COLLISION AVOIDANCE MECHANISMS FOR UAV

TAN JIE SIM

BACHELOR OF MECHATRONICS ENGINEERING WITH HONOURS
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

COLLISION AVOIDANCE MECHANISMS FOR UAV

TAN JIE SIM

A report submitted in partial fulfilment of the requirements for the degree of Bachelor of Mechatronics Engineering with Honours

Faculty of Electrical Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I declare that this thesis entitled "COLLISION AVOIDANCE MECHANISMS FOR UAV is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in the candidature of any other degree.

Signature	:	Sim
Name	:	TAN JIE SIM
Date	:	17/06/2024

APPROVAL

I hereby declare that I have checked this report entitled "Collision Avoidance Mechanisms for UAV", and in my opinion, this thesis fulfils the partial requirement to be awarded the degree of Bachelor of Mechatronics Engineering with Honours.

Signature	:	[illegible] FADILAH BINTI ABDUL AZIS PENSYARAH KANAN Fakulti Teknologi dan Kejuruteraan Elektrik Universiti Teknikal Malaysia Melaka
Supervisor Name	:	MADAM FADILAH BINTI ABDUL AZIS
Date	:	24 JUNE 2024

DEDICATIONS

To my beloved family and fellow friends.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest gratitude to the contribution of people in this report. There is no denying the fact that researching and writing the report is quite challenging. I am incredibly grateful that I can write this report with the support and guidance of my supervisors, family and fellow friends.

First and foremost, I would like to express my appreciation to my supervisor, Madam Fadilah Binti Abdul Azis. She provides a lot of guidance and mental support throughout the whole project. Her expertise and knowledge assist me in enhancing my project. I feel grateful for her willingness to offer help for my project.

In addition, I would like to convey my appreciation to my family for their physical and mental support throughout my whole project. Their encouragement acts as my guiding light, providing me with the strength to persevere through challenges, especially when facing problems in my project. I am deeply grateful for their presence in my life.

Last but not least, I would like to give a special thank you to my supportive fellow friends who are involved in my project. Without their assistance and motivation, this project will not be presented here.

In conclusion, I am deeply grateful to those who have contributed to the development of this project. Their contribution has led to the outcome of this project.

ABSTRACT

Unmanned Aerial Vehicles (UAV) is an aircraft which can fly autonomously without the presence of pilot. There has been significant development in UAV over the past few years. Nowadays, UAV is widely used in various applications such as delivery, monitoring and filming. There is a high probability that UAV may collide with objects because it always flies to an unpredictable environment. This shows the importance of a collision avoidance system with the correct algorithm for UAV when carrying out its mission. In real world applications, there are many inherent uncertainties during the flight of UAV. The aim of this project is to investigate the collision avoidance algorithms that are suitable for UAV. Analyzing the performance of APF algorithm under 3 different conditions is one of the objectives in this project. Collision avoidance algorithm coupled with distance sensors is a recommended safety improvement for UAVs in real world applications. To carrying out this project, a comparison between the algorithms is computed to choose a suitable algorithm for this project. Then, the selected algorithm is modelled on MATLAB. The performance of the algorithm in virtual environment with different numbers of obstacles, extreme goal position and wind disturbances are analyzed by calculating RMSE value. Some experiments are carried out to make a comparison between the ultrasonic sensor and infrared sensor. By the end of this project, APF algorithm is selected and modelled on MATLAB. In virtual simulations, APF algorithm effectively plans paths for UAV to reach goal position without colliding with obstacles. In environment with wind disturbances, the RMSE value increased by 36.5% if compared to environment without wind disturbances.

ABSTRAK

Kenderaan Udara Tanpa Pemandu (UAV) ialah pesawat yang boleh terbang secara autonomi tanpa kehadiran juruterbang. Terdapat perkembangan ketara dalam UAV sejak beberapa tahun kebelakangan ini. Kini, UAV digunakan secara meluas dalam pelbagai aplikasi seperti penghantaran, pemantauan dan penggambaran. Terdapat kebarangkalian tinggi bahawa UAV mungkin berlanggar dengan objek kerana ia sentiasa terbang ke persekitaran yang tidak dapat diramalkan. Ini menunjukkan kepentingan sistem pengelakan perlanggaran dengan algoritma yang betul untuk UAV semasa menjalankan misinya. Dalam aplikasi dunia sebenar, terdapat banyak ketidakpastian yang wujud semasa penerbangan UAV. Matlamat projek ini adalah untuk menyiasat algoritma pengelakan perlanggaran yang sesuai untuk UAV. Penganalisisan prestasi algoritma APF di bawah 3 keadaan berbeza adalah salah satu objektif dalam projek ini. Algoritma pengelakan perlanggaran ditambah dengan penderia jarak adalah penambahbaikan keselamatan yang disyorkan untuk UAV. Untuk melaksanakan projek ini, perbandingan antara algoritma dikira untuk memilih algoritma yang sesuai untuk projek ini. Kemudian, algoritma yang dipilih dimodelkan pada MATLAB. Prestasi algoritma dalam persekitaran maya dengan bilangan halangan yang berbeza, kedudukan matlamat yang melampau dan gangguan angin dianalisis dengan mengira nilai RMSE. Beberapa eksperimen dijalankan untuk membuat perbandingan antara sensor ultrasonik dan sensor inframerah. Menjelang akhir projek ini, algoritma APF dipilih dan dimodelkan pada MATLAB. Dalam simulasi maya, algoritma APF secara berkesan merancang laluan untuk UAV mencapai kedudukan matlamat tanpa bertembung dengan halangan. Dalam persekitaran dengan gangguan angin, nilai RMSE meningkat sebanyak 36.5% jika dibandingkan dengan persekitaran tanpa gangguan angin.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

U_{art}	-	Artificial potential energy
U_{goal}	-	Attractive potential energy
U_{obs}	-	Repulsive potential energy
x	-	Spatial position of UAV
x_d	-	Spatial position of goal
k_p	-	Attractive force gain coefficient
η	-	Repulsive force gain coefficient
ρ	-	Shortest distance to the obstacle O
ρ_0		Limit distance of the potential field influence
F_{goal}	-	Attractive force
F_{obs}	-	Repulsive force
$\frac{\partial \rho}{\partial x}$	-	Partial derivatives of variable shortest distance to the obstacle O in the spatial position of UAV
C_{ij}	-	Grid confidence matrix
β	-	Corresponding angle of grid
V	-	Velocity of ultrasonic waves
S	-	Distance of ultrasonic waves
$U(q)$	-	Artificial potential field exerted on UAV
$U_{att}(q)$	-	Attractive field exerted by goal
$U_{rep}(q)$	-	Repulsive field exerted by obstacle
$F(q)$	-	Resultant artificial force which moves the UAV
$F_{att}(q)$	-	Attractive force which generated by goal
$F_{rep}(q)$	-	Repulsive force which generated by obstacle
k_a	-	Positive coefficient of gravity for APF
q	-	Current position vector of UAV
q_d	-	Desired goal position vector
ρ_{goal}	-	Euclidean distance from UAV's current position to goal position
k_b	-	Repulsion gain coefficient
d	-	Distance between UAV and obstacle
d_0	-	Distance of obstacle repulsive force field
x_i	-	UAV's position in x-direction
x_{gi}	-	Goal position in x-direction
y_i	-	UAV's position in y-direction
y_{gi}	-	Goal position in y-direction
D	-	Maximum detection range
Y_n	-	Actual distance
X_n	-	Sensor reading

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

An Unmanned Aerial Vehicles (UAV) also referred to as drones. UAV is a type of aircraft that can be fly autonomously or controlled by remote without the presence of pilot [1]. A communication link, a ground control station, and unmanned aerial vehicles make up the Unmanned Aerial System. UAV can provide cloud-free and high-resolution images. Hence, it is widely used in various applications such as delivering, monitoring and filming.

Unmanned Aerial Vehicle (UAV) has undergone significant advancements over time. In the First World War, UAV was used as a flying bomb. Due to the limitations in technology, it cannot focus on the target accurately. In World War II, Pilotless Target Aircraft (PTA) was developed which can return after completing the mission. In the Cold War, a new UAV, "Lightning Bug" was developed. The advancements in the electronics of this UAV allow us to carry out real-time data transmission. In the 1990s, the development of UAV make it can gather information such as location of enemy [2]. Nowadays, UAV are equipped with cameras by merging the radio-controlled aircraft and smartphone technology. Hence, it is suitable to use for inspection and photography.

To enable an UAV to fly autonomously, some important features become main consideration of researchers when designing the UAV. Among these features, the computer algorithm is particularly important, as it enhances data processing and contributes to the overall performance of the UAV system. To stabilize the UAV flight, the algorithm can adjust the attitude and position of UAV according to the condition by tracking the UAV's flight data [3]. To ensure the safety of user, integrating sensors into the UAV is another important element. Since UAV always fly to an unpredictable condition, various types of sensors such as ultrasonic sensors, cameras and radar laser sensors are important for UAV to detect the obstacles surrounding it [4].

1.2 Motivation

Recently, UAV has rapidly gained widespread popularity around the globe. The worldwide UAV market is expected to expand significantly over the next several years. When comparing current UAV technology to earlier models, there have been numerous advancements. Consequently, it raises the UAV market around the world. The development of UAV technology also spurs my curiosity about looking into a UAV-related idea.

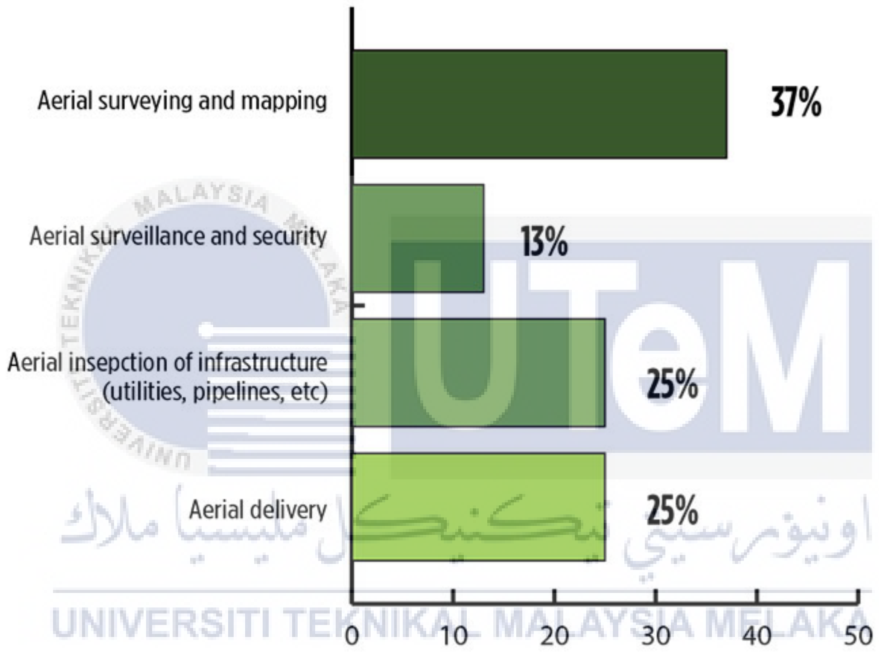


Figure 1.1 Application of UAV in 2019 [5]

Previously, UAV were mainly used for military purposes. Nowadays, advancement in UAV technology such as the integration of cameras on UAV are significantly broadening its scope of applications across various fields. For example, UAV can be used for gaming, mapping, cartograph, inspection, traffic monitoring, search and rescue purpose [6]. Figure 1.1 shows the statistics of the application of UAV in 2019.

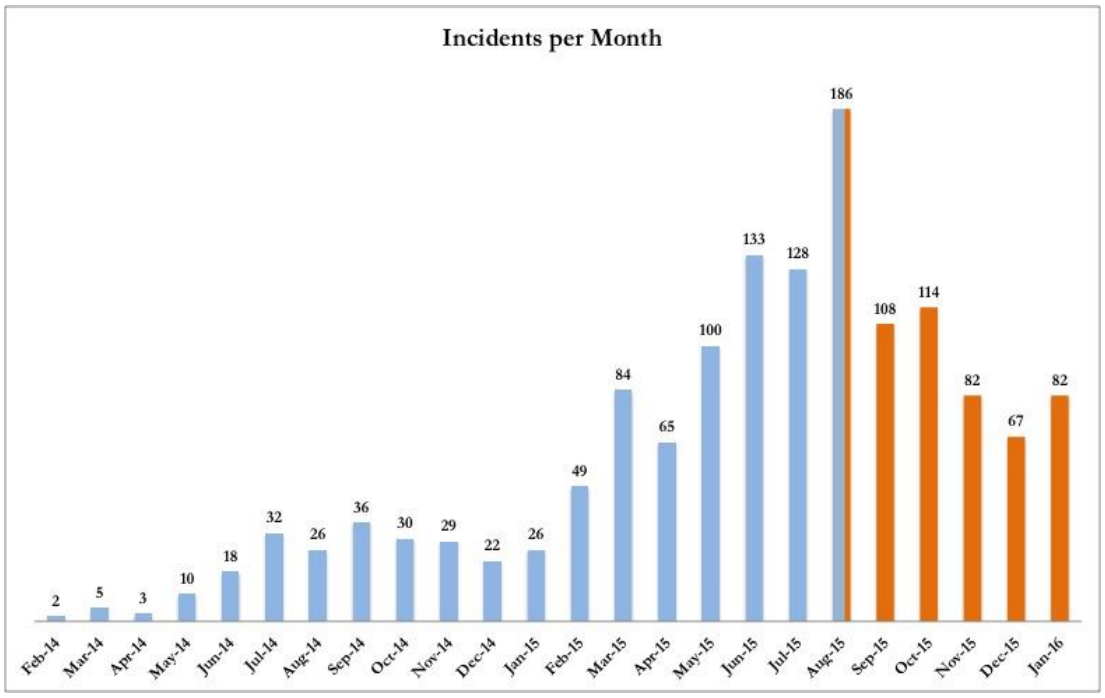


Figure 1.2 UAV incidents from 2014 to 2016 [7]

Although there are a lot of advancements in UAV technology, the accidents of UAV still cannot be avoided. Figure 1.2 illustrates the rising probability of UAV collision accidents between 2014 and 2016. My interest in finding out about the cause of the UAV's high collision risk has been awakened by the analysis. There are two main issues, which are hardware and software issues which may cause UAV accidents. The malfunctions of sensors like faulty GPS or ultrasonic sensors can disrupt flight stability and lead to disorientation or unexpected maneuvers. For software, UAV accidents happen when there are errors in the flight control software of UAV. The errors may affect the flight path and stability of UAV. Therefore, it can lead to erratic movements or crashes of UAV.

1.3 Problem Statement

The global demand for UAV is on the rise. With the increasing utilization of UAV across various sectors, certain limitations of UAV arise. UAV often navigates through unpredictable conditions, there is a risk that it may collide with objects. In [8], CNN and Aviation Safety Network stated that the number of deaths caused by commercial flight accidents is around few hundreds per year. Recently, researchers have been involved in the development of UAV. However, the probability of UAV accidents is still high.

To reduce the probability of UAV's accidents, obstacle avoidance system plays a significant role. This is because obstacle avoidance system consists of algorithms and sensors to identify the presence and location of obstacles. Integrating specified algorithms on the UAV can aid the UAV to determine the best route to reach the goal without colliding with obstacles. Autonomous flights always rely on algorithms to perform tasks such as object tracking and path planning. If there are errors in these algorithms, it may cause the UAV to deviate from the planned route or fail to react to various conditions appropriately. Finally, a collision occurs.

In addition, sensors are another important feature for UAV. The safety of UAV is highly relying on sensors. Sensors measure the surrounding condition of UAV. For example, a distance sensor detects the distance between the obstacles and UAV. After the sensor detects the obstacles, the data is sent to the flight controllers. After that, the controllers will calculate the distance between the UAV and the obstacles. Besides, the speed of the UAV decreases according to the distance [9]. Hence, a collision between UAV and objects is avoided.

1.4 Objective

The objectives of this project are:

1. To investigate the collision avoidance algorithm for UAV.

2. To model the Artificial Potential Field (APF) algorithm on MATLAB.

3. To analyze the performance of APF algorithm under 3 conditions, different number of obstacles, extreme goal position and wind disturbances.

4. To analyze the effectiveness of distance sensor.

1.5 Scope and Limitation

The main scope of this project is to explore the collision avoidance algorithm that is suitable for UAV. During the algorithm research, the application pros and cons of the algorithm are highlighted. In this project, the priority is given to algorithms which are suitable for path planning and collision avoidance for UAV. The algorithm is simulated on MATLAB by adding some real-world scenarios which affect the performance of the algorithm. The performance of the algorithm is analyzed through the simulation result of MATLAB. As an advancement of the collision avoidance mechanisms, distance sensors can be applied on UAV to detect the distance between the UAV and obstacles. In this project, the specifications of ultrasonic sensors HC-SR04 and infrared sensors KY-032 will be tested and compared. Then, a suitable distance sensor is selected.

The limitation for this project is that the designated virtual environment on MATLAB may not fully represent the real-world scenarios. The success obtained in the simulation may vary when implemented on the physical UAV which navigates in dynamic environments. Furthermore, the testing experiment of sensors is fully carried out in the indoor environment. When the selected sensor is applied on physical UAV, the performance of the sensor may be affected due to the unpredictable conditions such as variable weather and lighting.

1.6 K-Chart

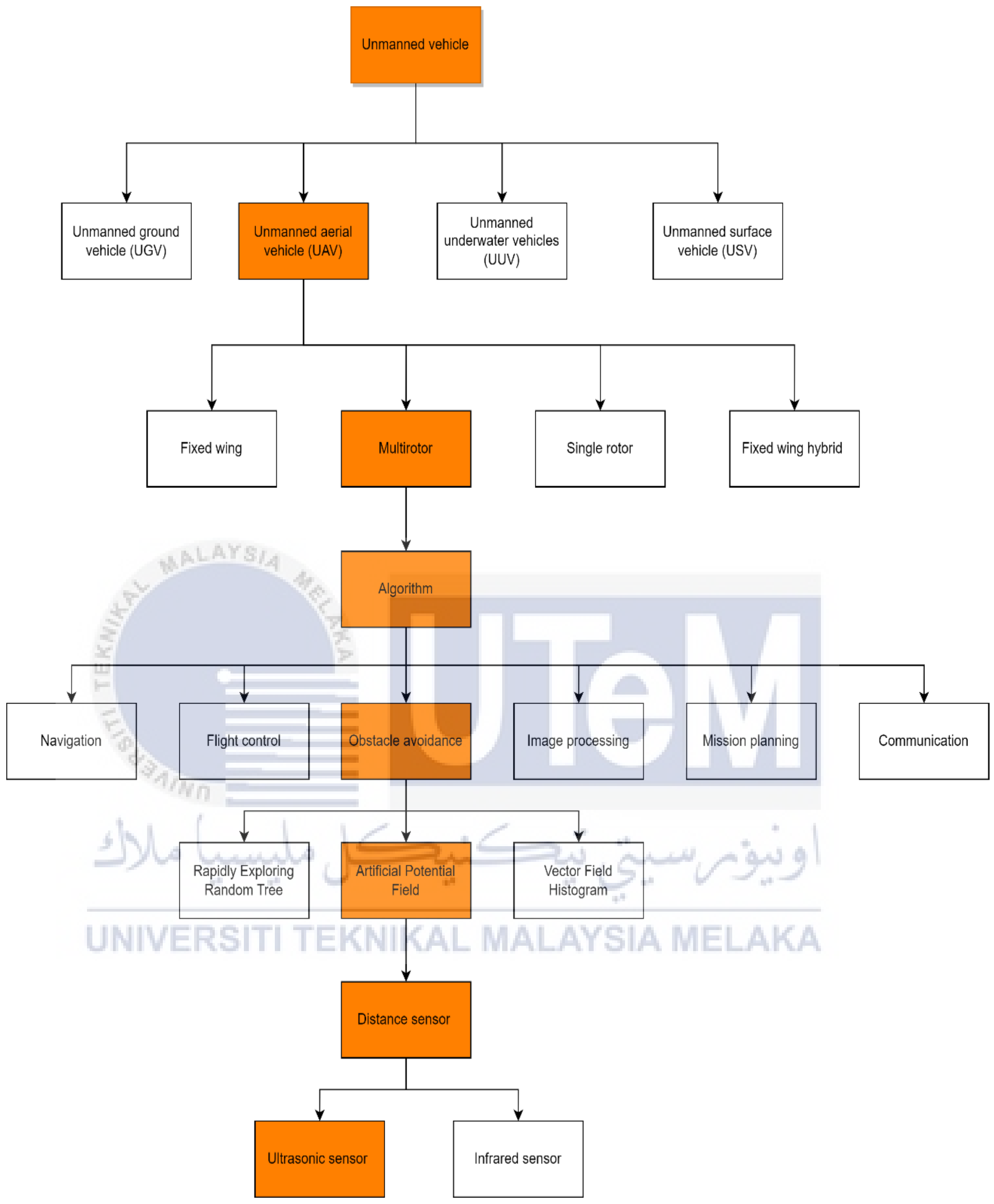


Figure 1.3 K-chart of Unmanned Vehicle

CHAPTER 2

LITERATURE REVIEW

2.1 Overview of UAV

Unmanned aerial vehicles (UAV) are aircraft without a human pilot on board [10]. Over the last few decades, UAVs have improved significantly, especially for its advancements in component technologies. There is a strong relation between UAV with our daily activities. Due to the advancements in their technology, UAV are used in various application such as inspecting pipelines, agriculture, surveillance and mapping.

Figure 2.1 UAV used for surveillance and mapping [10]

Figure 2.2 UAV used for spraying pesticide on crops [10]

There are many types of UAV available in the market. UAV can be categorized according to their size, range and properties. Table 2.1 shows that the classification of UAV is based on weight with their specification.

Table 2.1 Classification of UAV [11]

Type	**Maximum weight**	**Operating altitude (m)**	**Range (km)**	**Payload (kg)**	**Flight time (min)**	**Description**
Nano	200 g	50	5	< 0.2	6 -8	Easily remote and reach remote locations
Micro	2 kg	< 90	25	0.2-0.5	45	Operated on low altitudes with limited space for fuel and battery
Mini	20 kg	150 – 300	40	0.5-10.0	18	Maintain line of sight between aircraft and ground station
Small	25kg-150kg	< 1500	150	5.0-50.0	180	Operated at low to medium altitudes and longer loiter capabilities
Tactical	>150 kg	< 3000	200	25.0-200.0	1800	Operated at high altitudes, provide tracking or monitoring

There are four major types of UAV which are fixed wing, fixed wing hybrid, single rotor and multirotor.

Table 2.2 Types of UAV and its description [15]

UAV	Description
Fixed wing [12]	Features: Long endurance, fast flight speed Requirement: Special skills to operate, wide area to launch Application: Aerial mapping, inspection of pipelines
Fixed wing hybrid [13]	Features: vertical takeoff and landing (VTOL), long endurance flight Limitations: Still in development, not good at hover or forward flight Application: Delivery
Single rotor [13]	Features: VTOL, hover and long enfurance flight Requirement: Special skill to operate, heavier payload like LiDAR sensor
Multirotor [14]	Features: VTOL, hover and short endurance flight Limitations: Not suitable for longer distance monitoring due to limited speed, flight time, energy efficiency Applications: Photography, Video surveillance

The combination of airframe and a computer system, Flight Control System enables the UAV to fly autonomously [16]. The Flight Control System includes hardware and software architecture for UAV. For hardware, it consists of sensors like accelerometers and magnetometers, GPS and CPUs. Algorithms are software which is implemented in the flight control system of UAV. All these elements collaborate to fly the plane without human intervention. Figure 2.3 shows the main components that are embedded in the UAV system.

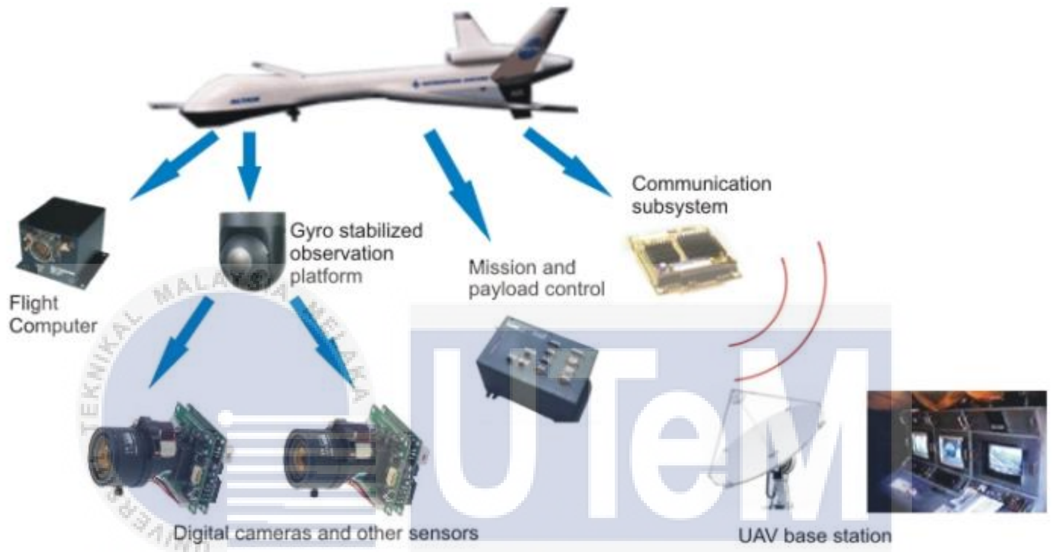

Flight Computer
Gyro stabilized observation platform
Mission and payload control
Communication subsystem
Digital cameras and other sensors
UAV base station

Figure 2.3 Main components of UAV system [16]

2.2 Previous Studies on Algorithm of UAV

Algorithms play an important role in the operation of UAV. Different algorithms are accessible to UAVs. An appropriate algorithm is selected depending on the specific requirements. An algorithm is a set of instructions that a computer or person can follow to complete a task. In UAV system, there are two types of computers which are flight controller and mission computer. Some basic algorithms are pre-loaded into flight controller's firmware to keep the UAV stable. Conversely, the algorithm which relates to special mission such as path planning is embedded in the mission computer. In this project, the algorithm that will be investigated is related to collision avoidance and path planning.

2.2.1 Artificial Potential Field

Artificial Potential Field (APF) is an algorithm which is developed by the Khatib in 1986 [17]. Due to simplicity, high efficiency and smooth trajectory generation of APF algorithm, it is widely used in UAV trajectory planning and obstacle avoidance [18]. The trajectory that generated by the APF is the smoothest and safest, but it is not the shortest trajectory. The basic concept of APF is applying attractive and repulsive force. The desired goal acts like an attractive pole while the obstacle acts like a repulsive pole. Khatib modeled the UAV and the target point as particles, and he regarded obstacles as circles. The analysis of the Artificial Potential Field (APF) model was conducted in two-dimensional space. At any position in the planned space, the direction of UAV's movement is determined by the resultant force field, a combination of the gravitational field from the target or goal and the repulsion field from the obstacles. By considering only a single obstacle present in space, the attractive and repulsive potential function can be represented as follows:

$$\boldsymbol{U_{art}(x) = U_{goal}(x) + U_{obs}(x)} \tag{2-1}$$

$$\boldsymbol{U_{goal}(x) = \frac{1}{2}k_p(x - x_d)^2} \tag{2-2}$$

$$\boldsymbol{U_{obs}(x)} = \begin{cases} \boldsymbol{0.5}\eta(\frac{\boldsymbol{1}}{\boldsymbol{\rho}} - \frac{\boldsymbol{1}}{\boldsymbol{\rho_0}})^2, \boldsymbol{\rho \leq \rho_0} \\ \boldsymbol{0} \quad\quad , \boldsymbol{\rho > \rho_0} \end{cases} \tag{2-3}$$

Table 2.3 Notation and its description for equation (2-1) to (2-3)

Notation	Description
$\boldsymbol{U_{art}}$	Artificial potential energy
$\boldsymbol{U_{goal}}$	Attractive potential energy
$\boldsymbol{U_{obs}}$	Repulsive potential energy
$\boldsymbol{x}$	Spatial position of UAV
$\boldsymbol{x_d}$	Spatial position of goal
$\boldsymbol{k_p}$	Attractive force gain coefficient
η	Repulsive force gain coefficient
$\boldsymbol{\rho}$	Shortest distance to the obstacle O
$\boldsymbol{\rho_0}$	Limit distance of the potential field influence

After computing the negative gradient of gravitational potential field function, the corresponding attractive force function and repulsive force function are determined as follows:

$$\boldsymbol{F_{goal}(x)} = -\boldsymbol{grad}[\boldsymbol{U_{goal}(x)}] = -\boldsymbol{k_p}(\boldsymbol{x} - \boldsymbol{x_d}) \tag{2-4}$$

$$\boldsymbol{F_{obs}(X)} = -\boldsymbol{grad}[\boldsymbol{U_{obs}(x)}]$$

$$= \begin{cases} \eta \left(\frac{\boldsymbol{1}}{\boldsymbol{\rho}} - \frac{\boldsymbol{1}}{\boldsymbol{\rho_0}}\right)\frac{\boldsymbol{1}}{\boldsymbol{\rho^2}}\frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{x}} \ , & \boldsymbol{\rho \le \rho_0} \\ \boldsymbol{0} \ , & \boldsymbol{\rho > \rho_0} \end{cases} \tag{2-5}$$

Table 2.4 Notation and its description for equation (2-4) to (2-5)

Notation	Description
$\boldsymbol{F_{goal}}$	Attractive force
$\boldsymbol{F_{obs}}$	Repulsive force
$\frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{x}}$	Partial derivatives of variable shortest distance to the obstacle O in the spatial position of UAV

When multiple obstacles present, the resultant force is:

$$\boldsymbol{F_{art}} = \boldsymbol{F_{goal}(x)} + \sum_{\boldsymbol{obs=1}}^{\boldsymbol{n}} \boldsymbol{F_{obs}(x)} \tag{2-6}$$

Although the APF algorithm proves effective in path planning and obstacle avoidance, it is susceptible to phenomena such as converging on local minimum points, target unreachability and trajectory jitter in narrow region. To overcome those issues, the conventional algorithm has been approved. Rostami et al. [19] have introduced that inserting a regulatory factor, ${R_A}^M$ into algorithm can overcome the local minima and unreachable target when the UAV is surrounded by obstacles issue. In this modified algorithm, the repulsive force, F_R is divided into two vector components, F_{R1} and F_{R2}. $F_{R1}$is the force that aligned with the direction from the UAV to the obstacle while F_{R2} is the force that aligned with the direction from the UAV to the target. As the UAV approaches the target, F_{R1} decreases more rapidly (order M) compared to F_{R2}. This rapid decrease in F_{R1} leads it to disappear, facilitating convergence based on F_{R2} and the attraction force. Hence, the regulatory factor, ${R_A}^M$ enables obstacle avoidance and convergence to the target. In [20], a gravitational function generated by obstacles into the repulsive function is introduced to solve the issue of unreachable target points and falling into local optimum. To ensure the UAV fly within the prescribed scope of path planning, a border function which can limit the flying area of UAV is implemented.

