Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
Information Technology
Department of Network Information

REAL-TIME OBJECT DETECTION SYSTEM USING UNMANNED AERIAL VEHICLES (UAVS) AND DEEP LEARNING

A Thesis

Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology-Information Networks

Mustafa Fahem Ibrahim Mohammed

Supervised by

Asst. Prof. Dr. Mehdi Ebady Manaa Mehdi

2022 A.D. **1444 A.H.**

Supervisor Certification

I certify that the thesis entitled (**REAL-TIME OBJECT DETECTION SYSTEM USING UNMANNED AERIAL VEHICLES (UAVS) AND DEEP LEARNING**) was prepared under my supervision at the department of Information Networks/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology-Information Networks.

Signature:

Supervisor Name: Asst. Prof. Dr. Mehdi Ebady Manaa

Date: / /2022

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "REAL-TIME OBJECT DETECTION SYSTEM USING UNMANNED AERIAL VEHICLES (UAVS) AND DEEP LEARNING" for debate by the examination committee.

Signature:

Prof. Dr. Saad Talib Hasson

Head of Information Networks Department

Date: / /2022

Certification of the Examination Committee

We hereby certify that we have studied the thesis entitled (**REAL-TIME OBJECT DETECTION SYSTEM USING UNMANNED AERIAL VEHICLES (UAVS) AND DEEP LEARNING**) presented by the student (**Mustafa Fahem ALBAGHDADI**) and examined him/her in its content and what is related to it, and that, in our opinion, it is adequate with (**Viva Result**) standing as a thesis for the degree of Master in Information Technology-Information Networks.

Signature:
Name:
Title:
Date: / / 2022
(**Chairman**)

Signature:
Name:
Title:
Date: / / 2022
(**Member**)

Signature:
Name:
Title:
Date: / / 2022
(**Member**)

Signature:
Name:
Title:
Date: / / 2022
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Title: Professor
Date: / / 2022
(**Dean of Collage of Information Technology**)

Dedication

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my parents whose words of encouragement and push for tenacity ring in my ears.

Acknowledgement

Foremost, praise be to God Almighty for giving me the motivation and ability to accomplish this thesis and its success.

I would like to express my sincere gratitude and thanks to my supervisor, **Asst. Prof. Dr. Mehdi Ebady Manaa** for the continuous support of my MSc study through his scientific suggestions, guidance, and assistance to carry out present research.

My great thank also goes to department of Information Networks/ College of Information Technology/ University of Babylon, and all the members of Information Networks department for their support and help.

My deepest gratitude goes to my dear parents, they have always been so close to me.

I would also like to thank my brother Dr. Ahmed Fahem Al Baghdadi for supporting me a lot and help me all the period of this thesis.

Mustafa Fahem Ibrahim Mohammed

Abstract

The applications of unmanned aerial vehicles (UAV) increase in recent periods. It began to play an essential role in various fields. It becomes necessary to develop appropriate systems capable of performing a specific automated function such as detecting and locating an object accurately.

There are multiple problem will be addressed in this thesis which are: Complex environments, hardness of stability with the existence of disturbances factors, difficulties of flying over some targets, real time detection and communication between quadcopter and base station, and localize a known target position.

The proposed system must achieve four essential objectives in order to perform the function with high accuracy. The first objective is to define a safe path within a multi-obstacle environment for the drone. The second objective is to detect a specific target. The third objective is to locate a known target and calculate its coordinates. The fourth goal is to build a secure communication network between the UAV and the ground station in order to transmit these coordinates in real-time.

The first objective is achieved by developing a path planning algorithm capable of finding the best path within a multi-obstacle three-dimensional environment. The second objective is achieved by using deep learning to detect objects through the You only look once (YOLO) algorithm. The third objective is achieved by developing an algorithm that determines the location of the target by knowing the location of the UAV and the direction vector of the camera. The fourth objective is achieved by establishing a 2.4 -frequency connection using the TCP protocol and using the RSA encryption algorithm.

A quadcopter is built and all the aforementioned methods were applied to it. The UAV was able to fly stably. Where the specific path is tracked and the target is detected with an accuracy of 93% based on YOLO algorithm. The

computational complexities of the proposed path planning algorithm was improved by around 27% from the nearest algorithm which are potential field algorithm. The localization algorithm was simulated the accuracy increased can be with the increasing of the number of views.

Declaration Associated with this Thesis

Some of the works presented in this thesis have been published as listed below.

[1] Mustafa Fahem Albaghdadi, Mehdi Ebady Manaa, “Unmanned Aerial Vehicles (UAVs) and Machine Learning for Detecting Objects in Real Time”, *Bulletin of Electrical Engineering and Informatics, 2022*.

[2] Mustafa Fahem ALBAGHDADI, Mehdi Ebady Manaa, Ahmed Fahem ALBAGHDADI , “A new algorithm of path planning in 3D environments for an implemented quadcopter robot”, 1st International Conference on Universities Promote Sustainable Development Goals , Turkey, 2022.

Table of Contents

List of Tables

List of Figures

Table of Abbreviation

Abbreviation	Definition
3D	Three-dimensional
A*	A Star
AI	Artificial intelligence
ASIC	Application-specific integrated circuits
CNN	Convolutional neural network
DFS	Depth-first search
DSP	Digital signal processors
FPGA	Field-programmable gate arrays
GA	Genetic algorithm
GPS	Global Positioning System
GPU	Graphics processing unit
GUI	Graphical user interface
GWO	Gray wolf optimization
mAP	Mean Average Precision
Mbps	Megabits per second
MCFO	Modified central force optimization
PSO	Particle swarm optimization
R-FCN	Region-based Fully Convolutional Network
RNN	Recurrent neural network
RRT	Rapidly exploring random tree
RSA	Rivest-Shamir-Adleman
SIFT	Scale and rotation invariant
SSD	Single-shot detector
TCP/IP	Transmission Control Protocol/Internet Protocol
UAV	Unmanned aerial vehicles
YOLO	You only look once

CHAPTER ONE

INTRODUCTION

1.1 Overview

Autonomous Unmanned aerial vehicle (UAV) have many applications such as rescue, surveillance, and others have sparked increased interest in recent years. Visual object identification is a key component in such UAV applications, and it's essential for developing completely autonomous systems. Object recognition faces many challenges that can be summarized and not limited to the following: Limited camera resolution, noise, continuous motion, and rapid data processing in real time [1].

Autonomous systems such as recognizing faces or objects, self-driving systems and voice recognition systems have spread recently, due to their many applications. Systems that rely on deep learning have spread more widely for recognition, detection, classification, and others. convolutional neural network (CNN) is regarded as a highly strong technology in the field of object identification and categorization. CNNs are hierarchical models that are biologically inspired. The CNN's structure usually starts with a feature extractor and ends with a classifier. A lot of progress has been achieved in the field of CNN-based object identification in recently [2] .

There are many object detection algorithms that can be adopted to obtain real time detection. It can be concluded simply that the algorithms achieve the detection process at different speeds, for example, the Region-based Fully Convolutional Network (R-FCN) and Region-based Convolutional Neural Network (R-CNN) algorithm depends on two stages in the detection, namely, determining the important area and then conducting the detection process. On the other hand, there are other algorithms that achieve the goal step by step and are more suitable in projects that require real-time detection. Especially when the target and

the camera are in constant motion, such as single-shot detector (SSD) and You only look once YOLO algorithms.

A three-dimensional (3D) path-planning algorithm attempts to determine the minimum collision-free path for the robot to move in the workspace with an obstacle. The minimum path requires the minimum cost according to an objective function and is not always the shortest path. 3D path planning is an essential tool for autonomous robots, owing to the fundamental role played by robots in the economy. The importance of 3D path planning appears in numerous aspects, such as when human safety depends on the robot reaching its target in minimum time or in industrial applications, repetitive tasks can improve and increase productivity. The input to the path planning (motion planning) algorithms are environment description and required function. The output of the path planning algorithms is the path that achieves the required task in the given environment. There are multiple applications of path planning such as video games in addition to modern world applications [3].

Autonomous robot applications increased rapidly. So, it is possible to say about path planning: It is a vital tool in the new world.

Object detection is used extensively in computer vision tasks, such as pedestrian identification, face recognition, face detection, and so on. Currently, the methodology to object recognition has essentially grown into two distinct categories: classical machine learning and deep learning. Traditional machine learning approaches must first create feature extraction, which is often artificially set and of which there are about three types. The general Hough transform suggested by Ballard D in 1981 may be considered as an effective method for the extraction of all geometric features. The second technique, such as the 1988-proposed

Harris corner detector, extracts object characteristics by identifying corners. To recognize objects, this approach pulls corner characteristics from two pictures and evaluates the degree of correlation between their points. The two aforementioned approaches are sensitive to the picture's geometric properties, i.e. a change in image size, rotation, or grayscale value may impact the final results. Lowe suggested SIFT (scale and rotation invariant) in 2004 as the third approach. The SIFT method is used to identify and characterize pictures' local characteristics. It considers a feature as an object, therefore picture rotation and scaling have no effect on the output. Deep learning approaches often rely less on successful feature engineering than classic machine learning methods, and thus outperform traditional machine learning methods in big data size training. With the fast growth of deep learning over the last decade, object identification has also joined the deep learning movement.

In 2014, R. Girshick and colleagues introduced the R-CNN model, the first method to effectively use deep learning to object identification. Fast R-CNN and Faster R-CNN were introduced in 2015 as an upgrade to R-CNN. Both of them have improved the calculation and training efficiency of R-CNN to some degree. Also in 2015, Redmon j et al. introduced the YOLO model. It enables real-time monitoring of objects. In 2017, YOLO version 2 was presented to optimize and increase the accuracy of the YOLO model. Mask R-CNN and YOLO version 3 were released in 2018 as improvements on Faster R-CNN and YOLO version 2, respectively [4].

1.2 Problem Statement

Due the developments in technology, software, and hardware, QUARD-COPTERs are becoming easier to use in commercial and

military applications. QUARD-COPTER may be utilized for a number of tasks, including monitoring, search and rescue, and more. As a result, a precise location of objects must be returned. The QUARD-COPTER encountered several challenges:

- Traffic and Obstacle Detection
- Navigation
- Communication
- Major terrain mapping difficulties
- Weather conditions and area to map

1.3 Aim of Thesis

The essential aim of this thesis is localizing a known object in a known area using drone and camera. This aim can be achieved by the following objectives :

- To Design and implement a QUAD-COPTER based on YOLO that use to detect object with high-currency and return the accurate position
- To develop QUAD-COPTER works independently with no connection to finish up its work and return to the base station.
- Develop Real-time position detection system to send data to base station.
- Built Extreme accurate detection from different views to the same object.
- Propose an algorithm to plan the path of a QUAD-COPTER within a three-dimensional environment with fixed obstacles.

1.4 Related Works

The related works can be classified into three categories as following.

1.4.1 Related works on object detection and localization

CARPK is the world's biggest drone view dataset, is generated by Meng-Ru Hsieh, Yen-Liang Lin (2018) [5], In a large-scale automobile counting job, it is a tough dataset for varied sceneries of parking lots. a novel method was developed for producing feasible region recommendations in the study, which takes advantage of spatial layout information for an item counting job with regularized structures. With past knowledge of item arrangement patterns, the learnt deep model can better count objects.

Yongxi Lu, Zeyangyi Wang (2018) [6], the authors of this paper looked into the difficulty of employing drones to find targets. They created a testbed to study real-world scenarios for this application. They discovered that persistent elements in the environment might have an unanticipated influence on the output of the observations after testing perception modules with modern computer vision algorithms. Such parameters have a considerable impact on the design of the search algorithm, according to the comprehensive simulations. Future initiatives include better describing the influence of persistent features, creating ways to automatically identify the presence of such structures to avoid over-modeling, and constructing more robust computer vision algorithms for target search and other drone-related applications.

P.J. Baeck, N. Lewyckyj (2019) [7], demonstrated a drone image processing chain that can recognize human being, location, and show the findings in a GIS context. The authors recognize that by train system

with bigger and dataset, they made a robust model can achieves their goals. Additionally, work must be done to enhance the model's run time. The research advances method for converting pixel coordinates in individual photographs to real-world locations. Because It don't work directly on the orthostatic, The authors get high accuracy duo to the multi reasons

Haotian Zhang, Gaoang W ang (2019) [8], Developed a platform for drones to locate and track objects in three-dimensional space. The researcher relied on the principles of CNN to identify things and reveal multiple things. the system resilience is demonstrated by its ability to handle the majority of drone scenarios, including occlusion management and camera quick movements. However, there are certain limitations to proposed work. Although that the quick camera movements do not influence tracking performance, they do affect group plane estimates.

1.4.2 Related work on Path planning

Path planning is essential for unmanned aerial vehicles (UAV). It is a complicated optimization problem that can allow them to compute the best possible path according to an algorithm autonomously and to achieve the requirements under constraints. There are many algorithms constructed to solve the problem of path planning. Most algorithms have evolved over time with multiple steps.

The potential field describes as space and potential hills at all obstacles. The robot in the potential field is moving toward the goal location while being repelled from obstacles in the workspace. According to the definition above, the mobile robot will move naturally to the goal point without collision [9][10][11]. The quality of the proposed path has been improved using the Particle swarm optimization (PSO) algorithm. A

local minima problem occurs when the equivalent forces equal zero. The local minimum drawbacks solved using the PSO algorithm [12][13][14].

The Probabilistic road map (PRM) algorithm in 1996 by Kavraki [15] takes random samples in free space and connects with nearby points after checking that the path between the new sample and each nearby point is collision-free. After that, the graph-search algorithm is to be applied to the resulting graph to obtain the suboptimal path.

The RRT algorithm by LaValle et al. 1998 [16] generates a node for each step. If the extension between the added node and nearest node does not intersect an obstacle, the new node is confirmed. On the other side, when it intersects an obstacle, it is erased. Thus, the RRT method requires a costly computation, because it must explore all regions, particularly when the workspace is cluttered. The workspace is sampled as a set of nodes, and the feasible path is searched [16][17][18]. Liang Yang et al. 2016 in [19] said that all Sampling-based algorithms, including the rapidly exploring random tree (RRT), dynamic domain.

It is possible to represent the path-planning problem as an optimization problem to solve it using the imperialist competitive algorithm [20]. Modified central force optimization (MCFO) algorithm proposed by Chen et al. 2016 for 3D motion planning use the principle of the PSO and the variation in the information of genetic algorithm (GA) are introduced into the central force optimization CFO to get better performance of CFO. Then an improved CFO algorithm is employed to explore the suboptimal track in 3D environments with present obstacles. The model of the resulting path is a group of broken lines, So the smoothness of the path is weak. The mutation operator is an essential fundamental operator in the GA used in order to avoid the local minimum trouble. The length of the resulting path from MCFO has

unnecessary additional length due to broken lines in the path. The increasing number of line segment lead to increase corner point. The robot must decrease its velocity to zero in order to not violate any dynamic constraint. These interrupts cause increasing flight time. These discontinuities removed by connecting each line segment by an arc (the segment of circular). The radius of the arc is calculated based on the speed and acceleration of the robot — this method of smoothness leads to violating dynamic constraints some times. The violation happens when the segments of lines very close to the obstacles. In this case, the algorithm ignores the smoothing [21].

1.4.3 Related work on UAV system

Markus Achtelik et al. 2009 [22] design and implement a complete system to control quad rotor aerial vehicle based on visual feedback. Active markers are designed and used in order to decrease the effect of light noise on the feedback system. The main properties of the system are simple setup and low cost. Adaptive Control system for a quadrotor copter used by Michael Achtelik et al. 2011 [23] in the presence of unmodeled dynamics, uncertain parameters, failure situations, or external disturbances, while Inkyu Sa in 2011 [24] presents control and modeling of an open-source quad rotor copter. Inkyu Sa et al 2012 [25] describes quad rotor copter system based on estimation, identification, and control.

Bernard et al. 2012 [26] provide a low-cost control system for the quad rotor copter to provide easy control for beginners and auto flying. M. Anwar et al. 2013 [27] present a swarm of UAV or quadrotor for object localization and tracking for military purposes. Modified Particle Swarm

Optimization algorithm used in the system to achieve better performance.

Simple graphical user interface (GUI) designed by Dirman Hanafi et al. 2013 [28] to build a quadrotor copter controlled by a remote-control system. The balancing system uses the FY90 controller and IMU 5DOF. The ultrasonic sensor is used for landing purpose.

Michael Y. Chen et al. 2013 [29] made a design of a quadrotor controller based on a smartphone and Arduino. The authors use a smartphone to generate navigation commands and avoid collisions. Ultrasonic sensors used in order to explore the unknown area independently. Mehdi Fatan et al 2013 [30] use adaptive PID controller in order to control altitude of the quadrotor. The genetic algorithm used to find optimal parameters In order to increase the efficiency of controlling the performance of its tasks and eliminate disturbance. A test of BCI controlling for a quad rotor aerial vehicle was made in three dimensions workspace by Karl LaFleur et al. 2013 [31].

Carreira et al. 2013 [32] provided an autonomous landing system for a quad rotor copter based on the onboard camera.

1.5 Thesis Layout

This rest of thesis contain the following chapters:

- **Chapter 2** demonstrates basic theory information for the important subjects that have a relation with the proposal of this thesis.
- **Chapter 3** presents four proposed algorithms and system in details.
- **Chapter 4** shows the results of all proposed algorithm together with simulation and implementation results of the quad copter.
- **Chapter 5** : Conclusions and future works.

CHAPTER TWO

Background and Algorithm Description

2.1 Introduction

This chapter contains the general description about path planning as well as all theoretical side of techniques and algorithms used in the proposed system.

2.2 Overview Motion Planning Problem

In recent years, robots have significant effects, especially interference with the current applications. It requires the development of robots in a way that enables them to perform their functions in perfect ways. As a result, many research problems have arisen, such as planning paths for mobile robots that will be addressed in this chapter.

The problem of finding a path is one of the most straightforward problems for humans. Humans can decide within a fraction of a second while this decision is a significant challenge for robots. The problem is to determine a safe robot path among two points: the start and end points. This problem is solved in several ways as described in the literature [33].

Moving safely between two or more points requires algorithms that run the job. Reducing distance is one of the most popular targets for algorithms because it affects other factors such as energy consumption and time. This chapter provides a comprehensive overview of path planning and types of algorithms [34].

There is an upcoming revolution in the field of robotics where robots have proven effective in many areas such as smart home systems [35], manufacturing plants, many others. It becomes vital to develop a

mechanism to work robots in order to perform their tasks optimally. In short, the robot must find solutions of the following three questions: Where is the robot? Where is the target? How to get there safely? Many techniques and algorithms are employed to achieve the best answers to the above questions.

- Localization: A significant thing that supports the robots to locate in workspace. Such as GPS [36], ultrasonic sensor [37], cameras and laser.
- Mapping: The robot needs to map the environment in which it moves. The map assists each robot to recognize directions quickly. That map could be stored in the memory of the robot as a matrix, for example, the robot can build it gradually whenever it detects a new site [38]
- Path planning: the robot must know the target location. targets may be absolute like latitude and longitude; they may be relative like room number [39].

Route planning is one crucial aspect that can answer the simple question about the location and paths. For applications that deal with mobile robots. The robot should have the ability to arrive at the target as well as bypassing the obstacles that are spread in the work-space while avoiding all obstacles and reach the target in the shortest possible way. A smooth track is one of the common goals addressed by some research in addition to the short path. Figure 2.1 describe different issues of the path planning.

Efficiency, accuracy, and safety are the most critical concerns to be taken into account in the matter of planning the paths. Each robot must

find its path in a short period and with the least amount of energy. Besides, it should avoid all obstacles and choose the best path to reach the target safely [40]

Planning a path in a large environment is a challenge because the problem difficulties increase. These studies address the most complicated problem. a widespread environment with obstacles in a scattered and complicated way [35].

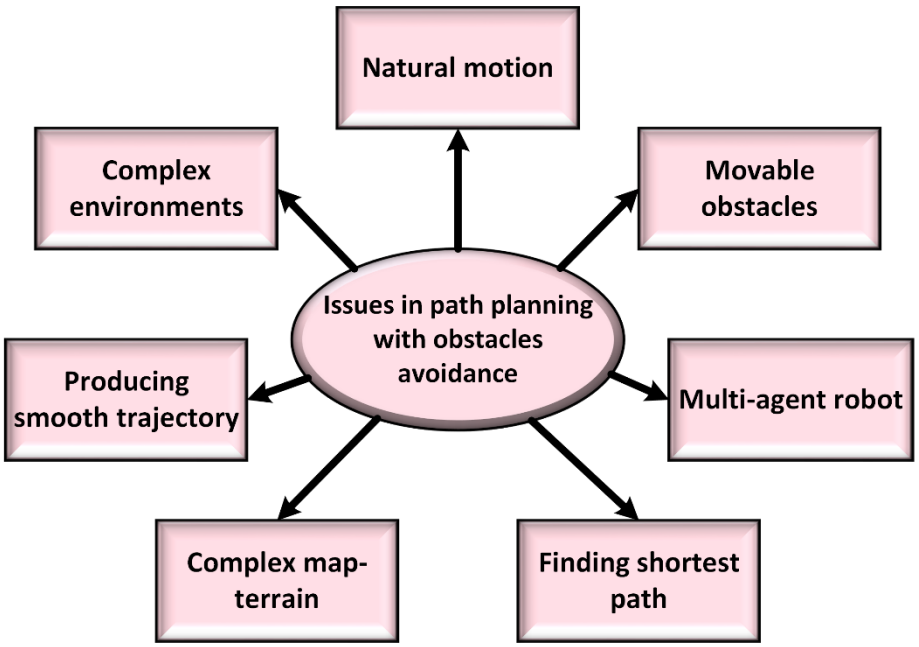


Figure 2.1: Various matters of route planning [40]

2.3 Problem Formulation

The path planning problem can be described as follows :-

- The robot within the region represented in the Euclidean space as R^2 or R^3.
- The Obstacles objects found within the region

- The location, orientation, and the geometric shape of the robot and obstacle are already obvious.
- The location of obstacles in the area is Identified.
- The path can be planned in the area Defined by a set of points

2.4 The Categories of the Motion Planning

There are three categories depending on the environment, algorithm, and the approach used to solve the problem and plan the path as shown in Figure 2.2 [40]

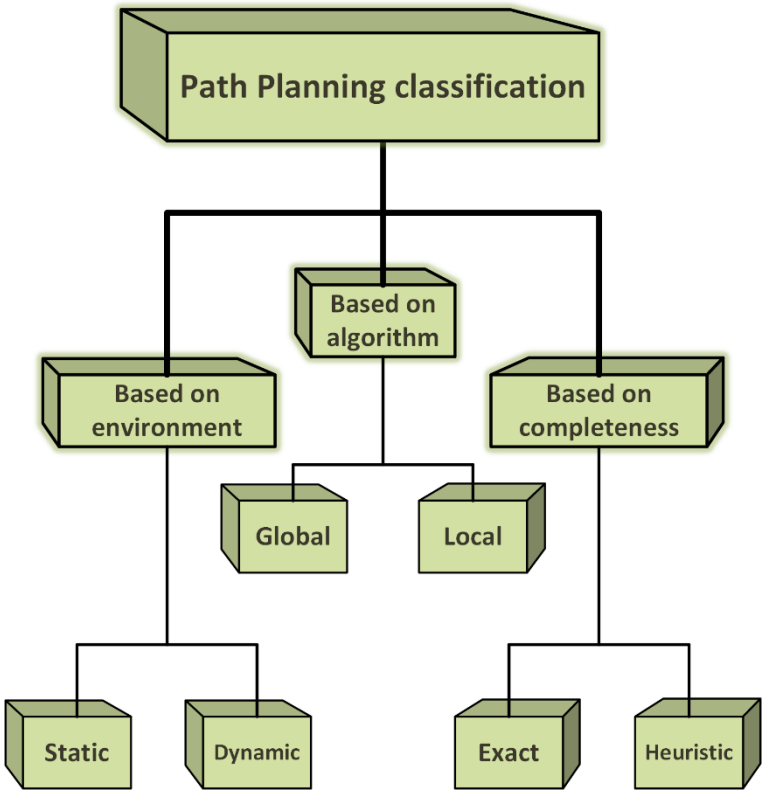


Figure 2.2: The Categories of the motion Planning [40]

Two basic kinds of environments are existing: static and dynamic. In stationary environments, obstacles and target points are fixed. In dynamic environments, the locations of obstacles and the target point are variable. New obstacles may appear in the road, or the location of the target may change. When the locations of obstacles and targets change, the process becomes more complex because the algorithm must react to real-time changes. Knowledge of the map has a significant role in determining the path. The path of the robot depends primarily on an essential reference, which is the map. The problem of mapping paths can be divided into two basic categories. In the first category, the robot knows the locations of the obstacles and the entire map this type called global route planning. In the second category, the robot does not identify accurate knowledge about the work-space. Therefore, it must sense it by different sensors and create a map during the real time at the exploration process, and this type is called the local route planning. Table (2-1) presents the differences between the Local and Global path planning [40].

Table 2-1: The differences between the Local and Global path planning.

Local	**Global**
Depend on sensors	Depend on Maps
Reactive navigation	Deliberative navigation
Fast response	Relatively slower response
incomplete given environment	Complete environment
generate path during moving	generate path before moving
Online	offline

There are two types of algorithms, whether it is accurate or heuristic, depending on the final completed path. The accurate or exact algorithm finds the optimal solution in the case that the path exists or gives an indication that there is no solution in the case that the path is not possible [40].

Algorithms can also be classified depending on the number of robots operating in the same area (single and multi-robot). Many applications need a group of robots to work together collaboratively. The algorithms here are responsible for providing a safe and short path for each robot to reach its target. The algorithm is responsible for not colliding any robots during work [40]. Static algorithm based on environment is the category used in the proposed system.

