*, Cambridge University Press, 2006, pp. 716–786.
- [6] R. Arkin, "Behavior-based robotics," MIP Press, 1998.
- [7] L. De Silva and H. Ekanayake, "Behavior-based robotics and the reactive paradigm: a survey," *Proc. 11th Int. Conf. Comput. Inf. Technol. ICCIT 2008*, pp. 36–43, 2008.
- [8] C. C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller. I," *IEEE Trans. Syst. Man. Cybern.*, vol. 20, no. 2, pp. 404–418, 1990.
- [9] A. Katbab, "Fuzzy logic and controller design-a review," in *Southeastcon '95. Visualize the Future., Proceedings., IEEE*, 1995, pp. 443–449.
- [10] K. Sinthipsomboon, I. Hunsacharonroj, J. Khedari, W. Pongaen, and P. Pratumswan, "A hybrid of fuzzy and fuzzy self-tuning pid controller for servo electro-hydraulic system," in *6th IEEE Conference on Industrial Electronics and Applications*, 2011, pp. 220–225.
- [11] H. Murakami and H. Seki, "Fuzzy algorithm based obstacle avoidance control of electric powered wheelchair using ultrasonic sensor," *IECON Proc. (Industrial Electron. Conf.)*, pp. 4251–4256, 2009.
- [12] A. Zhu and S. X. Yang, "A fuzzy logic approach to reactive navigation of behavior-based mobile robots," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 5045–5050, 2004.
- [13] H. Li and S. X. Yang, "A behavior-based mobile robot with a visual landmark-

recognition system,” IEEE/ASME Trans. Mechatronics, vol. 8, no. 3, pp. 390–400, 2003.

- [14] SHARP Corporation, “Distance measuring sensor unit,” Production. pp. 1–9.
- [15] MathWorks Inc., “Robotics System Toolbox™ User’s Guide R2016b,” 2016. [Online]. Available: https://www.mathworks.com/help/pdf_doc/robotics/robotics_ug.pdf.
- [16] MathWorks Inc., “Path following with obstacle avoidance in Simulink.” [Online]. Available: <https://www.mathworks.com/help/robotics/examples/path-following-with-obstacle-avoidance-in-simulink.html>.