2.2.2 Rapid-exploration Random Tree

Rapid-exploration Random Tree (RRT) algorithm is proposed by LaVall [21]. RRT algorithm is used as an obstacle avoidance path planning technique which can do obstacle avoidance path planning in real time and online. RRT algorithm can construct a safe and flyable path within a short period of time for UAVs in a variety of threat scenarios. The basic principle of RRT algorithm is representing an initial point as a root node. Subsequently, leaf nodes are added to generate a random extension tree. The process ends when the leaf node of random extension trees includes target node goal x. The tree expansion process is shown in Figure 2.4. To have a better understanding of the basic principle of RRT algorithm, its flowchart is represented in Figure 2.5.

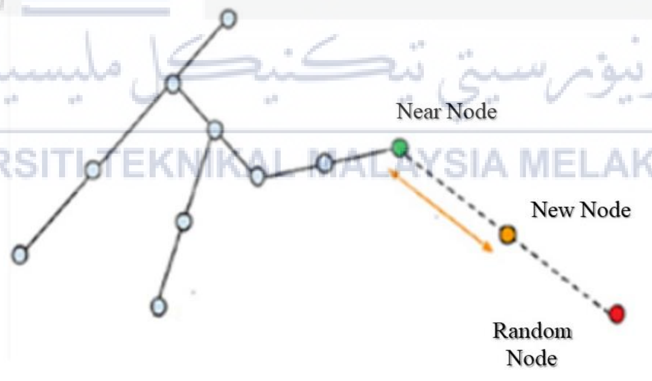


Figure 2.4 Tree expansion process for RRT algorithm [22]

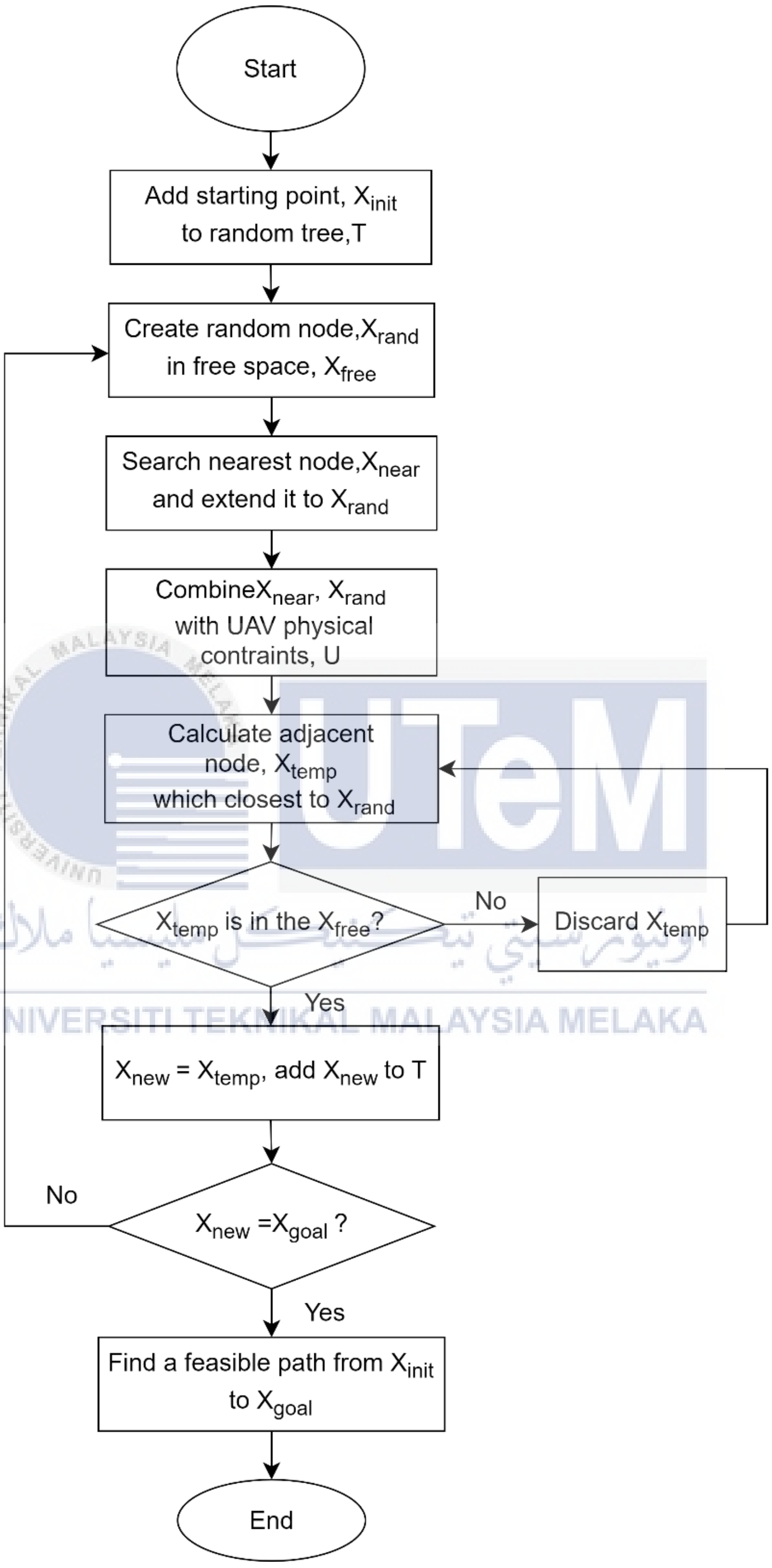

Start
Add starting point, X_{init} to random tree,T
Create random node,X_{rand} in free space, X_{free}
Search nearest node,X_{near} and extend it to X_{rand}
CombineX_{near}, X_{rand} with UAV physical contraints, U
Calculate adjacent node, X_{temp} which closest to X_{rand}
X_{temp} is in the X_{free}?
No
Discard X_{temp}
Yes
X_{new} = X_{temp}, add X_{new} to T
No
X_{new} =X_{goal} ?
Yes
Find a feasible path from X_{init} to X_{goal}
End

Figure 2.5 Flowchart of basic principle of RRT algorithm [21]

Yang et al. [23] proposed a RRT path optimization approach based on ant colony algorithm to obtain a global optimal solution. Pheromones are applied to the path discovered by the RRT. Then, the next expansion point is chosen based on pheromone concentration through roulette wheel selection. Through multiple iterations, an improved path is generated. In [24], a refined algorithm, named EPF-RRT, has been proposed. It combines the concept of environmental potential field and original RRT algorithm. The EPF-RRT guides the RRT growth towards the goal and avoid obstacles simultaneously. In [25], an improved algorithm is proposed which integrates the Artificial Potential Field (APF) algorithm with RRT algorithm. A path which is close to the optimal one will be generated within a shorter time by using the improved algorithm. The highlighted point of this algorithm is that some parts of the original path that is affected by dynamic obstacles will be discarded. After that, a new path will be generated from the current position of UAV to the goal point.

2.2.3 Vector Field Histogram

Vector Field Histogram (VFH) is a path planning which is proposed by Johann Borenstein [26]. The basic concept of VFH is a combination of grid method and artificial potential field method. By using VFH algorithm, a polar coordinate histogram H is built and an optimal region around the UAV is selected as the motion direction. The polar histogram is updated at different angular resolutions [27]. The process of the VFH algorithm is shown in Figure 2.6. According to Figure 2.7, an UAV is positioned at the center of an active window with gridded space. A grid confidence matrix, C_{ij} which represents the obstacle confidence present in each grid is generated. After that, C_{ij} is mapped to polar coordinate histogram H. A polar coordinate system is built with the position of UAV (p_{ux}, p_{uy}) as its center. Every grid (i, j) is associated with a vector directed towards it by the UAV. According to the vector, the corresponding angle β of the grid is calculated as in equation (2-7). The polar obstacle density, POD of the grid can also be determined as shown in equation (2-8).

$$\boldsymbol{\beta_{ij} = arctan\frac{y_j - p_{uy}}{x_i - p_{ux}}} \tag{2-7}$$

$$\boldsymbol{m_{ij} = {c_{ij}}^2(a - bd_{ij})} \tag{2-8}$$

Finally, an optimal motion direction is selected by referring to the distribution of obstacles in each sector in polar coordinate histogram H. The threshold T is used to filter out the peaks and valleys. Then, UAV will select either valley which is closest to intended motion direction.

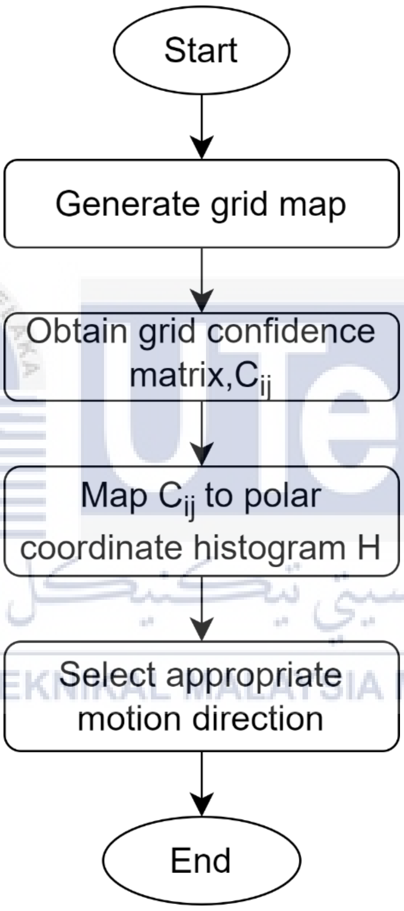

Start
Generate grid map
Obtain grid confidence matrix,C_{ij}
Map C_{ij} to polar coordinate histogram H
Select appropriate motion direction
End

Figure 2.6 Flowchart of basic principle of VFH algorithm [26]

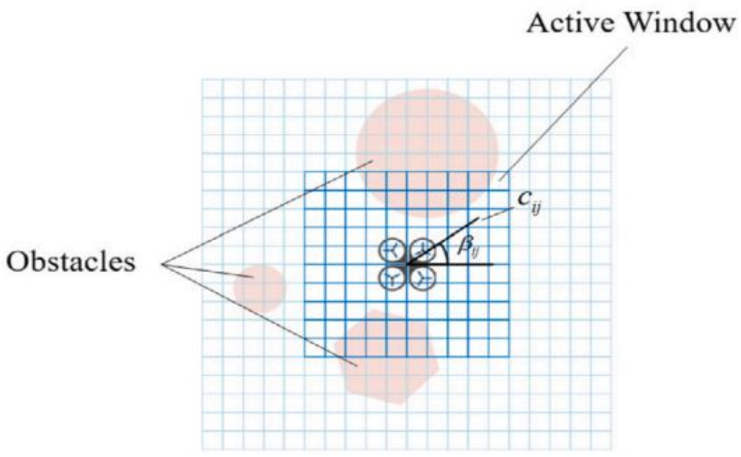


Figure 2.7 Grid map diagram [26]

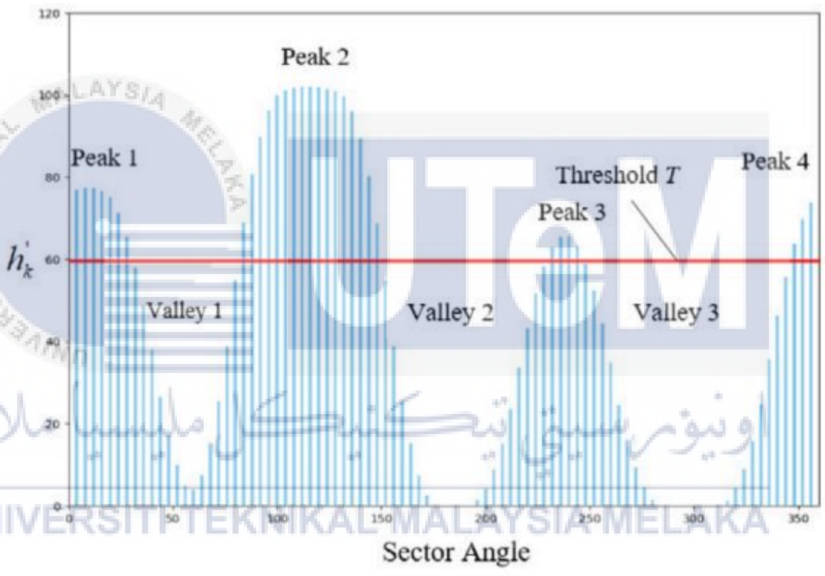


Figure 2.8 Polar coordinate histogram H [26]

In [28], an enhanced VFH algorithm is proposed by combining the concept of kinematic and dynamic constraints of the vehicle. A new active region is generated for VFH to ensure that the vehicle can reach all states within the specified region. Hence, smoother and collision-free trajectories are generated. H. Zhang et.al. [29] developed a Dubins-based improved vector field histogram (VFH) for fixed-wing UAV. A new path is inserted into the environment and a collision-free trajectory is generated. In [30], VFH algorithm is corporate with sensor by processing the sensor data. After that, the desired speed of drone flight is generated based on sensor data.

Table 2.5 Comparison between APF, RRT and VFH algorithm

Comparison	**Artificial Potential Field (APF)**	**Rapid-exploration Random Tree (RRT)**	**Vector Field Histogram (VFH)**
Founder	Khatib [17]	LaVall [21]	Johann Borenstein [26]
Principle to generate path	Apply potential field [18]	Build tree structure [21]	Analysis on polar coordinate histogram [27]
Adaptability to dynamic environments	High adaptability since continuous adjustment is applied to potential field [17]	Moderate adaptability, increase frequency of updating tree structure [25]	Low adaptability, reduce the reliability of the polar coordinate histogram [43]
Path Optimizing	May not generate optimized path due to potential local minimum [18]	Can generate a safe but not the shortest path within a short period of time [24]	Focus on generating a path without obstacle only [30]
Integration of sensor with algorithm	Data is used to generate repulsive forces in potential field [17]	Data is used to determine the configuration space and exploration of tree structure [23]	Data is used to construct polar coordinate histogram for selecting path without obstacles [30]

The pros and cons for three algorithms, APF, RRT and VFH are compared in Table 2.5. By comparing three algorithms, APF algorithm is simple to implement and can be used in dynamic environment but cannot generate an optimized path due to local minima. RRT algorithm can be used in dynamic environments but slow to converge on a good path. Hence, an optimal path may not be generated by applying RRT algorithm. VFH algorithm can be used for generating a path without colliding with obstacle but not efficient in dynamic environments.

In summary, since UAV always navigate through unpredictable environment, it needs real-time operation of algorithm which can rapidly respond to the changing environment. After comparing three algorithms, APF is more suitable to be applied in this project. The high adaptability of APF to various environments makes it is good choice for collision avoidance and path planning in UAV applications.

2.3 Sensor

2.3.1 Ultrasonic Sensor

Ultrasonic sensor is a type of distance sensor. This type of sensor is widely used in object detection and range detection due to its high efficiency [31]. It is used to measure the distance of objects in air through reflection of sound waves [32]. The time response of ultrasonic sensor depends on reflectance characteristics of the surface of detected object. Ultrasonic sensors can be used with the presence of microcontrollers or microprocessors such as Arduino and Raspberry Pi. Figure 2.9 shows the block diagram on the cooperation between ultrasonic sensors and other hardware with Arduino.

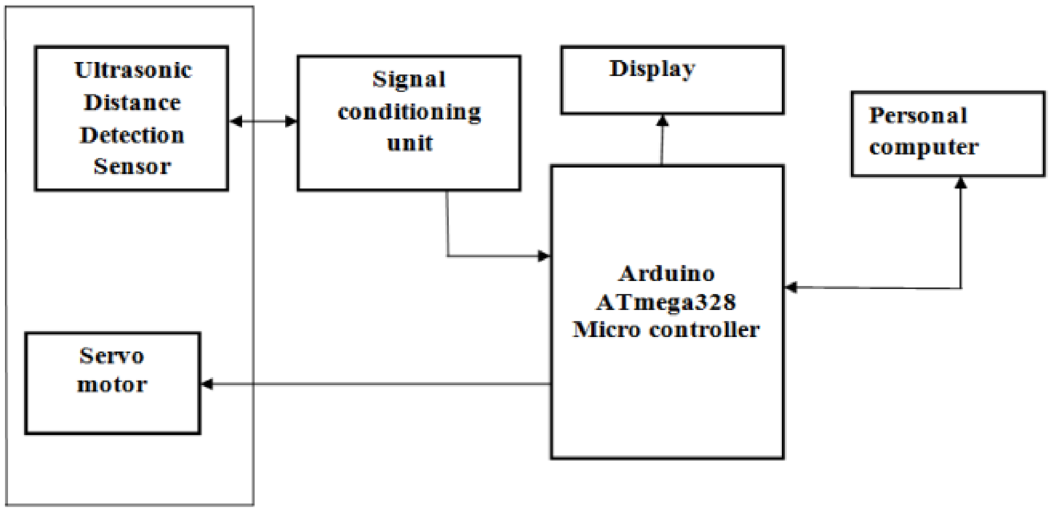

Ultrasonic Distance Detection Sensor
Signal conditioning unit
Display
Personal computer
Arduino ATmega328 Micro controller
Servo motor

Figure 2.9 Block diagram of ultrasonic distance detection with Arduino [32]

Ultrasonic sensor can detect objects which are located 2 cm-400 cm far from the sensor [33]. A 5V power supply is needed for sensor to operate. There are 4 connection pins on the ultrasonic sensor which are Vcc, Gnd, Trigger and Echo. For trigger input, it requires 10μs pulse as an input. Hence, an ultrasound at 40kHz will be sent out. Echo is an output pin which represents the time taken for ultrasonic sound return to sensor.

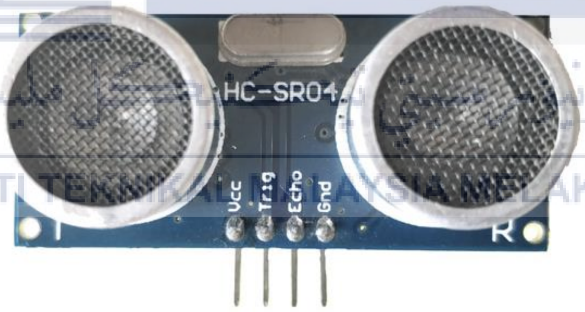


Figure 2.10 Pin configuration of ultrasonic sensor [33]

Ultrasonic ranging, phase detection, acoustic amplitude detection method and transit time detection are the methods used for ultrasonic sensor [34]. The most common method used is transit time detection in which detecting the transmission of ultrasonic wave to receiver's time. The working principle of ultrasonic sensor is shown in Figure 2.11. To obtain the distance of ultrasonic wave, the formula is in equation (2-9) with Δt is the time difference of ultrasonic pulse transmitting and receiving, $V = 340\ m/s$ is the ultrasonic velocity.

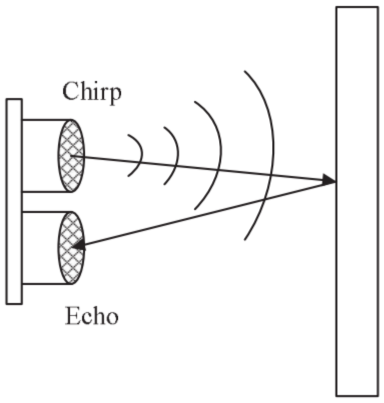


Figure 2.11 Working principle of ultrasonic sensor [34]

$$\boldsymbol{S = V.\frac{\Delta t}{2}} \tag{2-9}$$

In [35], an implementation of ultrasonic sensor, HCSR04 on UAV is proposed. There are a total of 4 ultrasonic sensors applied on the UAV system to detect the presence of obstacles at left, right, front, and back of UAV. In [36], HC-SR04 ultrasonic sensor is connected to Arduino Yun to get the distance between the sensor and the obstacle.

2.3.2 Infrared Sensor

Infrared sensor KY-032 is another type of distance sensor which is lighter in weight, less complexity and lower in cost [37]. It is widely used for obstacle avoidance. Infrared radiation is an electromagnetic wave with a frequency below the sensitivity range of the human eye. Infrared sensor is used to detect the presence of object through reflected infrared radiation [38]. It have an infrared transmitter and a receiver which form the sensor pair. The transmitter LED emits an infrared light with the frequency which can be detected by receiver LED. The receiving LED detects some of the signal back and triggers the digital on or off signal pin when a specific threshold distance has been detected.

There are 3 pins on infrared sensor which are Vcc, Gnd and OUT pin [40]. Vcc is the power supply input, GND is power supply ground and OUT is an output pin. The voltage supply for the infrared sensor is 5V DC while the current supply is 20mA. The detection range of the infrared sensor is up to 20 centimeters.

Figure 2.12 Infrared sensor [39]

In [40], four infrared (IR) sensors are utilized to prevent collisions by steering in the opposite direction. S. A. Daud et.al. [41] applied multiple infrared sensors to rebuild the shape of objects by detecting the changes in sensor displacement. Infrared sensors possess non-linear characteristics [42]. When measuring the distance, the angle of reflecting surface needs to be located directly to the sensor. The range resolution of infrared sensors is not as high as ultrasonic sensors. However, certain types of infrared sensors which are expensive perform better resolution at long distance. Besides, infrared sensors are always used to enhance the real-time response of a mobile robot due to its faster response.

Table 2.6 Comparison between ultrasonic and infrared sensor

Comparison	**Ultrasonic sensor**	**Infrared sensor**
Transmission medium	Emits ultrasonic sound waves [32]	Emits infrared light [38]
Working principle	Determine the distance by measuring time taken for sounds waves reflects from the obstacle [34]	Determine the presence of object by detecting the reflected infrared radiation [38]
Operating voltage	5V [33]	5V [39]
Detection range	2cm to 400cm [33]	Up to 20cm [39]
Factors that affect the accuracy of sensor	Affected by environmental factors such as temperature [44]	Affected by reflective properties of the surface of obstacles [44]
Response's speed	Longer response time due to speed of sound [38]	Shorter response time due to faster speed of light [38]

In summary, ultrasonic sensor is suggested to be integrated on the UAV due to its high detection range if compared to infrared sensor. The high detection range of ultrasonic sensor makes the obstacle which located further from sensor can be detected if compared to infrared sensor. Hence, flight controller can make adjustment on UAV to avoid collision with obstacles. Last but not least, high compability of APF algorithm and ultrasonic sensor can result an efficient collision avoidance mechanisms for UAV.

CHAPTER 3

METHODOLOGY

3.1 Table Structure of Project

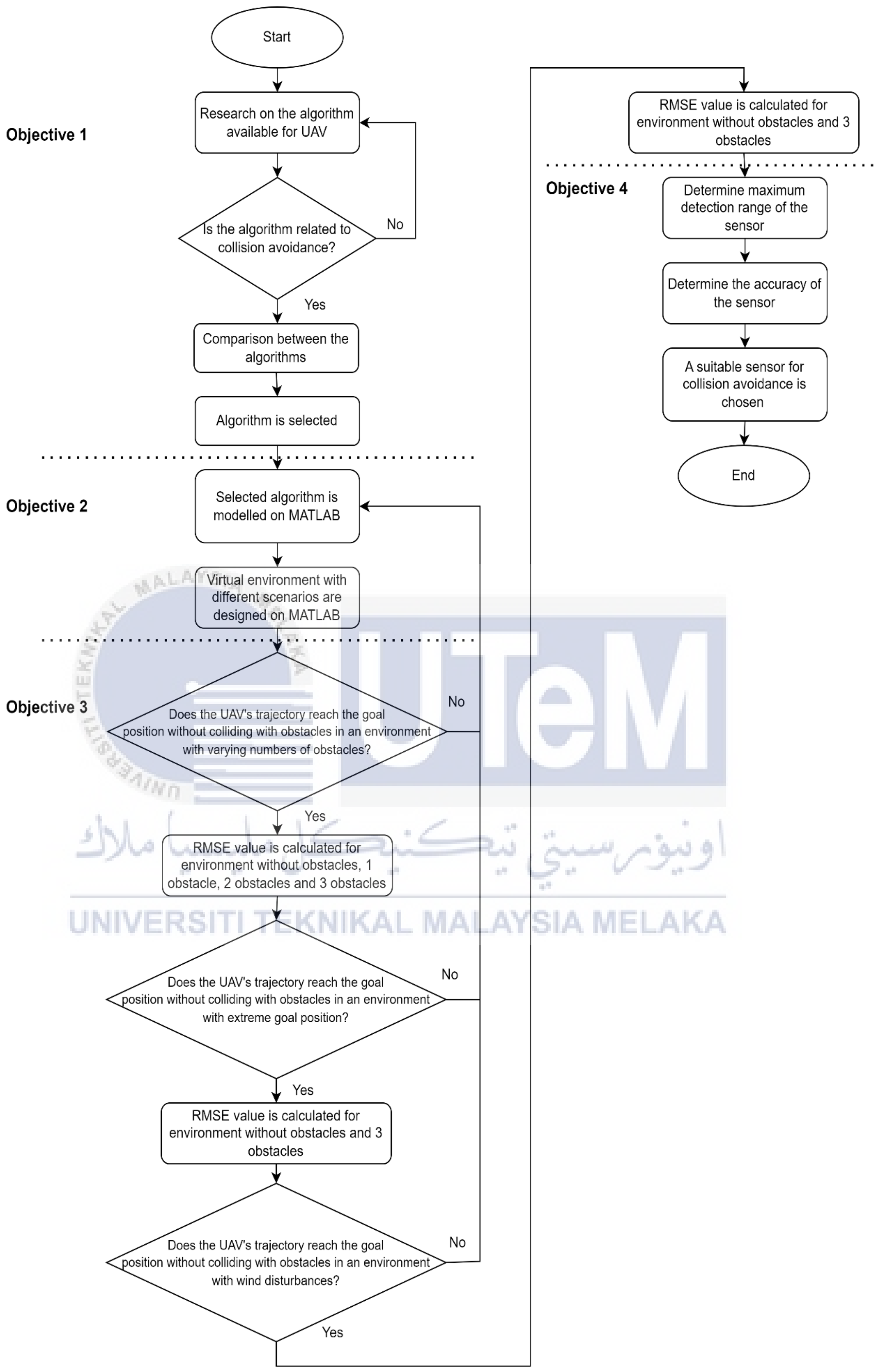
Table 3.1 displays the experiment or task that was conducted and mapping to every objective which stated previously. To achieve objective 1, an investigation on suitable algorithm for collision avoidance mechanisms of UAV was conducted. For objective 2, the APF algorithm was modelled on MATLAB and its performance was analyzed to achieve objective 3. For objective 4, experiments were carried out to determine the effectiveness of sensor.

Table 3.1 Table structure of the project

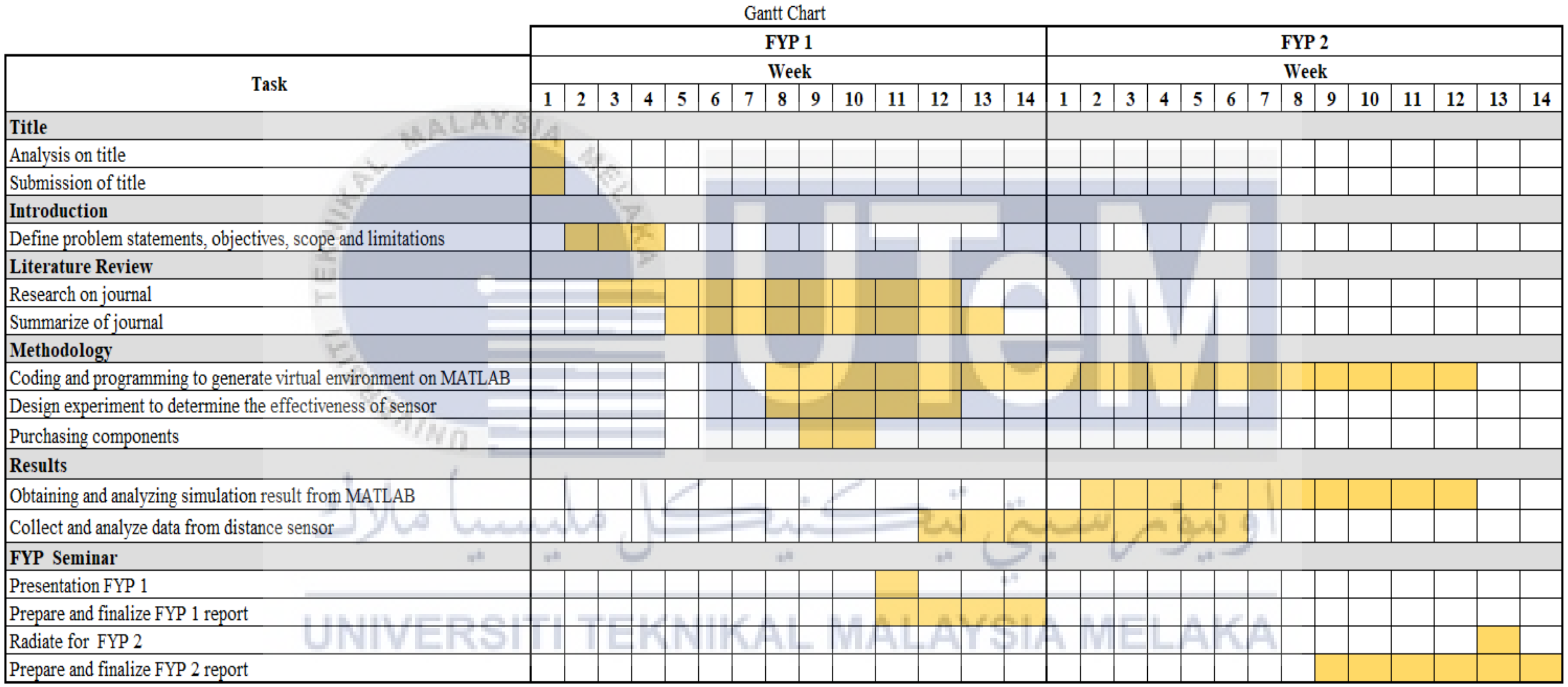
	Objective 1	Objective 2	Objective 3	Objective 4
Investigation of collision avoidance algorithm for UAV	√			
Modelling of APF algorithm on MATLAB		√		
Analyzation of APF algorithm's performance under conditions different numbers of obstacles, extreme goal position and wind disturbances.			√	
Experiment: Effectiveness of Sensor				√

3.2 Flow Chart of Project

The overall process of this project is summarized as shown in Figure 3.1. A previous study on collision avoidance algorithm for UAV was conducted. After that, a suitable algorithm was selected to be modelled on MATLAB. The forward step was designing a virtual environment with varying conditions, different number of obstacles, extreme goal position and wind disturbances to test the functionality of the selected algorithm, Artificial Potential Field (APF) algorithm. The performance of APF algorithm was analyzed by calculating the RMSE value. To enhance the collision avoidance mechanisms of UAV, a sensor was suggested to be implemented on the UAV for real time application. The maximum detection range and accuracy of ultrasonic sensor and infrared sensor were determined through experiment.