2.5 The Complexity of Path Planning

Realistically the robot is not just a point in space. The robot may be irregular, and the work environment can contain an unspecified number of dimensions. In a point representation of a robot, the number of dimensions is usually two or three, that facilitates the work of algorithms in finding the best path [41].

One of the most preferred methods to solve the problem of planning paths is to use smart techniques like A Star (A*) algorithm to lead the search to the appropriate area. The appropriate area is the area that contains the optimal solution. This method ensures that searches are not conducted in unrelated areas. Precise methods are time-consuming and cannot work in small areas or with low accuracy. It can be said that the problem of planning paths has polynomial complexity [42].

2.6 Approaches of Path Planning

It's began in the late 1960s and flourished in the 1980s [35]. A set of research has emerged to solve this problem in all types of the environments which are: static environment and dynamic environments). Many methods contribute and tried to find solutions to the problem of path planning. These initiatives or solutions can be broadly categorized into three basic categories, which are the classical approaches, the guiding approaches, and the graph-based approaches, as shown in Figure 2.3. The classical approach is a combination of classical methods such as roadmap and potential field. This category contains problems with time complexity and accuracy in optimization. The second category resolve the problems that face the first category. The third category relies on a chart in the layout of paths such as A*, and Dijkstra algorithm [40].

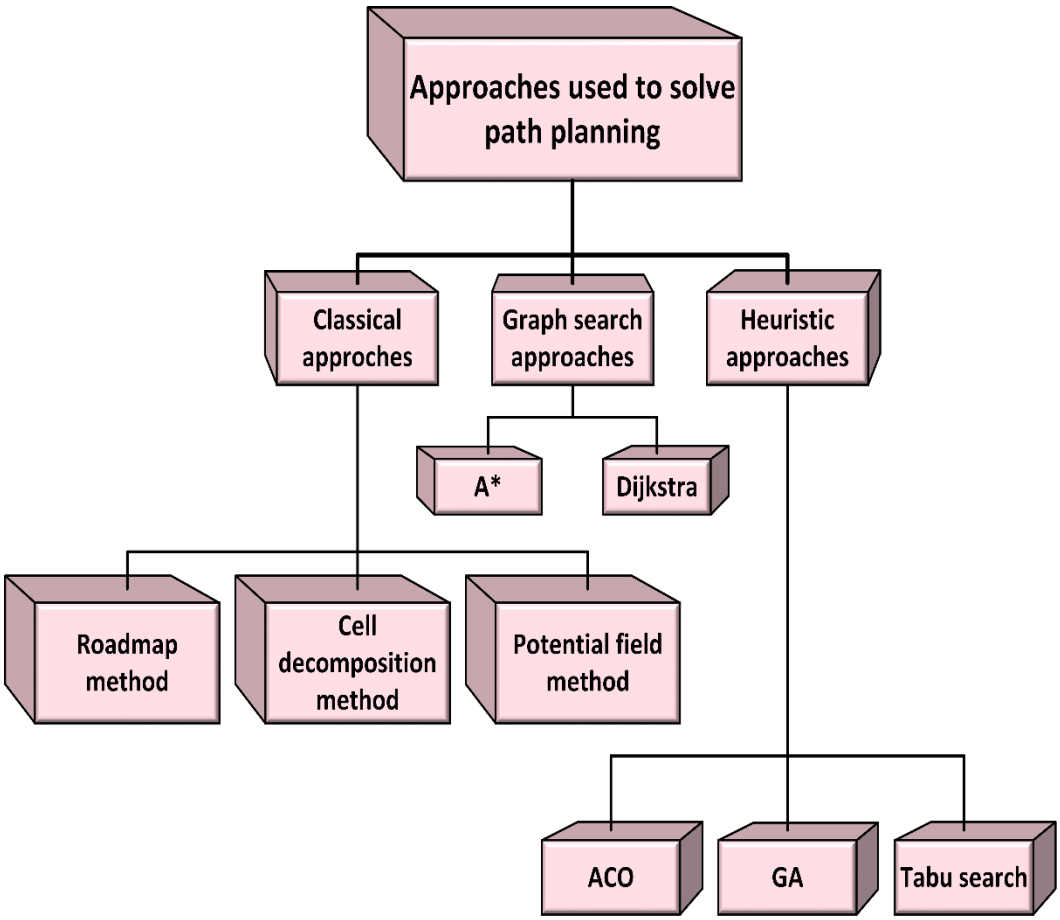


Figure 2.3:Approaches of path planning [40]

2.6.1 The Roadmap Approaches

The basic idea is to create a roadmap consisting of a set of lines. Each line connects two nodes. The line is installed as a road connecting the two nodes when it does not intersect any obstacle. A group of lines together represents a road map as shown in Figure 2.4. If there is a continuous route within the road map connecting the start and target points, then it is the chosen route. If more than one route is found within the roadmap linking the start and target points, then use an algorithm such as Dijkstra to choose the shortest or best route depending on the requirements of the problem [43][44][45].

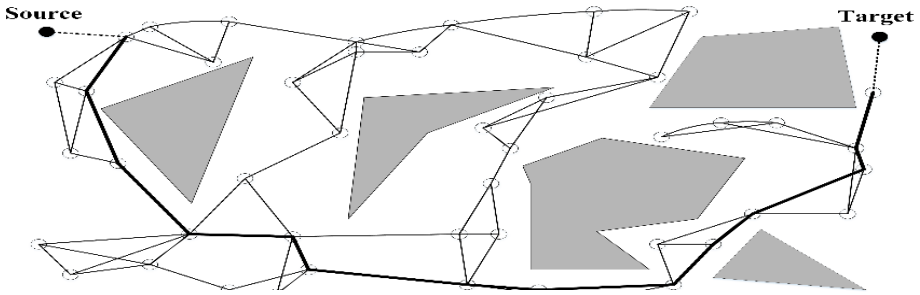
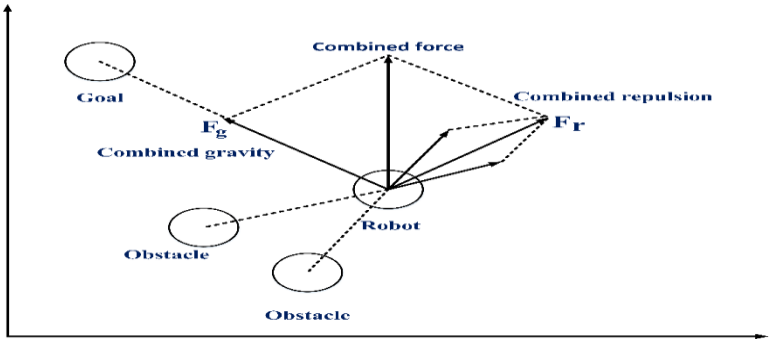


Figure 2.4: Roadmap approach of route planning [40]

2.6.2 Artificial Potential Field Algorithm

In this algorithm, it is assumed that there is a field consisting of two fundamental forces, namely attraction and repulsion. Force pulls the robot towards the target positioned at the target point. The other force is the power of repulsion. This force is positioned in the locations of obstacles. Between the forces of attraction and repulsion, the robot can find its way to the target without hitting obstacles. This algorithm has been developed many times to overcome its weaknesses (local minima). Figures 2.5 show the potential field approach of path planning [34].

Despite the ability of classical approaches to finding solutions and paths free from collisions, there are some limitations and drawbacks to their work. The most important of these constraints are long and complex calculations, especially in large and complex environments. The other downside to this approach of algorithms is their involvement in locally optimal solutions away from the global optimal solution [40] as shown in figure 2.5.



Figures 2.5 show the potential field approach of path planning [34].

2.7 Object Detection

The goal of object detection is to detect all instances of objects of one or more known classes, such as people, cars, or faces, in an image. Usually only a small number of objects are present in the image, but there are a very large number of possible locations and scales at which they can occur that need to be explored in some way. Each detection is reported with some kind of pose information [47][48]. This could be as simple as the object's position, a position and scale, a bounding box, or a segmentation mask. In other situations, the pose information is more detailed and includes the parameters of a linear or nonlinear transformation. For example, a face detector can calculate the positions of the eyes, nose, and mouth, as well as the bounding box of the face. The pose could also be defined by a three-dimensional transformation indicating the object's position in relation to the camera. Object recognition systems build a model for an object class from a set of training examples. In the case of a solid rigid object, only one example

may be required, but more generally, multiple training examples (often hundreds or thousands) are required to capture certain aspects of the variability of the class[49]. In general, less training data is needed when more information is available. The about class variability can be explicitly built into the model. However, it can be difficult to specify models that capture the wide variability found in images. An alternative approach is to use methods like Convolutional Neural Networks that learn class variability from large datasets [50][51].

Two-stage object detection refers to the use of algorithms that separate the object detection problem into the following two phases:

- Detecting possible object regions.
- Classifying the image in those regions into object classes.

Popular two-stage algorithms such as Fast-RCNN and Faster-RCNN often use a Region Proposal network that proposes regions of interest that may contain objects as shown in figure 2.6 [57][60].

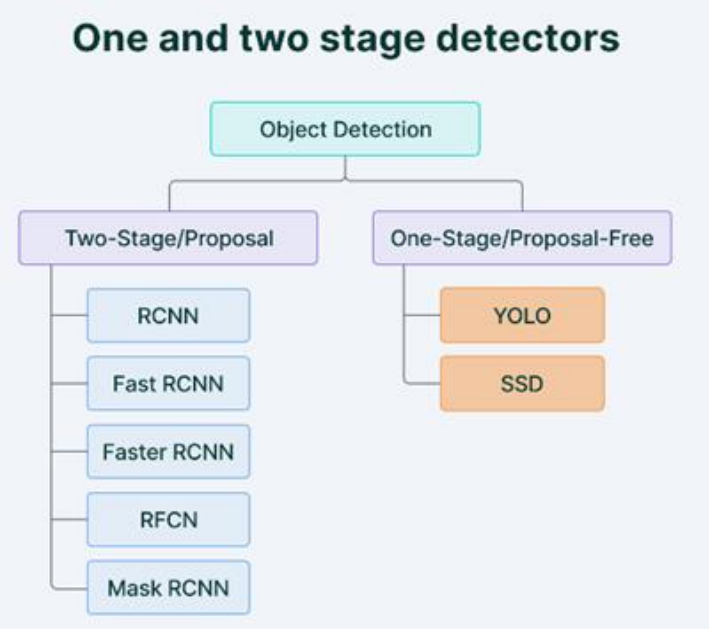


Figure 2.6: object detector [57]

2.7.1 YOLO Algorithm

You Only Look Once is the abbreviation for the term YOLO. This algorithm finds and identifies multiple objects in a photo (in real-time). YOLO performs object detection as a regression problem and provides the class probabilities of discovered photos.

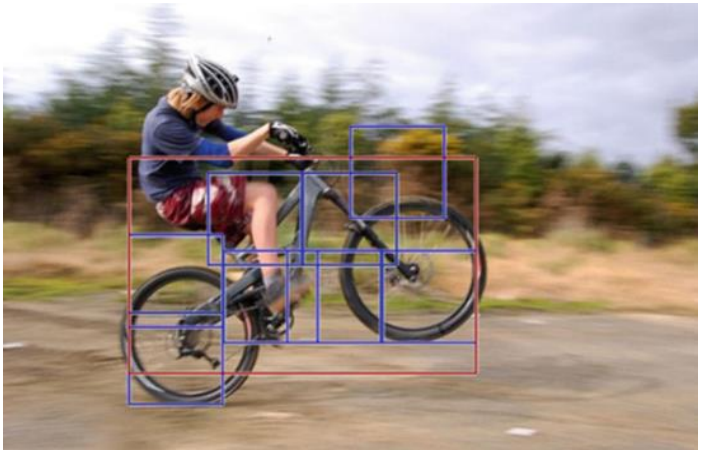


Figure 2.7: A bicycle detection using YOLO algorithm [61]

A classifier is then given the output of the RPN to classify the areas into classes. While this produces precise object identification results with a high mean Average Precision (mAP), it also causes several repetitions in the same picture, hence slowing down the algorithm's detection speed and inhibiting real-time detection [61][63].

Unlike previous object identification methods, which repurposed classifiers to do detection, YOLO proposes the usage of an end-to-end neural network that simultaneously predicts bounding boxes and class probabilities. YOLO produces state-of-the-art results in object identification by using a fundamentally different approach than existing

real-time object detection algorithms [64][66]. An example of bike detection that indicates the location of specific parts is shown in Figure 2.7.

2.7.2 YOLO methodology

The YOLO algorithm divides the picture into N grids, each of which has an equal-sized section of SxS. Each of these N grids is responsible for detecting and localizing the item included inside it. Similarly, these grids estimate B bounding box coordinates relative to their cell coordinates, as well as the item name and likelihood of the object's presence in the cell. Owing to the fact that both detection and recognition are handled by image cells, this method significantly reduces the amount of computing required, but also generates a large number of duplicate predictions due to many cells predicting the same item with various bounding box predictions. YOLO employs Non-Maximal Suppression to address this problem [63-65] as shown in figure 2.8.

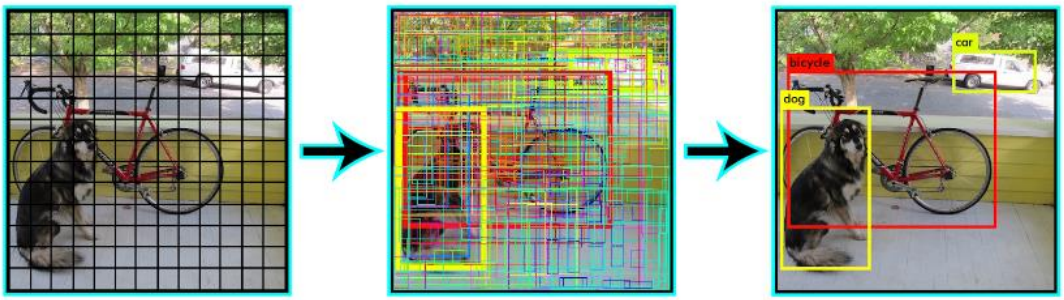


Figure 2.8: YOLO methodology [63]

during nonmaximal Suppression. YOLO does this by examining the

likelihood scores linked with each option and selecting the one with the highest score. After this, the bounding boxes with the biggest Intersection over Union with the current high probability bounding box are suppressed. This procedure is continued until the final enclosing boxes are achieved [66].

2.7.3 YOLO Steps

As shown in Figure 2.9, the Yolo algorithm divides the picture into grids such as a 3*3 grid.

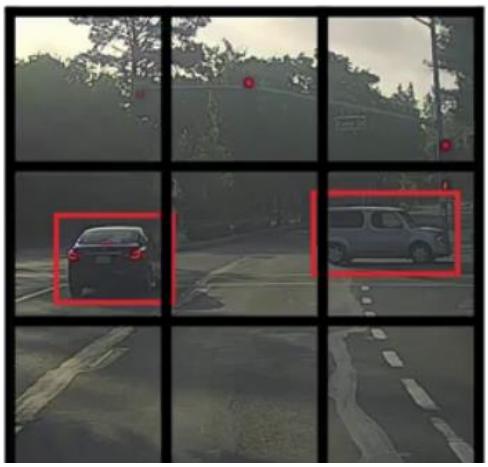


Figure 2.9:YOLO work example [63]

On each grid, image categorization and localization are applied. YOLO then forecasts the object's bounding boxes and their respective class probabilities (if any are found).

For the model to be trained, labeled data mused be given. Suppose each picture split into a 3 x 3 grid and there are a total of three classes for classification the items. Say the classes are respectively Pedestrian, Car, and Motorcycle. Figure 2.9 depicts that for each grid cell, the label y will be an eight-dimensional vector as shown in figure 2.10:

y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

Figure 2.10: eight-dimensional vector [63]

- PC determines if an item is present or absent in the grid (it is the probability)
- bx, by, bh, and bw define the bounding box if an item exists
- c1, c2, and c3 are the classes. Consequently, if the item is a vehicle, c2 will equal 1 and c1 and c3 will equal 0

YOLO process every cell in the grid, such as the first cell in figure 2.11

Figure 2.11: first cell [63]

This grid has no objects, therefore pc will be 0 and the y label for this grid will be empty will be as shown in figure 2.12.

y =	0
	?
	?
	?
	?
	?
	?
	?

Figure 2.12: eight-dimensional vector [63]

As there is no item in the grid, the values bx, by, bh, bw, c1, c2, and c3 include. Figure 2.13 depicts a second grid in which a vehicle (c2 = 1) is positioned.

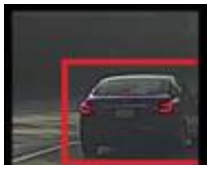


Figure 2.13: The cell that contains the object [63]

Before writing the y label for this grid, it is essential to understand how YOLO determines whether or not there is an item present. In the picture above, there are two items (two automobiles), hence YOLO will determine the midpoint of these two things and allocate these objects to the grid that includes the midpoint. The y label for the center-left grid containing the vehicle will be as seen in Figure 2.14. [60-63]

y =	1
	bx
	by
	bh
	bw
	0
	1
	0

Figure 2.14: eight-dimensional

Since there is an item in this grid, pc will equal 1. bx, by, bh, and bw will be computed relative to the current grid cell. Given that automobile is the second class, c2 = 1 whereas c1 and c3 = 0. Therefore, eight-dimensional output vector for each of the nine grids was obtained. This output will have the dimensions 3 by 3 by 8.

this model will be trained using both forward and backward propagation. During the testing step, the model was tested with some pictures and

perform forward propagation until the output y is obtained. To make things easy, discussed using a 3 × 3 grid, but in real-world situations, often bigger grids (perhaps 19 × 19) is used. Even if an item extends over many grids, it will only be allocated to the grid containing its midpoint. By increasing the number of grids, the likelihood of numerous items appearing in the same grid cell is reduced [65].

2.8 Embedded System

An embedded system is a microprocessor-based computer hardware and software system intended to execute a specific purpose, either independently or as a component of a larger system. An integrated circuit intended to do calculation for real-time processes is at the center.

From a single microcontroller to a suite of processors with associated peripherals and networks, and from no user interface to complicated graphical user interfaces, the spectrum of complexities is vast. The complexity of an embedded system varies considerably based on the function for which it is intended.

Applications of embedded systems include digital watches, microwaves, hybrid automobiles, and avionics. Up to 98 percent of all produced microprocessors are employed in embedded systems [67].

2.8.1 Embedded System Works

Microcontrollers or digital signal processors (DSP), application-specific integrated circuits (ASIC), field-programmable gate arrays (FPGA), graphics processing unit (GPU) technologies, and gate arrays manage embedded systems. Integrated into these processing systems are components designed to handle electrical and/or mechanical interfaces.

The firmware instructions for embedded systems are stored in read-only memory or flash memory chips and execute with restricted computer hardware capabilities. Through peripherals, embedded systems connect to the outside world, connecting input and output devices [67].

2.8.2 Structure of an Embedded System

Included in the structure of an embedded system are the following elements:

- The sensor detects the physical amount and transforms it to an electrical signal, which may subsequently be read by an embedded systems engineer or other electronic device. The detected amount is stored in memory by a sensor.
- A-D Converter: An A-D converter transforms the analog sensor signal into a digital signal.
- Processors and ASICs: Processors evaluate data to calculate output and store it in memory.
- A digital-to-analog converter transforms digital data given by the CPU into analog data.
- An actuator checks the output provided by the D-A Converter to the stored output and saves the authorized output [68].

2.8.3 Raspberry pi

As demonstrated in Figure 2.15, the Raspberry Pi is a low-cost, tiny, and portable computing board. It may connect to a computer display or television, keyboard, mouse, flash drive, etc. The Raspberry Pi contains

built-in software, such as Scratch, that allows users to program and create animations, games, and intriguing videos.

In addition, programmers may create scripts and applications using Python, the operating system's primary core language [8]. The Raspberry Pi B+ is the successor of Model B. In this work, Python was utilized to develop the script for client/server connection. In addition, there are enhancements like the addition of additional GPIO header pins, more USB ports, reduced power usage, etc. Model B+ is suggested for classroom learning because it offers more versatility than Model A, particularly for embedded applications that need minimal power, and has more USB ports than Model B [9].

Samba is open-source software that provides SMB/CIFS clients with a seamless file and print service [10]. Because it enables interoperability between Linux/Unix servers and Windows clients, this program is regarded as a useful and user-friendly innovation. Samba server is comprised of two essential programs, smbd and nmdb [11].

File and print services, authentication and authorisation, name resolution, and service announcement were Samba's four primary services. In this project, a Raspberry Pi is used as a server for remote file access by several PCs using Samba [67].

Figure 2.5: Raspberry pi [67]

2.8.4 GPIO in Raspberry pi

GPIO is an abbreviation for General Purpose Input Output. As seen in figure 2.16, the Raspberry Pi contains two rows of GPIO pins that serve as links to the real world. Output pins are analogous to switches that the Raspberry Pi may use to turn on or off devices, such as an LED light. However, it can also transmit signals to other devices. Input pins are similar to switches that may be turned on or off externally (like a light switch). However, it may also be sensor data or a signal from another device [67].

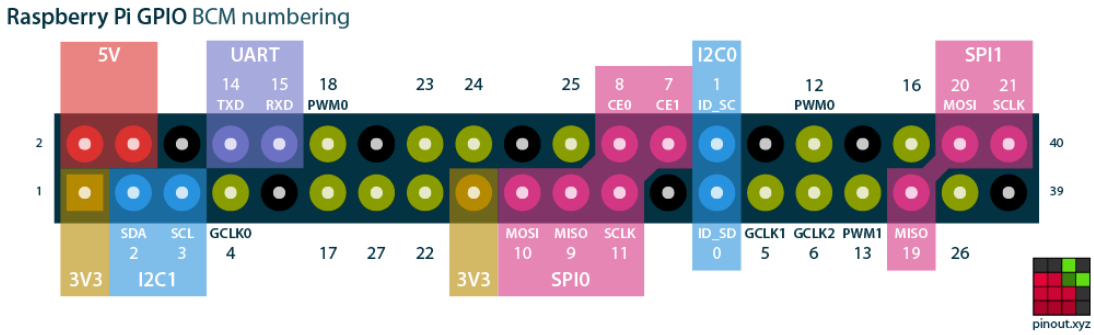


Figure 2.6 :GPIO Pins [67]

2.8.5 Arduino

Arduino is an open-source electronics platform centered on user-friendly hardware and software. Arduino boards can convert inputs - such as light on a sensor, a finger on a button, or a tweet - into outputs - such as operating a motor, lighting an LED, or posting anything online. Sending a series of instructions to the microcontroller on the board allows you to direct it. To do this, you use the Arduino programming language (based

on Wiring) and the Arduino Software (IDE), which is based on Processing.

Arduino is the brain of tens of thousands of projects throughout the years, ranging from ordinary things to complicated scientific apparatus. This open-source platform has attracted a global community of makers, including students, hobbyists, artists, programmers, and professionals, whose contributions have amassed an extraordinary amount of information that may be of tremendous assistance to beginners and specialists alike.

Arduino is created at the Ivrea Interaction Design Institute as a simple prototyping tool targeted for students with no prior experience in electronics or programming as shown in figure 2.17. As soon as it attracted a larger audience, the Arduino board began to evolve in response to new demands and problems, shifting from basic 8-bit boards to solutions for IoT applications, wearables, 3D printing, and embedded settings [69].

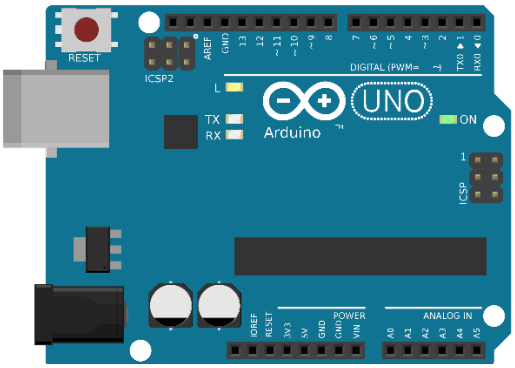


Figure 2.7: Arduino [69]

CHAPTER THREE

THE PROPOSED SYSTEM AND METHODOLOGY

3.1 Introduction

In this chapter, a detailed demonstration of how to extract the target object's location in 3d environment by using proposed algorithms.

- The n-Visibility Tree Algorithm (proposed).
- object detection using YOLOv5 algorithm.
- Localization algorithm.

The n-Visibility Tree may address the challenge of 3D environment path planning. This method may offer the robot or numerous robots in the environment with a smooth route. The YOLOv5 object detection method is used to identify the item in the picture. Based on the information obtained from the previous methods, a localization algorithm is used to locate the target.

3.2 General Overview of the Proposed System

The proposed system architecture includes two stages. Each stage consists of sub-steps in order to achieve the research objectives and meet its main aim. These stages find the best path and position detection stage.

The first stage is to find the best path using the n-Visibility Tree Algorithm. The output of this algorithm is the suboptimal path. The UAV flow this path to reach the target point.

The second stage is the position detection stage which encompasses many processes that are responsible for detecting any objects that appear in front of the UAV and sending that information encrypted to the base station which the last one shows the location on google map. This stage is to work with real-time detection of objects remotely and send data

instantly to the base station. Figure 3.1 shows the proposed system in general. Algorithm 3.1 demonstrate the overall description of the proposed system.