Start
Objective 1
Research on the algorithm available for UAV
Is the algorithm related to collision avoidance?
No
Yes
Comparison between the algorithms
Algorithm is selected
Objective 2
Selected algorithm is modelled on MATLAB
Virtual environment with different scenarios are designed on MATLAB
Objective 3
Does the UAV's trajectory reach the goal position without colliding with obstacles in an environment with varying numbers of obstacles?
No
Yes
RMSE value is calculated for environment without obstacles, 1 obstacle, 2 obstacles and 3 obstacles
Does the UAV's trajectory reach the goal position without colliding with obstacles in an environment with extreme goal position?
No
Yes
RMSE value is calculated for environment without obstacles and 3 obstacles
Does the UAV's trajectory reach the goal position without colliding with obstacles in an environment with wind disturbances?
No
Yes
RMSE value is calculated for environment without obstacles and 3 obstacles
Objective 4
Determine maximum detection range of the sensor
Determine the accuracy of the sensor
A suitable sensor for collision avoidance is chosen
End
Figure 3.1 Flow chart of project

3.3 Gantt Chart



Gantt Chart

Task	FYP 1														FYP 2													
	Week														Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Title																												
Analysis on title	■																											
Submission of title	■																											
Introduction																												
Define problem statements, objectives, scope and limitations		■	■	■																								
Literature Review																												
Research on journal			■	■	■	■	■	■	■	■	■	■																
Summarize of journal					■	■	■	■	■	■	■	■	■															
Methodology																												
Coding and programming to generate virtual environment on MATLAB								■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■		
Design experiment to determine the effectiveness of sensor								■	■	■	■	■																
Purchasing components									■	■																		
Results																												
Obtaining and analyzing simulation result from MATLAB																■	■	■	■	■	■	■	■	■	■	■		
Collect and analyze data from distance sensor												■	■	■	■	■	■	■	■	■	■							
FYP Seminar																												
Presentation FYP 1											■																	
Prepare and finalize FYP 1 report											■	■	■	■														
Radiate for FYP 2																											■	
Prepare and finalize FYP 2 report																							■	■	■	■	■	■

Figure 3.2 Gantt chart for project

3.4 Block Diagram

Figure 3.3 shows the block diagram of the collision avoidance system embedded with UAV system. Firstly, sensors were used to measure the distance between UAV and obstacles. The real-time data and distance measurements were sent to Arduino Uno. The collision avoidance algorithm was embedded in the Arduino Uno. An ideal path was generated by the algorithm according to the surrounding condition of UAV. Lastly, control signals which include the adjustment of UAV's flight parameter were sent from Arduino Uno to UAV system. Hence, collision with obstacles could be avoided by UAV.

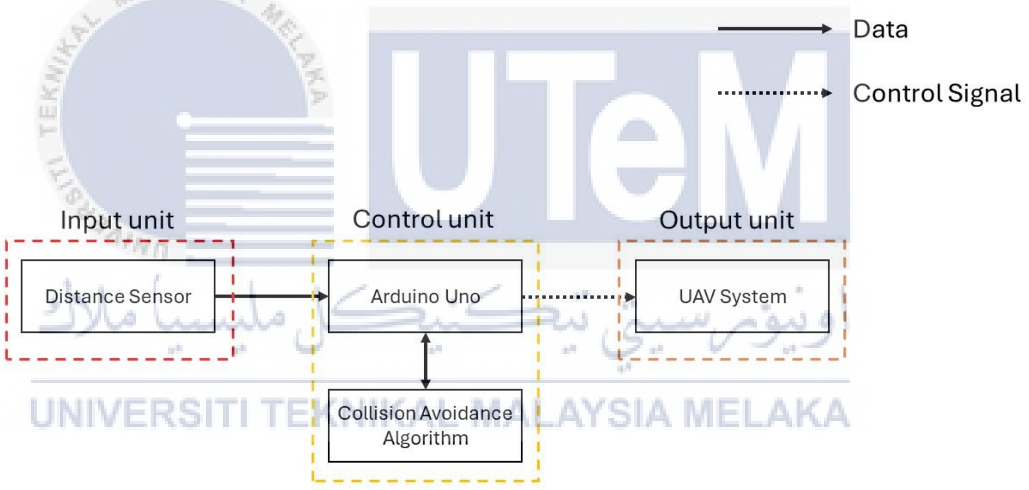


Figure 3.3 Block diagram for collision avoidance system with UAV system

3.5 Collision Avoidance Algorithm: Artificial Potential Field (APF)

The selected collision avoidance algorithm in this project is Artificial Potential Field (APF) algorithm. APF algorithm is an algorithm which is frequently used for path planning due to its safety and simplicity. Potential field is suitable for real-time applications. APF algorithm involves two forces which are attractive and repulsive force to perform the collision avoidance of UAV. Repulsive force is a force which is generated by obstacle while attractive force is generated by goals. The strength of the forces varies with the distance between UAV and obstacle or goal. By applying APF algorithm, the obstacle repels UAV while the goal attracts it. The resultant forces of the fields on UAV are used to determine the direction of UAV's motion.

When applying APF algorithm, the position vector of UAV is considered as $q = (x, y)^T$. The APF function is represented as equation (3-1).

$$\boldsymbol{U}(\boldsymbol{q}) = \boldsymbol{U_{att}}(\boldsymbol{q}) + \boldsymbol{U_{rep}}(\boldsymbol{q}) \qquad (3\text{-}1)$$

Table 3.2 Notation and its description for equation (3-1)

Notation	Description
$\boldsymbol{U}(\boldsymbol{q})$	Artificial potential field exerted on UAV
$\boldsymbol{U_{att}}(\boldsymbol{q})$	Attractive field exerted by goal
$\boldsymbol{U_{rep}}(\boldsymbol{q})$	Repulsive field exerted by obstacle

According to Figure 3.4, there is an attractive force, $\boldsymbol{F_{att}}$ which is direct towards to goal. This means that the attractive force is generated from goal to attract the UAV to move towards it. At the same time, there is a repulsive force, $\boldsymbol{F_{rep}}$ which is direct towards the opposite side of obstacle. This means that the repulsive force is generated from obstacle to repel the UAV. From equation (3-2), the force is the negative gradient of potential field. The resultant force, $\boldsymbol{F}$ is the combination force of attractive force, $\boldsymbol{F_{att}}$ and repulsive force, $\boldsymbol{F_{rep}}$. The direction of resultant force, $\boldsymbol{F}$ shows that the direction of motion of UAV.

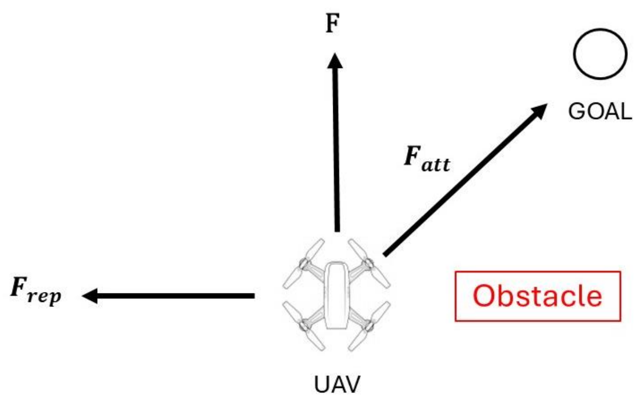


Figure 3.4 Resultant artificial force of potential function

$$\begin{aligned} F(q) &= -\nabla U(q) \\ &= -\nabla U_{att}(q) - \nabla U_{rep}(q) \\ &= F_{att}(q) + F_{rep}(q) \end{aligned} \tag{3-2}$$

Table 3.3 Notation and its description for equation (3-2)

Notation	Description
$F(q)$	Resultant artificial force which moves the UAV
$F_{att}(q)$	Attractive force which generated by goal
$F_{rep}(q)$	Repulsive force which generated by obstacle

The attractive field between UAV and goal is assembled to attract the UAV to the goal area. The attractive field between UAV and goal can be calculated by using the equation (3-3).

$$\begin{aligned} U_{att}(\mathbf{q}) &= \frac{1}{2} \times k_a \times \rho^2{}_{goal}(\mathbf{q}) \\ &= \frac{1}{2} k_a ||q - q_d||^2 \end{aligned} \tag{3-3}$$

Table 3.4 Noatation and its description for equation (3-3)

Notation	Description
k_a	Positive coefficient of gravity for APF
q	Current position vector of UAV
q_d	Desired goal position vector
ρ_{goal}	Euclidean distance from UAV's current position to goal position

The attractive force which is applied on the UAV can be calculated as the negative gradient of attractive potential field as shown in equation (3-4).

$$\begin{aligned} \boldsymbol{F_{att}}(\mathbf{q}) &= -\nabla \boldsymbol{U_{att}}(\boldsymbol{q}) \\ &= -\frac{1}{2}\, \boldsymbol{k_a}\, \boldsymbol{\rho^2}_{\boldsymbol{goal}}(\boldsymbol{q}) \\ &= -\boldsymbol{k_a}\,(\boldsymbol{q} - \boldsymbol{q_d}) \end{aligned} \tag{3-4}$$

From the aspect of potential field, UAV should be repelled away from obstacles. When the UAV is away from obstacles, the motion of UAV is not affected by obstacle. Hence, the repulsion potential field, $\boldsymbol{U_{rep}}(\boldsymbol{q})$ may considered as 0. When the UAV is close to the goal, repulsion potential field will gradually decrease. The repulsion potential field will become 0 when the UAV has reached the goal. The repulsion potential field which experienced by UAV can be calculated by using equation (3-5).

$$\boldsymbol{U_{rep}}(\mathbf{q}) = \begin{cases} \frac{1}{2}\, \boldsymbol{k_b}\, (\frac{1}{\boldsymbol{d}(\boldsymbol{q})} - \frac{1}{\boldsymbol{d_0}})^2 & , \quad \boldsymbol{d}(\boldsymbol{q}) \le \boldsymbol{d_0} \\ \mathbf{0} & , \quad \boldsymbol{d}(\boldsymbol{q}) \ge \boldsymbol{d_0} \end{cases} \tag{3-5}$$

Table 3.5 Notation and its description for equation (3-5)

Notation	**Description**
$\boldsymbol{k_b}$	Repulsion gain coefficient
$\boldsymbol{d}$	Distance between UAV and obstacle
$\boldsymbol{d_0}$	Distance of obstacle repulsive force field

Consider the configuration of obstacle which is closest to the latest position of UAV as $\boldsymbol{q_c} = (\boldsymbol{x_c}, \boldsymbol{y_c})$. The shortest distance between UAV and obstacles is considered as $\boldsymbol{d} = ||\boldsymbol{q} - \boldsymbol{q_c}||$ while the largest impact distance of obstacle to the UAV is considered as d_0. When the UAV is close to the obstacle, a repulsive force is exerted on the UAV. Conversely, there is no impact on UAV when the distance between the UAV and obstacle is greater than the largest impact distance, $\boldsymbol{d_0}$. Hence , the repulsive force can be considered as 0.

$$F_{rep}(q) = \begin{cases} k_b\left(\frac{1}{d(q)} - \frac{1}{d_0}\right)\left(\frac{1}{d^2(q)}\right)\left(\frac{\partial d(q)}{dx}\right) & , \quad d(q) \le d_0 \\ 0 & , \quad d(q) \ge d_0 \end{cases}$$

$$= \begin{cases} k_b\left(\frac{1}{d(q)} - \frac{1}{d_0}\right)\left(\frac{1}{d^2(q)}\right)\left(\frac{q - q_c}{||q - q_c||}\right) , & d(q) \le d_0 \\ 0 & , \quad d(q) \ge d_0 \end{cases} \tag{3-6}$$

According to the equation (3-6), the repulsive force which exerted on the UAV are the combination of its cartesian components which are repulsion force in x, F_{repx} and y direction, F_{repy}. The cartesian components of repulsion force, F_{repx} and F_{repy} can be calculated by applying equation (3-7) and (3-8).

$$F_{repx}(q) = \begin{cases} k_b\left(\frac{1}{d(q)} - \frac{1}{d_0}\right)\left(\frac{1}{d^2(q)}\right)\left(\frac{x - x_c}{||q - q_c||}\right) , & d(q) \le d_0 \\ 0 & , \quad d(q) \ge d_0 \end{cases} \tag{3-7}$$

$$F_{repy}(q) = \begin{cases} k_b\left(\frac{1}{d(q)} - \frac{1}{d_0}\right)\left(\frac{1}{d^2(q)}\right)\left(\frac{y - y_c}{||q - q_c||}\right) , & d(q) \le d_0 \\ 0 & , \quad d(q) \ge d_0 \end{cases} \tag{3-8}$$

Since there are n number of obstacles are designed in MATLAB environment, the artificial potential field, $U(q)$ and artificial force, $F(q)$ can be obtained as shown in equation (3-9) and (3-10) repectively.

$$U(\mathbf{q}) = U_{att}(q) + \sum_{i=1}^{n} U_{rep}(q) \tag{3-9}$$

$$F(\mathbf{q}) = F_{att}(q) + \sum_{i=1}^{n} F_{rep}(q) \tag{3-10}$$

By referring the mathematical equation of potential field and resultant force above, the APF algorithm is modelled and simulated on MATLAB.

3.6 Performance of Artificial Potential Field (APF) Algorithm

Artificial Potential Field (APF) algorithm is widely used in UAV systems for path planning and collision avoidance. According to research, some indirect factors may affect the performance of collision avoidance algorithm for UAV. As a consequence, UAV may collide with the obstacle. Hence, a framework to test the performance of the APF algorithm under varying conditions was outlined. There were 3 conditions which were number of obstacles, goal positioning and wind disturbance. The simulation was carried out on MATLAB which is a programming and numeric computing platform to analyze data, develop algorithms and create models.

3.6.1 Number of Obstacles

In practical application, UAV always fly into an environment containing an uncertain number of obstacles. The risk of collision may increase when UAV flies to an environment which contains high number of obstacles. The objective of this simulation is to evaluate how the APF algorithm performs when the number of obstacles between UAV's initial position and goal position is gradually increase. In the simulation environment, UAV is designed to fly to the goal position without collide with the static obtacle such as wall. The starting point of UAV is (0,0) and the final point which is the goal position is at (3.5, 3.5). Initially, UAV is simulated to fly in an environment without obstacles. After that, the number of obstacles is gradually increase from 1 obstacle until 3 obstacles. Figure 3.5 shows the UAV's simulation environment with different number of obstacles.

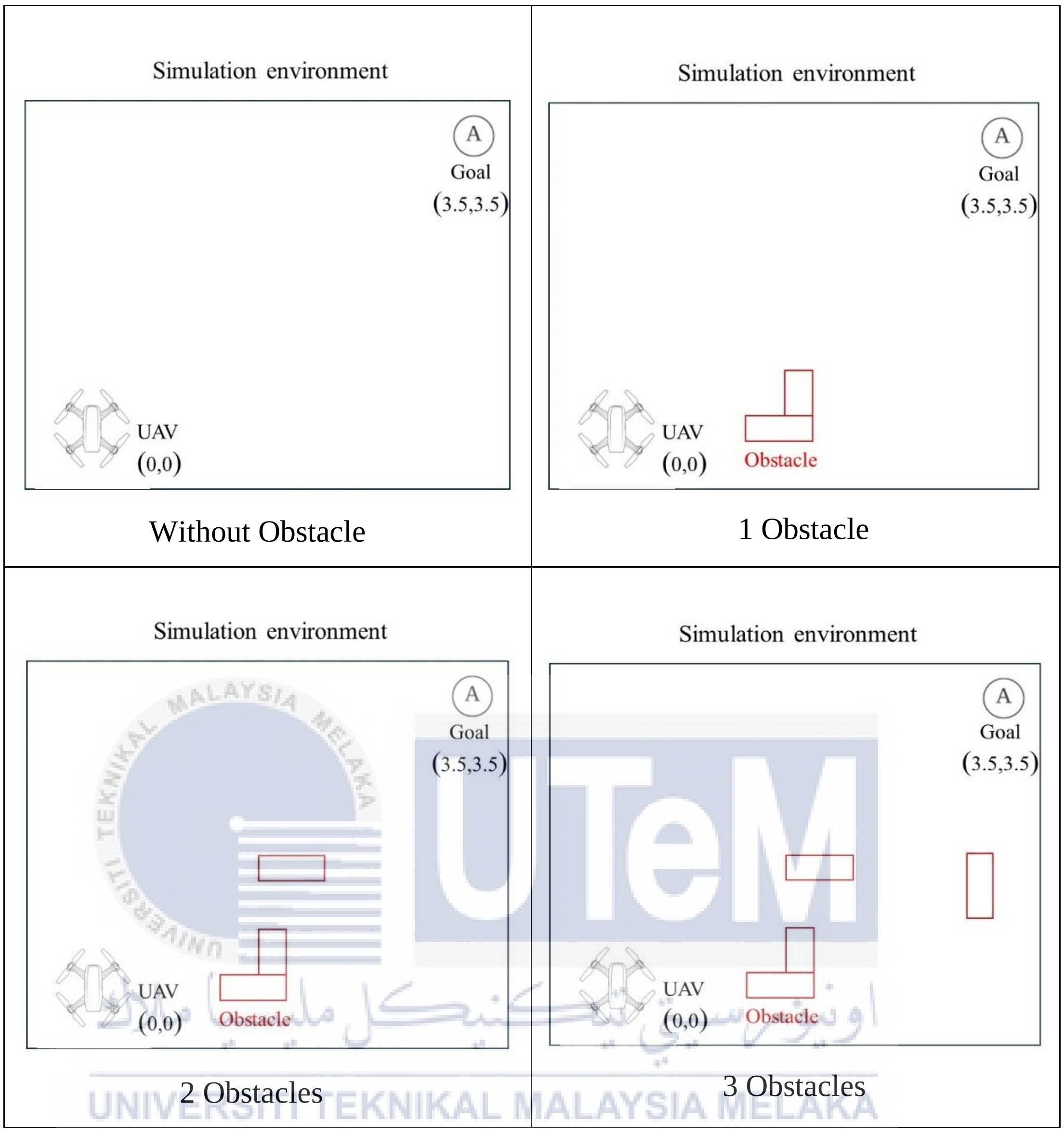


Figure 3.5 Simulation environment with different number of obstacles

3.6.2 Extreme Position of Goal

In real world application, goals are not frequently located at the same position. When designing the APF algorithm, it is crucial to consider a wide range of goal positions within the environment. Hence, this simulation is carried out to analyze the effectiveness of APF algorithm when goal is placed at extreme position within the environment. The starting position of UAV is at (0,0). In previous case, the goal position is at (3.5, 3.5). Conversely, in this case, the extreme position of goal is located at (100, 100). The simulation is carried out under 2 conditions which are without obstacles and with obstacles. The ability of the UAV to reach the goal is evaluated based on the simulation result.

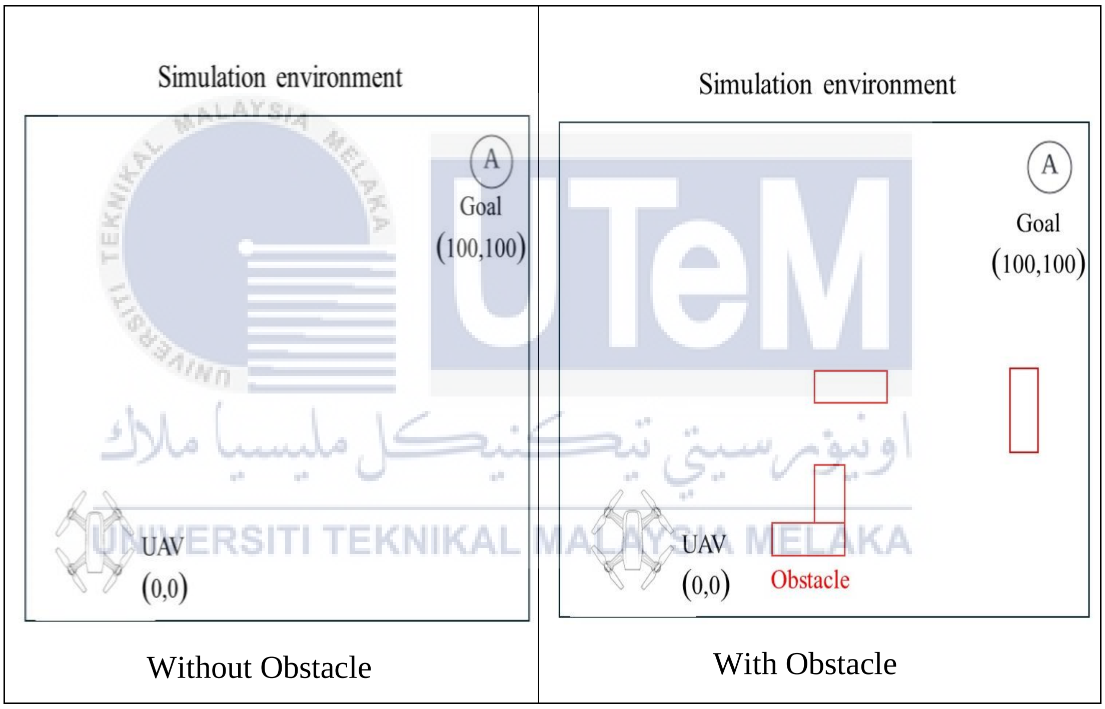


Figure 3.6 Simulation environment with extreme position of goal

3.6.3 Wind Disturbances

In real world application, there are many types of disturbance exists in the environment during the flight of UAV. The most common type of disturbance is wind disturbances. Wind may affects the UAV's trajectory by pushing UAV towards obstacles or change its intended path. Hence, it may leads to the collision of UAV and obstacle. Artificial Potential Field (APF) algorithm is required to account for wind forces to guide the UAV to its goal position accurately without deviating. In this simulation, UAV starts to fly from position (0,0) to goal position (3.5, 3.5). The simulation is carried out under 2 conditions which are without and with obstacles. The performance of APF algorithm is analyzed based on the simulation result.

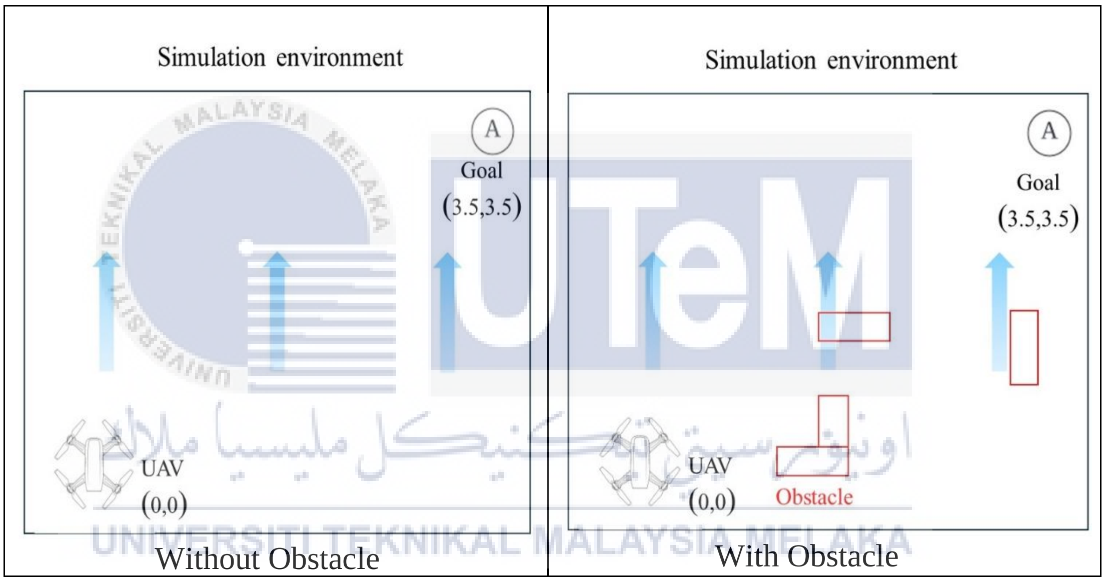


Figure 3.7 Simulation environment with wind disturbances

3.7 Experiment for Sensor

An ideal UAV system typically requires APF algorithm and sensors for optimal performance. APF algorithm provides a framework for path planning and obstacle avoidance. Sensors also play a crucial role in providing real-time environmental data to the UAV. By combining the APF algorithm with sensor inputs, the UAV can navigate safely and efficiently in dynamic uncertain environments.

For the following section, 2 experiments are carried out to analyze the effectiveness of the ultrasonic and infrared sensor. The maximum detection range of ultrasonic and infrared sensors are determined through Experiment 1 while the accuracy of ultrasonic and infrared sensors is tested through Experiment 2.

3.7.1 Experiment 1: Maximum detection range of sensor

3.7.1.1 Ultrasonic Sensor

Objective: To determine the maximum detection distance of the ultrasonic sensor**.**

Apparatus:

1. Breadboard
2. Arduino Uno
3. Ultrasonic Sensor HC-SR04
4. Jumper Wires
5. Cardboard (Obstacles)
6. Computer
7. Steel measuring tape

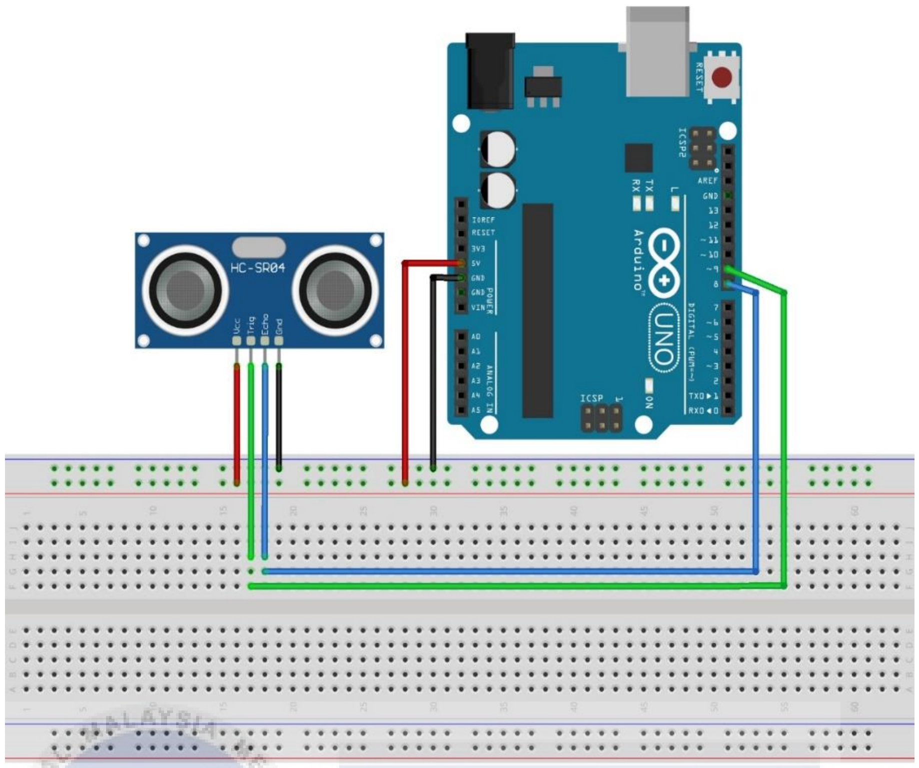


Figure 3.8: Wiring connection of ultrasonic sensor HC-SR04 and Adruino Uno

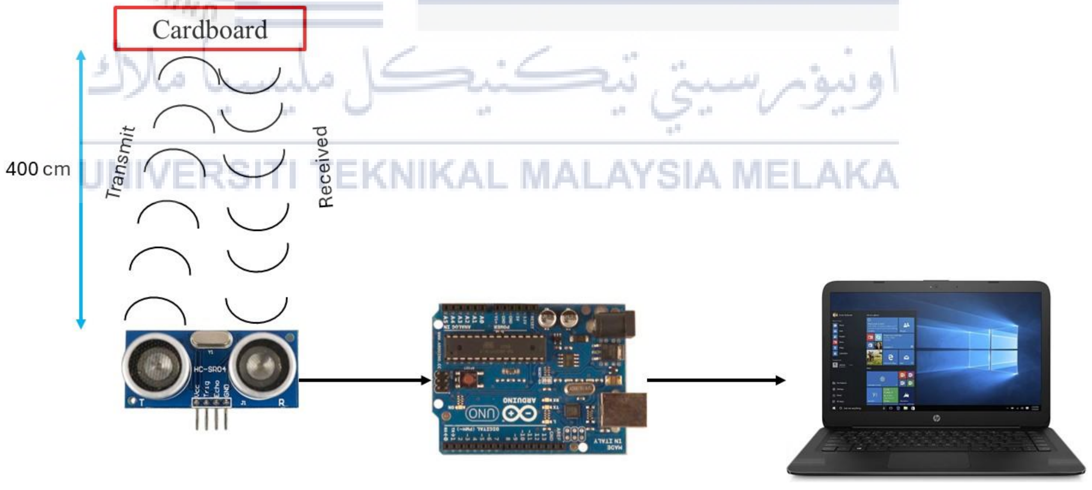


Figure 3.9: Experiment 1 setup for ultrasonic sensor

Procedures:

1. Ultrasonic sensor was connected with Arduino UNO as shown in Figure 3.8.
2. The Arduino Uno was connected with PC via USB connector.
3. Cardboard was placed at a distance of 400 cm in front of ultrasonic sensor.
4. Arduino code was written in Arduino software and uploaded to Arduino Uno.
5. Cardboard was moved towards the direction of ultrasonic sensor.
6. After uploading the code to Arduino Uno, the ultrasonic sensor reading was shown in Serial Monitor.
7. The ultrasonic sensor reading was recorded once the cardboard was detected by the ultrasonic sensor.
8. Step 3 to 7 were repeated 5 times.

3.7.1.2 Infrared (IR) sensor

Objective: To determine the maximum detection distance of infrared sensor.

Apparatus:

1. Breadboard
2. Arduino Uno
3. Infrared Sensor KY-032
4. Jumper Wires
5. Cardboard (Obstacles)
6. Computer
7. Steel Measure Tape

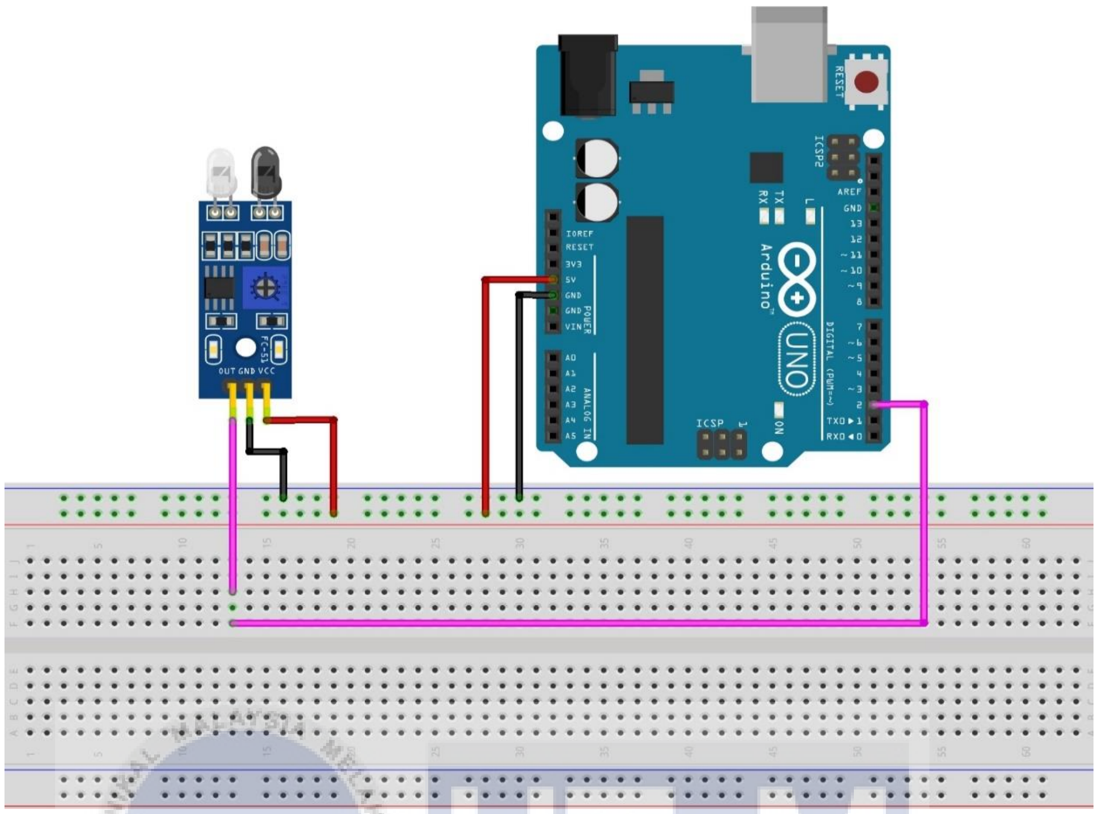


Figure 3.10: Wiring connection of infrared sensor KY-032 and Arduino Uno

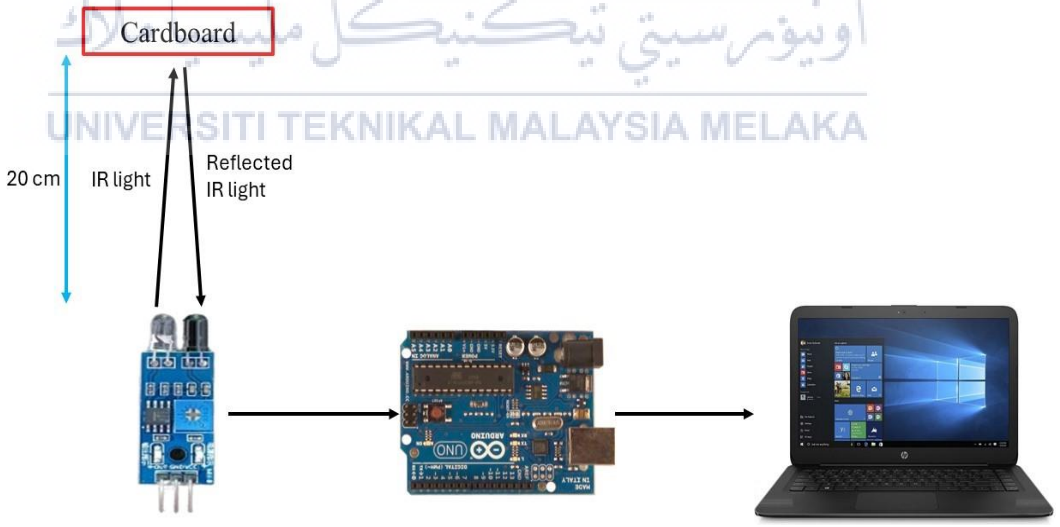


Figure 3.11: Experiment 1 setup for infrared sensor

Procedures:

1. Infrared sensor was connected with Arduino Uno as shown in Figure 3.10.
2. The Arduino Uno was connected with PC via USB connector.
3. Cardboard was placed at a distance of 20 cm from infrared sensor.
4. Arduino code was written in Arduino software and uploaded to Arduino Uno.
5. Cardboard was moved towards the direction of infrared sensor.
6. After uploading the code to Arduino Uno, the infrared sensor result was shown in Serial Monitor.
7. The infrared sensor result was recorded once the cardboard was detected by the infrared sensor.
8. Step 3 to 7 were repeated 5 times.

3.7.2 Experiment 2: Accuracy of sensor

3.7.2.1 Ultrasonic Sensor

Objective: To compare the reading of ultrasonic sensor with actual distance.

Apparatus:

1. Breadboard
2. Arduino Uno
3. Ultrasonic Sensor HC-SR04
4. Jumper Wires
5. Cardboard (Obstacles)
6. Computer
7. Steel measuring tape

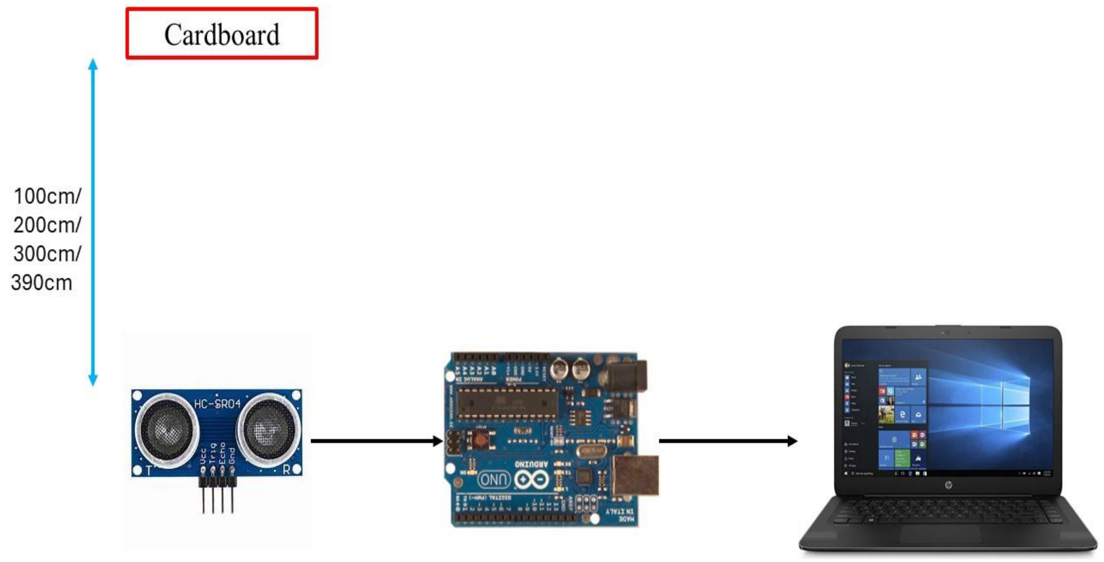


Figure 3.12: Experiment 2 setup for ultrasonic sensor

Procedures:

1. Ultrasonic sensor was connected with Arduino Uno on breadboard as shown in Figure 3.8.
2. The Arduino Uno was connected with PC via USB connector.
3. Cardboard was placed at a distance of 100 cm from ultrasonic sensor.
4. Arduino code was written in Arduino software and uploaded to Arduino Uno.
5. After uploading the code to Arduino Uno, ultrasonic sensor reading was shown in Serial Monitor.
6. The reading of ultrasonic sensor was recorded.
7. The percentage of errors between ultrasonic sensor reading and actual distance was calculated and analyzed.
8. Step 3 to 6 were repeated by replacing the distance between cardboard and ultrasonic sensor as 200 cm, 300 cm and 390cm.

3.7.2.2 Infrared Sensor

Objective: To evaluate the accuracy of infrared sensor with different incident angle.

Apparatus:

1. Breadboard
2. Arduino Uno
3. Infrared Sensor KY-032
4. Jumper Wires
5. Cardboard (Obstacles)
6. Computer
7. Steel measuring tape
8. Protractor

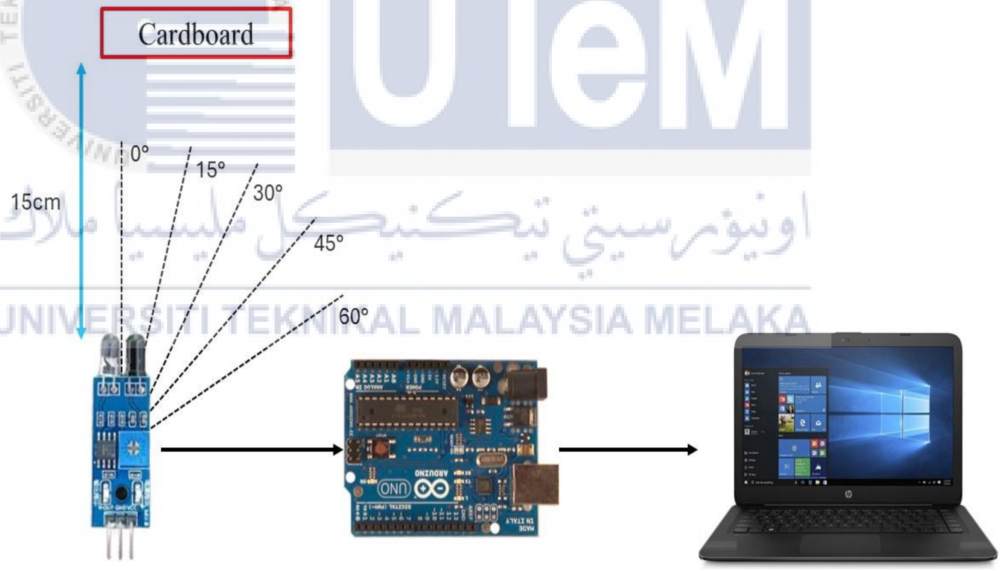


Figure 3.13: Experiment 2 setup for infrared sensor

Procedures:

1. Infrared sensor was connected with Arduino Uno on breadboard as shown in Figure 3.10.
2. The Arduino Uno was connected with PC via USB connector.
3. Cardboard was placed at a distance of 15 cm from infrared sensor.
4. The infrared sensor was located perpendicular, 0° to the cardboard.
5. Arduino code was written in Arduino software and uploaded to Arduino Uno.
6. After uploading the code to Arduino Uno, infrared sensor result was shown in Serial Monitor.
7. The infrared sensor result was recorded.
8. Step 4 to 7 were repeated by replacing the incident angle of infrared sensor as 15°, 30°, 45° and 60°.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Outline

There are 2 parts of results that will be shown in the following part. The first part is the MATLAB simulation result for the Artificial Potential Field algorithm's performance under conditions different number of obstacles, extreme position of goal and wind disturbances. The second part is the experiment result which can be used to evaluate the effectiveness of sensors. 2 experiments are carried out for ultrasonic sensors and infrared sensors respectively. The project outline for Chapter 4 is shown in Figure 4.1.

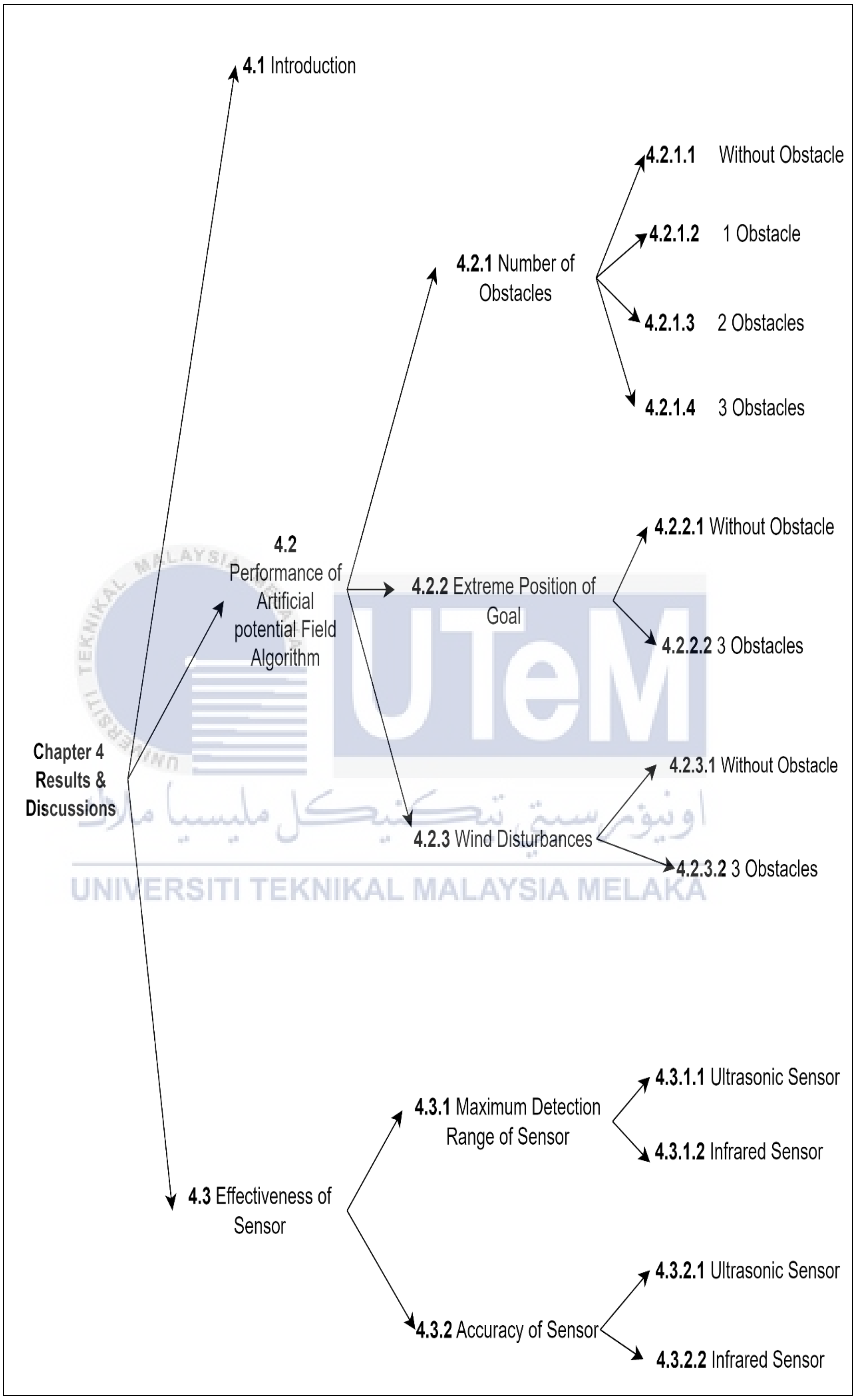


Figure 4.1 Outline for Chapter 4

4.2 Performance of Artificial Potential Field Algorithm (UAV Trajectory)

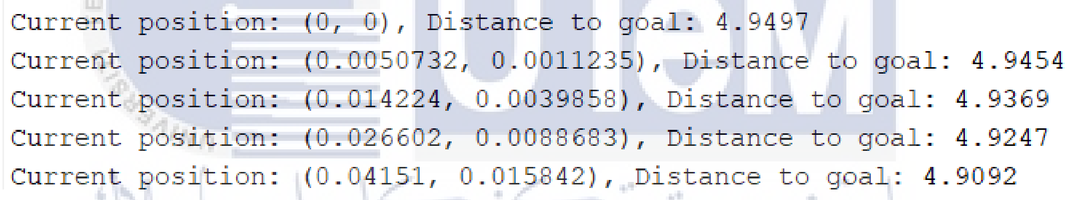
4.2.1 Number of Obstacle

In this part, the results shown are related to the 4 simulation environments which are without obstacle, 1 obstacle, 2 obstacles and 3 obstacles. The position of UAV and distance between UAV and goal position are updated from time to time in the MATLAB command window. The UAV's initial and final position for each conditions are recorded as shown in Table 4.1, Table 4.2, Table 4.3 and Table 4.4. UAV trajectory for each conditions are shown in Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.5.

4.2.1.1 Number of Obstacle: Without Obstacle

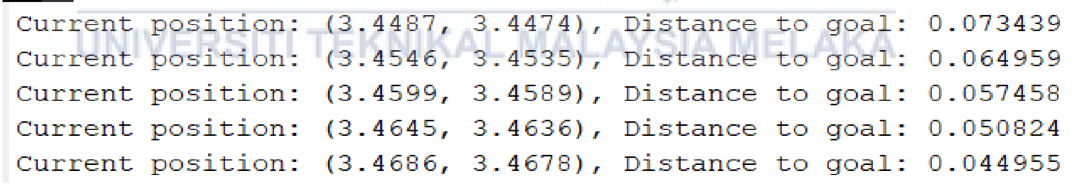
Table 4.1 UAV's position in environment without obstacle

Initial



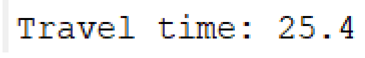
```
Current position: (0, 0), Distance to goal: 4.9497
Current position: (0.0050732, 0.0011235), Distance to goal: 4.9454
Current position: (0.014224, 0.0039858), Distance to goal: 4.9369
Current position: (0.026602, 0.0088683), Distance to goal: 4.9247
Current position: (0.04151, 0.015842), Distance to goal: 4.9092
```

Final



```
Current position: (3.4487, 3.4474), Distance to goal: 0.073439
Current position: (3.4546, 3.4535), Distance to goal: 0.064959
Current position: (3.4599, 3.4589), Distance to goal: 0.057458
Current position: (3.4645, 3.4636), Distance to goal: 0.050824
Current position: (3.4686, 3.4678), Distance to goal: 0.044955
```

Travel time



```
Travel time: 25.4
```

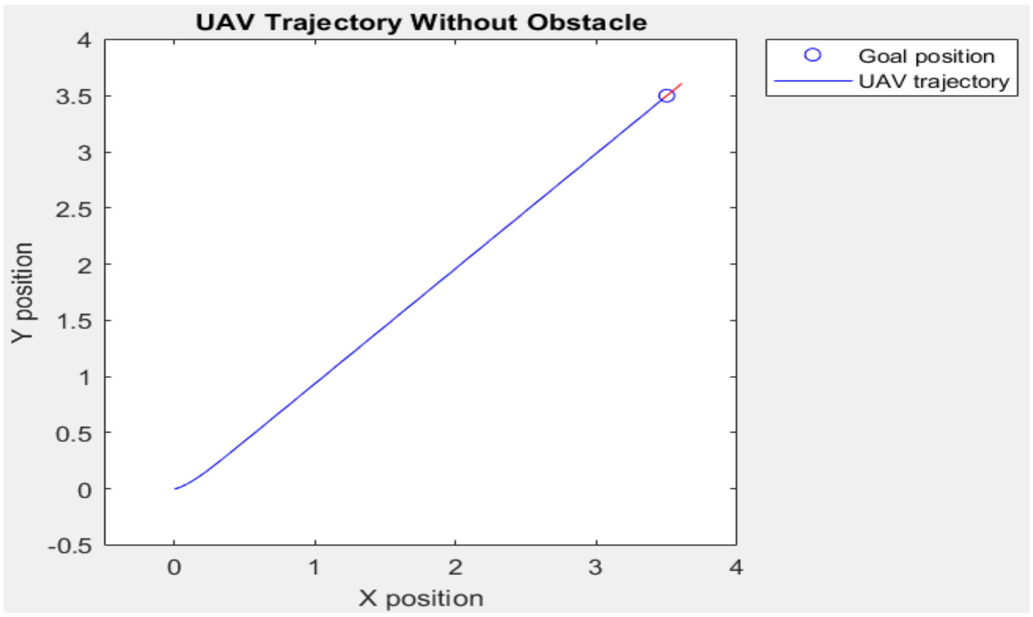


Figure 4.2 UAV trajectory without obstacle

According to Table 4.1, the travel time taken for UAV from its initial position (0,0) to goal position (3.5, 3.5) is 25.4 seconds. In Artificial Potential Field (APF) algorithm, an attractive force is generated from the goal. According to Figure 4.2, it shows that the UAV is attracted to the direction of goal due to the presence of attractive force. Since there are no obstacles, there is no repulsive field in this condition.

4.2.1.2 Number of Obstacle: 1 Obstacle

Table 4.2 UAV's position in environment with 1 obstacle

Initial

```
Current position: (0, 0), Distance to goal: 4.9497
Current position: (0.001402, 0.00058808), Distance to goal: 4.9483
Current position: (0.0061943, 0.0031597), Distance to goal: 4.9431
Current position: (0.013267, 0.0077459), Distance to goal: 4.9349
Current position: (0.021742, 0.014166), Distance to goal: 4.9244
```

Final

```
Current position: (3.4356, 3.4551), Distance to goal: 0.07856
Current position: (3.443, 3.4603), Distance to goal: 0.069488
Current position: (3.4496, 3.4648), Distance to goal: 0.061464
Current position: (3.4554, 3.4689), Distance to goal: 0.054367
Current position: (3.4606, 3.4725), Distance to goal: 0.048089
```

Travel time

```
Travel time: 27.1
```

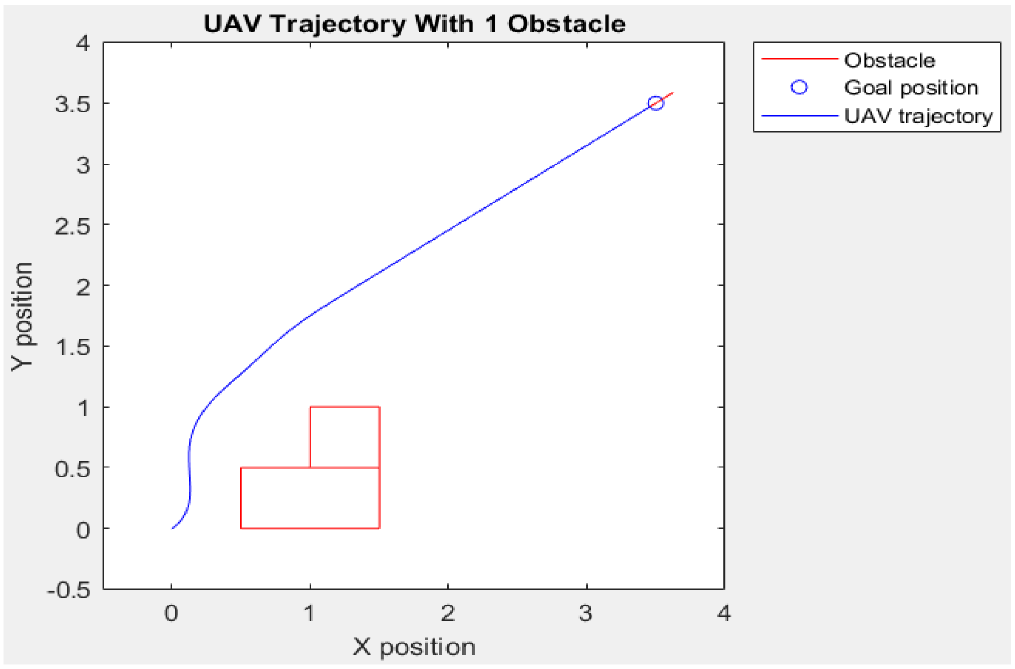


Figure 4.3 UAV trajectory with 1 obstacle

According to Table 4.2, the travel time taken for UAV from its initial position (0,0) to goal position (3.5, 3.5) is 27.1 seconds. Due to the presence of obstacle, UAV takes longer time to reach the goal position if compared to previous. In this condition, an attractive force and repulsive force is exerted on UAV. According to Figure 4.3, the initial position of UAV is close to the obstacle, hence a repulsive force is exerted on the UAV. Hence, UAV successfully avoids colliding with obstacle. Simultaneously, an attractive force is exerted on the UAV to guide it toward the goal. As UAV moves away from obstacle or close to goal position, repulsion potential field gradually decrease. The repulsion potential field becomes 0 when the UAV has reached the goal.

4.2.1.3 Number of Obstacle: 2 Obstacles

Table 4.3 UAV's position in environment with 2 obstacles

Initial

```
Current position: (0, 0), Distance to goal: 4.9497
Current position: (0.001402, 0.00058808), Distance to goal: 4.9483
Current position: (0.0061943, 0.0031597), Distance to goal: 4.9431
Current position: (0.013267, 0.0077459), Distance to goal: 4.9349
Current position: (0.021742, 0.014166), Distance to goal: 4.9244
```

Final

```
Current position: (3.466, 3.4355), Distance to goal: 0.072869
Current position: (3.4699, 3.443), Distance to goal: 0.064455
Current position: (3.4734, 3.4496), Distance to goal: 0.057012
Current position: (3.4765, 3.4554), Distance to goal: 0.050429
Current position: (3.4792, 3.4605), Distance to goal: 0.044606
```

Travel time

```
Travel time: 29.8
```

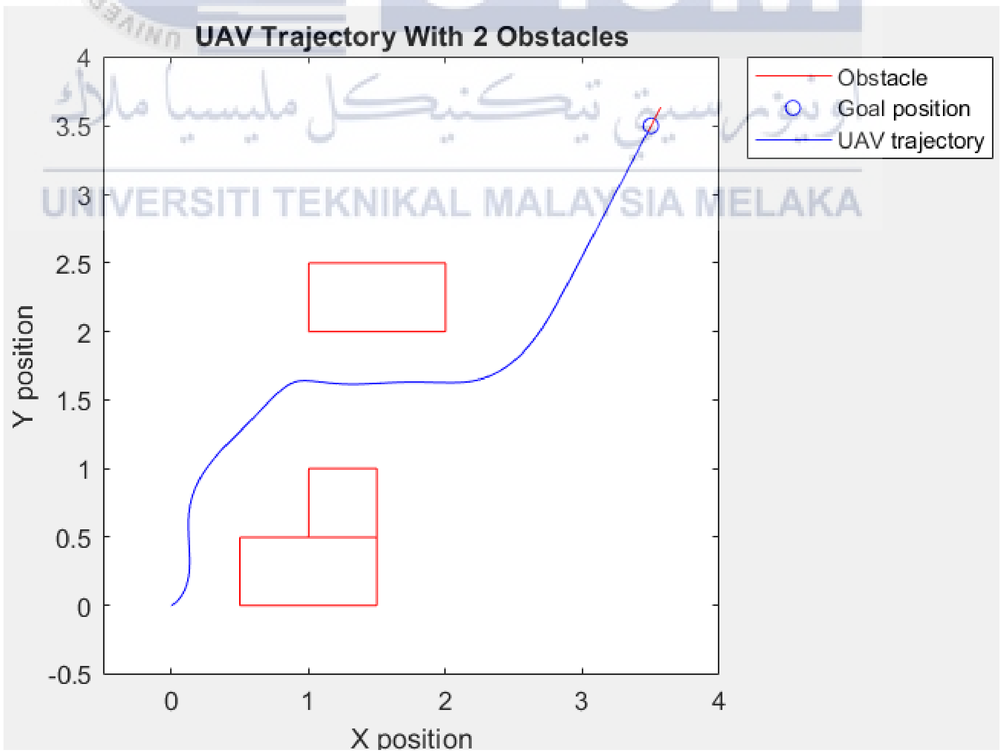


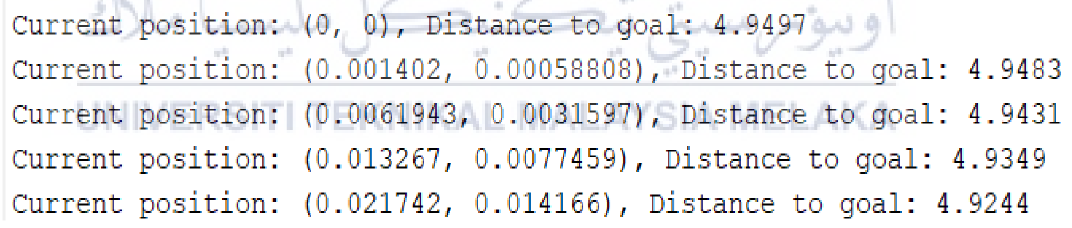
Figure 4.4 UAV trajectory with 2 obstacles

According to Table 4.3, the travel time taken for UAV from its initial position (0,0) to goal position (3.5, 3.5) is 29.8 seconds. Due to the higher number of obstacle, UAV takes longer time to reach the goal position if compared to previous. In this condition, an attractive force and repulsive force is exerted on UAV. According to Figure 4.4, there are 2 obstacles in the simulation environment. The overall repulsion potential field in the environment is higher than previous. The UAV's trajectory is altered by considering the repulsion potential field by both obstacles in simulation environment. The initial position of UAV is close to the first obstacle, hence a repulsive force is exerted on the UAV. Hence, UAV successfully avoids colliding with first obstacle. After that, UAV's trajectory is altered again due to the presence of second obstacle. Simultaneously, an attractive force is exerted on the UAV to guide it toward the goal. As UAV moves away from obstacle or close to goal position, repulsion potential field gradually decrease. The repulsion potential field become 0 when the UAV has reached the goal.