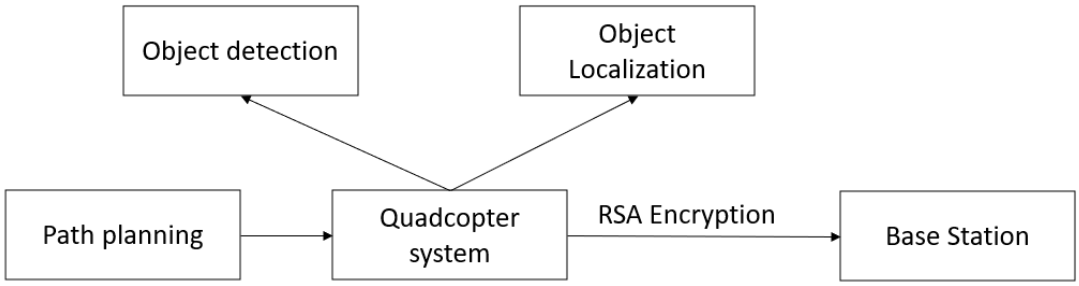


Figure 3.1: The proposed system in general

Algorithm (3.1): Pseudocode of the proposed system

INPUT: Target_type, search_area_point[] ;
OUTPUT: Target Position;
Begin
1- **Define search area**
2- **N visibility tree algorithm (start point, end point)// return the best path.**
3- **Object detection based on yolo**
4- **Localization algorithm (drone position [])// return target location**
5- **Encrupt data using RSA**
6- **Send info to base station**
End

3.3 The Proposed n-Visibility Tree Algorithm

It is a suggested approach for identifying the shortest route in densely populated areas. The strategy is evaluated using a spherical robot that travels in three-dimensional space. It is also believed that obstacles are spheres.

Two strategies are used to design the robot's path: travelling in a straight line to the obstacle's tangent point and going around obstacles using the smallest segment of a great circle. Multiple viable routes are so constrained. The minimal route is then identified based on the minimization of an n-visibility tree technique. The n-visibility tree technique is proposed for 3D route planning with obstacle avoidance for mobile robots.

In this method, it is assumed that the quadcopter robot was present. The desired alignment of the robot toward the target has been attained. The core premise of the depth-first search (DFS) method is to do a search around the target point. For searching, the DFS algorithm needs nodes. This work calculates the nodes using the equations produced from this method. The DFS is shown in Figure 3.2. The search starts at the first location and progresses deeper until there is no choice but to return. The standard DFS method needs knowledge of all nodes and connections beforehand. This study expanded the idea to make it compatible with the n-visibility tree technique. The proposed approach is aided by the DFS to create an undetermined number of nodes and lines.

First, the algorithm builds a full set of pathways from the source point (i.e., the robot's center) to the destination point. Each route is composed of straight lines and curved lines that link these straight lines. In the second stage, a searching algorithm is used on the n-visibility tree graph to discover the shortest route between the beginning and ending points. With the use of the next section's derived equations, DFS constructs a set of complete pathways between the source and destination sites.

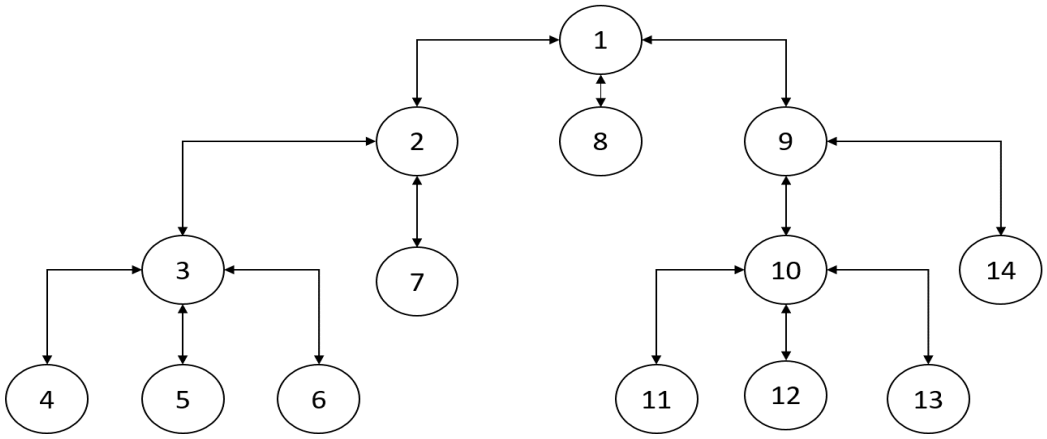


Figure 3.2: DFS algorithm: order in which nodes are visited.

The construction of a complete set of paths is the first step in this algorithm. This first step is essential because the algorithm complexity depends on the complexity of this part. The idea is to search for the safest and most direct path between two points, avoiding a collision. The minimum collision-free path that avoids obstacles is the tangent observed by the robot. In this thesis, the tangent is the path of the robot's center that causes its surface to be tangent with the obstacle. There are an infinite number of tangents can be used to avoid spherical obstacles. All of which satisfy the obstacle equation at the maximum circle observed by the robot. The circle points satisfy the robot surface equation. The radius of the seen circle from robot side is always less than the radius of the obstacle. If the target point is on the other side of the obstacle, its faced circle is also observed. These two circles must be connected by curves far from the surface of the obstacle, by r along the curve. An infinite number of curves can connect any two points. The required one is the minimum. The great circle principle is used to determine this minimum. Two other scenarios can be faced by the

algorithm when constructing a set of complete paths. First, when the robot avoids the first obstacle but may face another of the same size, in which case the solution is to construct a cylinder covering the two obstacles with a radius of R+r. The other scenario occurs when the second obstacle does not have the same size as the first avoided obstacle, and the tangents to each are required. These tangents resemble a section of a cone with the radius of each side is more than the obstacle by r to ensure that the robot will be in tangent with both obstacles during movement from one to another. Therefore, when n reaches infinity, the shape is a cylinder or cone. Otherwise, it is simply lined belonging to cylinders or cones. All details regarding the operation of this algorithm are explained in the following steps as well as the algorithm 3.2.

Algorithm (3.2): Pseudocode of proposed n-Visibility Tree Algorithm

INPUT: Start 3D point (P);
OUTPUT: Target 3D point;
Begin
1- The radius and 3d coordinate (P) of the Drone and obstacles
2- Obtain the nearest obstacle.
3- Calculate visible circle in 3d
4- While (any path intersects with any obstacle)
5- Replace the straight path with n paths to n points of the circle
6- End while
7- Avoid obstacles based on great circle principle
End

Step 1 : Acquire the information. r, m, p, and k indicate the robot's radius and 3D coordinates. the n value. The obstacle radii and coordinates are given (R_1: s_{11}, s_{12}, s_{13}; R_2: s_{21}, s_{22}, s_{23};. . . R_i: s_{i1}, s_{i2}, s_{i3}). The target coordinate is represented by (E, F, G), where i indicates the obstacle number.

Where $-\infty \leq t \leq \infty$

R_i : the radius of obstacle i

s_{i1} , s_{i2} , s_{i3} : 3D coordinate of obstacle i.

Step 2 : Derive an equation of a straight line between the source (drone center) and target position as:

$$t = \frac{x-m}{E-m} = \frac{y-p}{F-p} = \frac{z-k}{G-k} \quad \ldots\ldots (3.1)$$

Step 3 : Obtain the ID of the closest obstacle that crosses the preceding straight line. Replace this straight line with n tangent straight lines to the obstruction. When the Euclidean distance between the line and obstacle is smaller than the total of the drone and obstacle radii (R+r), an intersection occurs. Based on the source point and obstacle information (position and radius), n points are determined around the obstruction. When the drone travels in a straight path to these spots, it will collide with the obstruction. As n approaches infinity, a circle in 3D will be formed, as seen in Figure 3.3.

There are n spots inside the circle that must be chosen. First, as illustrated in Figure 3.4, a circle is determined by the intersection of the plane with the barrier after extending its radius by r to account for the drone diameter. As illustrated in Figure 3.5, the plane is formed by the intersection of two spheres: the first is the barrier after its radius is

increased by r, and the second is an assumed sphere centered at (m, p, k) with a radius provided by

$$Ras = S_{i1}^2 + S_{i2}^2 + S_{i3}^2 - m^2 - p^2 - k^2 - 2(R+r)^2 + (s_{i1} - m)^2 + (s_{i2} - p)^2 + (s_{i3} - k)^2 \quad \ldots\ldots (3.2)$$

Where

Ras: radius of assumed sphere.

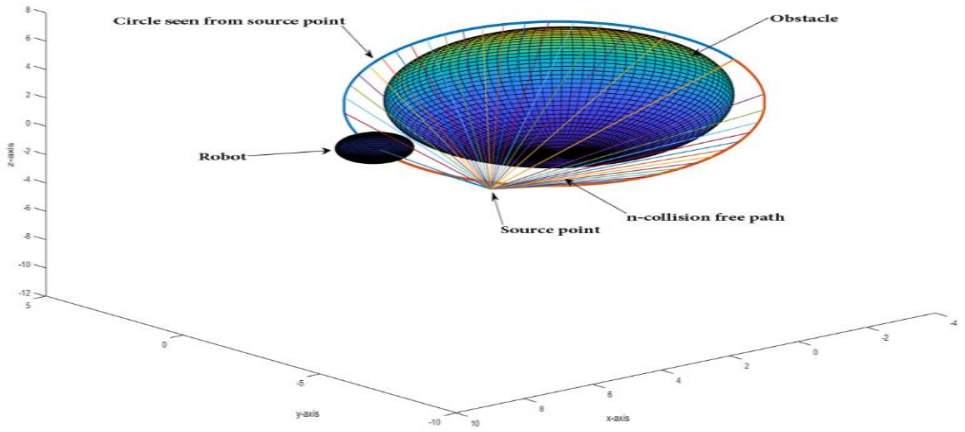


Figure 3.3: Circle around obstacle observed from drone

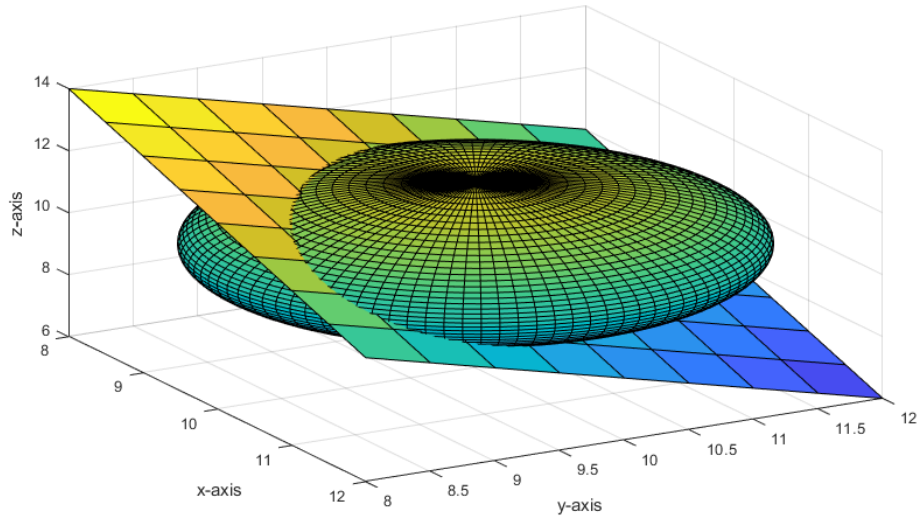


Figure 3.4: Sphere plane intersection.

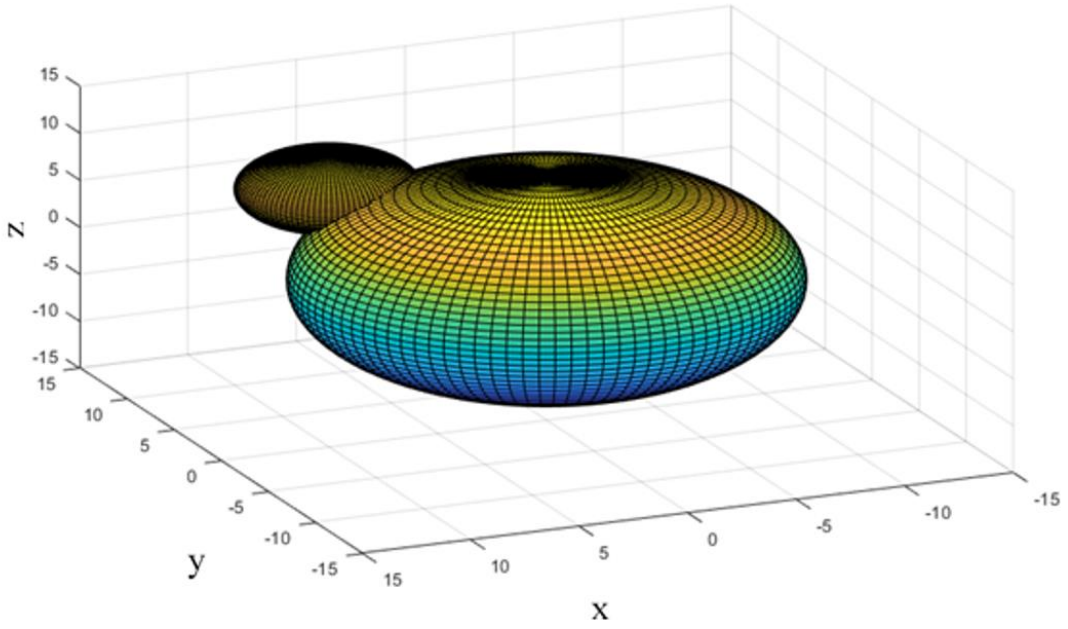


Figure 3.5: Two sphere intersected

Therefore, the intersection between these two spheres results a circle belong in the plane of Equation bellow:

$$(2\,m - s_{i1})X + (2\,p - s_{i2})Y + (2\,k - s_{i3})Z + Ras = 0 \qquad \ldots\ldots (3.3)$$

The source point is the supposed sphere's center, and its radius is Ras. The tiny sphere is the obstruction for which the tangent points must be determined. The resultant plane contacts the obstruction once its radius is increased by r. According to Equation, the outcome of the intersection is a three-dimensional circle (3.4).

$$(X - s_{i1})^2 + (Y - s_{i2})^2 + (Z - s_{i3})^2 = (R + r)^2$$
$$(2\,m - 2s_{i1})X + (2\,p - 2s_{i2})Y + (2\,k - 2s_{i3})Z + Ras = 0 \quad \ldots\ldots (3.4)$$

Uniformly spaced n points from the circle of Equation (3.4) are selected, as illustrated in Figure 3.6.

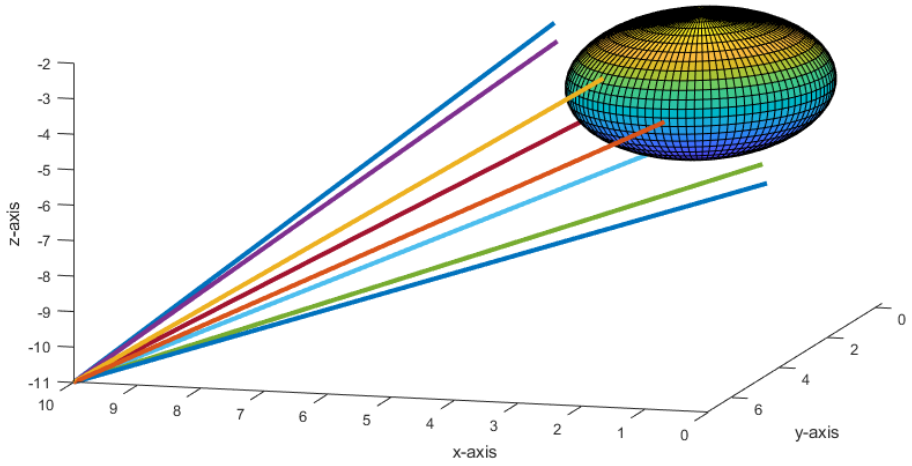


Figure 3.6: The n tangents paths around one obstacle from source point (n=8)

Step 4 : Repeat step 3 for each line, until there are no more intersects with any obstacles.

Step 5: As seen in Figure 3.7, derive n equations of straight lines between the goal point and the tangent of all obstacles previously identified in step 3. According to Equation, (3.5) each line has two points: the target point and the point chosen from the specified circle.

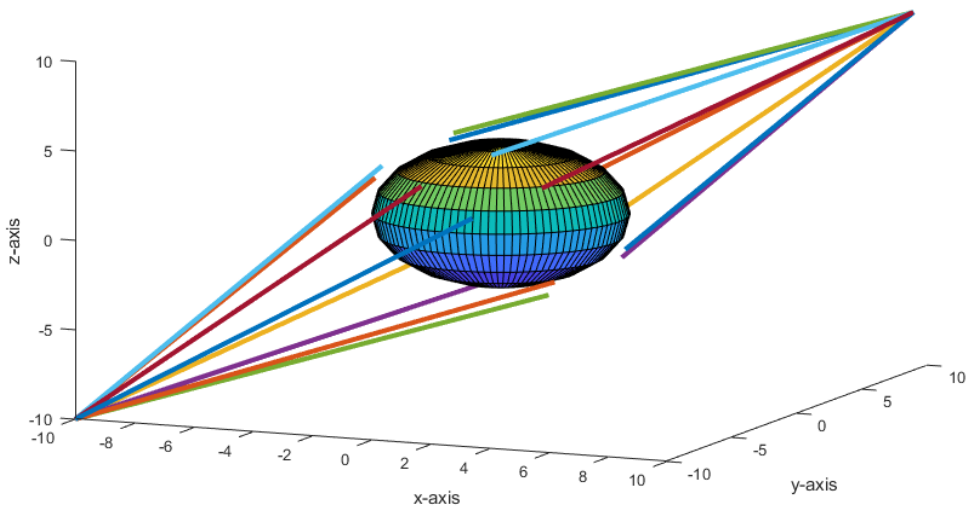


Figure 3.7: The n tangents paths from source and target points (n=8).

$$(X - s_{i1})^2 + (Y - s_{i2})^2 + (Z - s_{i3})^2 = (R + r)^2$$
$$(2\,E - s_{i1})X + (2\,F - s_{i2})Y + (2\,G - s_{i3})Z + Ras = 0 \qquad \ldots\ldots (3.5)$$

Step 6 : For each line derived in the previous step, if no intersection with it exists, confirm it. If it intersects with at least one obstacle, obtain the nearest intersected obstacle.

Step 7: A new obstacle must be avoided by the path tangent to each of the two obstacles (first where the drone exists and second is the obstacle intersects a straight path to the target). There are two cases for this scenario, as follows.

Case one: When both obstacles have the same radius, as indicated in Figure 3.8, n points are selected on each obstacle. First, the circle in 3D around each obstruction must be found by intersecting a plane with the obstruction and increasing its radius by r. To identify the plane equation, any point belonging to the plane and a vector perpendicular to the plane

from this point must be known. For each obstacle, a point that is the obstacle's center and the vector that is between obstacle centers is known. The resulting plane equations for each obstacle are derived as follows. First, the obstacle center is (s_{11}, s_{12}, s_{13}), and the radius after compensating for the drone radius is $(R + r)$. The second obstacle center is (s_{21}, s_{22}, s_{23}). The n points around the first obstacle can be selected from the calculated circle in Equation (3.6).

$$(X - s_{11})^2 + (Y - s_{12})^2 + (Z - s_{13})^2 = (R + r)^2 \quad \ldots\ldots (3.6)$$
$$(s_{21} - s_{11})(X - s_{11}) + (s_{22} - s_{12})(Y - s_{12}) + (s_{23} - s_{13})(Z - s_{13}) = 0$$

The n points around the second obstacle can be selected from the calculated circle in Equation (3.7)

$$(X - s_{21})^2 + (Y - s_{22})^2 + (Z - s_{23})^2 = (R + r)^2 \quad \ldots\ldots (3.7)$$
$$(s_{21} - s_{11})(X - s_{11}) + (s_{22} - s_{12})(Y - s_{12}) + (s_{23} - s_{13})(Z - s_{13}) = 0$$

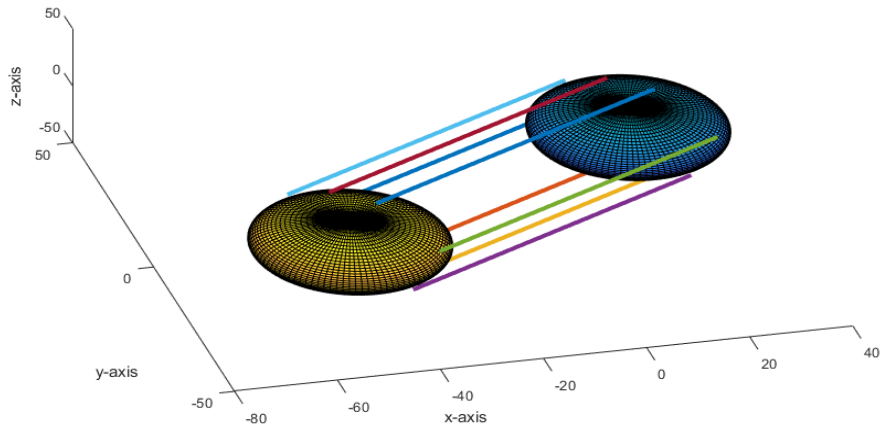


Figure 3.8: The n tangents paths between two same size spheres (n=8).

Case two: When the radii differ, as illustrated in Figure 3.9, n points are selected on each obstacle. the first obstacle center is (s_{11}, s_{12}, s_{13}), and the radius after incrementing by the drone radius is $R1$. The second obstacle center is (s_{21}, s_{22}, s_{23}), and its radius after increasing it by the drone radius is $R2$. The goal here is to determine another point satisfying a line equation that passes through (s_{11}, s_{12}, s_{13}) and (s_{21}, s_{22}, s_{23}). The point position is denoted by a, b, and c. From this point, it is possible to draw n lines that cause the drone to in tangent with both obstacles when moving. If $R1$ is less than $R2$, the points can be selected from the calculated circle using Equation (3.11).

$$dis2 = \sqrt{(s_{11} - s_{21})^2 + (s_{12} - s_{22})^2 + (s_{13} - s_{23})^2} \quad \ldots\ldots (3.8)$$

$$dis1 = \frac{R1\ dis2}{R2 - R1} \quad \ldots\ldots (3.9)$$

$$a = s_{11} + \frac{dis1(s_{11} - s_{21})}{dis2}$$
$$b = s_{12} + \frac{dis1(s_{12} - s_{22})}{dis2} \quad \ldots\ldots (3.10)$$
$$c = s_{13} + \frac{dis1(s_{12} - s_{22})}{dis2}$$

$$(X - s_{11})^2 + (Y - s_{12})^2 + (Z - s_{13})^2 = R1^2 \quad \ldots\ldots (3.11)$$
$$(2a - 2s_{11})X + (2b - 2s_{12})Y + (2c - 2s_{13})Z + dis = 0$$

$$dis = {s_{11}}^2 + {s_{12}}^2 + {s_{13}}^2 - a^2 - b^2 - c^2 - 2R1^2 + (s_{11} - a)^2 + (s_{12} - b)^2 + (s_{13} - c)^2 \quad \ldots..(3.12)$$

The points around the second obstacle are calculated using Equation (3.13)

$$(X - s_{21})^2 + (Y - s_{22})^2 + (Z - s_{23})^2 = R2^2 \quad \ldots..(3.13)$$

$$(2a - 2s_{21})X + (2b - 2s_{22})Y + (2c - 2s_{23})Z + dist2 = 0$$

$$dist2 = {s_{21}}^2 + {s_{22}}^2 + {s_{23}}^2 - a^2 - b^2 - c^2 - 2R2^2 + (s_{21} - a)^2 + (s_{22} - b)^2 + (s_{23} - c)^2 \quad \ldots..(3.14)$$

If $R1$ is greater than $R2$, a point is determined from Equation (3.17).

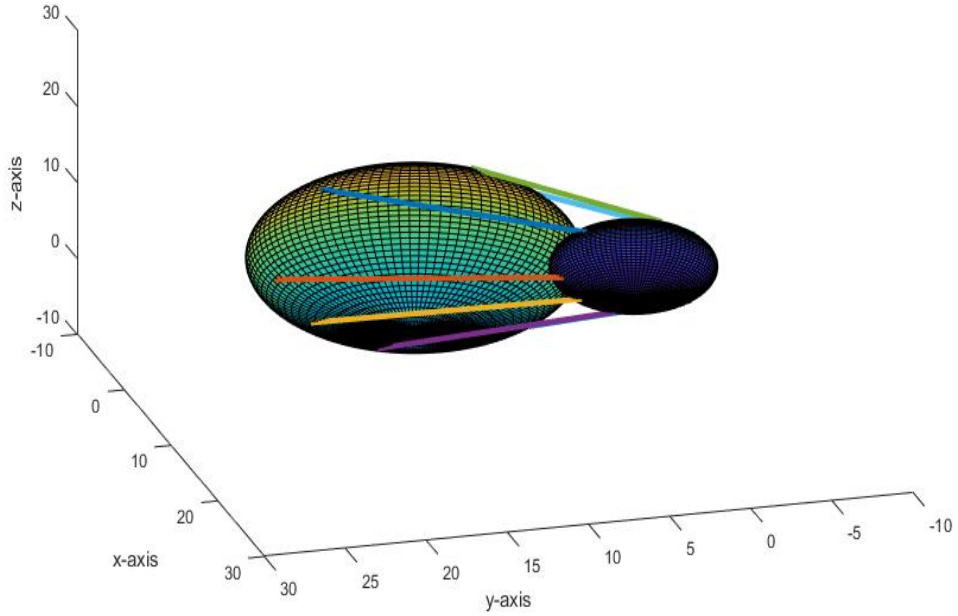


Figure 3.9: The n tangents paths between two different size spheres (n=8).

$$diso2 = \sqrt{(s_{21} - s_{11})^2 + (s_{22} - s_{12})^2 + (s_{23} - s_{13})^2} \quad \ldots..(3.15)$$

$$diso1 = \frac{R1\ dis2}{R1 - R2} \quad \ldots\ldots (3.16)$$

$$a = s_{21} + \frac{diso1(s_{21} - s_{11})}{diso2}$$

$$b = s_{22} + \frac{diso1(s_{22} - s_{12})}{diso2} \quad \ldots\ldots (3.17)$$

$$c = s_{23} + \frac{diso1(s_{22} - s_{12})}{diso2}$$

The same Equations (3.11) and (3.13) are used to determine the points around each obstacle.