4.2.1.4 Number of Obstacle: 3 Obstacles

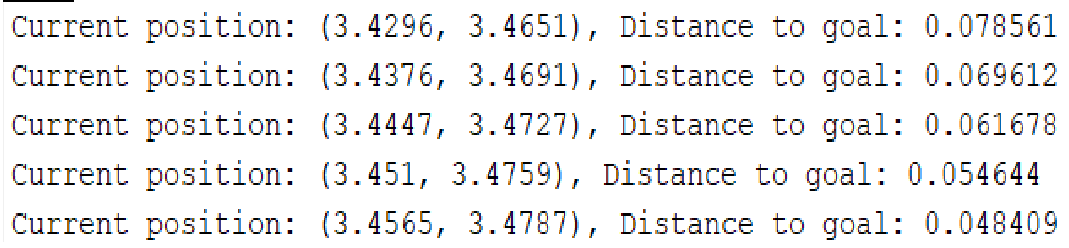
Table 4.4 UAV's position in environment with 3 obstacles

Initial



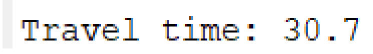
```
Current position: (0, 0), Distance to goal: 4.9497
Current position: (0.001402, 0.00058808), Distance to goal: 4.9483
Current position: (0.0061943, 0.0031597), Distance to goal: 4.9431
Current position: (0.013267, 0.0077459), Distance to goal: 4.9349
Current position: (0.021742, 0.014166), Distance to goal: 4.9244
```

Final



```
Current position: (3.4296, 3.4651), Distance to goal: 0.078561
Current position: (3.4376, 3.4691), Distance to goal: 0.069612
Current position: (3.4447, 3.4727), Distance to goal: 0.061678
Current position: (3.451, 3.4759), Distance to goal: 0.054644
Current position: (3.4565, 3.4787), Distance to goal: 0.048409
```

Travel time



```
Travel time: 30.7
```

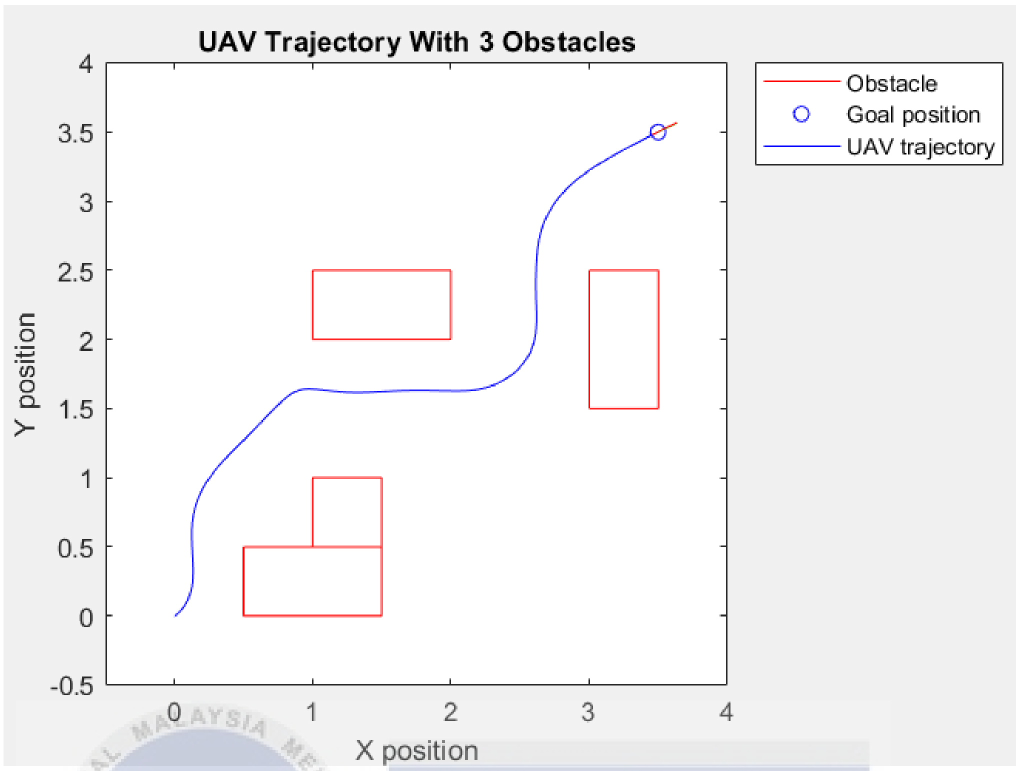


Figure 4.5 UAV trajectory with 3 obstacles

According to Table 4.4, the travel time taken for UAV from its initial position (0,0) to goal position (3.5, 3.5) is 30.7 seconds. Due to the higher number of obstacle, UAV takes longest time to reach the goal position if compared to previous. In this condition, an attractive force and repulsive force is exerted on UAV. According to Figure 4.5, there are 3 obstacles in the simulation environment. The overall repulsion potential field in the environment is the combination repulsion field from 3 obstacles. The UAV's trajectory is altered by considering the repulsion potential field by 3 obstacles in simulation environment. The initial position of UAV is close to the first obstacle, hence a repulsive force is exerted on the UAV. Therefore, UAV successfully avoids colliding with first obstacle. After that, UAV's trajectory is altered again due to the presence of second and third obstacle. Simultaneously, an attractive force is exerted on the UAV to guide it toward the goal. As UAV close to goal position, repulsion potential field gradually decrease. The repulsion potential field become 0 when the UAV has reached the goal.

According to the simulation result, the accuracy of UAV's positioning system can be determined by applying Root Mean Square Error as shown in equation (4-1).

$$\textit{Root Mean Square Error, RMSE}$$

$$= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x_{gi})^2 + (y_i - y_{gi})^2} \quad (4\text{-}1)$$

Table 4.5 Notation and its description for equation (4-1)

Notation	Description
x_i	UAV’s position in x-direction
x_{gi}	Goal position in x-direction
y_i	UAV’s position in y-direction
y_{gi}	Goal position in y-direction

Table 4.6 RMSE value for different number of obstacle

	i	UAV’s x-position x_i	UAV's y-position y_i	Goal's x-position x_{gi}	Goal's y-position y_{gi}	RMSE
Without Obstacle	1	3.4487	3.4474	3.5	3.5	0.05921
	2	3.4546	3.4535	3.5	3.5	
	3	3.4599	3.4589	3.5	3.5	
	4	3.4645	3.4636	3.5	3.5	
	5	3.4686	3.4678	3.5	3.5	
1 Obstacle	1	3.4356	3.4551	3.5	3.5	0.06330
	2	3.4430	3.4603	3.5	3.5	
	3	3.4496	3.4648	3.5	3.5	
	4	3.4554	3.4689	3.5	3.5	
	5	3.4606	3.4725	3.5	3.5	
2 Obstacles	1	3.4660	3.4355	3.5	3.5	0.05874
	2	3.4699	3.4430	3.5	3.5	
	3	3.4734	3.4496	3.5	3.5	
	4	3.4765	3.4554	3.5	3.5	
	5	3.4792	3.4605	3.5	3.5	
3 Obstacles	1	3.4296	3.4651	3.5	3.5	0.06349
	2	3.4376	3.4691	3.5	3.5	
	3	3.4447	3.4727	3.5	3.5	
	4	3.4510	3.4759	3.5	3.5	
	5	3.4565	3.4787	3.5	3.5	

According to Table 4.6, RMSE value is calculated for 4 simulation environments, without obstacles, 1 obstacle, 2 obstacle and 3 obstacles. The deviation of UAV from goal position can be determined through RMSE value. A lower RMSE value shows that the actual position of UAV is close to the goal position while a higher RMSE value shows that there is a discrepancy between the UAV's position and goal position. As the number of obstacles increases, the RMSE value increases. Higher number of obstacles generate more repulsive force towards UAV. Hence, UAV experiences greater deviation from its original path to avoid collision with obstacles. The deviations lead to higher value of RMSE.

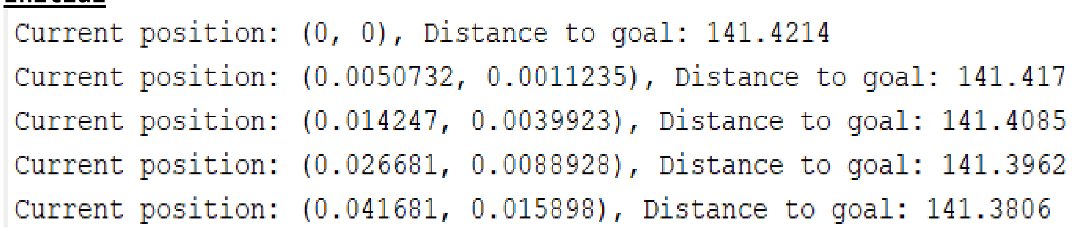
4.2.2 Extreme Position of Goal

In this part, the results shown are related to the 2 simulation environments which are without obstacle and with obstacles. The position of UAV and distance between UAV and goal position are updated from time to time in the MATLAB command window. The UAV's initial and final position for each conditions are recorded as shown in Table 4.7 and Table 4.8. UAV trajectory for each conditions are shown in Figure 4.6 and Figure 4.7.

4.2.2.1 Extreme Position of Goal: Without obstacle

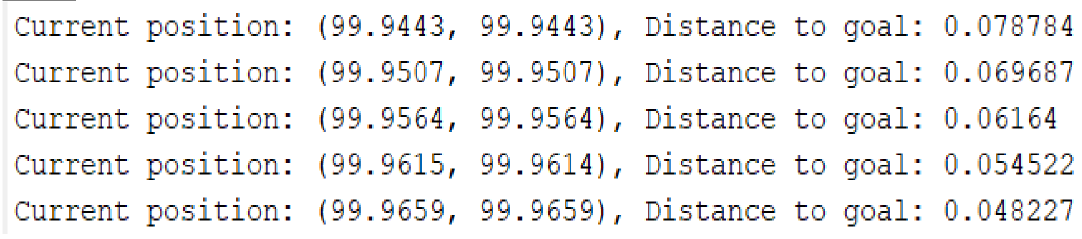
Table 4.7 UAV position in an obstacle free environment for extreme goal positions

Initial



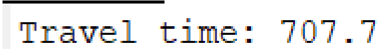
```
Current position: (0, 0), Distance to goal: 141.4214
Current position: (0.0050732, 0.0011235), Distance to goal: 141.417
Current position: (0.014247, 0.0039923), Distance to goal: 141.4085
Current position: (0.026681, 0.0088928), Distance to goal: 141.3962
Current position: (0.041681, 0.015898), Distance to goal: 141.3806
```

Final



```
Current position: (99.9443, 99.9443), Distance to goal: 0.078784
Current position: (99.9507, 99.9507), Distance to goal: 0.069687
Current position: (99.9564, 99.9564), Distance to goal: 0.06164
Current position: (99.9615, 99.9614), Distance to goal: 0.054522
Current position: (99.9659, 99.9659), Distance to goal: 0.048227
```

Travel time



```
Travel time: 707.7
```

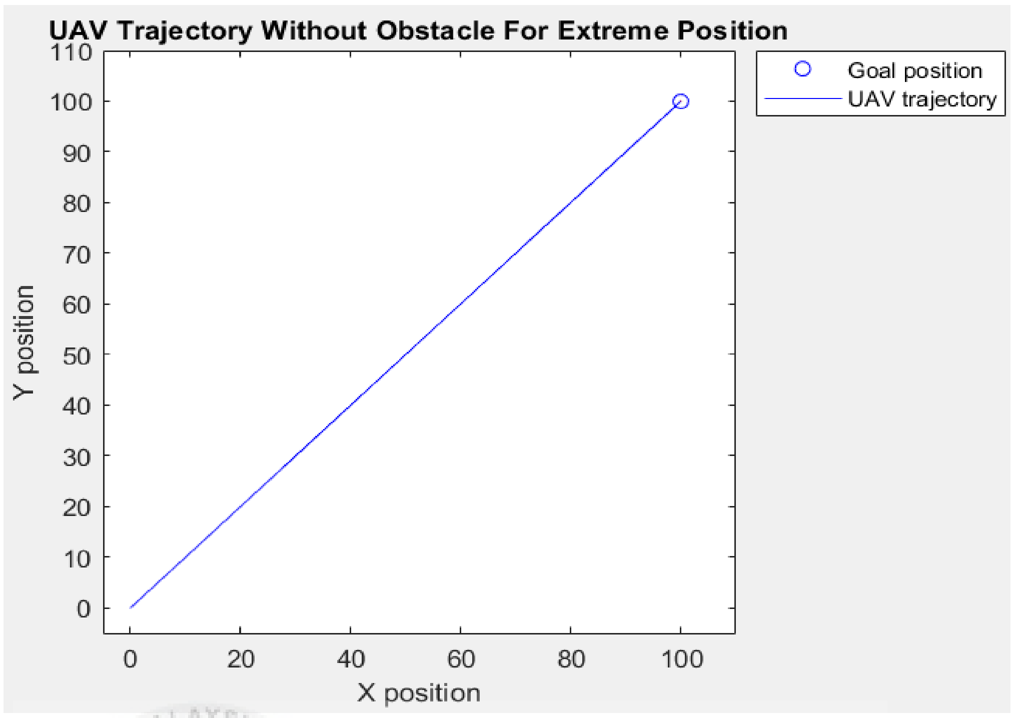


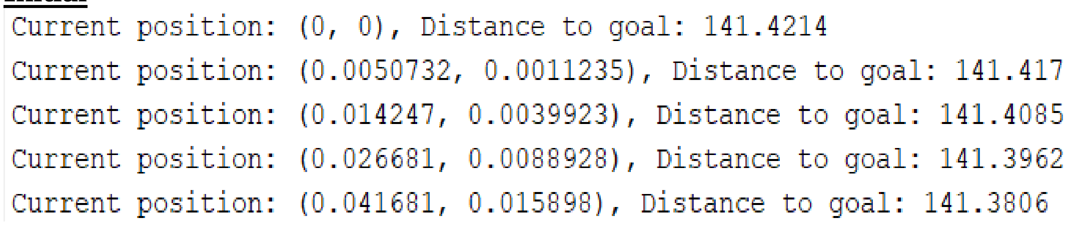
Figure 4.6 UAV trajectory without obstacle for extreme position of goal

According to Table 4.7, the travel time taken for UAV from its initial position (0,0) to goal position (100, 100) is 707.7 seconds which is around 11.795 minutes. In the Artificial Potential Field (APF) algorithm, the attractive force exerted by the goal becomes stronger as the distance between the UAV and the goal increases. This means that the APF algorithm is suitable for extreme position of goal. According to Figure 4.6, it shows that the UAV is attracted to the direction of goal due to the presence of attractive force. The attractive force is stronger when the UAV is far away from goal position. As UAV approaches the goal, the attractive force acting on UAV decreases. Hence, UAV can stopped precisely at the goal position. Since there are no obstacles in the simulation environment, there is no repulsive field in this condition.

4.2.2.2 Extreme Position of Goal: 3 Obstacles

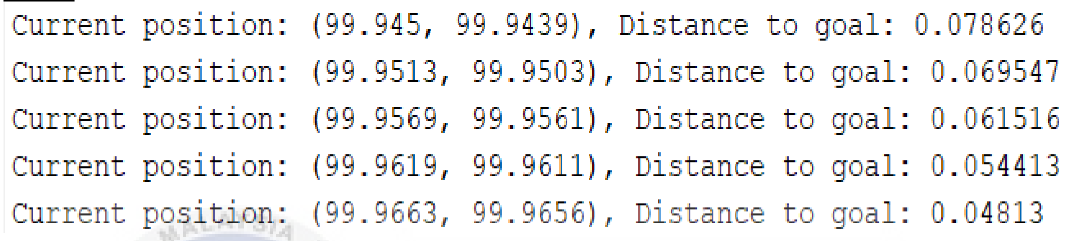
Table 4.8 UAV position in environment with obstacle for extreme goal position

Initial



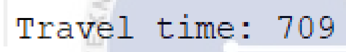
```
Current position: (0, 0), Distance to goal: 141.4214
Current position: (0.0050732, 0.0011235), Distance to goal: 141.417
Current position: (0.014247, 0.0039923), Distance to goal: 141.4085
Current position: (0.026681, 0.0088928), Distance to goal: 141.3962
Current position: (0.041681, 0.015898), Distance to goal: 141.3806
```

Final



```
Current position: (99.945, 99.9439), Distance to goal: 0.078626
Current position: (99.9513, 99.9503), Distance to goal: 0.069547
Current position: (99.9569, 99.9561), Distance to goal: 0.061516
Current position: (99.9619, 99.9611), Distance to goal: 0.054413
Current position: (99.9663, 99.9656), Distance to goal: 0.04813
```

Travel time



```
Travel time: 709
```

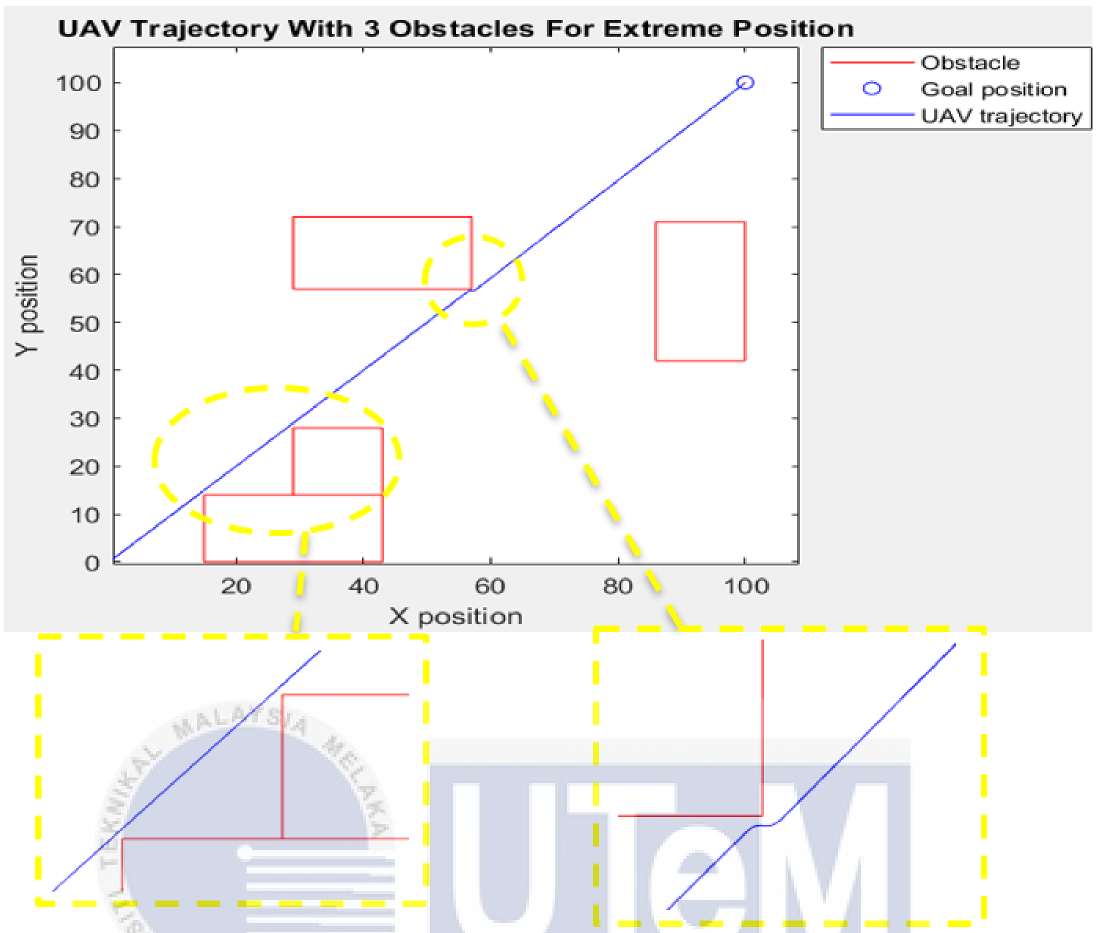


Figure 4.7 UAV trajectory with 3 obstacles for extreme position of goal

According to Table 4.8, the travel time taken for UAV from its initial position (0,0) to goal position (100, 100) is 709 seconds which is around 11.817 minutes. According to Figure 4.7, an attractive force and repulsive force is exerted on UAV. There are 3 obstacles in the simulation environment. The attractive force is stronger when the UAV is far away from goal position. To reach the goal position, UAV must consider the total repulsion potential field from 3 obstacles. By observing the simulation result, it shows that UAV successfully avoid colliding with obstacles due to the presence of repulsive force. When UAV is repelled away from obstacle, there is an attractive force which attracts UAV to move towards the direction of goal. As UAV approaches the goal, the attractive force acting on UAV decreases. Finally,UAV can stopped at the goal position.

Table 4.9 RMSE value for extreme position of goal

	i	UAV's x-position x_i	UAV's y-position y_i	Goal's x-position x_{gi}	Goal's y-position y_{gi}	RMSE
Without Obstacle	1	99.9443	99.9443	100	100	0.06351
	2	99.9507	99.9507	100	100	
	3	99.9564	99.9564	100	100	
	4	99.9615	99.9614	100	100	
	5	99.9659	99.9659	100	100	
3 Obstacles	1	99.9450	99.9439	100	100	0.06338
	2	99.9513	99.9503	100	100	
	3	99.9569	99.9561	100	100	
	4	99.9619	99.9611	100	100	
	5	99.9663	99.9656	100	100	

According to Table 4.9, RMSE value is calculated for 2 simulation environments, without obstacles and 3 obstacles when the goal is located at an extreme position. The deviation of UAV from goal position can be determined through RMSE value. A lower RMSE value shows that the actual position of UAV is close to the goal position while a higher RMSE value shows that there is a discrepancy between the UAV’s position and goal position. When the goal is located at an extreme position, UAV needs to take a long path to reach the goal. As UAV navigates over a long distance, small deviations are accumulated over a long path. Hence, the RMSE value when goal is located at (100,100) is higher than goal at (3.5, 3.5). The RMSE value for environment without obstacles is higher than environment with obstacle. This is because APF algorithm can be used in environments without obstacles. However, it cannot provide the most optimal path for the environment without obstacles. The working principle of APF algorithm is to generate an ideal path by balancing the attractive and repulsive force. In an environment without obstacles, it can cause UAV to overshoot the goal position. Hence, it can lead to higher RMSE value.

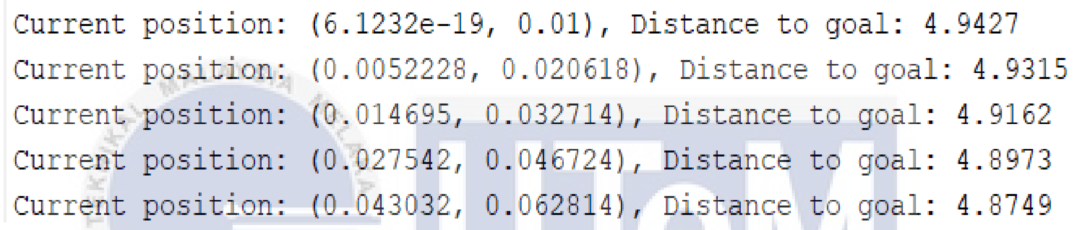
4.2.3 Wind Disturbance

In this part, the results shown are related to the 2 simulation environments which are without obstacle and with obstacles. The position of UAV and distance between UAV and goal position are updated from time to time in the MATLAB command window. The UAV's initial and final position for each conditions are recorded as shown in Table 4.10 and Table 4.11. UAV trajectory for each conditions are shown in Figure 4.8 and Figure 4.9.

4.2.3.1 Wind Disturbances: Without Obstacles

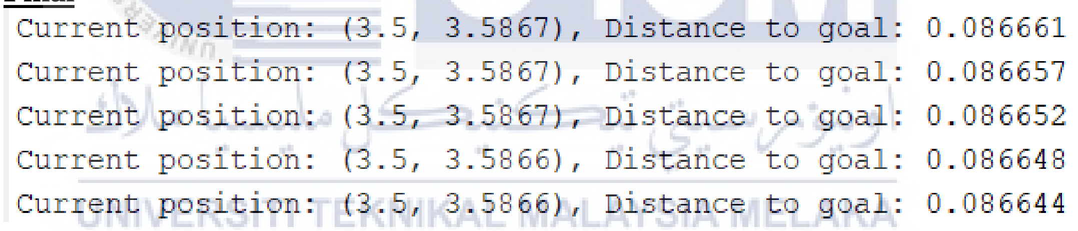
Table 4.10 UAV position in an obstacle free environment with wind disturbance

Initial



```
Current position: (6.1232e-19, 0.01), Distance to goal: 4.9427
Current position: (0.0052228, 0.020618), Distance to goal: 4.9315
Current position: (0.014695, 0.032714), Distance to goal: 4.9162
Current position: (0.027542, 0.046724), Distance to goal: 4.8973
Current position: (0.043032, 0.062814), Distance to goal: 4.8749
```

Final



```
Current position: (3.5, 3.5867), Distance to goal: 0.086661
Current position: (3.5, 3.5867), Distance to goal: 0.086657
Current position: (3.5, 3.5867), Distance to goal: 0.086652
Current position: (3.5, 3.5866), Distance to goal: 0.086648
Current position: (3.5, 3.5866), Distance to goal: 0.086644
```

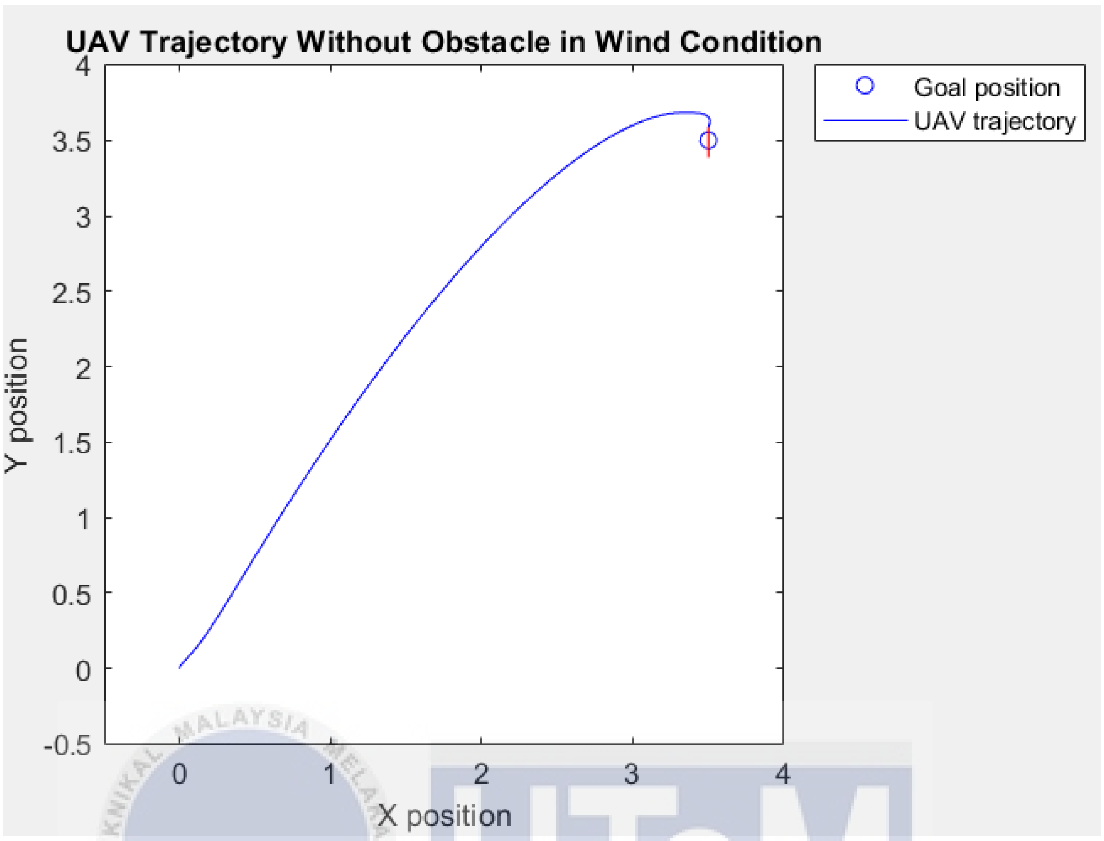


Figure 4.8 UAV trajectory without obstacle in wind condition

In practical application, some inherent uncertainties may arise during the flight of UAV. The inherent uncertainties such as wind can significantly affect the UAV's flight path. This is because wind can cause deviations from the intended path. In this simulation environment, there is a wind blowing from the south to the north. UAV needs to navigate from its initial position (0, 0) to the goal position (3.5, 3.5). According to Figure 4.8, UAV is attracted to the direction of goal due to the presence of attractive force. As UAV moves to the goal's direction, it can be seen that UAV deviates from its intented path due to the presence of wind. The Artificial Potential Field (APF) algorithm detects the deviation and increases the attractive force towards the goal to counteract the wind. Finally, UAV reaches the goal's position successfully.

4.2.3.2 Wind Disturbances: 3 Obstacles

Table 4.11 UAV position in an environment with 3 obstacles and wind disturbances

Initial

```
Current position: (6.1232e-19, 0.01), Distance to goal: 4.9427
Current position: (1.2246e-18, 0.02), Distance to goal: 4.9356
Current position: (0.0015686, 0.03046), Distance to goal: 4.9271
Current position: (0.0067119, 0.042613), Distance to goal: 4.9149
Current position: (0.014225, 0.056632), Distance to goal: 4.8997
```

Final

```
Current position: (3.5001, 3.5867), Distance to goal: 0.086742
Current position: (3.5001, 3.5867), Distance to goal: 0.086727
Current position: (3.5, 3.5867), Distance to goal: 0.086715
Current position: (3.5, 3.5867), Distance to goal: 0.086705
Current position: (3.5, 3.5867), Distance to goal: 0.086697
```

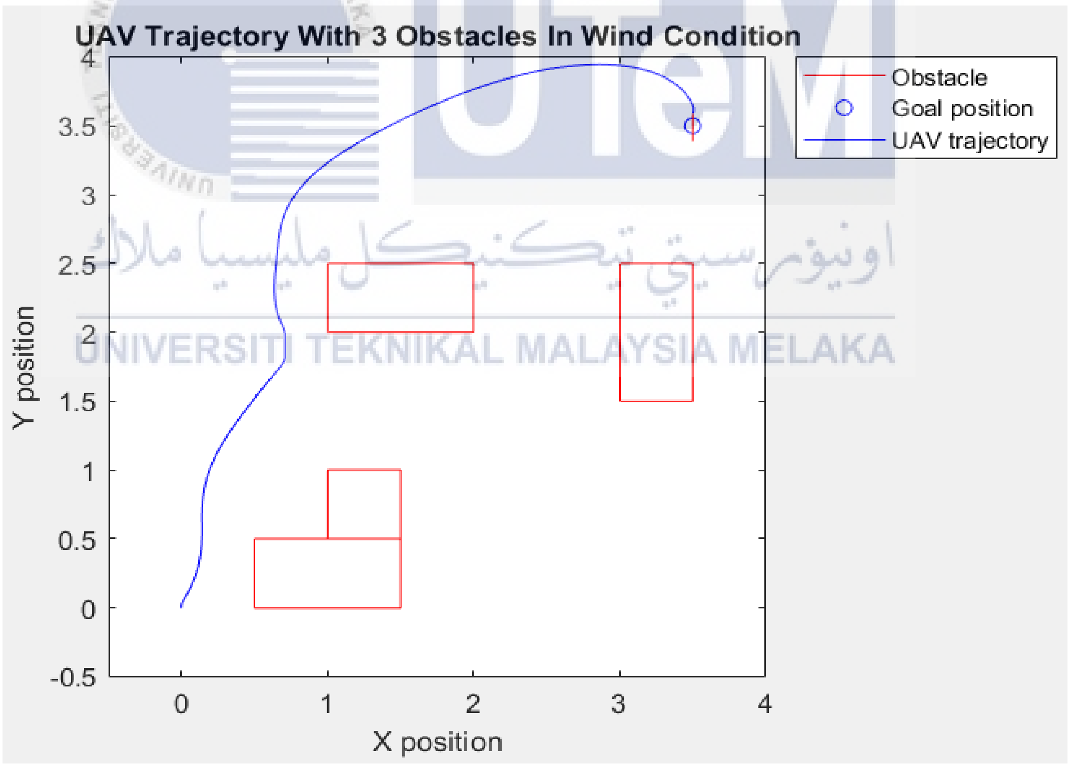


Figure 4.9 UAV trajectory with 3 obstacles in wind condition

According to Figure 4.9, there is a wind blowing from the south to the north and there are 3 obstacles present in the simulation environment. Hence, UAV is required to navigate from its initial position (0,0) towards its designated goal position (3.5, 3,5) while simultaneously avoiding obstacles in its path. At the same time, UAV needs to counteract wind disturbances. When UAV starts to navigate, there is a repulsive force exerted on UAV to repel the UAV away from first obstacle. From the previous simulation environment without wind disturbances, UAV navigates between the first and second obstacles. However, in this condition, the trajectory of UAV has been altered due to the presence of wind disturbance. This is because the wind can cause deviation of UAV's trajectory from its intended path. Finally, a path as shown in Figure 4.9 is generated by APF algorithm after considering the presence of 3 obstacles and wind disturbances.