Step 8: For each line between points in Step 7, if it intersects any obstacle, delete it and return to Step 7 after obtaining the identity of the nearest obstacle of interest.

Step 9: As seen in Figure 3.10, draw arcs from the locations on either obstacle side using the short segment of the great circle method. After expanding its radius by r, the great circle in 3D may be found by intersecting a plane with the obstruction. A link a central point and two other locations was desired. Using a cross-product between the vectors from the origin and the two desired locations, the plane's was obtained by perpendicular vector. The resultant intersection is a large circle that connects the two desired spots. These two points split the great circle into two halves of varying length. The smallest segment is what is needed. A great circle is described by the intersection of the sphere and the plane in equation (3.18). Po and Pq are the 3D desired points to link, while,s-1. represents the obstacle's center.

$$(X - s_{11})^2 + (Y - s_{12})^2 + (Z - s_{13})^2 = (r + R)^2$$

$$\big((Po_2 - s_{12})(Pq_3 - s_{13}) - (Po_3 - s_{13})(Pq_2 - s_{12})\big)(X - s_{11})$$

$$\big((Po_1 - s_{11})(Pq_3 - s_{13}) - (Po_3 - s_{13})(Pq_1 - s_{11})\big)(Y - s_{12}) +$$
$$\big((Po_1 - s_{11})(Pq_2 - s_{12}) - (Po_2 - s_{12})(Pq_1 - s_{11})\big)(Z - s_{13}) = 0$$
.....(3.18)

Step 10: Repeat Steps 5 to 9 until no line intersects an obstacle. Therefore, all lines will have reached the target point as shown in figure 3.10.

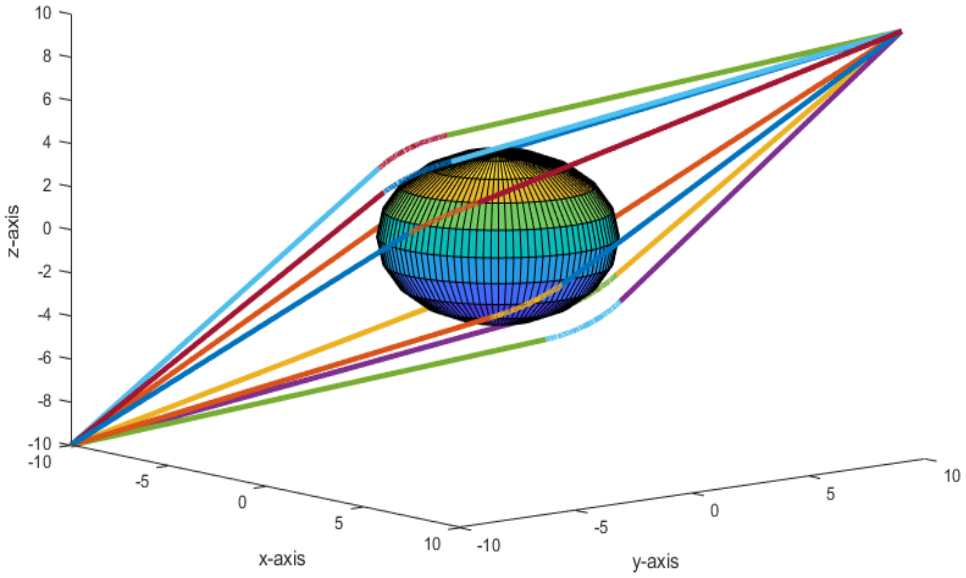


Figure 3.10: One to one connection curves between n straight lines.

The construction of a complete set of paths is the most crucial thing of this algorithm because only need to choose one of the paths in the next section. So the complexity of this algorithm depends hugely in this section. As seen in steps above it is possible to say its regular algorithm, each obstacle can be avoided by n paths, So it is easy to make some conclusions about time complexity and optimality which are: time complexity increase with an increase in the number of obstacles or n

parameter or both of them, So the complexity can be controlled by control of the n parameter, each path calculated by equations in steps above to be suboptimal and smooth as much as possible.

3.4 Search algorithm using Dijkstra

As explained previously, n nodes point around each obstacle. Because many paths or curves avoid obstacles, each set contains n curves. Additionally, an unknown number of sets exist. This number depends on how many obstacles intersect the tangents between the target point and obstacle. Otherwise, the distances between points are used as weights. The Euclidian distance of a straight line between any two points on a sphere is proportional to the distance of the short part of the great circle between these points. This information is used to calculate the weights without the need for determining the curve distance in 3D. Thus, several nodes and weights are gotten (i.e., distances between points). Dijkstra algorithm used to choose the minimum complete path from the set of paths constructed by DFS.

Algorithm (3.3): Pseudocode of Dijkstra Algorithm

INPUT: Tree of paths, weights;
OUTPUT: Best path;
Begin
1- Read Tree of paths;
2- Apply Dijkstra procedure
3- Sort the results;
4- Get the min weight path;
5- Return the best path
End

3.5 The Modeling of Obstacles

The obstacles modeling is a vital portion of path planning. The way used to describe the obstacles effects on the results path optimality and complexity. The accurate modeling method can clarify the difficulty. In order to represent any complicated obstacle, multiple spheres must be used. A complex obstacle like building must be entirely inside spheres. The distribution of spheres has to achieve the above condition. The main affected parameters are radii of spheres and distance between them. Long radius causes less complexity because it will need fewer spheres to represent a complicated obstacle, but on another side, it decreases the optimality of the path. A short radius increases optimality and complexity at the same time. The second parameter (distance between spheres) must achieve the following condition in equation (3.18) in order to cover obstacles completely.

The intersection of two spheres generates a circle (AB) with a diameter has shown in Figure 3.11. The diameter h must be more than the maximum diameter of an obstacle. It can be calculated from the equation (3.19).

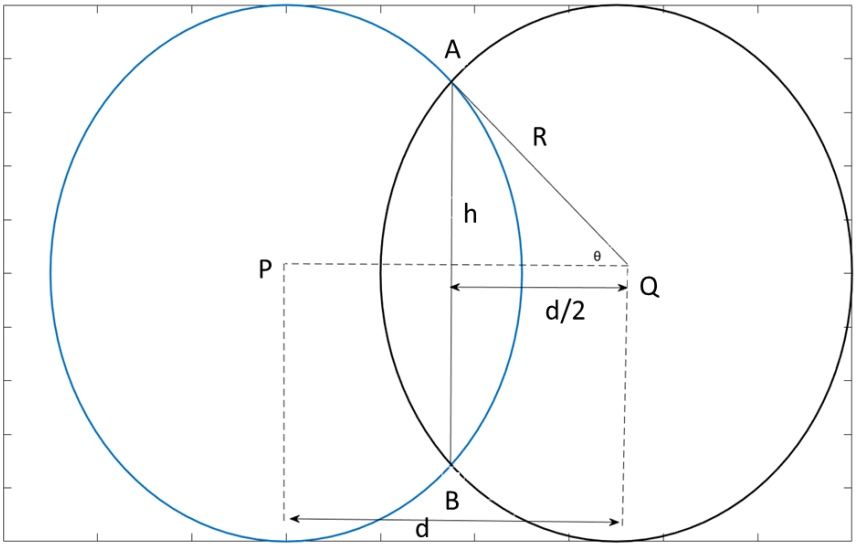


Figure 3.11:The intersection of two-spheres for complex obstacles

$$h > 2\sqrt{R^2 - \frac{d^2}{4}} \qquad \ldots\ldots(3.19)$$

Where R: radius of the sphere, d distance between centers of spheres,

Finally, the user has a choice to represent a complicated obstacle in any method. He can choose their radius, distance between any two spheres. The only condition is obstacle must be entirely inside spheres. There is another method for representation. Each building or any other obstacle represent by some rectangular blocks. Now replace each block by a sphere with a radius equal or more than the longest diameter of the block.

For example, let's choose rectangular building centered at (x=50, y=3) has length in x-axis =40m, width in y-axis= 10, height in z-axis= 28. Now let us imagine the building consists of four columns; the dimensions of each column are [10 x 10 x 28]. First column centered at

(35, 3). The maximum diameter of this column in x, y cross-section is h=14.14. Now let us choose the radius of each sphere R=9. Based on h and R, equation (3.19) can used to calculate d= 11 or less. Let us choose d=10. so three spheres is needed to represent first column of building centered at (35,3,5),(35,3,15),and(35,3,25). To represent other parts of the building, shift these spheres by 10, 20, and 30 in the x-direction. So 12 spheres required to representing the supposed building.

the procedure above can be used to model the environments by intersecting spheres as shown in Figures 3.12 to 3.16.

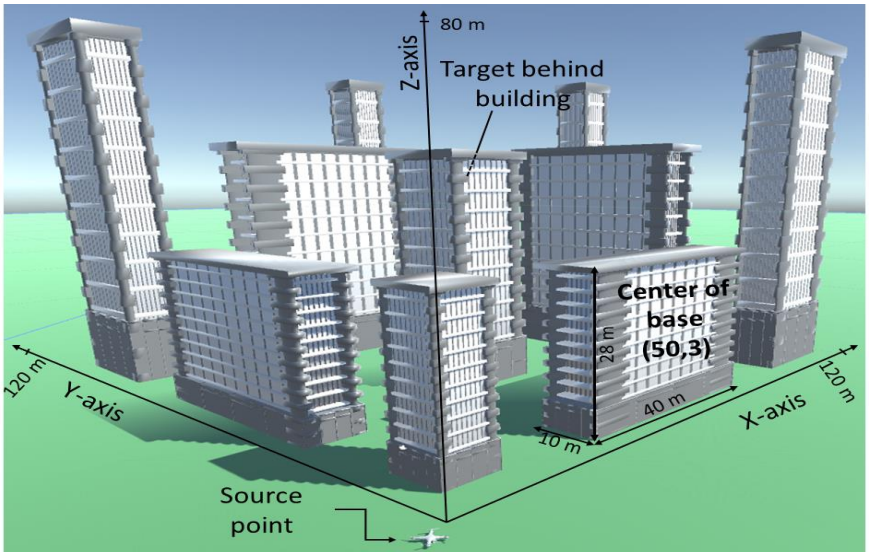


Figure 3.12:Buildings environment (vision a) representations.

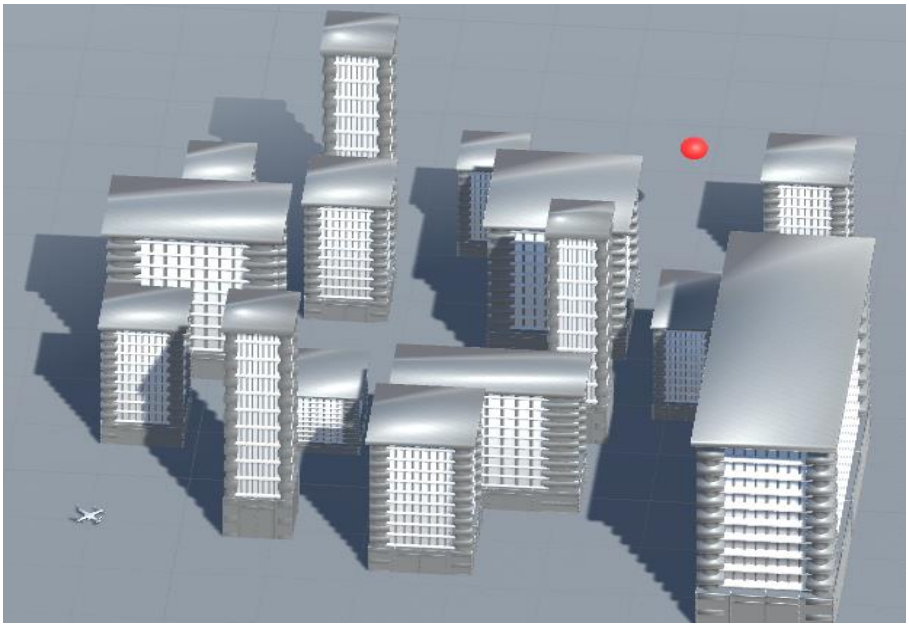


Figure 3.13: Buildings environment (vision b) representations.

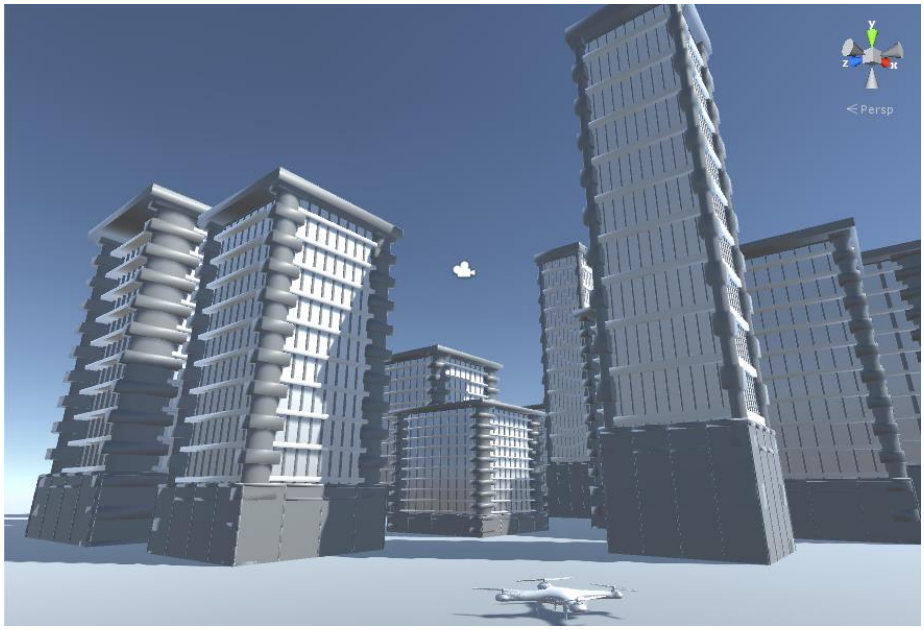


Figure 3.14: Buildings environment (vision c).representations.

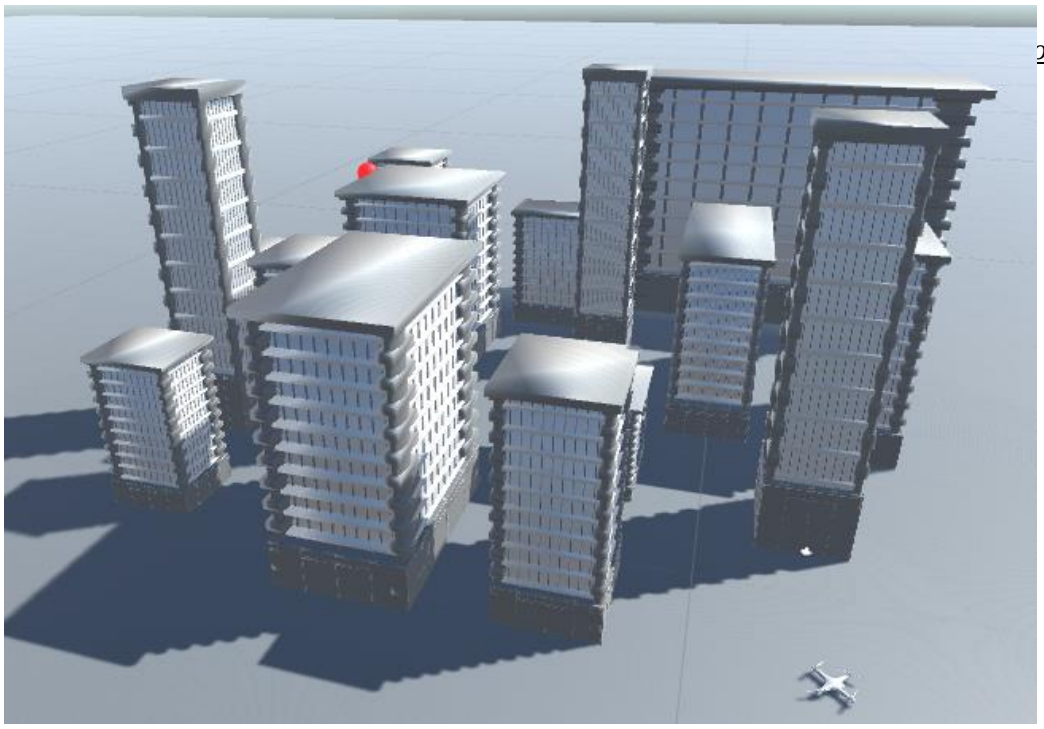


Figure 3.15: Buildings environment (vision d). representations.

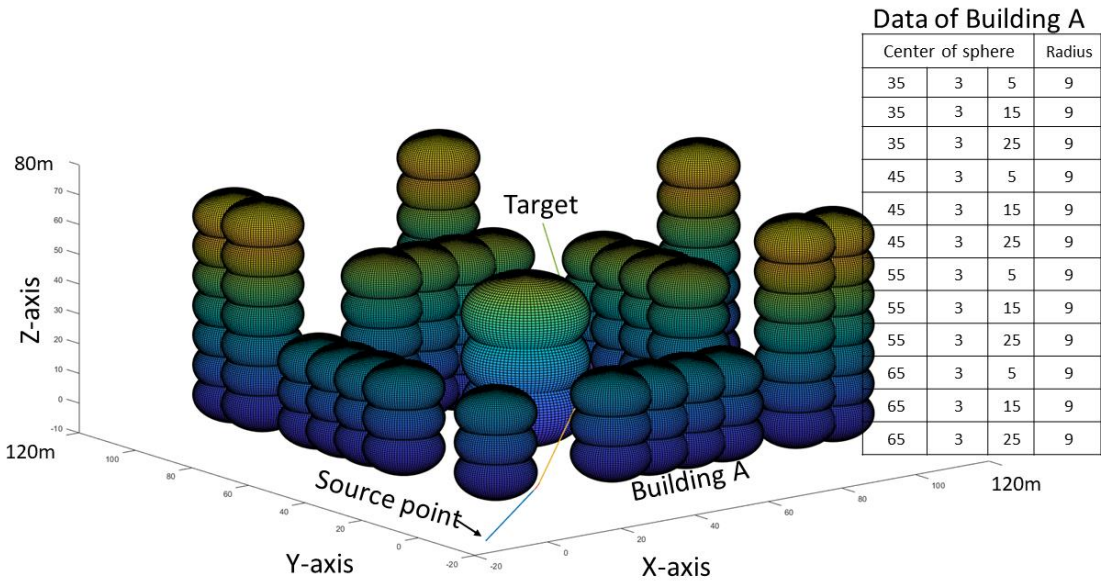


Figure 3.16: Buildings environment modeling.

Another example of palm trees environment as shown in figure 3.17 also, the modeling of this environment based on the proposed method as shown in figure 3.18.

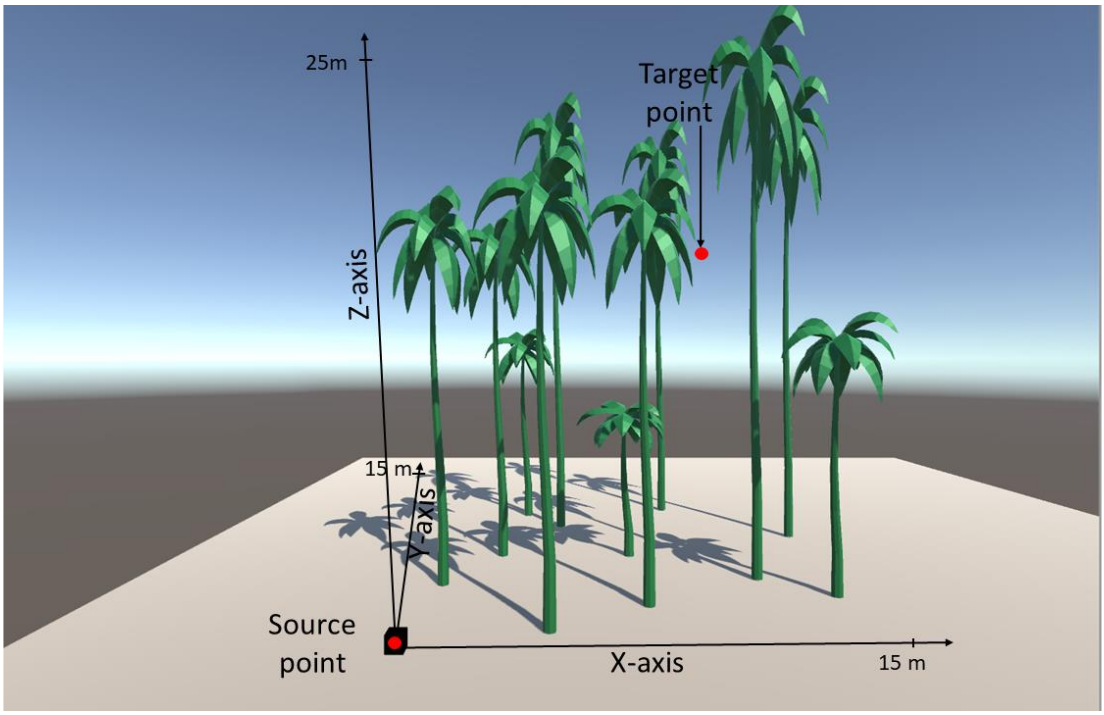


Figure 3.17: Palms environment representations.

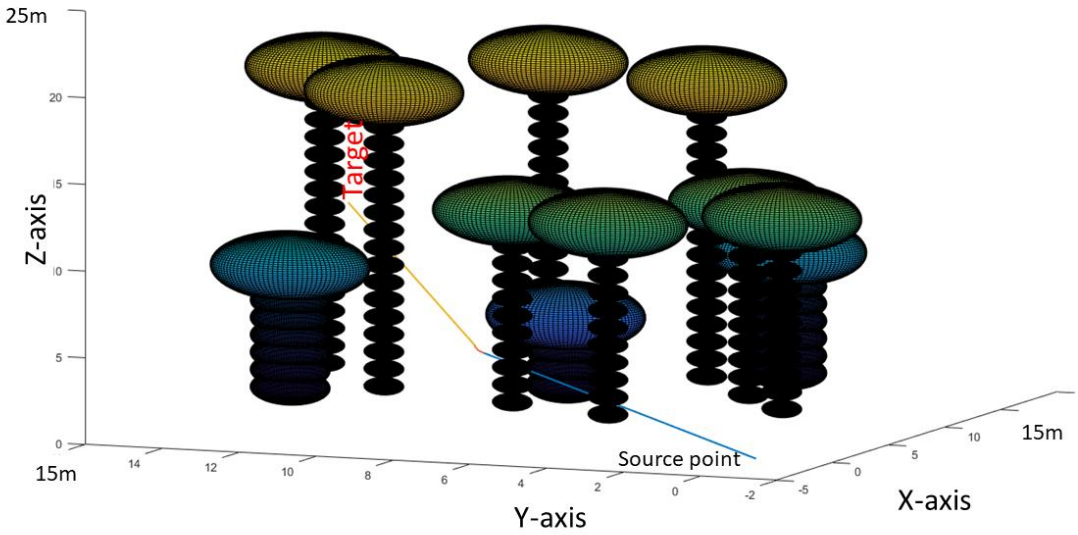


Figure 3.18: Palms environment modelling

3.6 Object Detection Using YOLO Algorithm

The dataset and training will be presented in this section as following.

3.6.1 Dataset Collections

The use of a database is suitable for the purpose of the model to be created is one of the essential things. In the case of the proposed system, the ready-made dataset was not found. It is necessary to create a dataset commensurate with the actual need of the proposed system. Since the goal is to detect objects from top view projection using a drone, a camera attached to a drone was used to record the required images and then label them manually. About 32,000 images were created in this way in different environmental conditions. In addition to the different conditions, effects were added using filters to obtain a more comprehensive dataset. Figure 3.19 represents a real picture of the drone while it was capturing images. figures 3.20 shows examples of images installed in the dataset that was created in an aforementioned manner.

Figure 3.19: Drone during collect dataset

Figure 3.20: Examples of collected dataset

The dataset that has been created, in order to train the algorithm, it must be in YAML format, so this set of images must be converted to this format as in Figure 3.21.

YAML contains two directories that are train and valid as well as text file named data.yaml in the root path.

data.yaml present: classes name, number of classes, and train and valid path directory.

The train directory consists of two directories. the first one is the images directory which contains image files, and the second directory is named Labels which contained label files.

The valid directory has same structure of the train directory

Figure 3.21: YAML format

3.6.2 Training Model

The YOLO algorithm is trained to detect objects on the collected dataset, and the colors of the images and the angles of taking pictures were changed to find new images to suit all possible conditions.

The dataset was divided into 80 percent for training and 20 percent for validation, where the number of images became 24000 images for training the model and 8000 images for validation the performance of the model

The large image size is a challenge in this field as it requires a massive amount of processing to work on in training and testing. Since the proposed system works on the Raspberry Pi processor, a solution to this problem must be found because this processor depends on the CPU, not the graphics processing unit. One of the most popular ways to reduce the size of images is the Max Polling method. The method described has been applied to reduce the size to 640 x 640 pixels.

3.7 Localization

The process of locating a specific object can be achieved through a set of projections. The number of views should not be less than two if the height is known. On the other hand, the least acceptable number of projections is three to find the location and altitude. An increasing number of projections leads to an increase in accuracy.