Table 4.12 RMSE value for wind disturbances

	i	UAV's x-position x_i	UAV's y-position y_i	Goal's x-position x_{gi}	Goal's y-position y_{gi}	RMSE
Without Obstacle	1	3.5000	3.5867	3.5	3.5	0.08666
	2	3.5000	3.5867	3.5	3.5	
	3	3.5000	3.5867	3.5	3.5	
	4	3.5000	3.5866	3.5	3.5	
	5	3.5000	3.5866	3.5	3.5	
3 Obstacles	1	3.5001	3.5867	3.5	3.5	0.08670
	2	3.5001	3.5867	3.5	3.5	
	3	3.5000	3.5867	3.5	3.5	
	4	3.5000	3.5867	3.5	3.5	
	5	3.5000	3.5867	3.5	3.5	

According to Table 4.12, RMSE value is calculated for 2 simulation environments, without obstacles and 3 obstacles with wind disturbances. The deviation of UAV from goal position can be determined through RMSE value. A lower RMSE value shows that the actual position of UAV is close to the goal position while a higher RMSE value shows that there is a discrepancy between the UAV's position and goal position. With wind disturbances, the RMSE value is higher than the condition without wind disturbances. Wind can cause UAV to drift off its intended path making it difficult to maintain a stable flight. Therefore, APF algorithm is designed to counteract wind effects. From the simulation result, it shows that UAV reaches the goal position successfully despite the wind condition.

4.3 Effectiveness of Sensor

4.3.1 Experiment 1: Maximum Detection Range of Sensor

This experiment is carried out to determine the maximum detection range of ultrasonic sensors and infrared sensors. The cardboard is moved towards the direction of the sensor. Once the sensor detects the presence of cardboard, that is the maximum detection range of sensor. The results are taken from serial monitor of Arduino software. To increase the accuracy of the result, 5 readings of maximum detection range of sensors are taken. After that, the average maximum detection range is calculated by using the equation (4-2). Lastly, a graph is generated based on the results obtained.

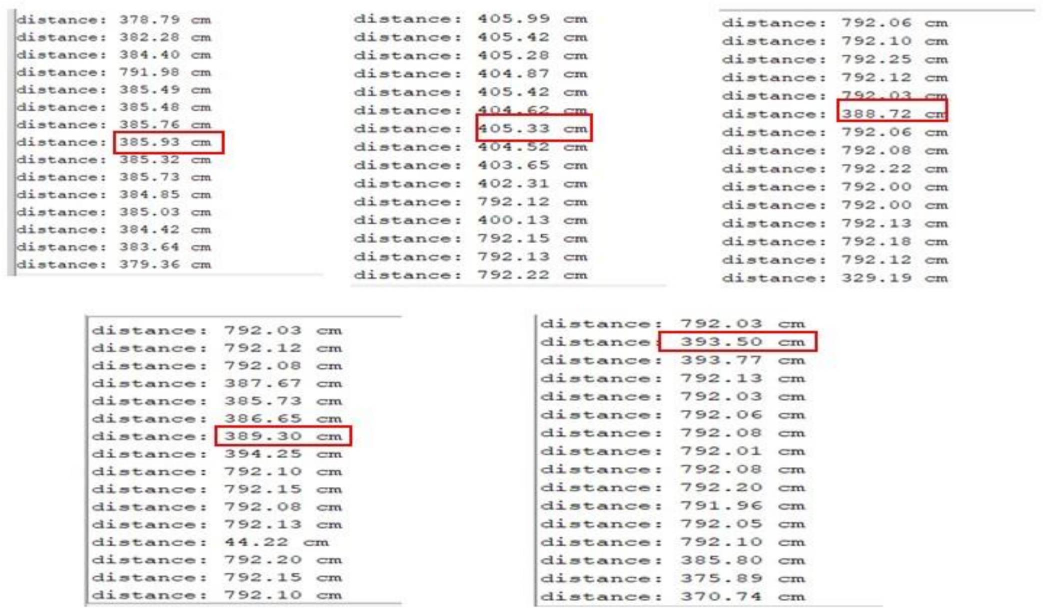
$$\textit{Average maximum detection range}$$

$$= \frac{\textit{Sum of maximum detection range, } D_T}{\textit{total number of reading taken}} \qquad (4\text{-}2)$$

$$= \frac{D_1 + D_2 + D_3 + D_4 + D_5}{5}$$

4.3.1.1 Ultrasonic Sensor

The maximum detection range of ultrasonic sensor is obtained from serial monitor of Arduino IDE and recorded in Table 4.13. The average maximum detection range is calculated by applying equation (4-2). After that, a graph is generated as shown in Figure 4.11 based on the reading obtained from Table 4.13.



```
distance: 378.79 cm
distance: 382.28 cm
distance: 384.40 cm
distance: 791.98 cm
distance: 385.49 cm
distance: 385.48 cm
distance: 385.76 cm
distance: 385.93 cm
distance: 385.32 cm
distance: 385.73 cm
distance: 384.85 cm
distance: 385.03 cm
distance: 384.42 cm
distance: 383.64 cm
distance: 379.36 cm
```

```
distance: 405.99 cm
distance: 405.42 cm
distance: 405.28 cm
distance: 404.87 cm
distance: 405.42 cm
distance: 404.62 cm
distance: 405.33 cm
distance: 404.52 cm
distance: 403.65 cm
distance: 402.31 cm
distance: 792.12 cm
distance: 400.13 cm
distance: 792.15 cm
distance: 792.13 cm
distance: 792.22 cm
```

```
distance: 792.06 cm
distance: 792.10 cm
distance: 792.25 cm
distance: 792.12 cm
distance: 792.03 cm
distance: 388.72 cm
distance: 792.06 cm
distance: 792.08 cm
distance: 792.22 cm
distance: 792.00 cm
distance: 792.00 cm
distance: 792.13 cm
distance: 792.18 cm
distance: 792.12 cm
distance: 329.19 cm
```

```
distance: 792.03 cm
distance: 792.12 cm
distance: 792.08 cm
distance: 387.67 cm
distance: 385.73 cm
distance: 386.65 cm
distance: 389.30 cm
distance: 394.25 cm
distance: 792.10 cm
distance: 792.15 cm
distance: 792.08 cm
distance: 792.13 cm
distance: 44.22 cm
distance: 792.20 cm
distance: 792.15 cm
distance: 792.10 cm
```

```
distance: 792.03 cm
distance: 393.50 cm
distance: 393.77 cm
distance: 792.13 cm
distance: 792.03 cm
distance: 792.06 cm
distance: 792.08 cm
distance: 792.01 cm
distance: 792.08 cm
distance: 792.20 cm
distance: 791.96 cm
distance: 792.05 cm
distance: 792.10 cm
distance: 385.80 cm
distance: 375.89 cm
distance: 370.74 cm
```

Figure 4.10 Result of maximum detection range of ultrasonic sensor

Table 4.13 Maximum detection range of ultrasonic sensor

	Ultrasonic sensor					
	1	**2**	**3**	**4**	**5**	**Average**
Maximum detection range, D (cm)	385.93	405.33	388.72	389.30	393.50	392.56

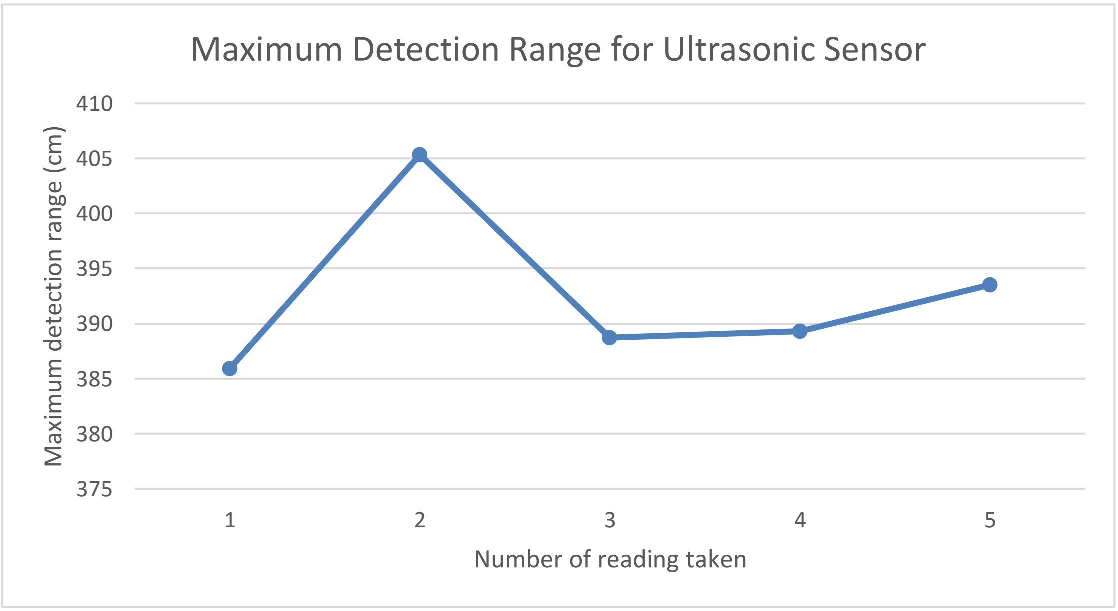
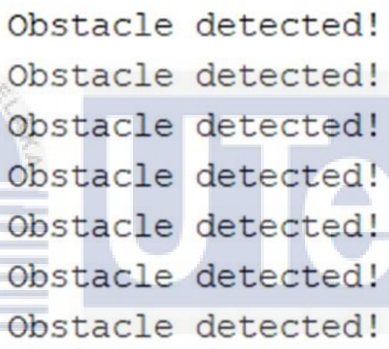


Figure 4.11 Graph of maximum detection range for ultrasonic sensor

From Table 4.13, it shows that the maximum detection range for ultrasonic sensors is between 385.93cm to 405.33cm. The average maximum detection range for the ultrasonic sensor is 392.56 cm.

4.3.1.2 Infrared Sensor

According to the result as shown in Figure 4.12, "Obstacle detected" is shown on the serial monitor for each reading. The maximum detection range of infrared sensor is measured and recorded in Table 4.14. The average maximum detection range is calculated by applying equation (4-2). After that, a graph is generated as shown in Figure 4.13 based on the reading obtained from Table 4.14.



```
Obstacle detected!
Obstacle detected!
Obstacle detected!
Obstacle detected!
Obstacle detected!
Obstacle detected!
Obstacle detected!
```

Figure 4.12 Result of maximum detection range of infrared sensor

Table 4.14 Maximum detection range of infrared sensor

	Infrared sensor					
	1	**2**	**3**	**4**	**5**	**Average**
Maximum detection range, D (cm)	16.0	15.8	15.9	15.5	16.1	15.86

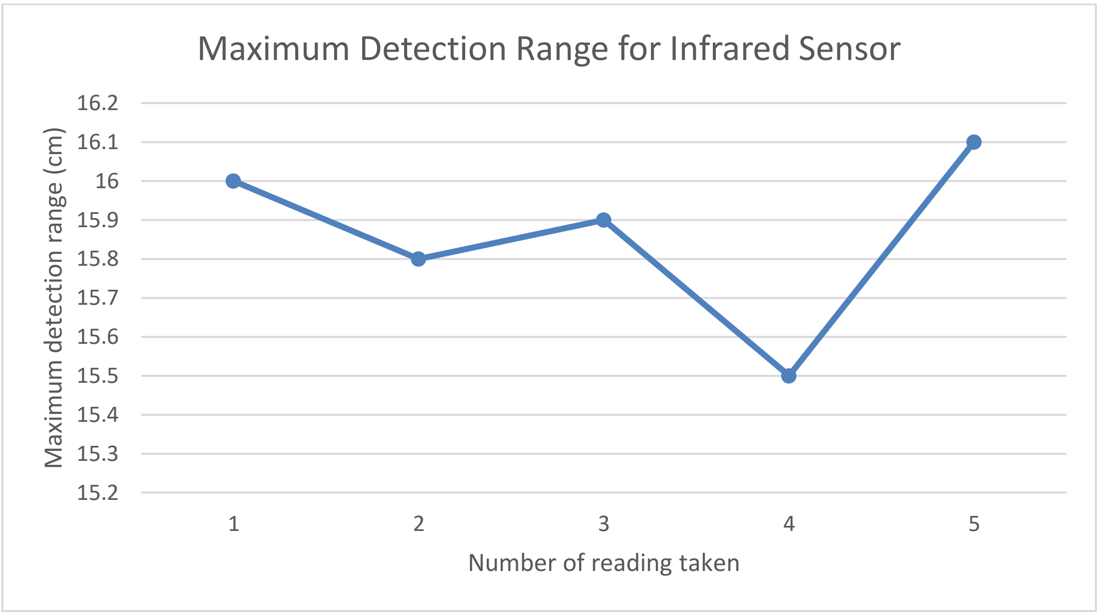


Figure 4.13 Graph of maximum detection range for infrared sensor

From Table 4.14, it shows that the maximum detection range for infrared sensors is between 15.8cm to 16.1cm. The average maximum detection range for infrared sensors is 15.86cm.

By comparing maximum detection range for both sensors, ultrasonic sensors are more suitable to detect obstacles further distance from the sensor. The early detection of obstacle gives UAV more time to react and plan a safe path around the obstacle.

4.3.2 Experiment 2: Accuracy of sensor

The experiment is carried out to determine the accuracy of ultrasonic and infrared sensors. The cardboard is located at a certain distance from the sensors. The sensor readings are shown in the Serial Monitor of Arduino software.

4.3.2.1 Ultrasonic Sensor

The ultrasonic sensor is located at 4 different distances from the cardboard which are 100cm, 200cm, 300cm, and 390cm. Since the maximum detection distance obtained from previous experiment of ultrasonic sensor is 392.56cm, the last distance is set as

390cm to ensure the accuracy of the result. Then, the results are tabulated. To determine the accuracy of sensor, the error between the actual distance and sensor reading is required to be calculated first by using the equation (4-3).

$$\boldsymbol{Percentage\ of\ Error}$$

$$= \frac{\boldsymbol{Actual\ distance, Y_n - sensor\ reading, X_n}}{\boldsymbol{Actual\ distance, Y_n}} \times \mathbf{100\%} \tag{4-3}$$

After that, the average percentage error is calculated by using the equation (4-4).

$$\boldsymbol{Average\ percentage\ error}$$

$$= \frac{\boldsymbol{Sum\ of\ percentage\ error}}{\boldsymbol{total\ number\ of\ reading\ taken}} \tag{4-4}$$

$$= \frac{\boldsymbol{Sum\ of\ percentage\ error}}{\mathbf{4}}$$

The accuracy of sensor is computed by using equations (4-5).

$$\boldsymbol{Accuracy\ of\ sensor}$$

$$= \mathbf{100\%} - \boldsymbol{average\ percentage\ error, \%error} \tag{4-5}$$

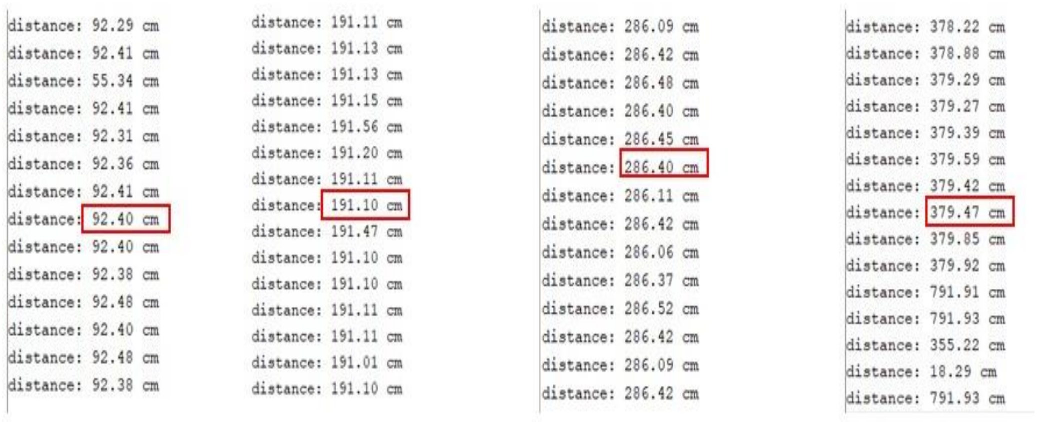

```
distance: 92.29 cm     distance: 191.11 cm     distance: 286.09 cm     distance: 378.22 cm
distance: 92.41 cm     distance: 191.13 cm     distance: 286.42 cm     distance: 378.88 cm
distance: 55.34 cm     distance: 191.13 cm     distance: 286.48 cm     distance: 379.29 cm
distance: 92.41 cm     distance: 191.15 cm     distance: 286.40 cm     distance: 379.27 cm
distance: 92.31 cm     distance: 191.56 cm     distance: 286.45 cm     distance: 379.39 cm
distance: 92.36 cm     distance: 191.20 cm     distance: 286.40 cm     distance: 379.59 cm
distance: 92.41 cm     distance: 191.11 cm     distance: 286.11 cm     distance: 379.42 cm
distance: 92.40 cm     distance: 191.10 cm     distance: 286.42 cm     distance: 379.47 cm
distance: 92.40 cm     distance: 191.47 cm     distance: 286.06 cm     distance: 379.85 cm
distance: 92.38 cm     distance: 191.10 cm     distance: 286.37 cm     distance: 379.92 cm
distance: 92.48 cm     distance: 191.10 cm     distance: 286.52 cm     distance: 791.91 cm
distance: 92.40 cm     distance: 191.11 cm     distance: 286.42 cm     distance: 791.93 cm
distance: 92.48 cm     distance: 191.11 cm     distance: 286.09 cm     distance: 355.22 cm
distance: 92.38 cm     distance: 191.01 cm     distance: 286.42 cm     distance: 18.29 cm
                       distance: 191.10 cm                             distance: 791.93 cm
```

Figure 4.14 Result of accuracy of ultrasonic sensor

Table 4.15: Accuracy of ultrasonic sensor

No.	**Actual distance, Yn (cm)**	**Ultrasonic sensor reading, Xn (cm)**	**Percentage of error (%)**
1	100	92.40	7.60
2	200	191.10	4.45
3	300	286.40	4.53
4	390	379.47	2.70
Average percentage error, % error			4.82
Accuracy			95.18

A graph is generated to compare the difference between the actual distance and sensor reading.

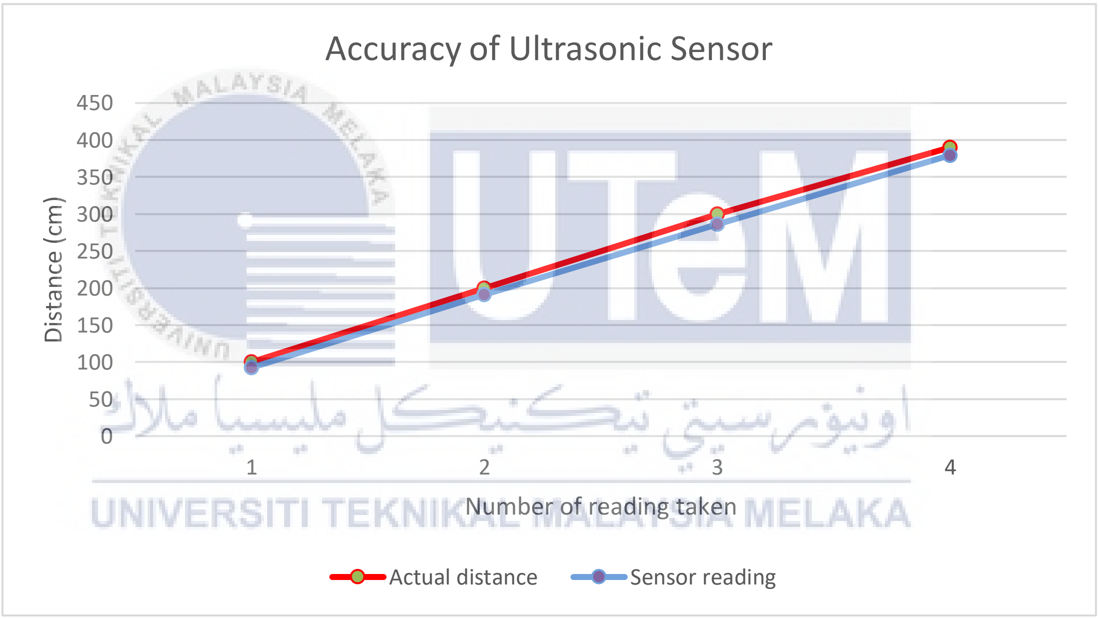
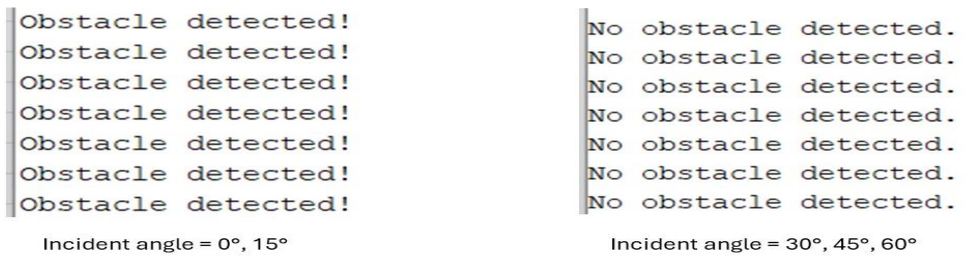


Figure 4.15: Graph of accuracy of ultrasonic sensor

From the results obtained, there is only a small deviation between the ultrasonic sensor reading and actual distance. The accuracy of ultrasonic sensor is 95.18%. It shows the high reliability of ultrasonic sensors in detecting the presence of obstacles. Hence, ultrasonic sensors are suitable distance sensors to be used in the collision avoidance mechanisms for UAV.

4.3.2.2 Infrared Sensor

In previous experiment, the maximum detection range of ultrasonic sensor is 15.86cm. The infrared sensor is located at a distance of 15cm from the cardboard. The result of of the experiment is shown in Figure 4.16. The different incident angle (0°, 15°, 30°, 45° and 60°) of infrared sensor is recorded in Table 4.16.



```
Obstacle detected!
Obstacle detected!
Obstacle detected!
Obstacle detected!
Obstacle detected!
Obstacle detected!
Obstacle detected!
```

```
No obstacle detected.
No obstacle detected.
No obstacle detected.
No obstacle detected.
No obstacle detected.
No obstacle detected.
No obstacle detected.
```

Figure 4.16 Result of different incident angle of infrared sensor

Table 4.16 Different incident angle of infrared sensor

Incident angle	**Detection of Obstacle**
0°	Obstacle is detected.
15°	Obstacle is detected.
30°	No obstacle is detected.
45°	No obstacle is detected.
60°	No obstacle is detected.

According to Table 4.16, the obstacle is detected when the incident angle is 0° and 15° while the obstacle is not detected when the incident angle is 30°, 45° and 60°. Infrared sensor detects the presence of obstacles through reflected infrared light. The accuracy of the infrared sensor is the highest when the infrared sensor is placed at the perpendicular to the cardboard. This is because infrared sensors receive the maximum amount of reflected infrared radiation from the cardboard. As the incident angle of infrared sensors increases, the infrared radiation cannot be reflected from cardboard to infrared sensor. Hence, the accuracy of infrared sensors decreases. As a result, the risk of UAV colliding with obstacles is higher.

After comparing the accuracy of both sensors, ultrasonic sensors are more suitable to be used for obstacle detection. Since there are many inherent uncertainties present in UAV navigation environment, utilizing ultrasonic sensors can reduce the factors contributing to sensor inaccuracy.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, all objectives of this project are achieved. This project is mainly focused on the collision avoidance mechanisms which are suitable for UAV. The collision avoidance algorithm which is suitable to be used in this project is investigated from the previous studies. The specifications of each algorithm have been analyzed. In this project, the algorithm that is decided to be modelled on MATLAB is Artificial Potential Field (APF) algorithm. Several virtual environments such as different number of obstacles, extreme position of goal and wind disturbances have been designed on MATLAB. All the simulation results show that the UAV can reach the goal position without colliding with obstacles for each condition. In an environment with a different number of obstacles, the RMSE value is between 0.05874 and 0.06349. When the goal is located at extreme position, the range of RMSE value is similar with previous which is between 0.06338 to 0.06351. However, in the virtual environment with wind disturbances, it shows the highest RMSE value which is between 0.08666 to 0.08670. This shows that there is a discrepancy between UAV's position and goal position. Lastly, ultrasonic sensors with 95.18% accuracy is selected to be integrated with the collision avoidance algorithm as an advancement of the collision avoidance mechanisms for UAV.

5.2 Recommendation

For future work, the APF algorithm and ultrasonic sensor are recommended to be integrated to apply on a physical UAV system. The Artificial Potential Field (APF) model on MATLAB is translated into a compatible programming language for UAV's onboard computer. Hence, a functioning UAV system which can avoid obstacles is developed. To improve the accuracy and efficiency of UAV's navigation, the APF algorithm can be optimized by refining the potential field equations or incorporating additional sensor data for better decision making.

REFERENCES

[1] Singhal, G., Bansod, B., & Mathew, L. (2018, November 27). *Unmanned Aerial Vehicle Classification, Applications and Challenges: A Review*. https://doi.org/10.20944/preprints201811.0601.v1

[2] M. Palik and M. Nagy, "Brief history of UAV development," Repüléstudományi Közlemények, vol. 31, no. 1, pp. 155–166, 2019, doi: 10.32560/rk.2019.1.13.

[3] J. Dong and Y. Zhang, "Optimization of Autonomous UAV Control Technology based on Computer Algorithms," 2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, 2022, pp. 194-197, doi: 10.1109/AEECA55500.2022.9919083.

[4] N. A. Singh and M. Borschbach, "Effect of external factors on accuracy of distance measurement using ultrasonic sensors," 2017 International Conference on Signals and Systems (ICSigSys), Bali, Indonesia, 2017, pp. 266-271, doi: 10.1109/ICSIGSYS.2017.7967054.

[5] T. Murfin, "UAV Report: Growth trends & opportunities for 2019 - GPS World," *GPS World - The Business and Technology of Global Navigation and Positioning*, Oct. 01, 2018. https://www.gpsworld.com/uav-report-growth-trends-opportunities-for-2019/

[6] N. Muchiri and S. Kimathi, "A Review of Applications and Potential Applications of UAV," 2016 Annual Conference on Sustainable Research and Innovation, Nyeri, Kenya, 2016, pp. 280–283.

[7] "Analysis of New Drone Incident Reports," Analysis of New Drone Incident Reports, Mar. 28, 2016. https://dronecenter.bard.edu/analysis-3-25-faa-incidents/

[8] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Unmanned Aerial Vehicles (UAVs): Collision Avoidance Systems and Approaches," IEEE Access, vol. 8, pp. 105139–105155, 2020, doi: 10.1109/access.2020.3000064.

[9] Meng Guanglei and Pan Haibing, "The application of ultrasonic sensor in the obstacle avoidance of quad-rotor UAV," 2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC), Nanjing, China, 2016, pp. 976-981, doi: 10.1109/CGNCC.2016.7828918.

[10] V. Kangunde, R. S. Jamisola, and E. K. Theophilus, "A review on drones controlled in real-time," International Journal of Dynamics and Control, vol. 9, no. 4, pp. 1832–1846, Jan. 2021, doi: 10.1007/s40435-020-00737-5.

[11] M. Sivakumar and N. M. TYJ, "A Literature Survey of Unmanned Aerial Vehicle Usage for Civil Applications," Journal of Aerospace Technology and Management, vol. 13, 2021, doi: 10.1590/jatm.v13.1233.

[12] "SpyLite - Fixed-wing UAV by BlueBird Aero Systems | DirectIndustry," Fixed-wing UAV - SpyLite - BlueBird Aero Systems - civilian / mapping / for photogrammetry. https://www.directindustry.com/prod/bluebird-aero-systems/product-61783-1311527.html

[13] Uyanik, Ilyas & Wesley, Avinash. (2019). Next Generation Gas Emission Monitoring System. 10.2118/195015-MS.

[14] Unknown, "Introduction to Multi rotors," *UAV Society: Introduction to Multi rotors*, May 27, 2014. http://uav-society.blogspot.com/2014/05/introduction-to-multi-rotors.html

[15] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of Unmanned Aerial Vehicles — A Survey," *Journal of Industrial Information Integration*, vol. 16, p. 100106, Dec. 2019, doi: 10.1016/j.jii.2019.100106.

[16] E. Pastor, J. Lopez and P. Royo, "A Hardware/Software Architecture for UAV Payload and Mission Control," 2006 ieee/aiaa 25TH Digital Avionics Systems Conference, Portland, OR, USA, 2006, pp. 1-8, doi: 10.1109/DASC.2006.313738.

[17] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," Engineering Science and Technology, an International Journal, vol. 40, p. 101343, Apr. 2023, doi: 10.1016/j.jestch.2023.101343.

[18] J. Sun, J. Tang, and S. Lao, “Collision Avoidance for Cooperative UAVs With Optimized Artificial Potential Field Algorithm,” *IEEE Access*, vol. 5, pp. 18382–18390, 2017, doi: 10.1109/access.2017.2746752.

[19] S. M. H. Rostami, A. K. Sangaiah, J. Wang, and X. Liu, “Obstacle avoidance of mobile robots using modified artificial potential field algorithm,” EURASIP Journal on Wireless Communications and Networking, vol. 2019, no. 1, Mar. 2019, doi: 10.1186/s13638-019-1396-2.

[20] Y. Sun, W. Chen, and J. Lv, “Uav Path Planning Based on Improved Artificial Potential Field Method,” *2022 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, Sep. 2022, Published, doi: 10.1109/iccnea57056.2022.00031.

[21] W. Xinggang, G. Cong and L. Yibo, "Variable probability based bidirectional RRT algorithm for UAV path planning," The 26th Chinese Control and Decision Conference (2014 CCDC), Changsha, China, 2014, pp. 2217-2222, doi: 10.1109/CCDC.2014.6852537.

[22] Iram Noreen, Amna Khan, and Zulfiqar Habib, “A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms,” *IJCSNS International Journal of Computer Science and Network Security*, vol. VOL.16, no. No.10, Oct. 2016.

[23] F. Yang *et al.*, “Obstacle Avoidance Path Planning for UAV Based on Improved RRT Algorithm,” *Discrete Dynamics in Nature and Society*, vol. 2022, pp. 1–9, Jan. 2022, doi: 10.1155/2022/4544499.

[24] H. Yang, Q. Jia and W. Zhang, "An Environmental Potential Field Based RRT Algorithm for UAV Path Planning," 2018 37th Chinese Control Conference (CCC), Wuhan, China, 2018, pp. 9922-9927, doi: 10.23919/ChiCC.2018.8483453.

[25] H. Wang, Z. Sun, D. Li and Q. Jin, "An Improved RRT Based 3-D Path Planning Algorithm for UAV," 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 2019, pp. 5514-5519, doi: 10.1109/CCDC.2019.8832661.

[26] X. Fu, C. Zhi, and D. Wu, “Obstacle avoidance and collision avoidance of UAV swarm based on improved VFH algorithm and information sharing strategy,” Computers & Industrial Engineering, vol. 186, p. 109761, Dec. 2023, doi: 10.1016/j.cie.2023.109761.

[27] K. Ding, M. Chen and K. Yong, "Self-Adjusting Angular Resolution-based Obstacle Avoidance for Inspection Unmanned Aerial Vehicles in Nuclear Power Stations," 2022 41st Chinese Control Conference (CCC), Hefei, China, 2022, pp. 3730-3735, doi: 10.23919/CCC55666.2022.9901870.