The above goal is theoretically achieved by finding the point of intersection of the straight lines directed to the target from two or three locations, respectively.

The same goal above is practically achieved by finding the point that moves away from the two vectors with the least displacement. This principle is applied in practice since the vectors are not likely to intersect because the accuracy for determining the vector is never perfect.

Locating a UAV's physical position in line with a real or virtual coordinate system is known as localization. When a direct measurement of the UAV's position is unavailable, localization is critical . The accuracy of the estimated location information at a particular point in time is used to assess the performance of a system that uses localization. In this thesis, the software is built to calculate the object Tx, Ty, and Tz coordination, as shown in Figure 3.22 Based on the drone x, y and z coordination and the camera angle in vertical and horizontal (theta , psi)

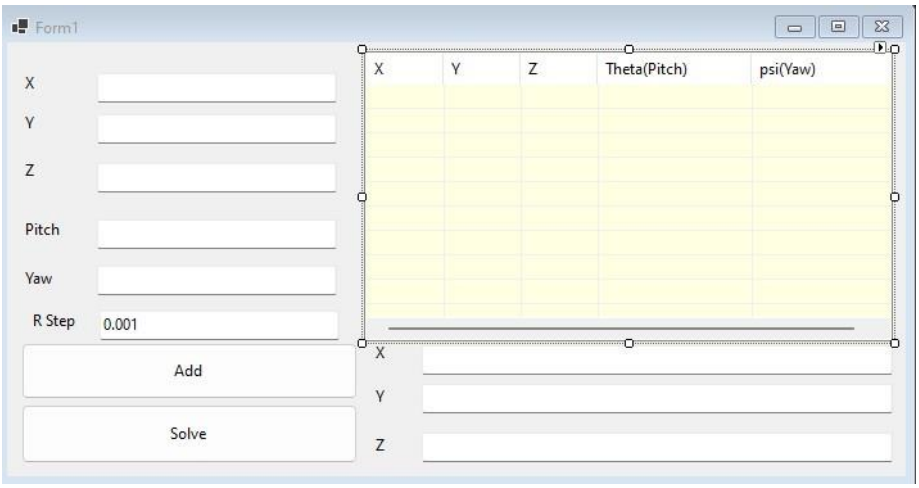


Figure 3.22: Localization in UAV's systems

Both X, Y, and Z are needed to be position coordinates. Path denotes the value of the Lateral Axis (Pitch), while Yaw denotes the value of the Vertical Axis. After the completion of the form and the addition of the values to the right pane, these values need to be solved.

The results are shown in the x, y, and z coordinates, which correspond to the goal location. Basically, when the drone moves in more than one direction in order to get a precise and steady position to the target, the intersection of these points is found by taking the average of these locations. Consequences for obtaining target coordinates may be found in the following modules.

$$\text{Deltax} = \text{r} * \cos(\text{th}) * \sin(\text{psi}) \quad (3.20)$$

$$\text{Deltay} = \text{r} * \cos(\text{th}) * \cos(\text{psi}) \quad (3.21)$$

$$\text{Deltaz} = \text{r} * \sin(\text{th}) \quad (3.22)$$

New x, y, z values are shown using the following modules:

$$Xnew = x + Deltax \quad (3.23)$$

$$Ynew = y + Deltay \quad (3.24)$$

$$Znew = z + Deltaz \quad (3.25)$$

The result of the above modules is a straight line that starts from the drone and ends up in ∞ , passing through the target position. The straight line is divided into radius r which is utilized to calculate the average of obtained target points. However, the average shows the closest point to the target position that the drone has captured from different trends .

3.8 Quad Copter Implementation

The hardware of the quad rotor consists of multiple components, which are the frame, Autopilot (Arduino mega 2560), motors, electric speed

controllers, propellers, GPS, compass, gyroscope, power manager, telemetry system, battery, and Barometric Pressure Sensor (for height measurement). Each component responsible for some purposes, as explained in the following subsections.

3.8.1 Frame

Its 450mm fiber glass unassembled frame for a quad rotor with GPS holder, as shown in Figure 3.23. It is light and durable. So, it is suitable for the system requirements.

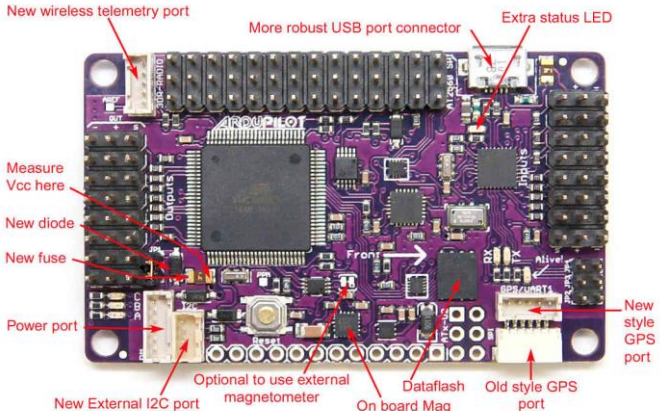
Figure 3.23: Frame of the Quadcopter.

3.8.2 Ardupilot

It is the central part of the quad rotor that represented by Arduino mega. Its almost connected with all other parts. It receives information from the RF receiver, telemetry, Gyroscope, GPS, compass, barometric pressure sensor, and other optional sensors. The main output of the autopilot is

the PWM signals that control the rotational speed of all four motors, which lead to control of the status of the quad rotor.

Arduino Mega 2560 is used in this work as an autopilot. The brand name of this autopilot is APM2.5. as shown in Figure 3.24. The program of this autopilot can be written using the Arduino C language. Mission Planner windows application used to communicate with the autopilot as shown in Figure 3.25.



Fiaure 3.24: APM ArduPilot

Figure 3.25: Mission Planner.

3.8.3 Motors

The motors shown in Figure 3.26 are used in the quad rotor are brushless DC motors. The used motors are responsible for generating the required thrust force and three-dimensional torque.

Figure 3.26: Motor of the Quadcopter

3.8.4 Electronic Speed Controllers

The electronic speed control or ESC shown in Figure 3.27 is a circuit responsible for controls the speed of the motors. It may also provide reversing of the motor and dynamic braking. It is controlled using a PWM signal generated by the autopilot

*Figure 3.**27**: ESC of the Quadcopter*

3.8.5 GPS and Compass

As demonstrated in Figure 3.28, GPS and compass are offered in a single package. The quad rotor is equipped with a GPS to determine its

position (longitude and latitude). It is a vital need for the autonomous quadrotor. The quad rotor's heading is determined by the compass inside the quad rotor.

Figure 3.28: GPS and compass Module With holder

3.8.6 Gyroscope and Accelerometer

On the same board as a 3-axis accelerometer and 3-axis gyroscope is a Digital Motion Processor capable of executing complicated 9-axis Motion Fusion algorithms. In order to balance the torques on their related axes, each motor of the quad rotor spins in a different direction than its neighboring motors. Therefore, the tail rotor is unnecessary.

Using the MPU6050 seen in Figure 3.29 or other variants as a tilting measurement sensor on a quadrotor, two main issues are identified in this study. In order to correctly characterize the issue, certain information regarding the sensor will be presented first. This sensor's output is used to compute acceleration along the three principal axes X, Y, and Z, as well as rotational velocities around the same axes. In order

to precisely control the quadrotor helicopter. The control system requires a tilting value, not a rotating speed. This may be determined by integrating the angular velocity to get the angle in all three directions. Everything seems to be in order, but it is a persistent reality that all sensors include an inaccuracy, although a very little one. This inaccuracy in measuring velocity leads to an error in calculating the angle, resulting in a cumulative error, as seen in Figure 3.30, since the current angle is dependent on the prior angle as a starting point. This accumulated mistake results in significant inaccuracies over time, as seen in Figure 3.31, causing the quadrotor to become unstable.

Fiaure 3.29: MPU6050

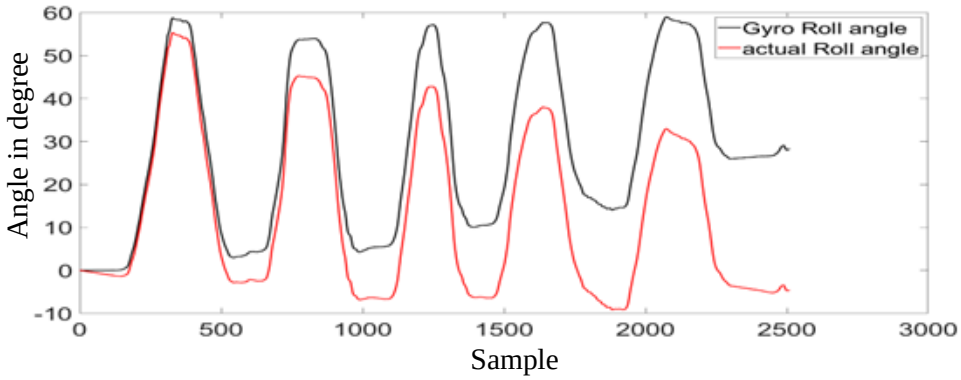


Figure 3.30: Real and measured roll angle out of Gyro

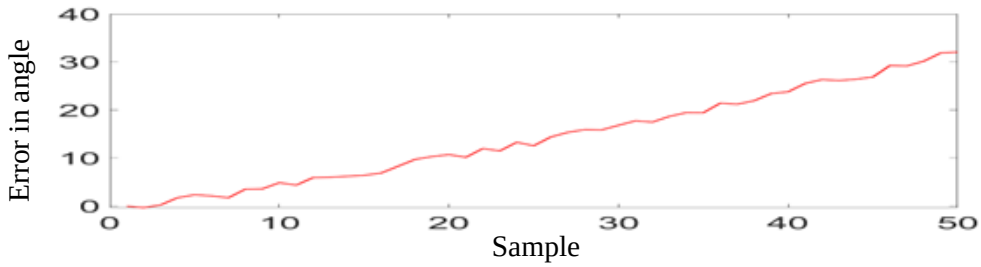


Figure 3.31: Gyro error signal

Using the MPU 6050's built-in accelerometer, it is possible to determine the tilt angle of the quad rotor around all three axes. This method immediately measures the angle without requiring an integration step. This method addresses the issue of cumulative mistakes. The accelerometer exhibits an additional issue. It is very sensitive to vibration, and as a result, the data it generates are rendered useless in the event of a quick vibration, such as that created by the motors in the quad rotor, as seen in Figure 3.32.

Now, the second issue that must be resolved will be illustrated. As illustrated in Figure 3.33, if the nose of the gyro (quad rotor copter) is pitched up 45 degrees, the pitch angle is increased as intended. Imagine what occurs when the yaw of the quad rotor spins ninety degrees in the clockwise direction, as seen in Figure 3.34c. Currently, the precise pitch axis of the quad rotor is horizontal, however the roll angle has risen. Because there is no angular motion in the pitch or roll direction, the pitch and roll angles detected by the gyro do not change (as they should).

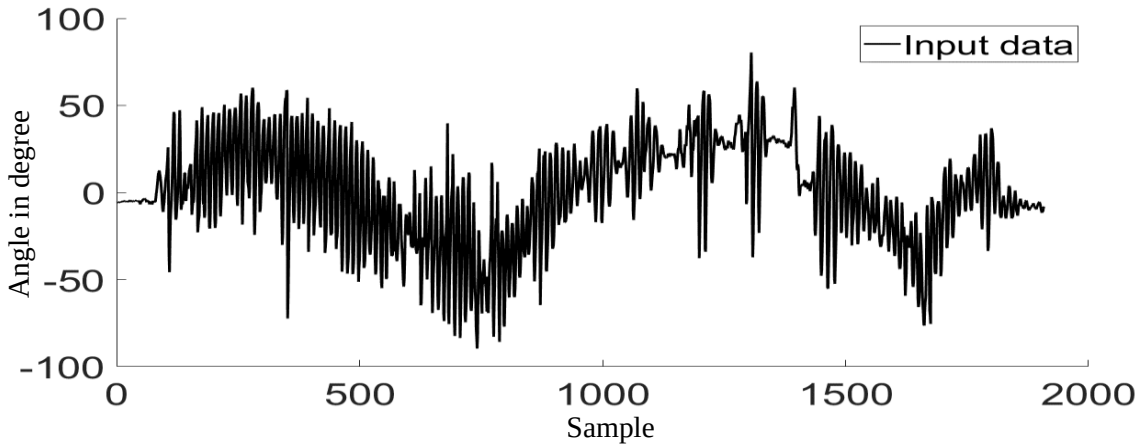


Figure 3. **32***: roll angle calculated from Accelerometer sensor*

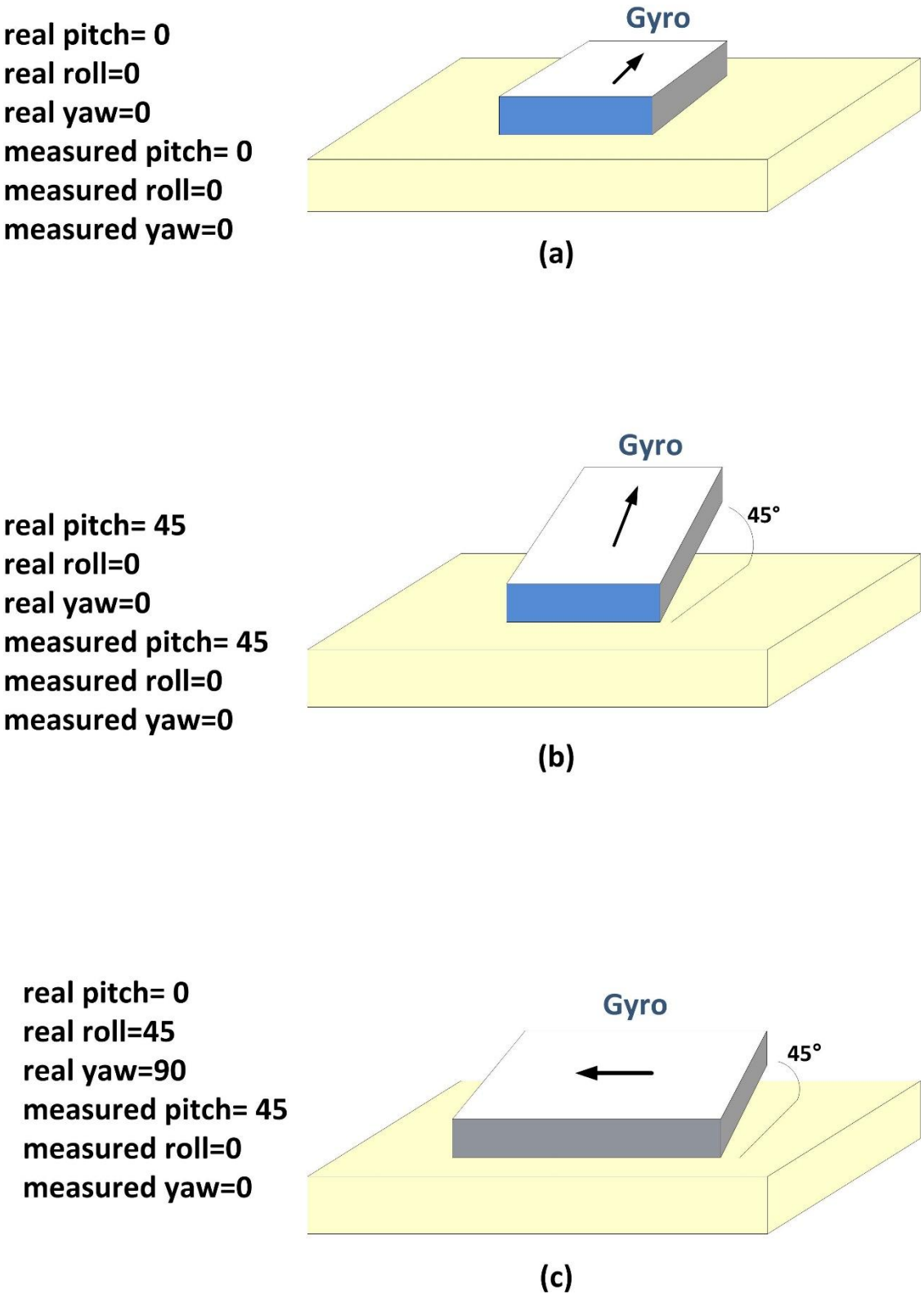


Figure 3.33: Gyro yaw problem during tilt.

MPU sensor on the quadrotor copter. This section will explore the possible remedies to the aforementioned issues.

The first issue was resolved by using the complementary filter, as seen in Figure 3.34. As seen in Figure 3.35, a simple collecting technique was employed between the data supplied by the gyroscope and the consequent data from the accelerometer at various speeds.

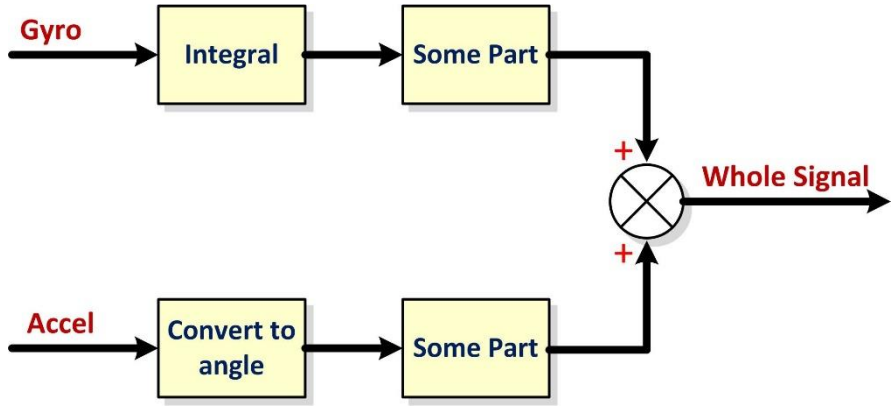


Figure 3.34: Simple design of the complementary filter

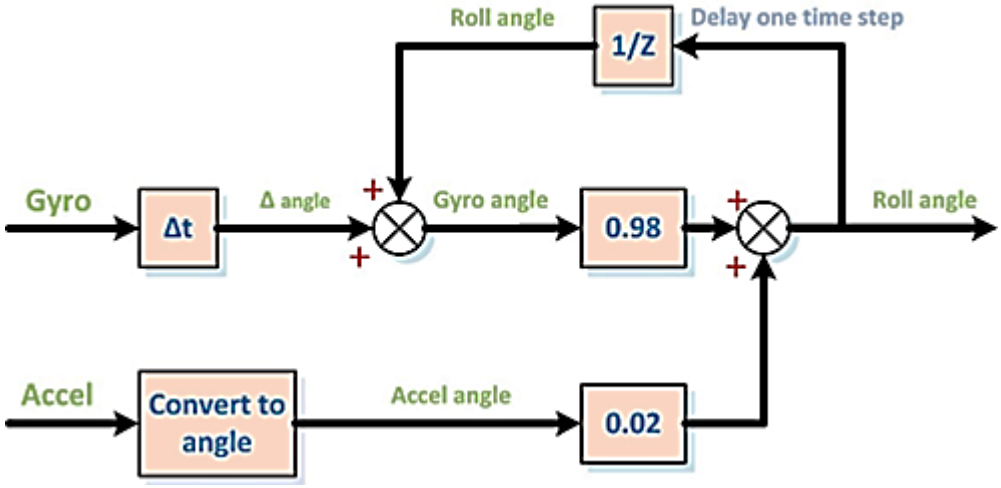


Figure 3.35: The Complementary filter used in this work

The gyroscope generates the majority of the data, whereas the accelerometer generates a minor fraction. Since the gyroscope data

includes a cumulative inaccuracy, a little portion of the accelerometer data may eliminate this problem. On the other hand, the vibration that affects the accelerometer when it operates independently has a minor influence currently due to its small proportion, which hardly affects the final tilting angle. Based on the goal function, a Gray Wolf Optimization GWO method is utilized to find the ratio of each component. As shown by Equation 3.36, the goal function consists of two sub-objective functions: phase shift delay and accelerometer noise. Figure 3.37 depicts the output roll angle of the complementary filter. The shift delay in the signal is negligible. This signal is superior than a signal received from each of them separately (gyro or accelerometer).

$$Obj = k * \ average\ phase\ shift\ delay \ + \ (1 - k) * ripple \quad \ldots\ldots (3.26)$$

Where

Obj: objective function.

k: a parameter that indicates phase shift delay impact.

$ripple$: ripple factor occurs on the output.

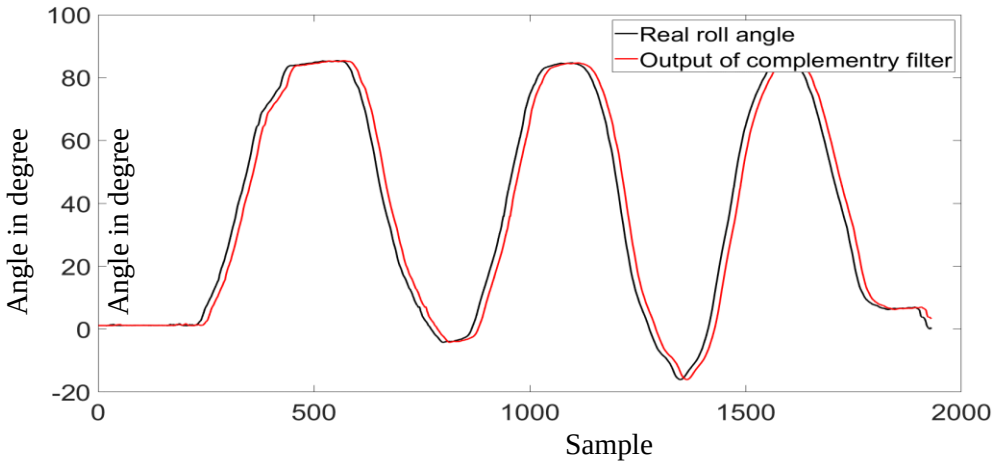


Figure 3.36: Real and measured roll angle out of optimized Complementary filter.

The second difficulty with the accelerometer data was resolved by developing a FIR filter for the accelerometer sensor data, therefore removing a significant portion of the noise caused by the vibration. The filter rendered the sensor data relevant and useable for the quadrotor aircraft. Figure 3.38 illustrates the magnitude response of the low pass FIR filter built for this article. The optimal order is 100, and the minimum frequency is 2 Hertz. The roll angle or frequency of data is always less than 2 hertz. Figure 3.37 displays the output of the filter.

The second difficulty outlined in the preceding section may be solved by linking the Yaw axis with the roll and pitch axes. When the Yaw axis monitors a rotation, the roll angle is transferred to the pitch angle and vice versa. However, what is the mathematical formula for converting roll angle to pitch angle? Based on certain measures, the connection is not linear. Figure 3.40 illustrates the relationship between the change in pitch angle and yaw rotation for every five degrees of yaw movement. Now, it is straightforward to identify that the form closely resembles the

sine function. Figure 3.38 and the formulae indicate that the function for transferring the pitch and roll angles is a sine function (3.39).

$$Roll\ angle\ =\ roll\ angle - pitch\ angle\ *\sin(yaw\ angle)$$

$$Pitch\ angle\ =\ Pitch\ angle\ +\ roll\ angle\ *\ \sin(yaw\ angle \quad \ldots\ldots (3.27)$$

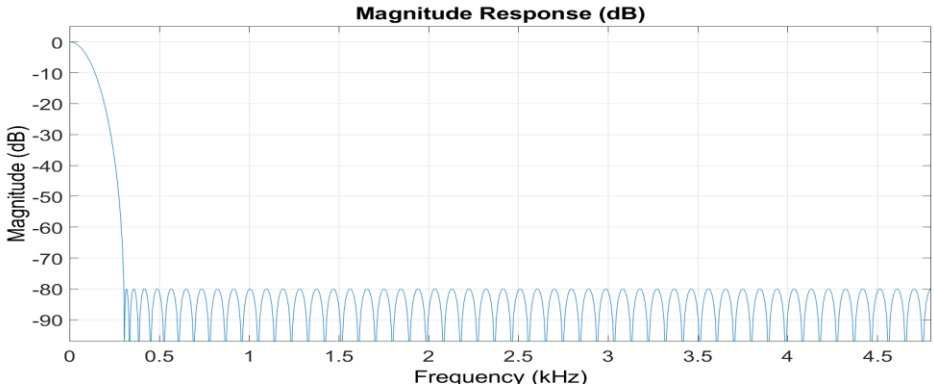


*Figure 3.**37**:Magnitude response of FIR filter*

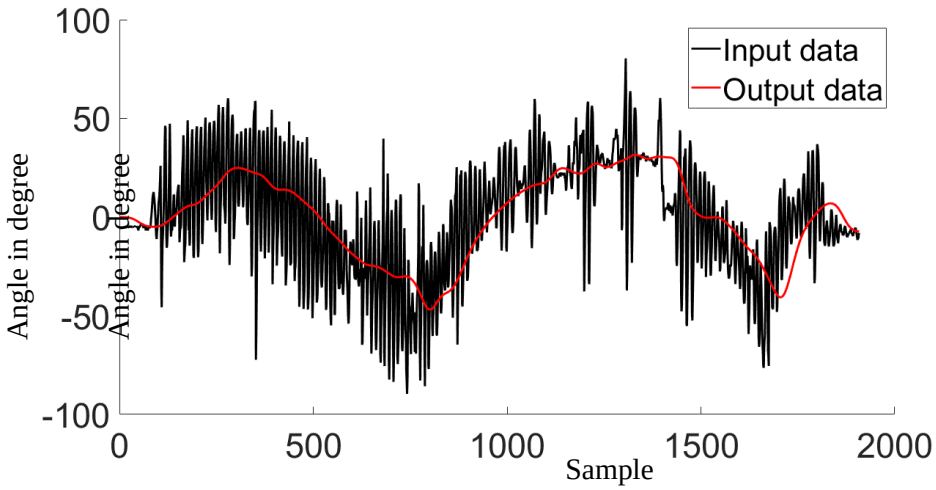


*Figure 3.**38**: Input and output of FIR filter*

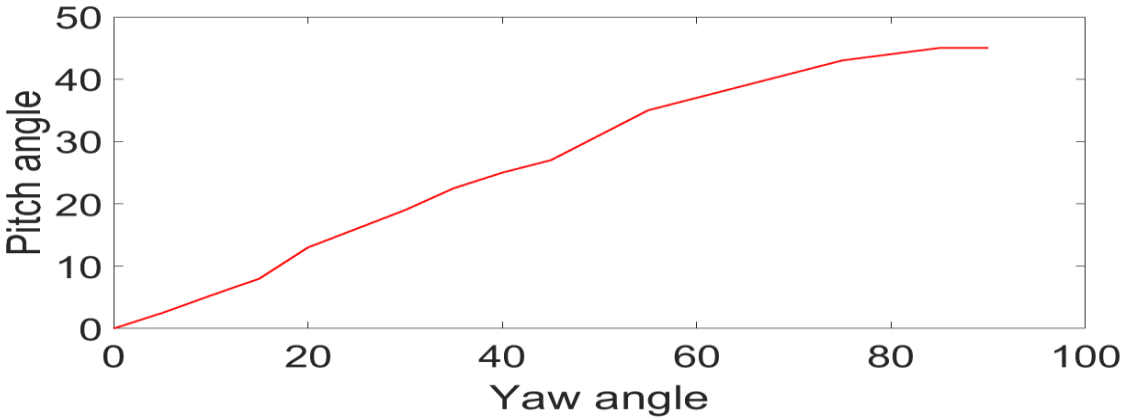


*Figure 3.**39**: pitch angle versus the Yaw rotation*

3.8.7 Building of the Quad Rotor

The first step in building a quad rotor involves properly Installations the frame (plate and arms). Most electronic parts are placed on the plate. The gyroscope is positioned at the center of the quad rotor as much as possible in order to reduce turbulence by the vibrations generated by the motors. Electronic and mechanical parts are distributed in a balanced manner to maintain the center of gravity of the quad rotor in the middle of it. The electronic parts were connected, as shown in Figure 3.40. Figures 3.41 and 3.42 shows the practically executed quad rotor. This quad rotor can track the path produced by the algorithms proposed in this thesis. In general, the quad rotor that implemented in this thesis can do the following tasks.

- **Tracking** : The practically implemented quad rotor in this thesis can receive three-dimensional points to track them sequentially without the need to control them remotely by a human being.
- **Simplicity**: It can be easily controlled remotely. The quad rotor automatically balances itself as a result of the good use of the gyroscope and GPS.

Fixed location and wind resistance: It can resist the movement of the wind regardless of its direction. The quad rotor maintains its position and resists the wind depending on the GPS and the compass

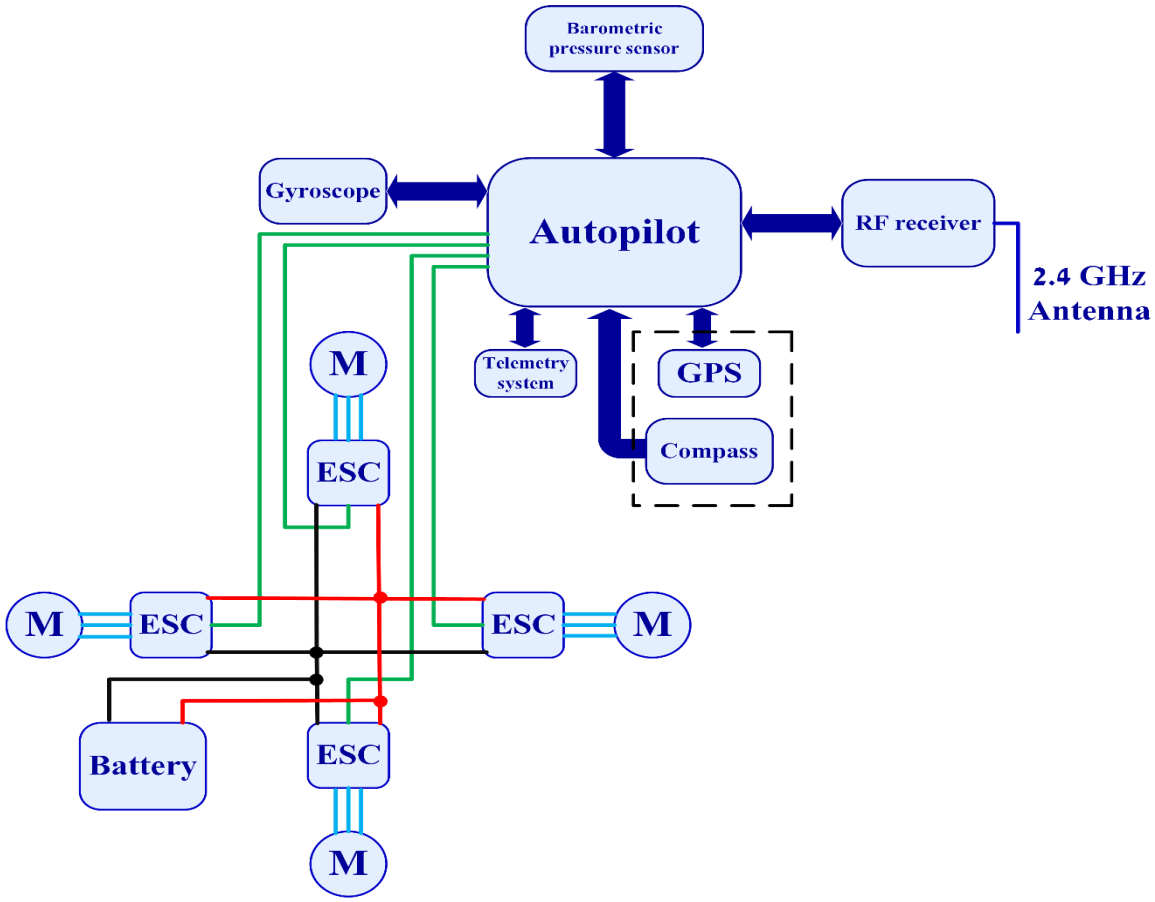


Figure 3.40: The electronics connection of the quadrotor

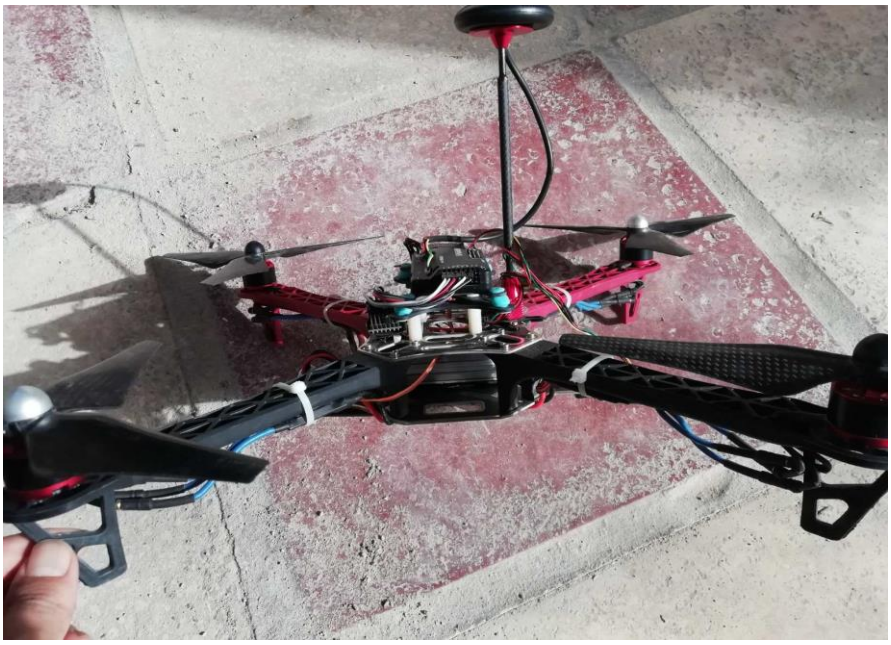


*Figure 3.4**1**: Implemented quadcopter view 1.*

*Figure 3.4**2**: Implemented quadcopter view 2.*

3.9 Encryption Using RSA Algorithm

The information that is calculated in the plane is the location of the target, which is very sensitive information, so this data was encrypted using the RSA algorithm. The schema relies on two keys, the public key and the private key. The plane encrypts the location of the target and sends it to the ground station as shown in Figure 3.43, and the ground station contains the key. Private where you can decode this sensitive information and display it. A 256-bit key algorithm was used to ensure data encryption. Only The position of the target encrypted with RSA. The encrypted data size is around 10 KB.

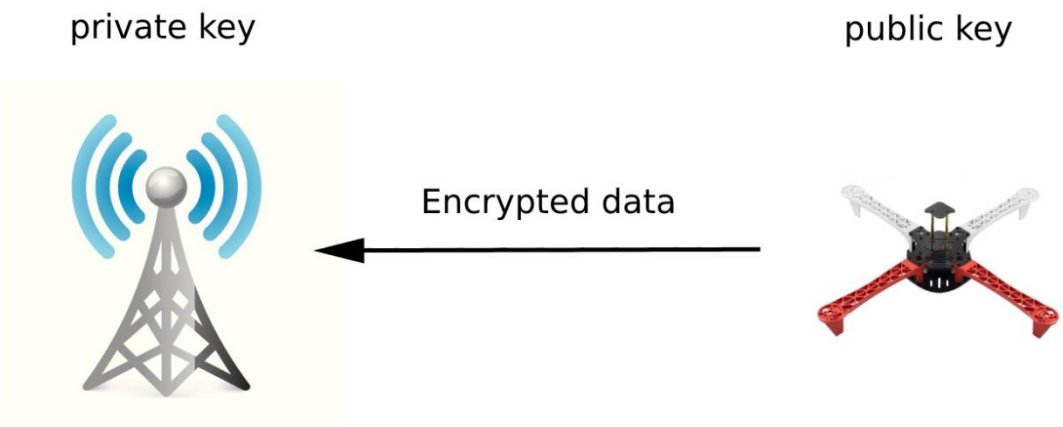


Figure 3.43: Encryption using RSA

CHAPTER FOUR

The Results and discussion

4.1 Introduction

This chapter presents the results obtained in this work, including the results of the proposed algorithms as well as the results for quadcopter simulation and practical results. The results are discussed in this chapter. In this chapter, detailed results for each of the proposed algorithms will also be presented. In addition to the results of comparison with another recent research. The results of the practical side will also be presented in this chapter

4.2 The n-Visibility Tree Algorithm Results

The implemented simulation was tested for multiple cases and different numbers of obstacles. Various obstacle sizes were used. Figure 4.1 illustrates the connection curves between points, as observed by the source and target sides. Each point from the source side is connected to all other n points on the target sides. The minimum path belongs to the sets of paths in Figure 4.1. According to the n-visibility tree algorithm, the minimum path is illustrated in Figure 4.2.

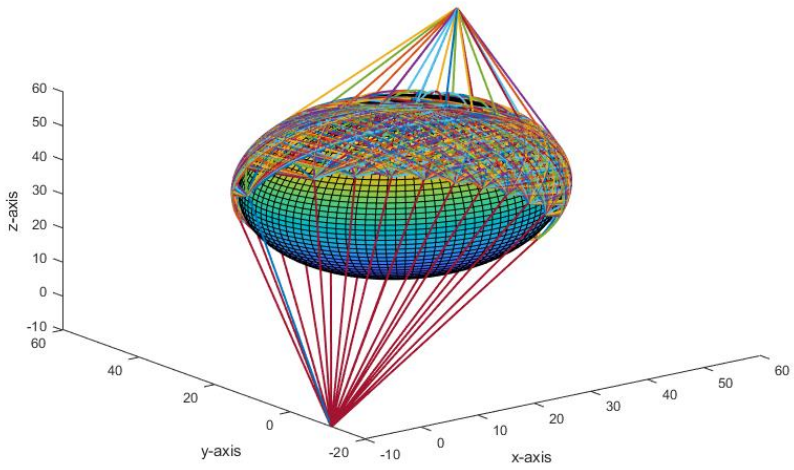


Figure 4.1: One to all connection curves n=20

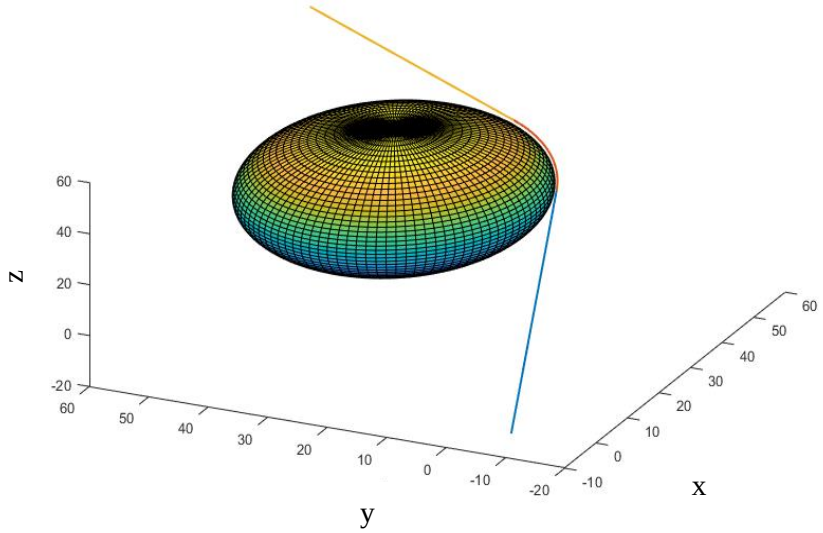


Figure 4.2: Optimal path of one obstacle

Figures 4.3 illustrate a complete set of paths when two obstacles of the same size exist and face the direct path of the drone to its target. In Figure 4.3, use n=4, which means that there are four points and four lines constructed around each obstacle. Figure 4.5 illustrates the minimum path of Figure 4.4 according to the n-visibility tree algorithm.

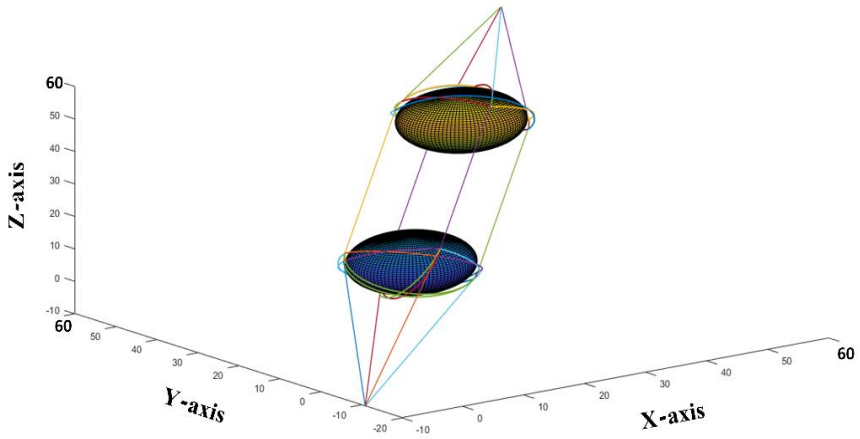


Figure 4.3: Complete set of paths with two same size obstacles n=4

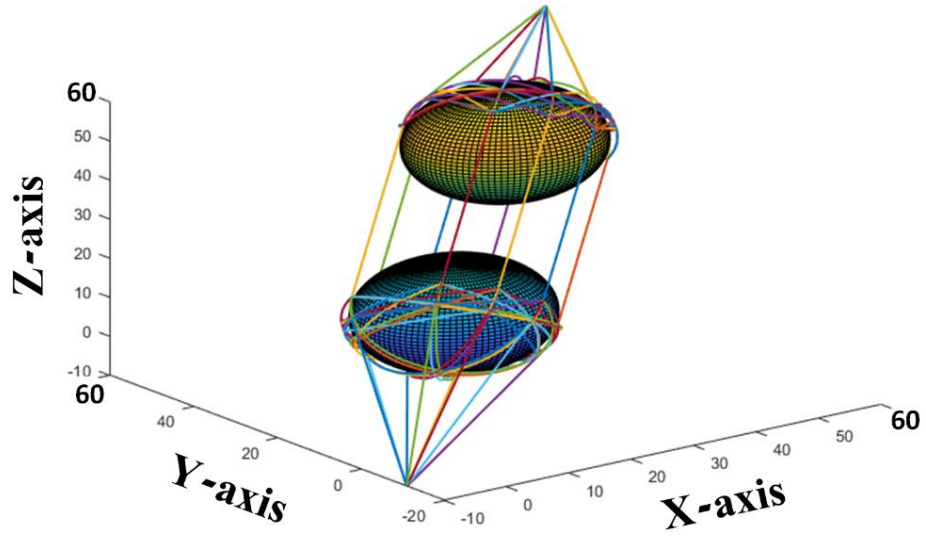


Figure 4.4: Complete set of paths with two same size obstacles n=8

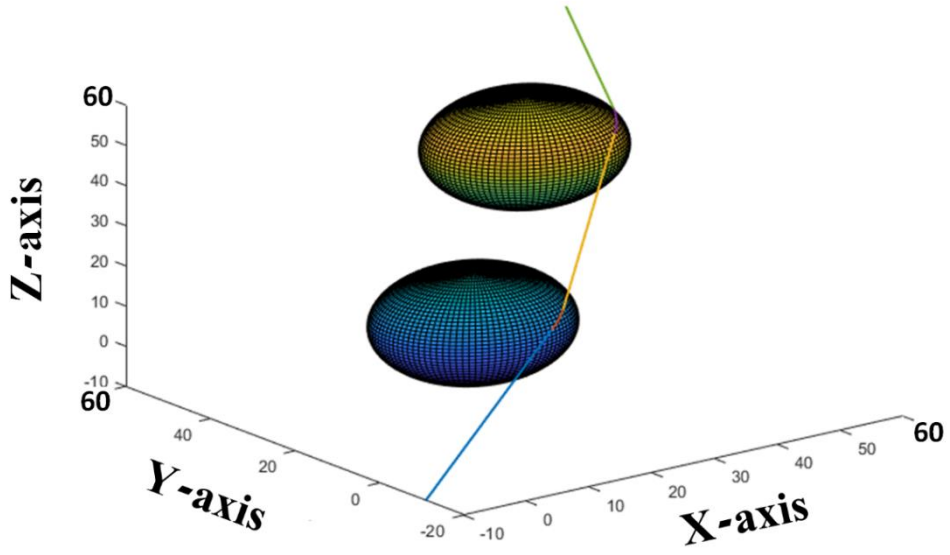


Figure 4.5: Optimal path

Figures 4.6 and 4.7illustrate a complete set of paths when two different-sized obstacles exist. Figure 4.6 uses n=4, which means there are four points and four lines constructed around each obstacle. Figure 4.8 illustrates the minimum path of Figure 4.6 according to the n-visibility tree algorithm.

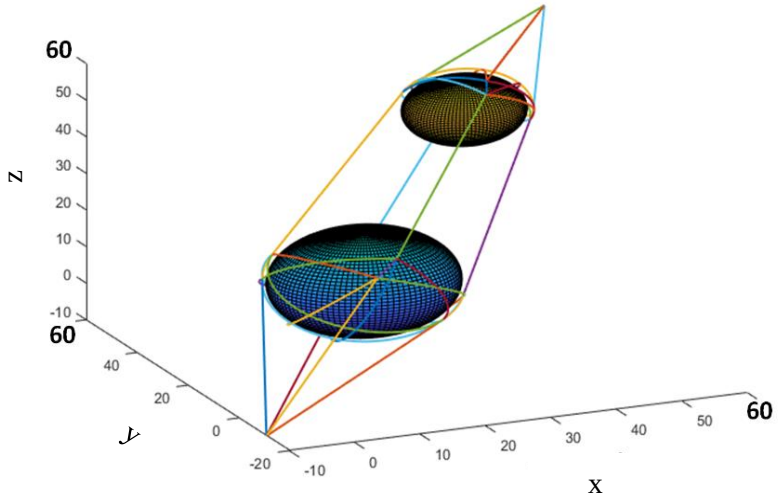


Figure 4.6: Complete set of paths with to different size obstacles n=4

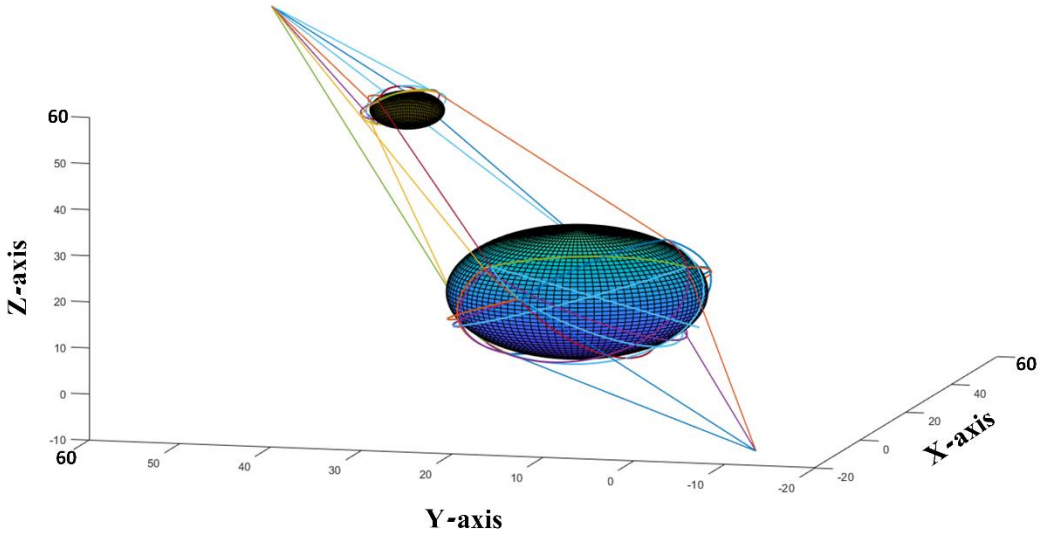


Figure 4.7: Complete set of paths with to different size obstacles n=4

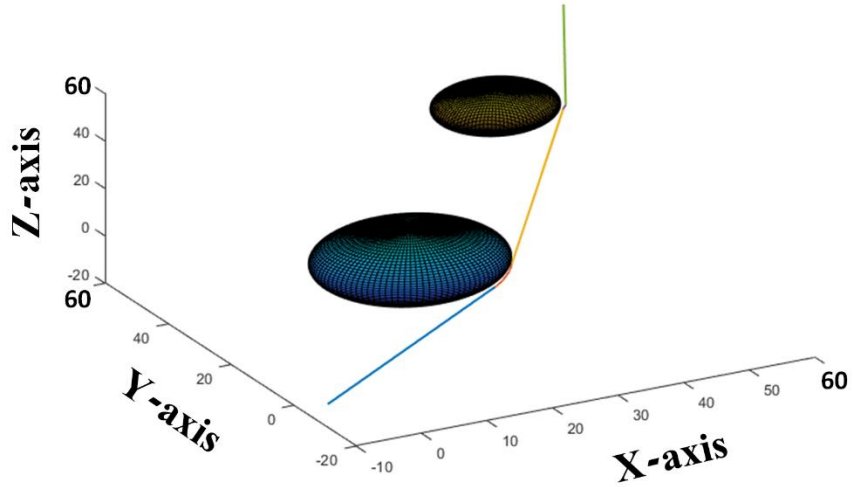


Figure 4.8: Optimal path

Figure 4.9 demonstrates a random environment consisting of 40 random-sized obstacles. The n-visibility tree algorithm draws a set of complete paths from the source point to the target. The optimal path illustrated in Figure 4.10 belongs to the set of paths in Figure 4.9. Figures 4.11 and 4.12 depict a random environment consisting of 50 random-sized obstacles. The n-visibility tree algorithm draws a set of complete paths from the source point to the target. The optimal path illustrated in Figure 4.13 belongs to the set of paths in Figure 4.12. The spherical obstacles can intersect with one another to construct additional obstacle shapes, such as buildings and towers, as illustrated in Figures 4.14 and 4.15. The proposed n-visibility tree algorithm is used in this scenario to determine the minimum path. Figure 4.15 illustrates some of the complete paths during the operation of the proposed algorithm.