[28] P. Qu, J. Xue, L. Ma and C. Ma, "A constrained VFH algorithm for motion planning of autonomous vehicles," 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea (South), 2015, pp. 700-705, doi: 10.1109/IVS.2015.7225766.

[29] H. Zhang et al., "Path planning for fixed-wing UAVs based on expert knowledge and improved VFH in cluttered environments," 2022 IEEE 17th International Conference on Control & Automation (ICCA), Naples, Italy, 2022, pp. 255-260, doi: 10.1109/ICCA54724.2022.9831848.

[30] Q. Liang, Z. Wang, Y. Yin, W. Xiong, J. Zhang, and Z. Yang, “Autonomous aerial obstacle avoidance using LiDAR sensor fusion,” PLOS ONE, vol. 18, no. 6, p. e0287177, Jun. 2023, doi: 10.1371/journal.pone.0287177.

[31] J. Park, Y. Je, H. Lee, and W. Moon, “Design of an ultrasonic sensor for measuring distance and detecting obstacles,” Ultrasonics, vol. 50, no. 3, pp. 340–346, Mar. 2010, doi: 10.1016/j.ultras.2009.10.013.

[32] Latha, N. A., Murthy, B. R., & Kumar, K. B. (2016). Distance sensing with ultrasonic sensor and Arduino. International Journal of Advance Research, Ideas and Innovations in Technology, 2(5), 1-5.

[33] V. A. Zhmud, N. O. Kondratiev, K. A. Kuznetsov, V. G. Trubin, and L. V. Dimitrov, “Application of ultrasonic sensor for measuring distances in robotics,” Journal of Physics: Conference Series, vol. 1015, p. 032189, May 2018, doi: 10.1088/1742-6596/1015/3/032189.

[34] Meng Guanglei and Pan Haibing, "The application of ultrasonic sensor in the obstacle avoidance of quad-rotor UAV," 2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC), Nanjing, China, 2016, pp. 976-981, doi: 10.1109/CGNCC.2016.7828918.

[35] S. Bhardwaj, A. Warbhe, and B. Raj Kumar, "Sensor System Implementation for Unmanned Aerial Vehicles," Indian Journal of Science and Technology, vol. 8, no. S2, p. 7, Jan. 2015, doi: 10.17485/ijst/2015/v8is2/57790.

[36] J. S. G. Guerrero, A. F. C. González, J. I. H. Vega, and L. A. N. Tovar, "Instrumentation of an Array of Ultrasonic Sensors and Data Processing for Unmanned Aerial Vehicle (UAV) for Teaching the Application of the Kalman Filter," Procedia Computer Science, vol. 75, pp. 375–380, 2015, doi: 10.1016/j.procs.2015.12.260.

[37] L. Di, H. Chao and Y. Chen, "A two-stage calibration method for low-cost UAV attitude estimation using infrared sensor," Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, QingDao, China, 2010, pp. 137-142, doi: 10.1109/MESA.2010.5552079.

[38] G. Benet, F. Blanes, J. E. Simó, and P. Pérez, "Using infrared sensors for distance measurement in mobile robots," Robotics and Autonomous Systems, vol. 40, no. 4, pp. 255–266, Sep. 2002, doi: 10.1016/s0921-8890(02)00271-3.

[39] WatElectronics, "IR Sensor : Circuit, Types, Working Principle & Its Applications," WatElectronics.com, Jul. 21, 2021. https://www.watelectronics.com/ir-sensor/

[40] N. Gageik, P. Benz and S. Montenegro, "Obstacle Detection and Collision Avoidance for a UAV With Complementary Low-Cost Sensors," in IEEE Access, vol. 3, pp. 599-609, 2015, doi: 10.1109/ACCESS.2015.2432455.

[41] S. A. Daud, S. S. Mohd Sobani, M. H. Ramiee, N. H. Mahmood, P. L. Leow and F. K. Che Harun, "Application of Infrared sensor for shape detection," 2013 IEEE 4th International Conference on Photonics (ICP), Melaka, Malaysia, 2013, pp. 145-147, doi: 10.1109/ICP.2013.6687095.

[42] Tarek Mohammad, “Using Ultrasonic and Infrared Sensors for Distance Measurement,” World Academy of Science, Engineering and Technology, vol. Vol:3, Mar. 2009.

[43] H. Liu, X. Li, M. Fan, G. Wu, W. Pedrycz and P. Nagaratnam Suganthan, "An Autonomous Path Planning Method for Unmanned Aerial Vehicle Based on a Tangent Intersection and Target Guidance Strategy," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 4, pp. 3061-3073, April 2022, doi: 10.1109/TITS.2020.3030444.

[44] B. Mustapha, A. Zayegh and R. K. Begg, "Ultrasonic and Infrared Sensors Performance in a Wireless Obstacle Detection System," 2013 1st International Conference on Artificial Intelligence, Modelling and Simulation, Kota Kinabalu, Malaysia, 2013, pp. 487-492, doi: 10.1109/AIMS.2013.89.

APPENDICES

APPENDIX A GANTT CHART

Gantt Chart

Task	FYP 1														FYP 2													
	Week														Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Title																												
Analysis on title	🟨																											
Submission of title	🟨																											
Introduction																												
Define problem statements, objectives, scope and limitations		🟨	🟨	🟨																								
Literature Review																												
Research on journal			🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨																
Summarize of journal					🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨															
Methodology																												
Coding and programming to generate virtual environment on MATLAB								🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨		
Design experiment to determine the effectiveness of sensor								🟨	🟨	🟨	🟨	🟨																
Purchasing components									🟨	🟨																		
Results																												
Obtaining and analyzing simulation result from MATLAB																🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨		
Collect and analyze data from distance sensor												🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨								
FYP Seminar																												
Presentation FYP 1											🟨																	
Prepare and finalize FYP 1 report											🟨	🟨	🟨	🟨														
Radiate for FYP 2																											🟨	
Prepare and finalize FYP 2 report																							🟨	🟨	🟨	🟨	🟨	🟨

APPENDIX B MATLAB CODE FOR OBSTACLE-FREE ENVIRONMENT

```
clear all; close all; clc;

% Initial position and orientation
x = 0;
y = 0;
theta = 0;

% Goal position
x_goal = 3.5;
y_goal = 3.5;
position_accuracy = 0.05;

% APF parameters
zeta = 1.1547;
eta = 0.0732;
dstar = 0.3;
Qstar = 0.75;

% Parameters related to kinematic model
error_theta_max = deg2rad(45);
v_max = 0.2;
Kp_omega = 1.5;
omega_max = 0.5*pi;

figure(1);
t = 1;
dT = 0.1;
t_max = 5000;
X = zeros(1,t_max);
Y = zeros(1,t_max);
X(1) = x;
Y(1) = y;

while norm([x_goal y_goal] - [x y]) > position_accuracy || t > t_max
% Calculate Attractive Potential
   if norm([x y]-[x_goal y_goal]) <= dstar
      nablaU_att =  zeta*([x y]-[x_goal y_goal]);
   else
      nablaU_att = dstar/norm([x y]-[x_goal y_goal]) * zeta*([x y]-[x_goal y_goal]);
   end

% Calculate final potential
   nablaU = nablaU_att;
```

```
% Calculate reference value of linear velocity (v_ref) and orientation (theta_ref)
    theta_ref = atan2(-nablaU(2), -nablaU(1));

    error_theta = theta_ref - theta;
    if abs(error_theta) <= error_theta_max
        alpha = (error_theta_max - abs(error_theta)) / error_theta_max;
        v_ref = min( alpha*norm(-nablaU), v_max );
    else
        v_ref = 0;
    end

% Simple kinematic mobile robot model
% Omitted dynamics.
    omega_ref = Kp_omega * error_theta;
    omega_ref = min( max(omega_ref, -omega_max), omega_max);
    theta = theta + omega_ref * dT;
    x = x + v_ref*cos(theta) * dT;
    y = y + v_ref*sin(theta) * dT;

    t = t + 1;

% Obtain the current position and distance between UAV and goal
    disp("Current position: (" + x + ", " + y + "), Distance to goal: " + norm([x_goal
y_goal] - [x y]));

    % Archive and plot it
    X(t) = x;
    Y(t) = y;
    cla;
    daspect([1 1 1]);
    xlim([-0.5,4]);  ylim([-0.5 4]);
    box on; hold on;
    plot(x_goal, y_goal, 'ob');
% Plot traveled path
    plot(X(1:t), Y(1:t), '-b');
% Plot reference orientation of the robot
    plot([x x+0.2*cos(theta_ref)], [y y+0.2*sin(theta_ref)], '-g');
% Plot orientation of the robot
    plot([x x+0.2*cos(theta)], [y y+0.2*sin(theta)], '-r');
    xlabel('X position');
    ylabel('Y position');
    title('UAV Trajectory Without Obstacle');
    legend('Goal position','UAV trajectory', 'Location','northeastoutside');
    drawnow;
    pause(dT);
end
t = t*dT; % scale from iterations to [s]

disp("Travel time: " + t);
```

APPENDIX C MATLAB CODE FOR ENVIRONMENT WITH 1 OBSTACLE

```
clear all; close all; clc;

% Initial position and orientation
x = 0;
y = 0;
theta = 0;

% Goal position
x_goal = 3.5;
y_goal = 3.5;
position_accuracy = 0.05;

% APF parameters
zeta = 1.1547;
eta = 0.0732;
dstar = 0.3;
Qstar = 0.75;

% Parameters related to kinematic model
error_theta_max = deg2rad(45);
v_max = 0.2;
Kp_omega = 1.5;
omega_max = 0.5*pi;

% Generate obstacles
obst1_points = [ linspace(1,2,100) linspace(2,2,100) linspace(2,1,100)
linspace(1,1,100)
                 linspace(1,1,100)   linspace(1,1.5,100)   linspace(1.5,1.5,100)
linspace(1.5,1,100) ];
obst1_points(1,:) = obst1_points(1,:) - 0.5;
obst1_points(2,:) = obst1_points(2,:) - 1;

obst2_points = [ linspace(1.5,2,100) linspace(2,2,100) linspace(2,1.5,100)
linspace(1.5,1.5,100)
                 linspace(1.5,1.5,100)   linspace(1.5,2,100)   linspace(2,2,100)
linspace(2,1.5,100) ];
obst2_points(1,:) = obst2_points(1,:) - 0.5;
obst2_points(2,:) = obst2_points(2,:) - 1;

figure(1);
t = 1;
dT = 0.1;
t_max = 1000;
X = zeros(1,t_max);
Y = zeros(1,t_max);
X(1) = x;
Y(1) = y;
```

```
while norm([x_goal y_goal] - [x y]) > position_accuracy || t > t_max
% Calculate Attractive Potential
   if norm([x y]-[x_goal y_goal]) <= dstar
      nablaU_att =  zeta*([x y]-[x_goal y_goal]);
   else
      nablaU_att = dstar/norm([x y]-[x_goal y_goal]) * zeta*([x y]-[x_goal y_goal]);
   end

% Find the minimum distance from the obstacle
   [obst1_idx, obst1_dist] = dsearchn(obst1_points', [x y]);
   [obst2_idx, obst2_dist] = dsearchn(obst2_points', [x y]);

% Calculate Repulsive Potential
   nablaU_rep = [0 0];
   if obst1_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst1_dist) * 1/obst1_dist^2)*([x y] -
[obst1_points(1,obst1_idx)  obst1_points(2,obst1_idx)]);
   end
   if obst2_dist <= Qstar  && ~inpolygon(x,y,obst2_points(1,:),obst2_points(2,:))
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst2_dist) * 1/obst2_dist^2)*([x y] -
[obst2_points(1,obst2_idx)  obst2_points(2,obst2_idx)]);
   end

% Calculate final potential
   nablaU = nablaU_att+nablaU_rep;

% Calculate reference value of linear velocity (v_ref) and orientation (theta_ref)
   theta_ref = atan2(-nablaU(2), -nablaU(1));

   error_theta = theta_ref - theta;
   if abs(error_theta) <= error_theta_max
      alpha = (error_theta_max - abs(error_theta)) / error_theta_max;
      v_ref = min( alpha*norm(-nablaU), v_max );
   else
      v_ref = 0;
   end

% Simple kinematic mobile robot model
% Omitted dynamics.
   omega_ref = Kp_omega * error_theta;
   omega_ref = min( max(omega_ref, -omega_max), omega_max);
   theta = theta + omega_ref * dT;
   x = x + v_ref*cos(theta) * dT;
   y = y + v_ref*sin(theta) * dT;

   t = t + 1;
% Obtain the current position and distance between UAV and goal
   disp("Current position: (" + x + ", " + y + "), Distance to goal: " + norm([x_goal
y_goal] - [x y]));
```

```
% Archive and plot it
    X(t) = x;
    Y(t) = y;
    cla;
    daspect([1 1 1]);
    xlim([-0.5 4]);  ylim([-0.5 4]);
    box on; hold on;
    plot(obst1_points(1,:), obst1_points(2,:), '-r');
    plot(obst2_points(1,:), obst2_points(2,:), '-r');
    plot(x_goal, y_goal, 'ob');
% Plot traveled path
    plot(X(1:t), Y(1:t), '-b');
% Plot reference orientation of the robot
    plot([x x+0.2*cos(theta_ref)], [y y+0.2*sin(theta_ref)], '-g');
% Plot orientation of the robot
    plot([x x+0.2*cos(theta)], [y y+0.2*sin(theta)], '-r');
    xlabel('X position');
    ylabel('Y position');
    title('UAV Trajectory With 1 Obstacle');
    legend('Obstacle','', 'Goal position','UAV trajectory', 'Location','northeastoutside');
    drawnow;
    pause(dT);
end
t = t*dT; % scale from iterations to [s]

disp("Travel time: " + t);
```

APPENDIX D MATLAB CODE FOR ENVIRONMENT WITH 2 OBSTACLES

```
clear all; close all; clc;

% Initial position and orientation
x = 0;
y = 0;
theta = 0;

% Goal position
x_goal = 3.5;
y_goal = 3.5;
position_accuracy = 0.05;

% APF parameters
zeta = 1.1547;
eta = 0.0732;
dstar = 0.3;
Qstar = 0.75;

% Parameters related to kinematic model
error_theta_max = deg2rad(45);
v_max = 0.2;
Kp_omega = 1.5;
omega_max = 0.5*pi;

% Generate obstacles
obst1_points = [ linspace(1,2,100) linspace(2,2,100) linspace(2,1,100)
linspace(1,1,100)
                  linspace(1,1,100)  linspace(1,1.5,100)  linspace(1.5,1.5,100)
linspace(1.5,1,100) ];
obst1_points(1,:) = obst1_points(1,:) - 0.5;
obst1_points(2,:) = obst1_points(2,:) - 1;

obst2_points = [ linspace(1.5,2,100) linspace(2,2,100) linspace(2,1.5,100)
linspace(1.5,1.5,100)
                  linspace(1.5,1.5,100)  linspace(1.5,2,100)  linspace(2,2,100)
linspace(2,1.5,100) ];
obst2_points(1,:) = obst2_points(1,:) - 0.5;
obst2_points(2,:) = obst2_points(2,:) - 1;

obst3_points = [ linspace(1.5,2.5,100) linspace(2.5,2.5,100) linspace(2.5,1.5,100)
linspace(1.5,1.5,100)
                  linspace(2.5,2.5,100)  linspace(2.5,3,100)  linspace(3,3,100)
linspace(3,2.5,100) ];
obst3_points(1,:) = obst3_points(1,:) - 0.5;
obst3_points(2,:) = obst3_points(2,:) - 0.5;

figure(1);
```

```
t = 1;
dT = 0.1;
t_max = 1000;
X = zeros(1,t_max);
Y = zeros(1,t_max);
X(1) = x;
Y(1) = y;

while norm([x_goal y_goal] - [x y]) > position_accuracy || t > t_max
% Calculate Attractive Potential
   if norm([x y]-[x_goal y_goal]) <= dstar
      nablaU_att =  zeta*([x y]-[x_goal y_goal]);
   else
      nablaU_att = dstar/norm([x y]-[x_goal y_goal]) * zeta*([x y]-[x_goal y_goal]);
   end

% Find the minimum distance from the obstacle
   [obst1_idx, obst1_dist] = dsearchn(obst1_points', [x y]);
   [obst2_idx, obst2_dist] = dsearchn(obst2_points', [x y]);
   [obst3_idx, obst3_dist] = dsearchn(obst3_points', [x y]);

% Calculate Repulsive Potential
   nablaU_rep = [0 0];
   if obst1_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst1_dist) * 1/obst1_dist^2)*([x y] -
[obst1_points(1,obst1_idx)  obst1_points(2,obst1_idx)]);
   end
   if obst2_dist <= Qstar  && ~inpolygon(x,y,obst2_points(1,:),obst2_points(2,:))
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst2_dist) * 1/obst2_dist^2)*([x y] -
[obst2_points(1,obst2_idx)  obst2_points(2,obst2_idx)]);
   end
   if obst3_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst3_dist) * 1/obst3_dist^2)*([x y] -
[obst3_points(1,obst3_idx)  obst3_points(2,obst3_idx)]);
   end

% Calculate final potential
   nablaU = nablaU_att+nablaU_rep;

% Calculate reference value of linear velocity (v_ref) and orientation (theta_ref)
   theta_ref = atan2(-nablaU(2), -nablaU(1));

   error_theta = theta_ref - theta;
   if abs(error_theta) <= error_theta_max
      alpha = (error_theta_max - abs(error_theta)) / error_theta_max;
      v_ref = min( alpha*norm(-nablaU), v_max );
   else
      v_ref = 0;
   end
```

```
% Simple kinematic mobile robot model
% Omitted dynamics.
    omega_ref = Kp_omega * error_theta;
    omega_ref = min( max(omega_ref, -omega_max), omega_max);
    theta = theta + omega_ref * dT;
    x = x + v_ref*cos(theta) * dT;
    y = y + v_ref*sin(theta) * dT;

    t = t + 1;

% Obtain the current position and distance between UAV and goal
    disp("Current position: (" + x + ", " + y + "), Distance to goal: " + norm([x_goal
y_goal] - [x y]));

% Archive and plot it
    X(t) = x;
    Y(t) = y;
    cla;
    daspect([1 1 1]);
    xlim([-0.5 4]);  ylim([-0.5 4]);
    box on; hold on;
    plot(obst1_points(1,:), obst1_points(2,:), '-r');
    plot(obst2_points(1,:), obst2_points(2,:), '-r');
    plot(obst3_points(1,:), obst3_points(2,:), '-r');
    plot(x_goal, y_goal, 'ob');
% Plot traveled path
    plot(X(1:t), Y(1:t), '-b');
% Plot reference orientation of the robot
    plot([x x+0.2*cos(theta_ref)], [y y+0.2*sin(theta_ref)], '-g');
% Plot orientation of the robot
    plot([x x+0.2*cos(theta)], [y y+0.2*sin(theta)], '-r');
    xlabel('X position');
    ylabel('Y position');
    title('UAV Trajectory With 2 Obstacles');
    legend('Obstacle',",",'Goal position','UAV trajectory', 'Location','northeastoutside');
    drawnow;
    pause(dT);
end
t = t*dT; % scale from iterations to [s]

disp("Travel time: " + t);
```

APPENDIX E MATLAB CODE FOR ENVIRONMENT WITH 3 OBSTACLES

```
clear all; close all; clc;

% Initial position and orientation
x = 0;
y = 0;
theta = 0;

% Goal position
x_goal = 3.5;
y_goal = 3.5;
position_accuracy = 0.05;

% APF parameters
zeta = 1.1547;
eta = 0.0732;
dstar = 0.3;
Qstar = 0.75;

% Parameters related to kinematic model
error_theta_max = deg2rad(45);
v_max = 0.2;
Kp_omega = 1.5;
omega_max = 0.5*pi;

% Generate obstacles
obst1_points = [ linspace(1,2,100) linspace(2,2,100) linspace(2,1,100)
linspace(1,1,100)
                  linspace(1,1,100)   linspace(1,1.5,100)   linspace(1.5,1.5,100)
linspace(1.5,1,100) ];
obst1_points(1,:) = obst1_points(1,:) - 0.5;
obst1_points(2,:) = obst1_points(2,:) - 1;

obst2_points = [ linspace(1.5,2,100) linspace(2,2,100) linspace(2,1.5,100)
linspace(1.5,1.5,100)
                  linspace(1.5,1.5,100)   linspace(1.5,2,100)   linspace(2,2,100)
linspace(2,1.5,100) ];
obst2_points(1,:) = obst2_points(1,:) - 0.5;
obst2_points(2,:) = obst2_points(2,:) - 1;

obst3_points = [ linspace(1.5,2.5,100) linspace(2.5,2.5,100) linspace(2.5,1.5,100)
linspace(1.5,1.5,100)
                  linspace(2.5,2.5,100)   linspace(2.5,3,100)   linspace(3,3,100)
linspace(3,2.5,100) ];
obst3_points(1,:) = obst3_points(1,:) - 0.5;
obst3_points(2,:) = obst3_points(2,:) - 0.5;
```

```
obst4_points = [ linspace(3.5,4,100) linspace(4,4,100) linspace(4,3.5,100)
linspace(3.5,3.5,100)
                    linspace(2.5,2.5,100)  linspace(2.5,3.5,100)  linspace(3.5,3.5,100)
linspace(3.5,2.5,100) ];
obst4_points(1,:) = obst4_points(1,:) - 0.5;
obst4_points(2,:) = obst4_points(2,:) - 1;

figure(1);
t = 1;
dT = 0.1;
t_max = 1000;
X = zeros(1,t_max);
Y = zeros(1,t_max);
X(1) = x;
Y(1) = y;

while norm([x_goal y_goal] - [x y]) > position_accuracy || t > t_max
% Calculate Attractive Potential
   if norm([x y]-[x_goal y_goal]) <= dstar
      nablaU_att =  zeta*([x y]-[x_goal y_goal]);
   else
      nablaU_att = dstar/norm([x y]-[x_goal y_goal]) * zeta*([x y]-[x_goal y_goal]);
   end

% Find the minimum distance from the obstacle
   [obst1_idx, obst1_dist] = dsearchn(obst1_points', [x y]);
   [obst2_idx, obst2_dist] = dsearchn(obst2_points', [x y]);
   [obst3_idx, obst3_dist] = dsearchn(obst3_points', [x y]);
   [obst4_idx, obst4_dist] = dsearchn(obst4_points', [x y]);

% Calculate Repulsive Potential
   nablaU_rep = [0 0];
   if obst1_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst1_dist) * 1/obst1_dist^2)*([x y] -
[obst1_points(1,obst1_idx)  obst1_points(2,obst1_idx)]);
   end
   if obst2_dist <= Qstar  && ~inpolygon(x,y,obst2_points(1,:),obst2_points(2,:))
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst2_dist) * 1/obst2_dist^2)*([x y] -
[obst2_points(1,obst2_idx)  obst2_points(2,obst2_idx)]);
   end
   if obst3_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst3_dist) * 1/obst3_dist^2)*([x y] -
[obst3_points(1,obst3_idx)  obst3_points(2,obst3_idx)]);
   end
   if obst4_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst4_dist) * 1/obst4_dist^2)*([x y] -
[obst4_points(1,obst4_idx)  obst4_points(2,obst4_idx)]);
   end
```

```
% Calculate final potential
   nablaU = nablaU_att+nablaU_rep;

% Calculate reference value of linear velocity (v_ref) and orientation (theta_ref)
   theta_ref = atan2(-nablaU(2), -nablaU(1));

   error_theta = theta_ref - theta;
   if abs(error_theta) <= error_theta_max
      alpha = (error_theta_max - abs(error_theta)) / error_theta_max;
      v_ref = min( alpha*norm(-nablaU), v_max );
   else
      v_ref = 0;
   end

% Simple kinematic mobile robot model
% Omitted dynamics.
   omega_ref = Kp_omega * error_theta;
   omega_ref = min( max(omega_ref, -omega_max), omega_max);
   theta = theta + omega_ref * dT;
   x = x + v_ref*cos(theta) * dT;
   y = y + v_ref*sin(theta) * dT;
   t = t + 1;

% Obtain the current position and distance between UAV and goal
   disp("Current position: (" + x + ", " + y + "), Distance to goal: " + norm([x_goal
y_goal] - [x y]));

% Archive and plot it
   X(t) = x;
   Y(t) = y;
   cla;
   daspect([1 1 1]);
   xlim([-0.5 4]);  ylim([-0.5 4]);
   box on; hold on;
   plot(obst1_points(1,:), obst1_points(2,:), '-r');
   plot(obst2_points(1,:), obst2_points(2,:), '-r');
   plot(obst3_points(1,:), obst3_points(2,:), '-r');
   plot(obst4_points(1,:), obst4_points(2,:), 'r');
   plot(x_goal, y_goal, 'ob');
% Plot traveled path
   plot(X(1:t), Y(1:t), '-b');
% Plot reference orientation of the robot
   plot([x x+0.2*cos(theta_ref)], [y y+0.2*sin(theta_ref)], '-g');
% Plot orientation of the robot
   plot([x x+0.2*cos(theta)], [y y+0.2*sin(theta)], '-r');
   xlabel('X position');
   ylabel('Y position');
   title('UAV Trajectory With 3 Obstacles');
   legend('Obstacle',",",",'Goal position','UAV trajectory', 'Location','northeastoutside');
   drawnow;
```

```
    pause(dT);
end

t = t*dT; % scale from iterations to [s]

disp("Travel time: " + t);
```

APPENDIX F MATLAB CODE FOR EXTREME GOAL POSITION IN OBSTACLE-FREE ENVIRONMENT

```
clear all; close all; clc;

% Initial position and orientation
x = 0;
y = 0;
theta = 0;

% Goal position
x_goal = 100;
y_goal = 100;
position_accuracy = 0.05;

% APF parameters
zeta = 1.1547;
eta = 0.0732;
dstar = 0.3;
Qstar = 0.75;

% Parameters related to kinematic model
error_theta_max = deg2rad(45);
v_max = 0.2;
Kp_omega = 1.5;
omega_max = 0.5*pi;

figure(1);
t = 1;
dT = 0.1;
t_max = 10000;
X = zeros(1,t_max);
Y = zeros(1,t_max);
X(1) = x;
Y(1) = y;

while norm([x_goal y_goal] - [x y]) > position_accuracy || t > t_max
% Calculate Attractive Potential
   if norm([x y]-[x_goal y_goal]) <= dstar
      nablaU_att =  zeta*([x y]-[x_goal y_goal]);
   else
      nablaU_att = dstar/norm([x y]-[x_goal y_goal]) * zeta*([x y]-[x_goal y_goal]);
   end

% Calculate final potential
   nablaU = nablaU_att;
```

```
% Calculate reference value of linear velocity (v_ref) and orientation (theta_ref)
   theta_ref = atan2(-nablaU(2), -nablaU(1));

   error_theta = theta_ref - theta;
   if abs(error_theta) <= error_theta_max
      alpha = (error_theta_max - abs(error_theta)) / error_theta_max;
      v_ref = min( alpha*norm(-nablaU), v_max );
   else
      v_ref = 0;
   end

% Simple kinematic mobile robot model
% Omitted dynamics.
   omega_ref = Kp_omega * error_theta;
   omega_ref = min( max(omega_ref, -omega_max), omega_max);
   theta = theta + omega_ref * dT;
   x = x + v_ref*cos(theta) * dT;
   y = y + v_ref*sin(theta) * dT;

   t = t + 1;

% Obtain the current position and distance between UAV and goal
   disp("Current position: (" + x + ", " + y + "), Distance to goal: " + norm([x_goal
y_goal] - [x y]));

% Archive and plot it
   X(t) = x;
   Y(t) = y;
   cla;
   daspect([1 1 1]);
   xlim([-5 110]);  ylim([-5 110]);
   box on; hold on;
   plot(x_goal, y_goal, 'ob');
% Plot traveled path
   plot(X(1:t), Y(1:t), '-b');
% Plot reference orientation of the robot
   plot([x x+0.2*cos(theta_ref)], [y y+0.2*sin(theta_ref)], '-g');
% Plot orientation of the robot
   plot([x x+0.2*cos(theta)], [y y+0.2*sin(theta)], '-r');
   xlabel('X position');
   ylabel('Y position');
   title('UAV Trajectory Without Obstacle For Extreme Position');
   legend('Goal position','UAV trajectory', 'Location','northeastoutside');
   drawnow;
   pause(dT);
end
t = t*dT; % scale from itetations to [s]

disp("Travel time: " + t);
```