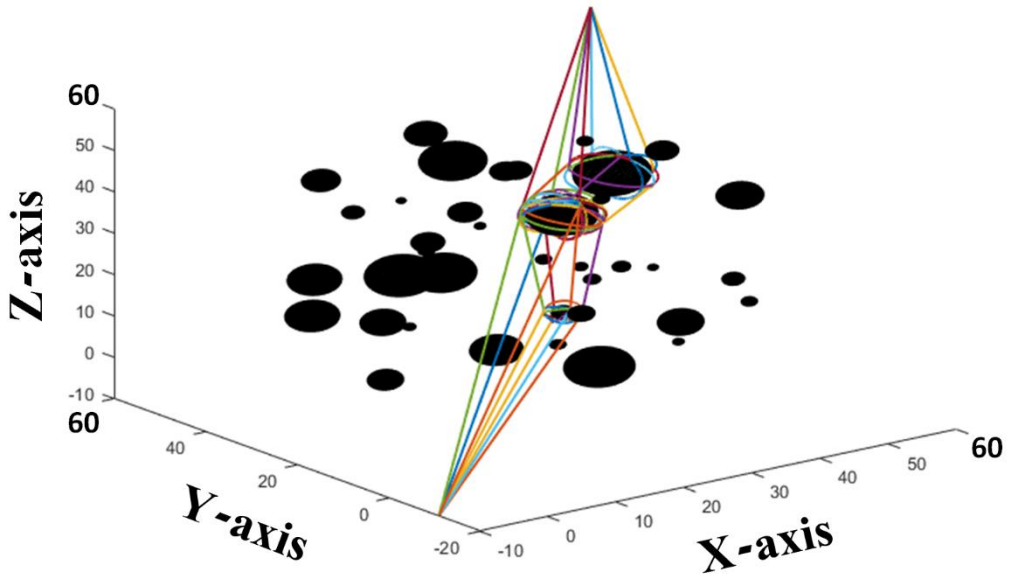


Figure 4.9: Complete set of paths with to random size forty obstacles n=4

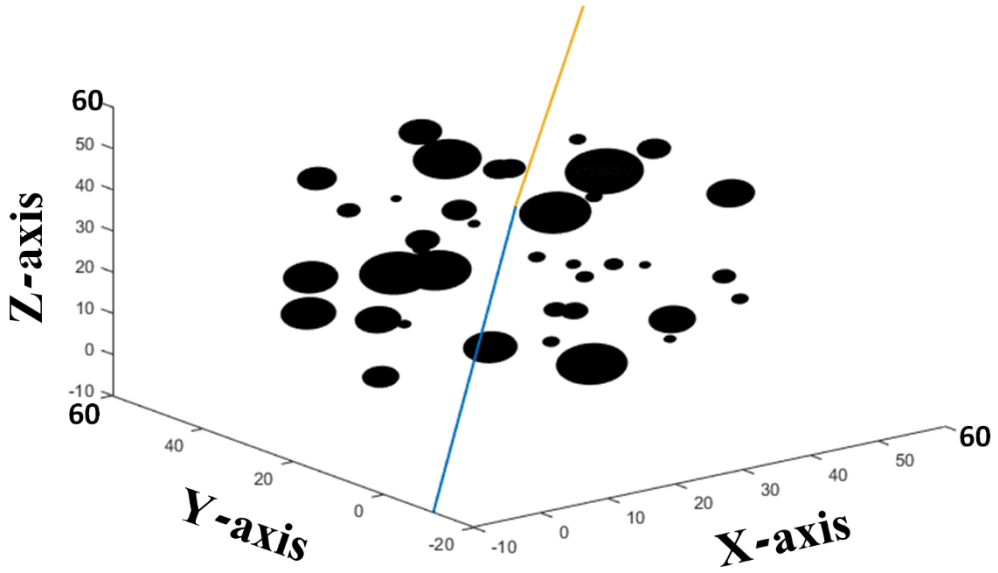


Figure 4.10: Optimal path

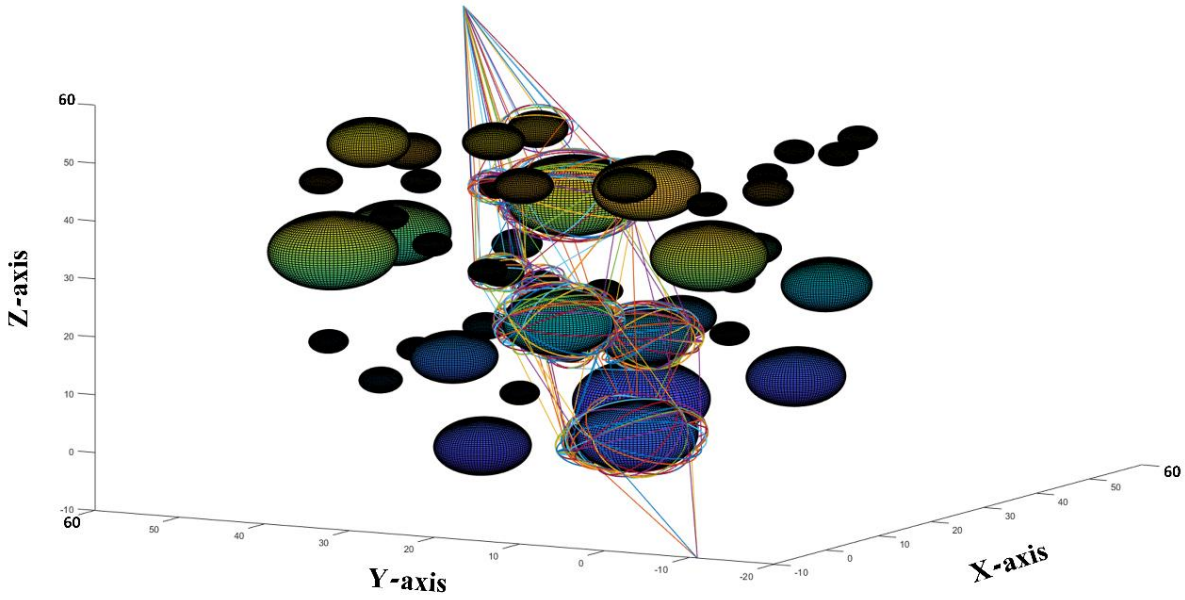


Figure 4.11: Complete set of paths with to random size fifty obstacles n=4

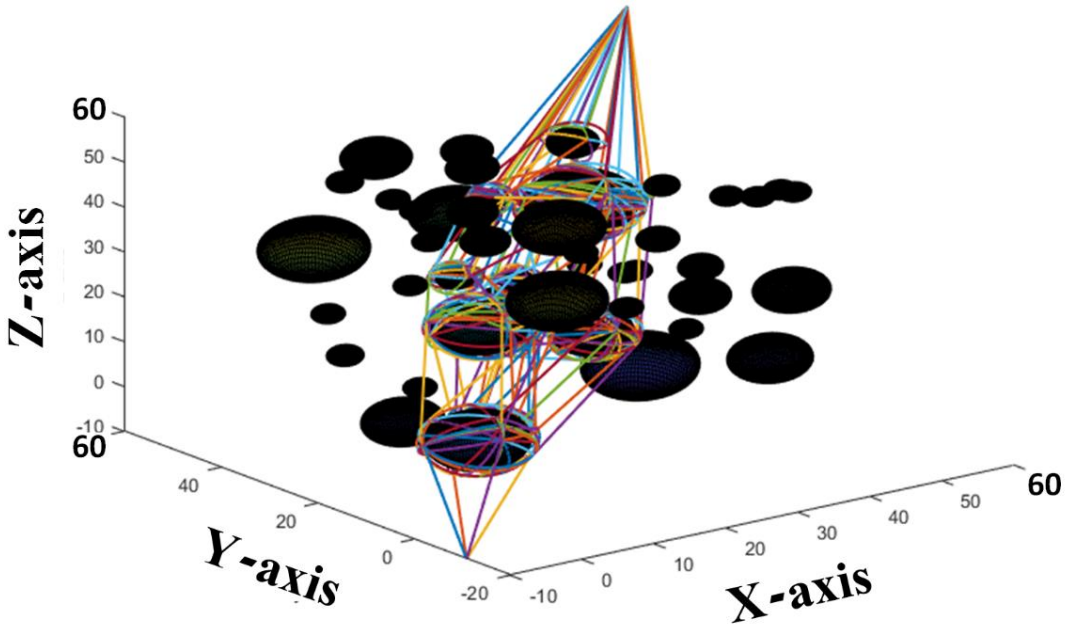


Figure 4.12: Complete set of paths with to random size fifty obstacles n=4

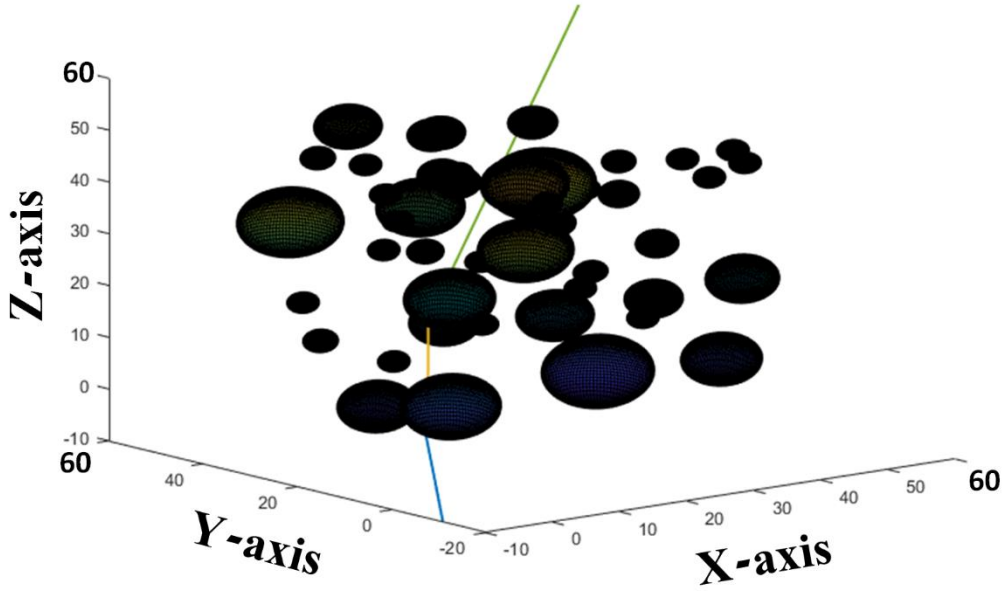


Figure 4.13: Optimal path

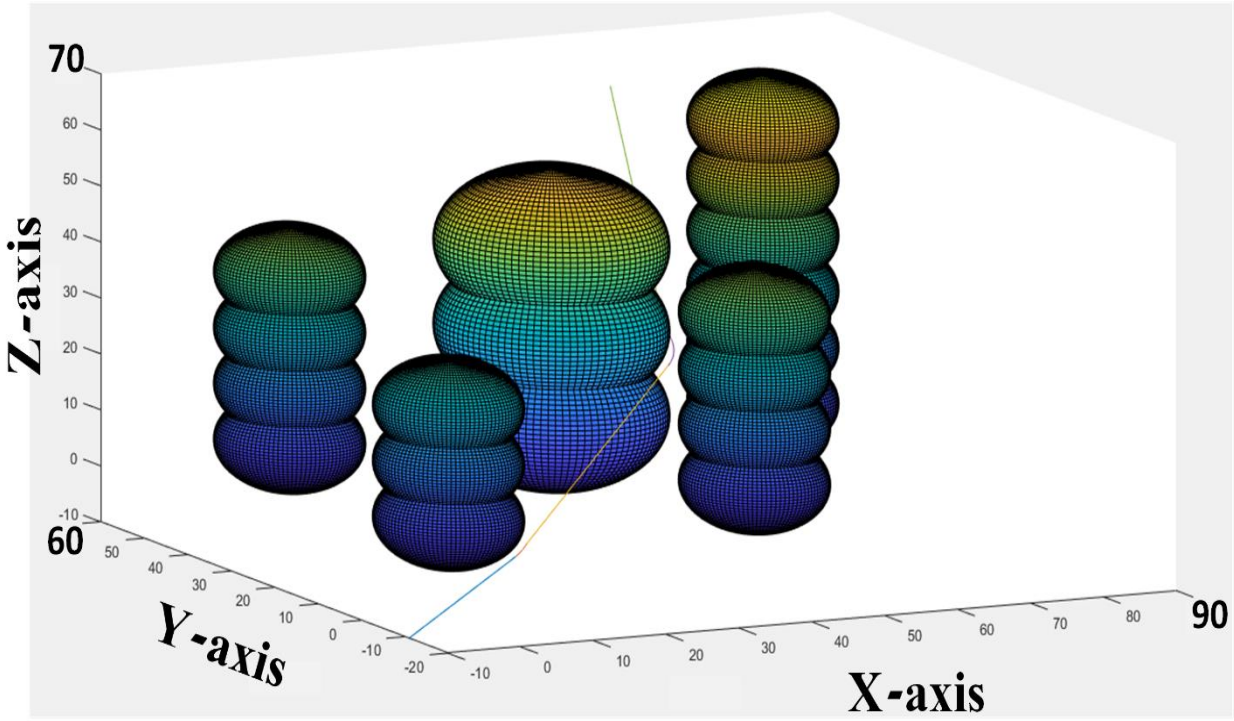


Figure 4.14: building representation by spheres

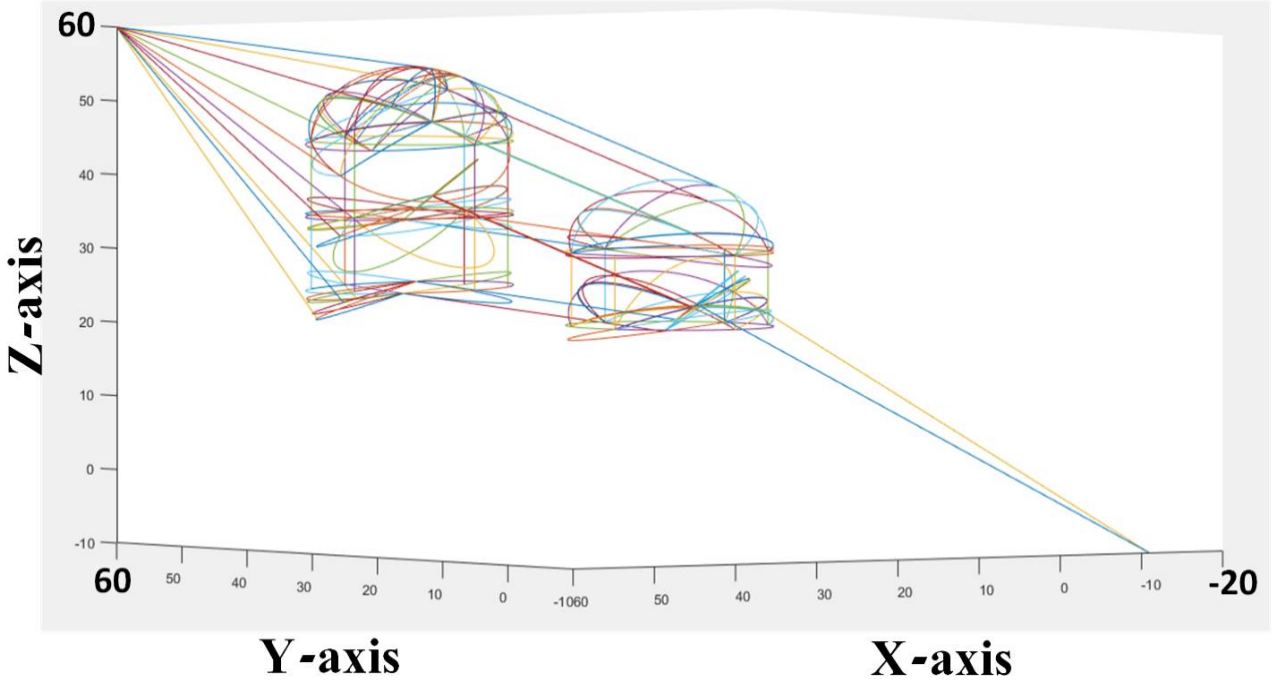


Figure 4.15: some of complete paths of figure

Figure 4.16 and 4.17 represent a complex environment tested in this algorithm. These environments represented in Figures 4.18 and 4.19 respectively by spherical building units to use the n-visibility tree algorithm for solving the path planning problem. The line in Figures 4.18 and 4.19 represent the minimum path for the drone.

There are three parameters affect on the performance of algorithm n, r, and R. Increasing n leads to an increase in the optimality (decrease the path length) and complexity (increase the computation time), as shown in Figure 4.20. This experiment was performed using the following conditions: Dimensions are (70 * 70 * 70) meter, n is variable, R+r = 10, and Number of obstacles =10.

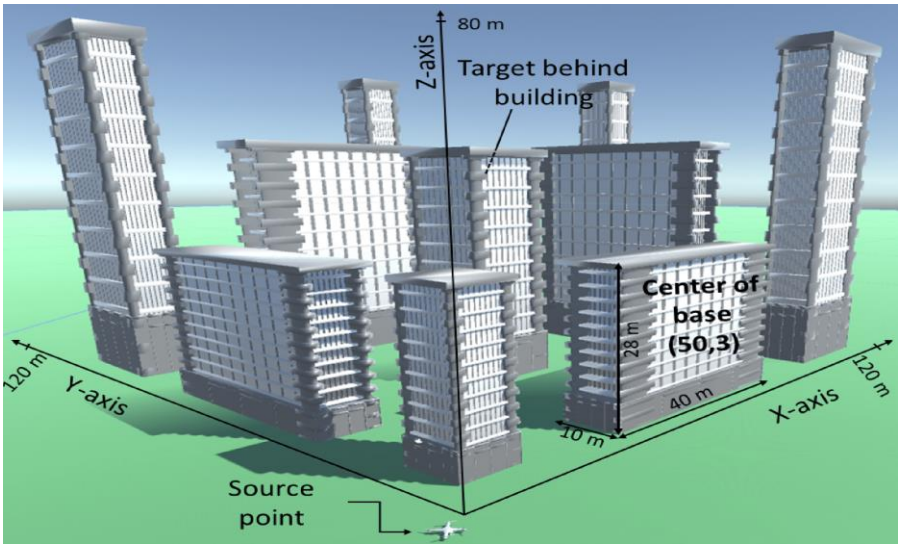


Figure 4.16: Complex environments (buildings)

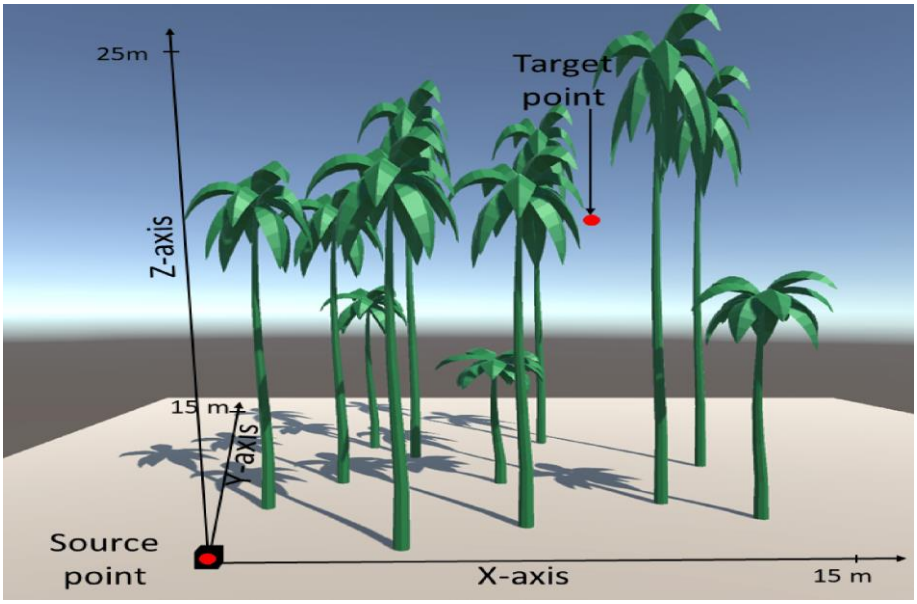


Figure 4.17: Complex environments (Palms)

Data of Building A

Center of sphere			Radius
35	3	5	9
35	3	15	9
35	3	25	9
45	3	5	9
45	3	15	9
45	3	25	9
55	3	5	9
55	3	15	9
55	3	25	9
65	3	5	9
65	3	15	9
65	3	25	9