APPENDIX G MATLAB CODE FOR EXTREME GOAL POSITION IN ENVIRONMENT WITH 3 OBSTACLE

```
clear all; close all; clc;

% Initial position and orientation
x = 0;
y = 0;
theta = 0;

% Goal position
x_goal = 100;
y_goal = 100;
position_accuracy = 0.05;

% APF parameters
zeta = 1.1547;
eta = 0.0732;
dstar = 0.3;
Qstar = 0.75;

% Parameters related to kinematic model
error_theta_max = deg2rad(45);
v_max = 0.2;
Kp_omega = 1.5;
omega_max = 0.5*pi;

% Generate obstacles
obst1_points = [ linspace(29,57,100) linspace(57,57,100) linspace(57,29,100)
linspace(29,29,100)
                     linspace(29,29,100)   linspace(29,43,100)   linspace(43,43,100)
linspace(43,29,100) ];
obst1_points(1,:) = obst1_points(1,:) - 14;
obst1_points(2,:) = obst1_points(2,:) - 29;

obst2_points = [ linspace(43,57,100) linspace(57,57,100) linspace(57,43,100)
linspace(43,43,100)
                     linspace(43,43,100)   linspace(43,57,100)   linspace(57,57,100)
linspace(57,43,100) ];
obst2_points(1,:) = obst2_points(1,:) - 14;
obst2_points(2,:) = obst2_points(2,:) - 29;

obst3_points = [ linspace(43,71,100) linspace(71,71,100) linspace(71,43,100)
linspace(43,43,100)
                     linspace(71,71,100)   linspace(71,86,100)   linspace(86,86,100)
linspace(86,71,100) ];
obst3_points(1,:) = obst3_points(1,:) - 14;
obst3_points(2,:) = obst3_points(2,:) - 14;
```

```
obst4_points = [ linspace(100,114,100) linspace(114,114,100) linspace(114,100,100)
linspace(100,100,100)
                        linspace(71,71,100)   linspace(71,100,100)   linspace(100,100,100)
linspace(100,71,100) ];
obst4_points(1,:) = obst4_points(1,:) - 14;
obst4_points(2,:) = obst4_points(2,:) - 29;

figure(1);
t = 1;
dT = 0.1;
t_max = 10000;
X = zeros(1,t_max);
Y = zeros(1,t_max);
X(1) = x;
Y(1) = y;

while norm([x_goal y_goal] - [x y]) > position_accuracy || t > t_max
% Calculate Attractive Potential
   if norm([x y]-[x_goal y_goal]) <= dstar
      nablaU_att =  zeta*([x y]-[x_goal y_goal]);
   else
      nablaU_att = dstar/norm([x y]-[x_goal y_goal]) * zeta*([x y]-[x_goal y_goal]);
   end

% Find the minimum distance from the obstacle
   [obst1_idx, obst1_dist] = dsearchn(obst1_points', [x y]);
   [obst2_idx, obst2_dist] = dsearchn(obst2_points', [x y]);
   [obst3_idx, obst3_dist] = dsearchn(obst3_points', [x y]);
   [obst4_idx, obst4_dist] = dsearchn(obst4_points', [x y]);

% Calculate Repulsive Potential
   nablaU_rep = [0 0];
   if obst1_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst1_dist) * 1/obst1_dist^2)*([x y] -
[obst1_points(1,obst1_idx)  obst1_points(2,obst1_idx)]);
   end
   if obst2_dist <= Qstar  && ~inpolygon(x,y,obst2_points(1,:),obst2_points(2,:))
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst2_dist) * 1/obst2_dist^2)*([x y] -
[obst2_points(1,obst2_idx)  obst2_points(2,obst2_idx)]);
   end
   if obst3_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst3_dist) * 1/obst3_dist^2)*([x y] -
[obst3_points(1,obst3_idx)  obst3_points(2,obst3_idx)]);
   end
   if obst4_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst4_dist) * 1/obst4_dist^2)*([x y] -
[obst4_points(1,obst4_idx)  obst4_points(2,obst4_idx)]);
   end
```

```
% Calculate final potential
   nablaU = nablaU_att+nablaU_rep;

% Calculate reference value of linear velocity (v_ref) and orientation (theta_ref)
   theta_ref = atan2(-nablaU(2), -nablaU(1));

   error_theta = theta_ref - theta;
   if abs(error_theta) <= error_theta_max
      alpha = (error_theta_max - abs(error_theta)) / error_theta_max;
      v_ref = min( alpha*norm(-nablaU), v_max );
   else
      v_ref = 0;
   end

% Simple kinematic mobile robot model
% Omitted dynamics.
   omega_ref = Kp_omega * error_theta;
   omega_ref = min( max(omega_ref, -omega_max), omega_max);
   theta = theta + omega_ref * dT;
   x = x + v_ref*cos(theta) * dT;
   y = y + v_ref*sin(theta) * dT;

   t = t + 1;

% Obtain the current position and distance between UAV and goal
   disp("Current position: (" + x + ", " + y + "), Distance to goal: " + norm([x_goal
y_goal] - [x y]));

% Archive and plot it
   X(t) = x;
   Y(t) = y;
   cla;
   daspect([1 1 1]);
   xlim([-5 110]);  ylim([-5 110]);
   box on; hold on;
   plot(obst1_points(1,:), obst1_points(2,:), '-r');
   plot(obst2_points(1,:), obst2_points(2,:), '-r');
   plot(obst3_points(1,:), obst3_points(2,:), '-r');
   plot(obst4_points(1,:), obst4_points(2,:), 'r');
   plot(x_goal, y_goal, 'ob');
% Plot traveled path
   plot(X(1:t), Y(1:t), '-b');
% Plot reference orientation of the robot
   plot([x x+0.2*cos(theta_ref)], [y y+0.2*sin(theta_ref)], '-g');
% Plot orientation of the robot
   plot([x x+0.2*cos(theta)], [y y+0.2*sin(theta)], '-r');
   xlabel('X position');
   ylabel('Y position');
   title('UAV Trajectory With 3 Obstacles For Extreme Position');
   legend('Obstacle',",",",'Goal position','UAV trajectory', 'Location','northeastoutside');
```

```
    drawnow;
    pause(dT);
end

t = t*dT; % scale from itetations to [s]

disp("Travel time: " + t);
```

APPENDIX H MATLAB CODE FOR OBSTACLE-FREE ENVIRONMENT WITH WIND DISTURBANCES

```
clear all; close all; clc;

% Initial position and orientation
x = 0;
y = 0;
theta = 0;

% Goal position
x_goal = 3.5;
y_goal = 3.5;
position_accuracy = 0.05;

% APF parameters
zeta = 1.1547;
eta = 0.0732;
dstar = 0.3;
Qstar = 0.75;

% Parameters related to kinematic model
error_theta_max = deg2rad(45);
v_max = 0.2;
Kp_omega = 1.5;
omega_max = 0.5*pi;

% Additional parameters for wind
wind_speed = 0.1;  % Wind speed
wind_direction = deg2rad(90);  % Wind direction in radians

figure(1);
t = 1;
dT = 0.1;
t_max = 1000;
X = zeros(1,t_max);
Y = zeros(1,t_max);
X(1) = x;
Y(1) = y;

while norm([x_goal y_goal] - [x y]) > position_accuracy && t <= t_max
% Calculate Attractive Potential
    if norm([x y]-[x_goal y_goal]) <= dstar
        nablaU_att =  zeta*([x y]-[x_goal y_goal]);
    else
        nablaU_att = dstar/norm([x y]-[x_goal y_goal]) * zeta*([x y]-[x_goal y_goal]);
    end
```

```
% Calculate final potential
    nablaU = nablaU_att;

% Calculate reference value of linear velocity (v_ref) and orientation (theta_ref)
    theta_ref = atan2(-nablaU(2), -nablaU(1));

    error_theta = theta_ref - theta;
    if abs(error_theta) <= error_theta_max
        alpha = (error_theta_max - abs(error_theta)) / error_theta_max;
        v_ref = min( alpha*norm(-nablaU), v_max );
    else
        v_ref = 0;
    end

% Simple kinematic mobile robot model
% Omitted dynamics.
    omega_ref = Kp_omega * error_theta;
    omega_ref = min( max(omega_ref, -omega_max), omega_max);

% Calculate wind effect on velocity components
    wind_vx = wind_speed * cos(wind_direction);
    wind_vy = wind_speed * sin(wind_direction);

% Update velocity components with wind effect
    vx_with_wind = v_ref * cos(theta) + wind_vx;
    vy_with_wind = v_ref * sin(theta) + wind_vy;

% Update UAV position with wind-disturbed velocity components
    x = x + vx_with_wind * dT;
    y = y + vy_with_wind * dT;

% Update orientation
    theta = theta + omega_ref * dT;

    t = t + 1;

% Obtain the current position and distance between UAV and goal
    disp("Current position: (" + x + ", " + y + "), Distance to goal: " + norm([x_goal
y_goal] - [x y]));

% Archive and plot it
    X(t) = x;
    Y(t) = y;
    cla;
    daspect([1 1 1]);
    xlim([-0.5,4]);  ylim([-0.5 4]);
    box on; hold on;
    plot(x_goal, y_goal, 'ob');
% Plot traveled path
    plot(X(1:t), Y(1:t), '-b');
```

```
% Plot reference orientation of the robot
    plot([x x+0.2*cos(theta_ref)], [y y+0.2*sin(theta_ref)], '-g');
% Plot orientation of the robot
    plot([x x+0.2*cos(theta)], [y y+0.2*sin(theta)], '-r');
    xlabel('X position');
    ylabel('Y position');
    title('UAV Trajectory Without Obstacle in Wind Condition');
    legend('Goal position','UAV trajectory', 'Location','northeastoutside');
    drawnow;
    pause(dT);
end

t = t * dT; % scale from iterations to [s]

disp("Travel time: " + t);
```

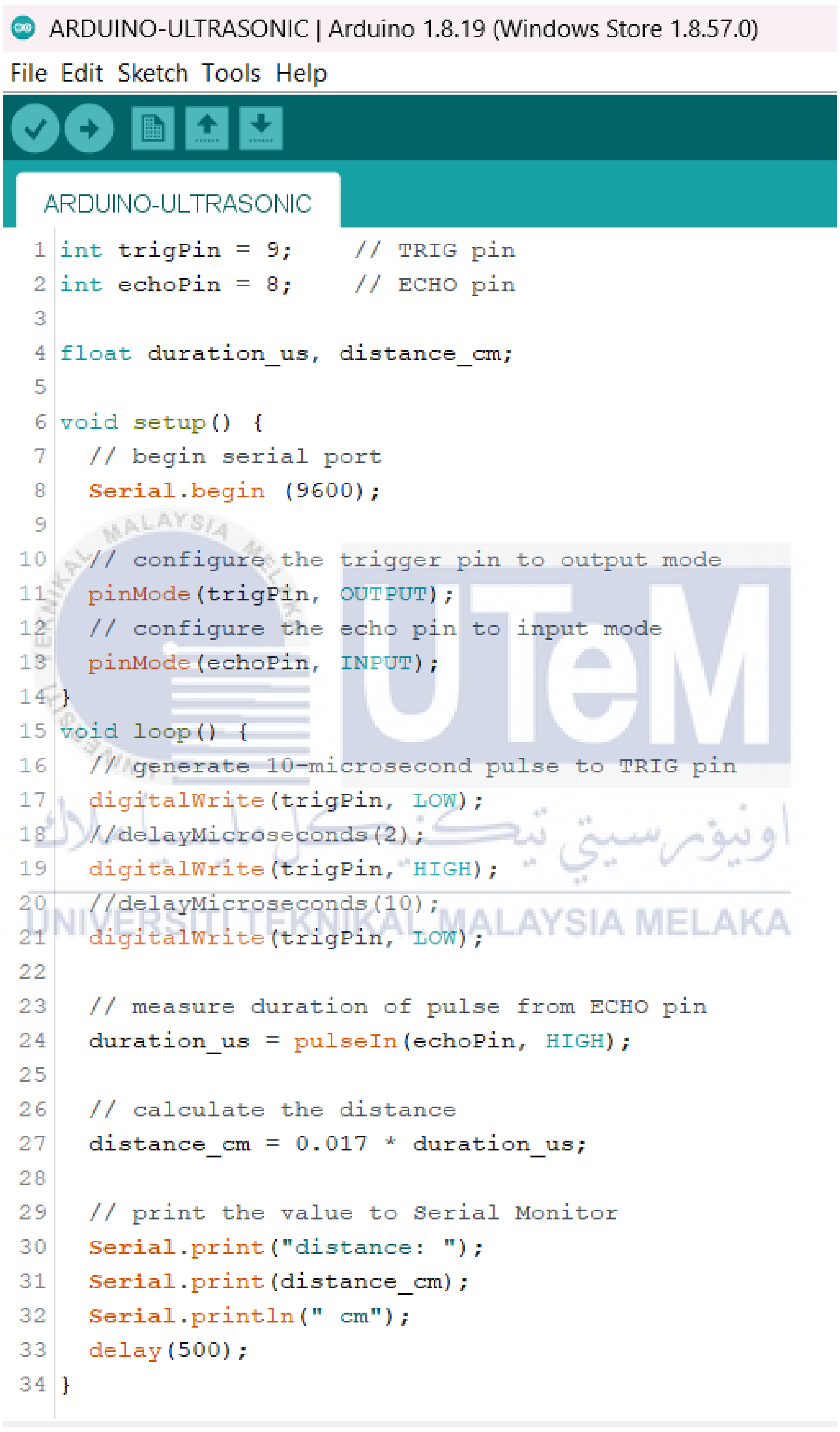
APPENDIX I MATLAB CODE FOR ENVIRONMENT WITH WIND DISTURBANCES AND 3 OBSTACLES

```
clear all; close all; clc;

% Initial position and orientation
x = 0;
y = 0;
theta = 0;

% Goal position
x_goal = 3.5;
y_goal = 3.5;
position_accuracy = 0.05;

% APF parameters
zeta = 1.1547;
eta = 0.0732;
dstar = 0.3;
Qstar = 0.75;

% Parameters related to kinematic model
error_theta_max = deg2rad(45);
v_max = 0.2;
Kp_omega = 1.5;
omega_max = 0.5*pi;

% Generate obstacles
obst1_points = [ linspace(1,2,100) linspace(2,2,100) linspace(2,1,100)
linspace(1,1,100)
               linspace(1,1,100)   linspace(1,1.5,100)   linspace(1.5,1.5,100)
linspace(1.5,1,100) ];
obst1_points(1,:) = obst1_points(1,:) - 0.5;
obst1_points(2,:) = obst1_points(2,:) - 1;

obst2_points = [ linspace(1.5,2,100) linspace(2,2,100) linspace(2,1.5,100)
linspace(1.5,1.5,100)
               linspace(1.5,1.5,100)   linspace(1.5,2,100)   linspace(2,2,100)
linspace(2,1.5,100) ];
obst2_points(1,:) = obst2_points(1,:) - 0.5;
obst2_points(2,:) = obst2_points(2,:) - 1;

obst3_points = [ linspace(1.5,2.5,100) linspace(2.5,2.5,100) linspace(2.5,1.5,100)
linspace(1.5,1.5,100)
               linspace(2.5,2.5,100)   linspace(2.5,3,100)   linspace(3,3,100)
linspace(3,2.5,100) ];
obst3_points(1,:) = obst3_points(1,:) - 0.5;
obst3_points(2,:) = obst3_points(2,:) - 0.5;
```

```
obst4_points = [ linspace(3.5,4,100) linspace(4,4,100) linspace(4,3.5,100)
linspace(3.5,3.5,100)
            linspace(2.5,2.5,100)   linspace(2.5,3.5,100)   linspace(3.5,3.5,100)
linspace(3.5,2.5,100) ];
obst4_points(1,:) = obst4_points(1,:) - 0.5;
obst4_points(2,:) = obst4_points(2,:) - 1;

% Additional parameters for wind
wind_speed = 0.1;  % Wind speed
wind_direction = deg2rad(90);  % Wind direction in radians

figure(1);
t = 1;
dT = 0.1;
t_max = 5000;
X = zeros(1,t_max);
Y = zeros(1,t_max);
X(1) = x;
Y(1) = y;

while norm([x_goal y_goal] - [x y]) > position_accuracy && t <= t_max
% Calculate Attractive Potential
   if norm([x y]-[x_goal y_goal]) <= dstar
      nablaU_att =  zeta*([x y]-[x_goal y_goal]);
   else
      nablaU_att = dstar/norm([x y]-[x_goal y_goal]) * zeta*([x y]-[x_goal y_goal]);
   end

% Find the minimum distance from the obstacle
   [obst1_idx, obst1_dist] = dsearchn(obst1_points', [x y]);
   [obst2_idx, obst2_dist] = dsearchn(obst2_points', [x y]);
   [obst3_idx, obst3_dist] = dsearchn(obst3_points', [x y]);
   [obst4_idx, obst4_dist] = dsearchn(obst4_points', [x y]);

% Calculate Repulsive Potential
   nablaU_rep = [0 0];
   if obst1_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst1_dist) * 1/obst1_dist^2)*([x y] -
[obst1_points(1,obst1_idx)  obst1_points(2,obst1_idx)]);
   end
   if obst2_dist <= Qstar  && ~inpolygon(x,y,obst2_points(1,:),obst2_points(2,:))
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst2_dist) * 1/obst2_dist^2)*([x y] -
[obst2_points(1,obst2_idx)  obst2_points(2,obst2_idx)]);
   end
   if obst3_dist <= Qstar
      nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst3_dist) * 1/obst3_dist^2)*([x y] -
[obst3_points(1,obst3_idx)  obst3_points(2,obst3_idx)]);
   end
   if obst4_dist <= Qstar
```

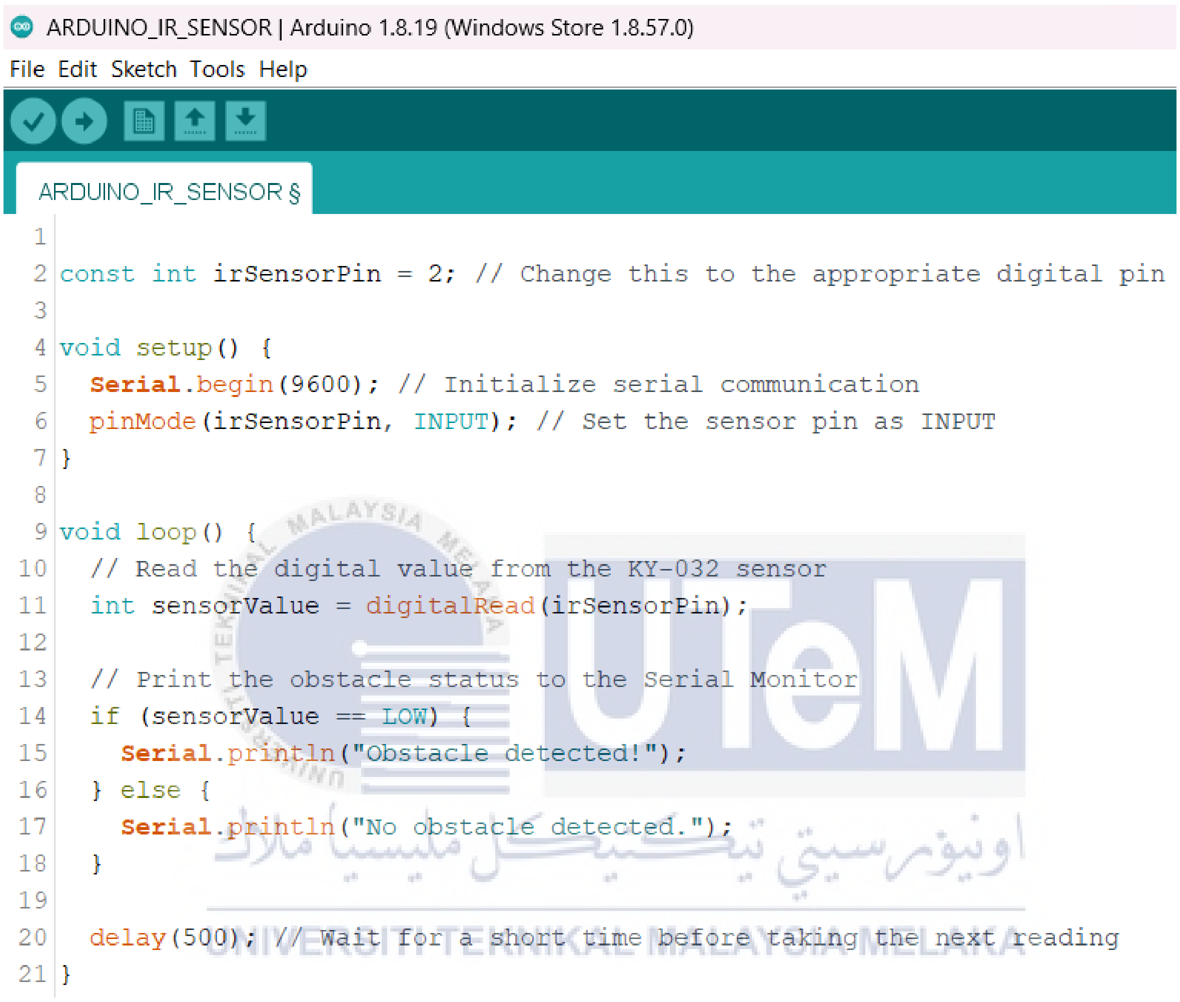
```
        nablaU_rep = nablaU_rep + (eta*(1/Qstar - 1/obst4_dist) * 1/obst4_dist^2)*([x y] - [obst4_points(1,obst4_idx)  obst4_points(2,obst4_idx)]);
    end

% Calculate final potential
    nablaU = nablaU_att+nablaU_rep;

% Calculate reference value of linear velocity (v_ref) and orientation (theta_ref)
    theta_ref = atan2(-nablaU(2), -nablaU(1));

    error_theta = theta_ref - theta;
    if abs(error_theta) <= error_theta_max
        alpha = (error_theta_max - abs(error_theta)) / error_theta_max;
        v_ref = min( alpha*norm(-nablaU), v_max );
    else
        v_ref = 0;
    end

% Simple kinematic mobile robot model
% Omitted dynamics.
    omega_ref = Kp_omega * error_theta;
    omega_ref = min( max(omega_ref, -omega_max), omega_max);

% Calculate wind effect on velocity components
    wind_vx = wind_speed * cos(wind_direction);
    wind_vy = wind_speed * sin(wind_direction);

% Update velocity components with wind effect
    vx_with_wind = v_ref * cos(theta) + wind_vx;
    vy_with_wind = v_ref * sin(theta) + wind_vy;

% Update UAV position with wind-disturbed velocity components
    x = x + vx_with_wind * dT;
    y = y + vy_with_wind * dT;

% Update orientation
    theta = theta + omega_ref * dT;

    t = t + 1;

% Obtain the current position and distance between UAV and goal
    disp("Current position: (" + x + ", " + y + "), Distance to goal: " + norm([x_goal y_goal] - [x y]));

% Archive and plot it
    X(t) = x;
    Y(t) = y;
    cla;
    daspect([1 1 1]);
    xlim([-0.5,4]);  ylim([-0.5 4]);
    box on; hold on;
    plot(obst1_points(1,:), obst1_points(2,:), '-r');
    plot(obst2_points(1,:), obst2_points(2,:), '-r');
    plot(obst3_points(1,:), obst3_points(2,:), '-r');
    plot(obst4_points(1,:), obst4_points(2,:), 'r');
    plot(x_goal, y_goal, 'ob');
% Plot traveled path
    plot(X(1:t), Y(1:t), '-b');
% Plot reference orientation of the robot
    plot([x x+0.2*cos(theta_ref)], [y y+0.2*sin(theta_ref)], '-g');
% Plot orientation of the robot
    plot([x x+0.2*cos(theta)], [y y+0.2*sin(theta)], '-r');
    xlabel('X position');
    ylabel('Y position');
    title('UAV Trajectory With 3 Obstacles In Wind Condition');
    legend('Obstacle',",",",'Goal position','UAV trajectory', 'Location','northeastoutside');
    drawnow;
    pause(dT);
end

t = t * dT; % scale from iterations to [s]

disp("Travel time: " + t);
```

APPENDIX J CODE TO CONNECT ULTRASONIC SENSOR WITH ARDUINO UNO


ARDUINO-ULTRASONIC | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
ARDUINO-ULTRASONIC

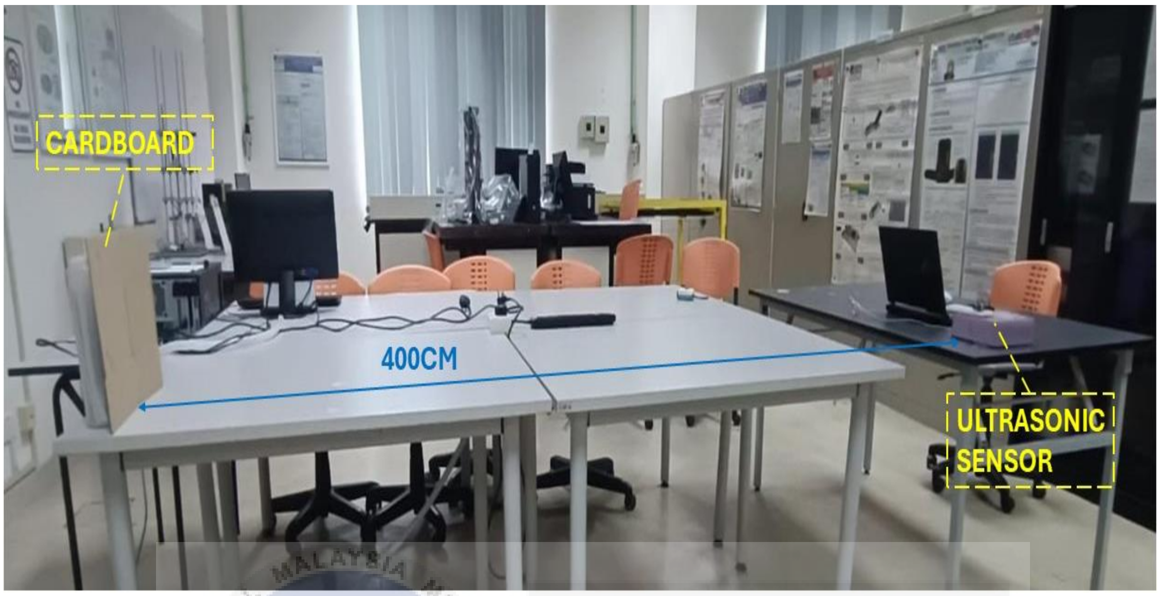
```
int trigPin = 9;     // TRIG pin
int echoPin = 8;     // ECHO pin

float duration_us, distance_cm;

void setup() {
  // begin serial port
  Serial.begin (9600);

  // configure the trigger pin to output mode
  pinMode(trigPin, OUTPUT);
  // configure the echo pin to input mode
  pinMode(echoPin, INPUT);
}
void loop() {
  // generate 10-microsecond pulse to TRIG pin
  digitalWrite(trigPin, LOW);
  //delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  //delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // measure duration of pulse from ECHO pin
  duration_us = pulseIn(echoPin, HIGH);

  // calculate the distance
  distance_cm = 0.017 * duration_us;

  // print the value to Serial Monitor
  Serial.print("distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");
  delay(500);
}
```

APPENDIX K CODE TO CONNECT INFRARED SENSOR WITH ARDUINO UNO

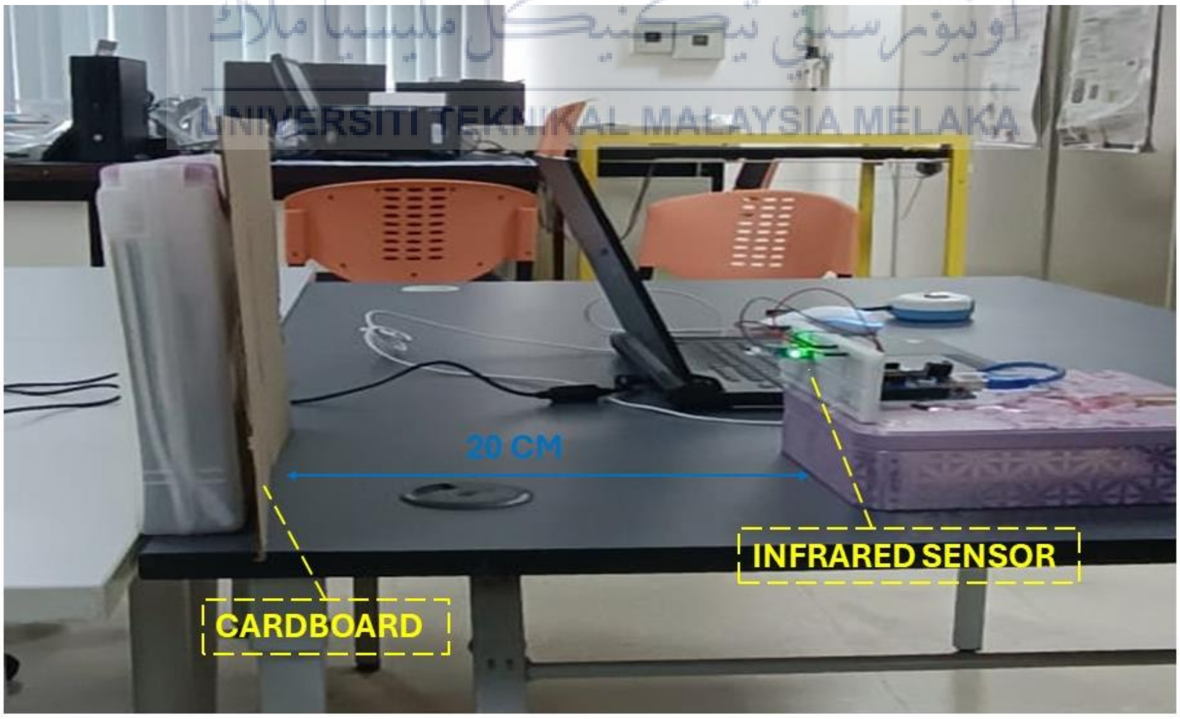

ARDUINO_IR_SENSOR | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
ARDUINO_IR_SENSOR §

```

const int irSensorPin = 2; // Change this to the appropriate digital pin

void setup() {
  Serial.begin(9600); // Initialize serial communication
  pinMode(irSensorPin, INPUT); // Set the sensor pin as INPUT
}

void loop() {
  // Read the digital value from the KY-032 sensor
  int sensorValue = digitalRead(irSensorPin);

  // Print the obstacle status to the Serial Monitor
  if (sensorValue == LOW) {
    Serial.println("Obstacle detected!");
  } else {
    Serial.println("No obstacle detected.");
  }

  delay(500); // Wait for a short time before taking the next reading
}
```

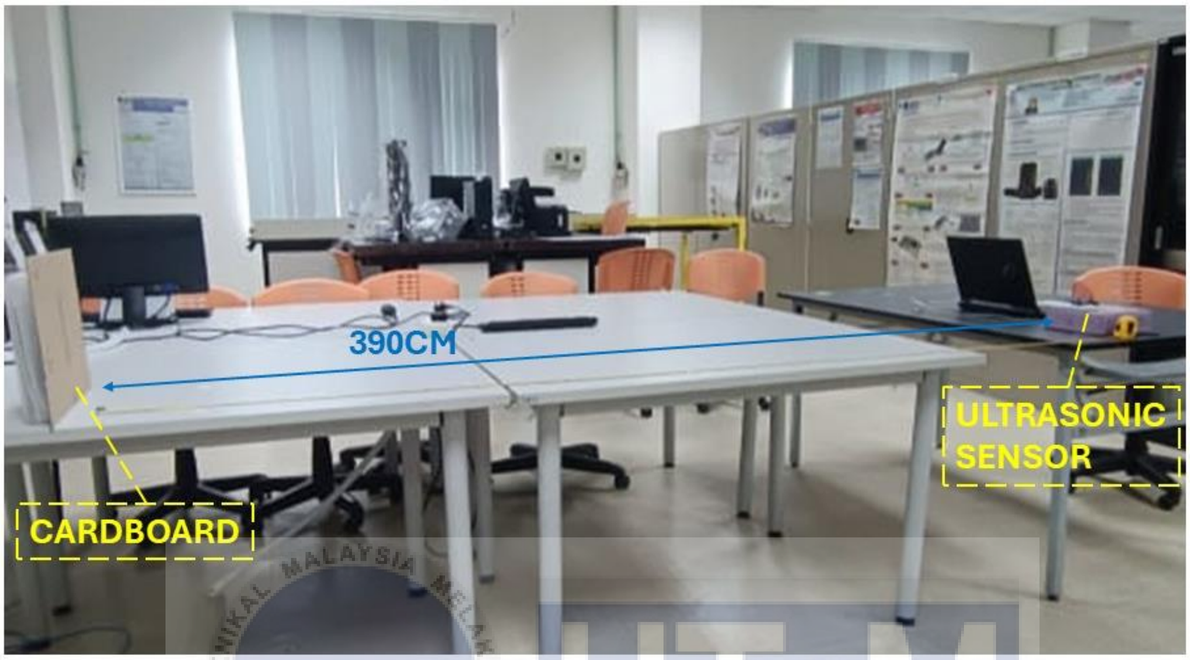
APPENDIX L EXPERIMENT SETUP FOR MAXIMUM DETECTION RANGE OF ULTRASONIC SENSOR



APPENDIX M EXPERIMENT SETUP FOR MAXIMUM DETECTION RANGE OF INFRARED SENSOR



APPENDIX N EXPERIMENT SETUP FOR ACCURACY OF ULTRASONIC SENSOR



APPENDIX O EXPERIMENT SETUP FOR ACCURACY OF INFRARED SENSOR