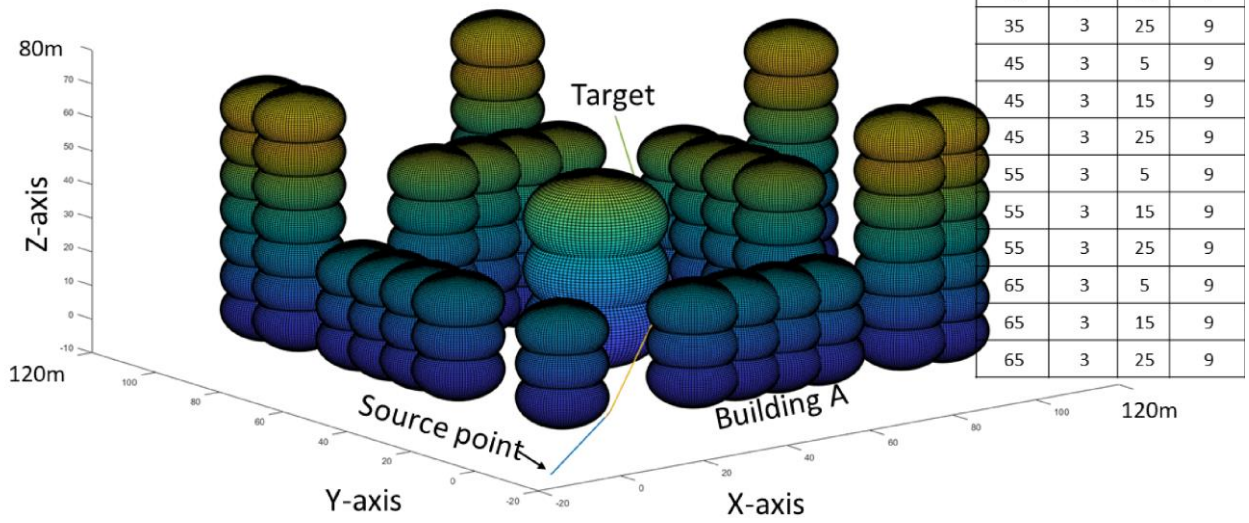


Figure 4.18: complex environment representation and minimum path for buildings

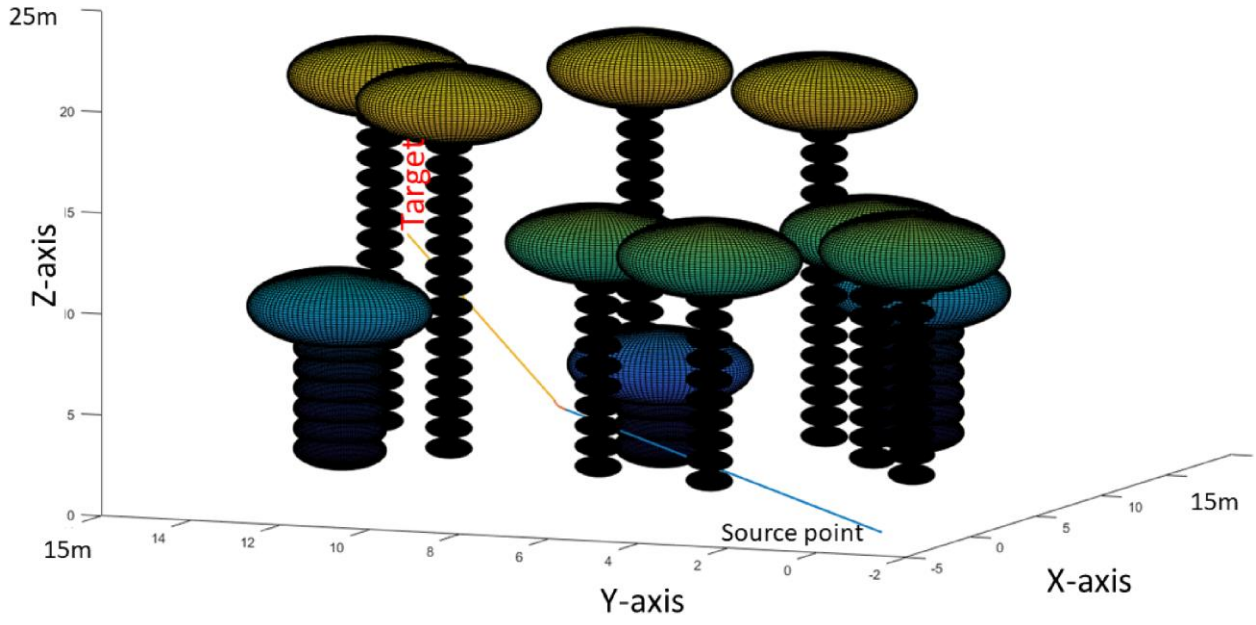


Figure 4.19: complex environment representation and minimum path for palms

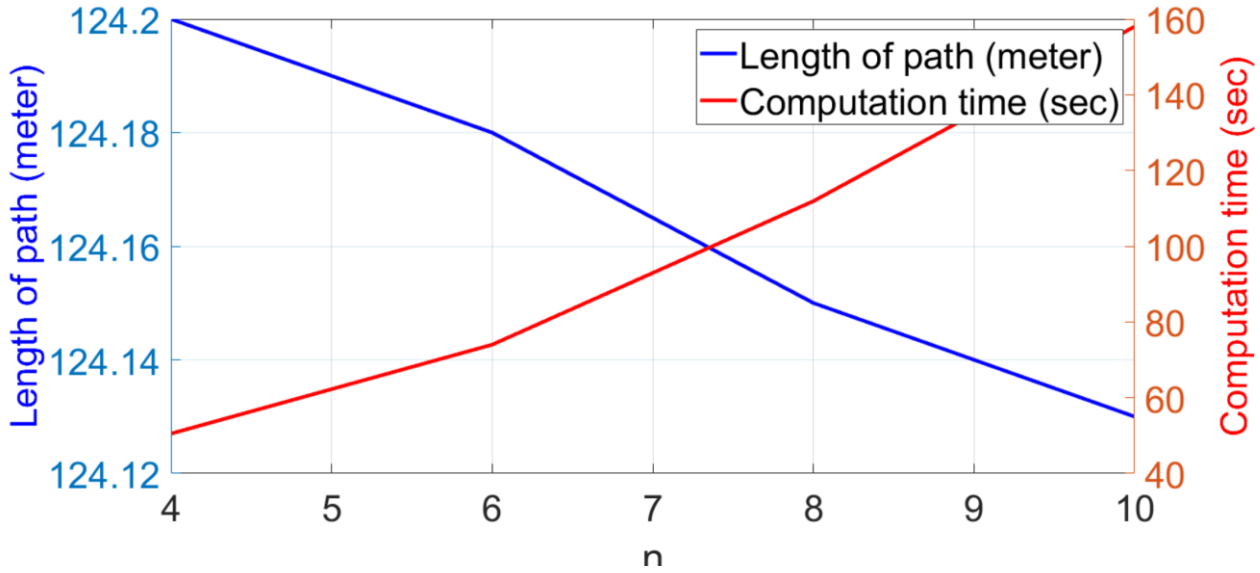


Figure 4.20: Path length and Computation time related to n values with R+r=10

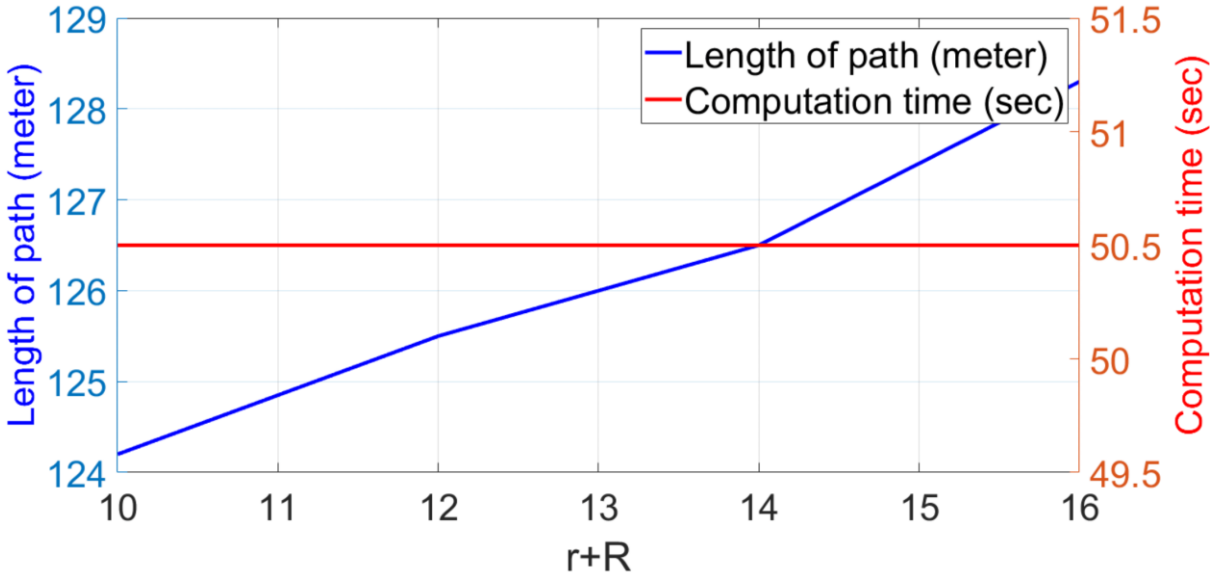


Figure 4.21: Path length and Computation time related to r+R with n=4

4.2.1 Comparisons and discussions of path planning

The introduced N-visibility tree algorithm was compared to two other algorithms and demonstrated an improvement in accuracy for determining the minimum path and computational time. Figure 4.22 illustrates the comparison results between the n-visibility tree algorithm, the RRT* algorithm, and the 3D PF algorithm. The x-axis represents the number of obstacles, whereas the y-axis represents the average processing times in seconds. Figure 4.23 illustrates the comparison results between the n-visibility tree algorithm, the RRT* algorithm, and the 3D PF algorithm. The x-axis represents the number of obstacles inside the workspace, whereas the y-axis represents the average lengths. It can be observed from comparing Figures 4.22 and 4.23 that there is a clear difference between algorithms that use random optimized searches, such as the RRT series, and those using equations based on n-obstacle tangents to determine. the minimum path (i.e., n-visibility tree algorithm).

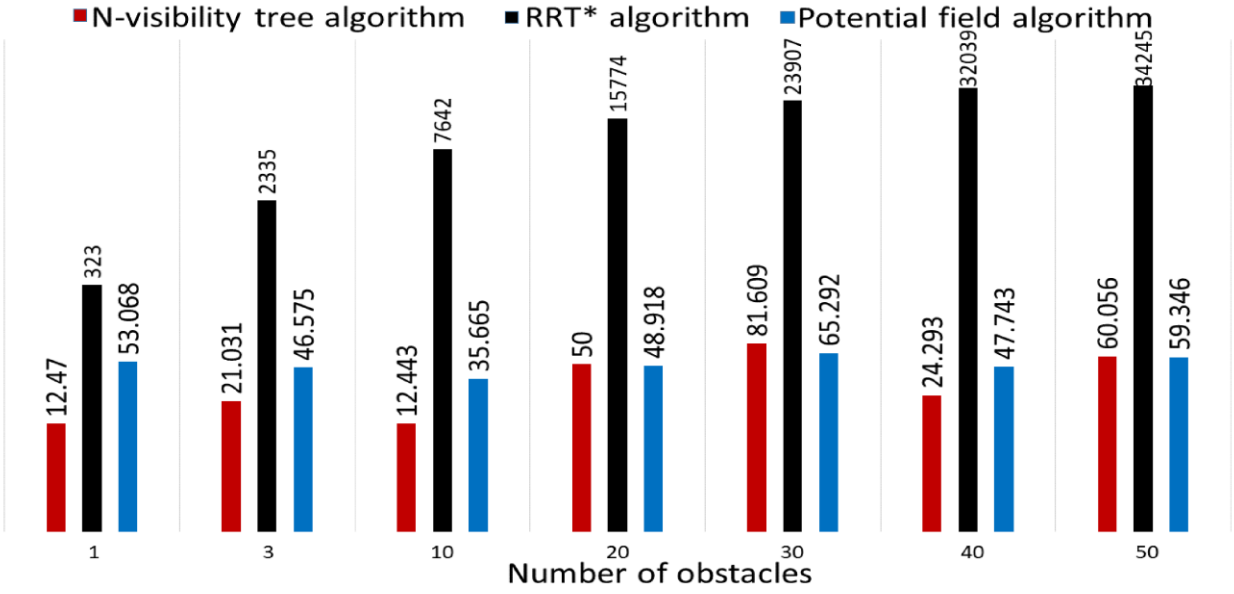


Figure 4.22: Comparison between algorithms based on computational time in sec

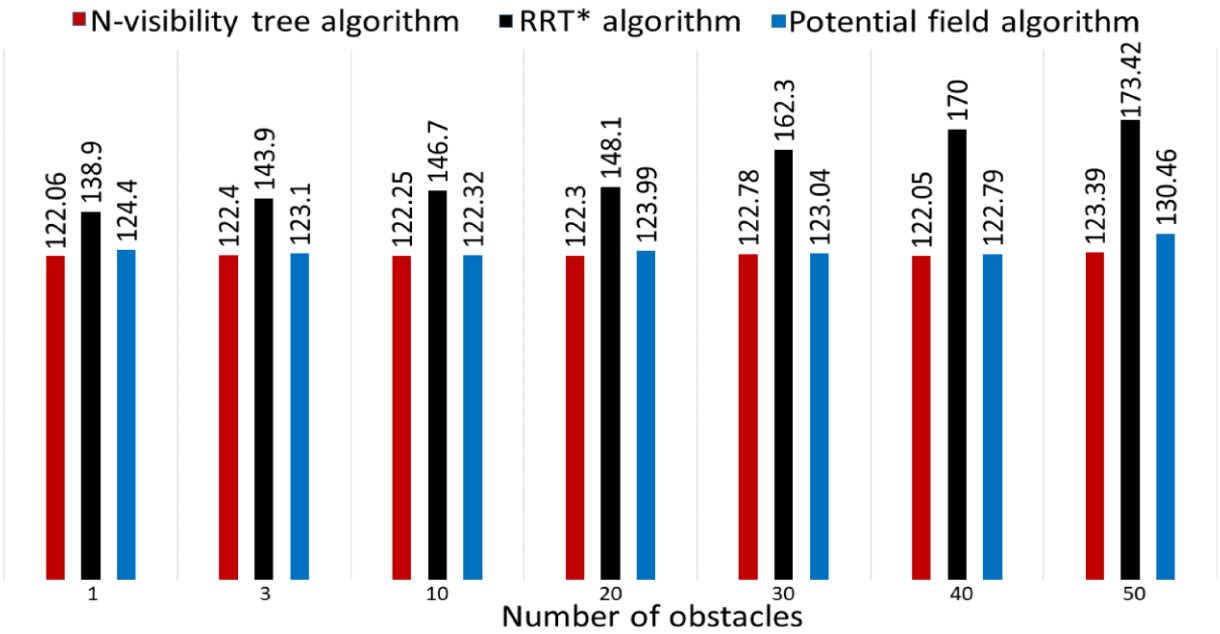


Figure 4.23: Comparison between algorithms based on length of path in meter

The introduced algorithm is compared to four other algorithms, and demonstrates an improvement in accuracy for determining the minimum path and computational time. Figures (4.24) and (4.25) illustrates the comparison results between the n-visibility tree algorithm, PRM [18], Wavefront [5],Potential field [8] and COSPS algorithm [5] based on path length and computation time.

The accuracy many algorithms like PRM, Wavefront, and COSPS algorithm depend on w which represents a distance between any point and its neighboring point. The simulation results of the suggested approach, PRM, Wavefront, Potential field, and COSPS algorithm are described on the basis of the difference in the considered obstacles and points using maps and w diagrams.. where the map is randomly distributed of 18 obstacles in the workspace 2000*2000*2000. w changes from 200 to 100. The accuracy and computational time depend

mainly on the number of points used to represent the grid of the workspace. The number of points depends on the distance (w). So, the results are affected by varying w. Basically, decreasing w lead to an increase in the points in the space which leads in turn to increase both accuracy and computational time. The results in Figures 4.24 and 4.25 show that the proposed algorithm produces the shortest path in all cases. When w is small, the length of the paths becomes very close but the computational time for the proposed algorithm has significant computation time reduction. On the other hand, when w is wide, The COSPS algorithm outperforms the proposed algorithm by computing time while losing in the length of the path due to decreasing accuracy. It can be concluded from the comparisons that the proposed algorithm and potential field algorithm doesn't depend on w. Note that the simulation of the proposed algorithm was performed with n=4.

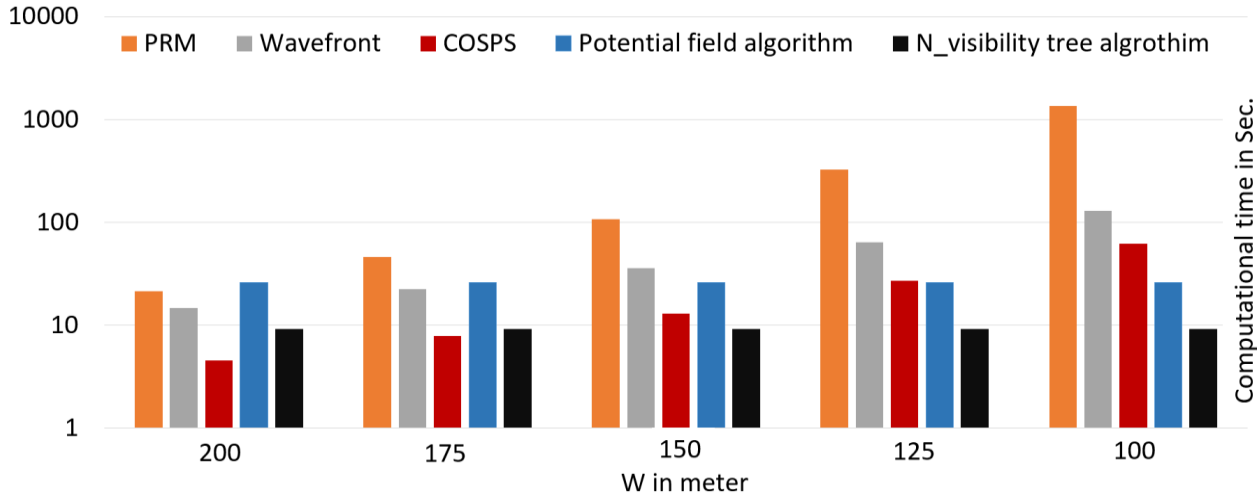


Figure 4.24: Comparison between algorithms based on computational time in sec

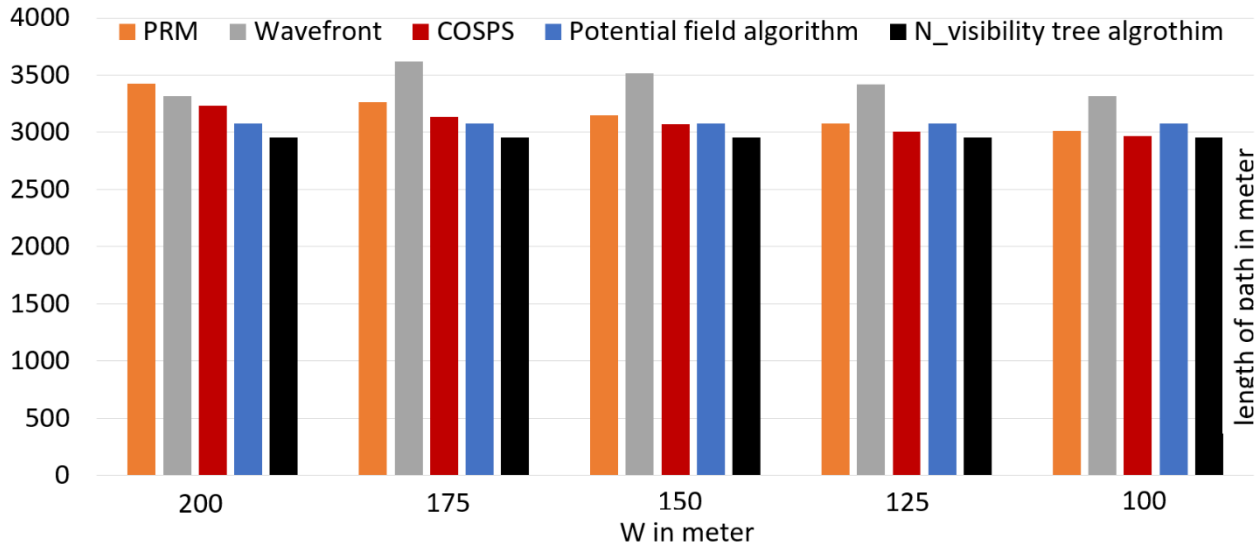


Figure 4.25: Comparison between algorithms based on length of path in meter

4.2.2 Simulation and Practical Work Results

The quadcopter was simulated in Matlab to get the response of the system. The responses in x, y and z directions as explained in Figures 4.26 , 4.27, and 4.28 respectively. In the x-axis step response, there is a small disturbance at time step 20. This disturbance caused by the response of the quadcopter to the y-axis at the same time, step 20. In the same manner, there two disturbances in the response of the z-axis in the time step 7 and 20. This happens due to the response of the quadcopter to the x-axis and y-axis. These interactions between the response in different directions are a natural result for the nonlinearity of the quadcopter system. A 3D path is loaded to the simulation to generate a result path from the simulation. Both wanted and actual paths explained in Figure 4.27. It's clear from the resulting path; It is very close to the desired path generated by the algorithms. There are only small divergent

in the corners of the path. Figures 4.28, 4.29 and 4.30 represent a response of quadcopter in each x,y and z-axis views.

The implemented quadcopter in the air explained in figures 4.31 to 4.33

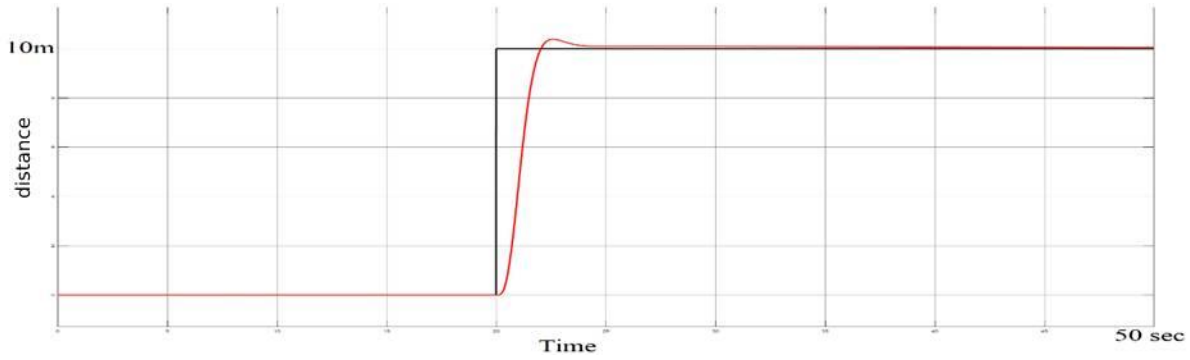


Figure 4.26: Step response of quadcopter in z axis

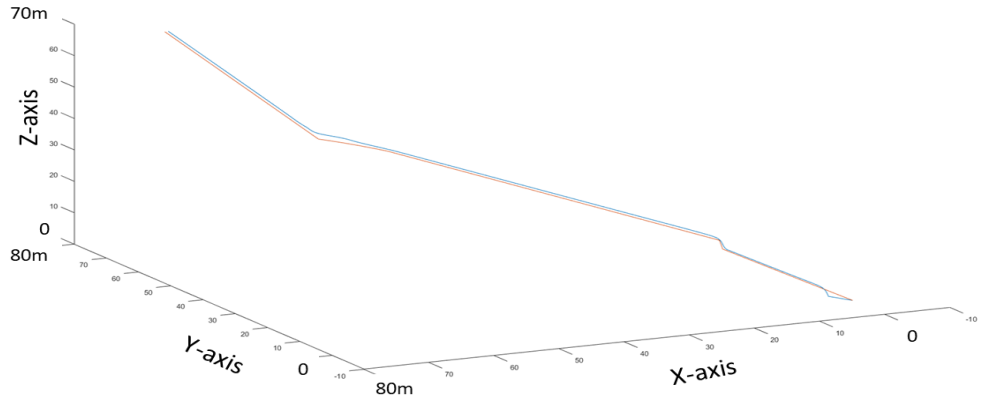


Figure 4.27: Algorithm and simulation paths

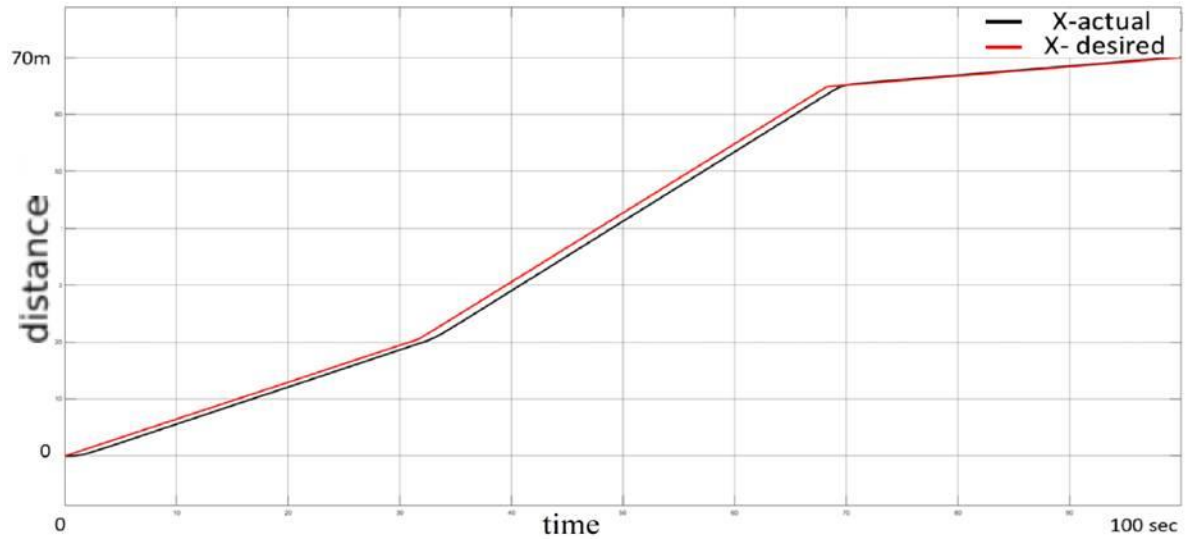


Figure 4.28: Path response of quadcopter in x axis

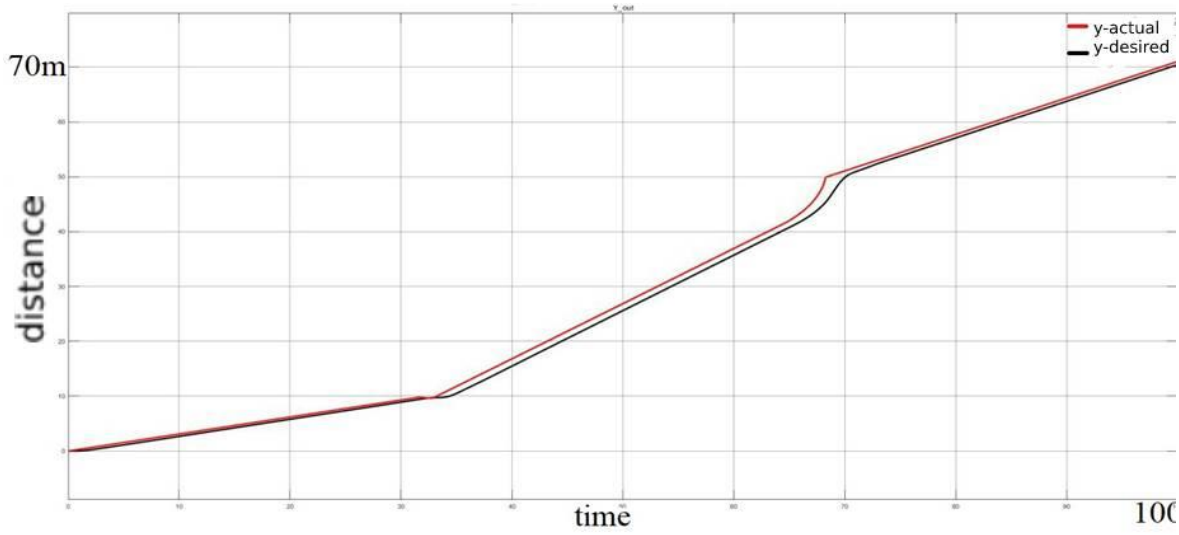


Figure 4.29: Path response of quadcopter in y axis

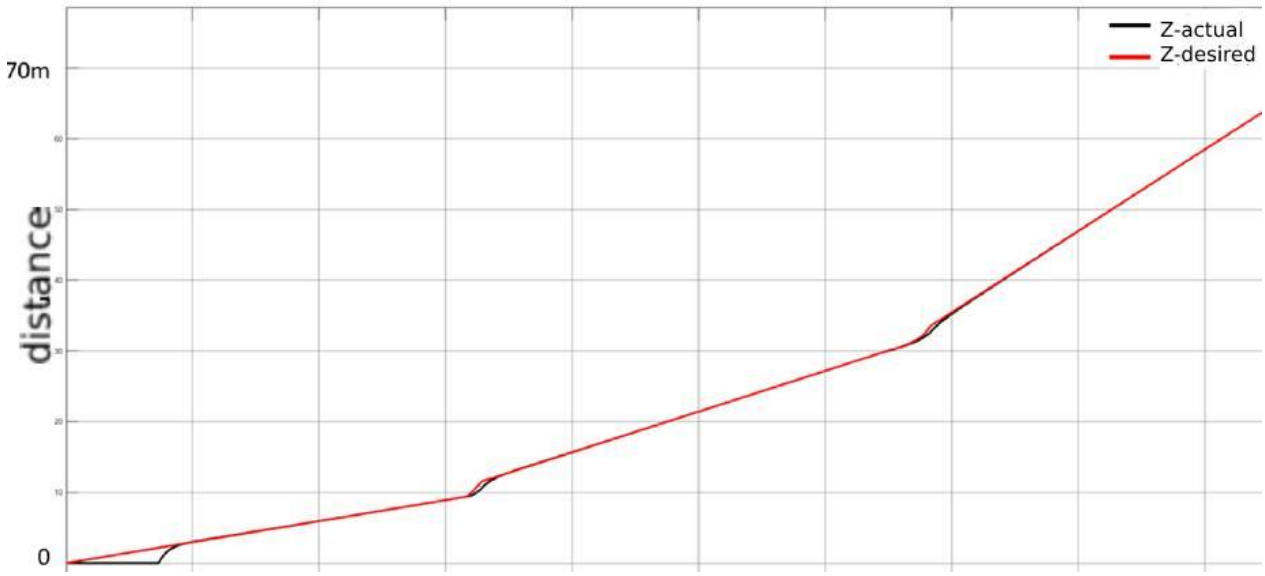


Figure 4.30: Path response of quadcopter in z axis

Figure 4.31: quad rotor flying in the sky

Figure 4.32: quad rotor start flying

Figure 4.33: quad rotor fly far into the sky

The most effective parameter on the compass is the interference with the magnetic field that produces due to the main power line of the motors. Figure 4.34 shows the interference when the internal compass is used. it is clear that the interference is huge. there are several methods to reduce interference like covering the high-power wires by a grounded conductor. This process reduced the interference to be in Figure 4.38. but it is still unacceptable interference. The interference was overcome by using an external compass to be in Figure 4.36.

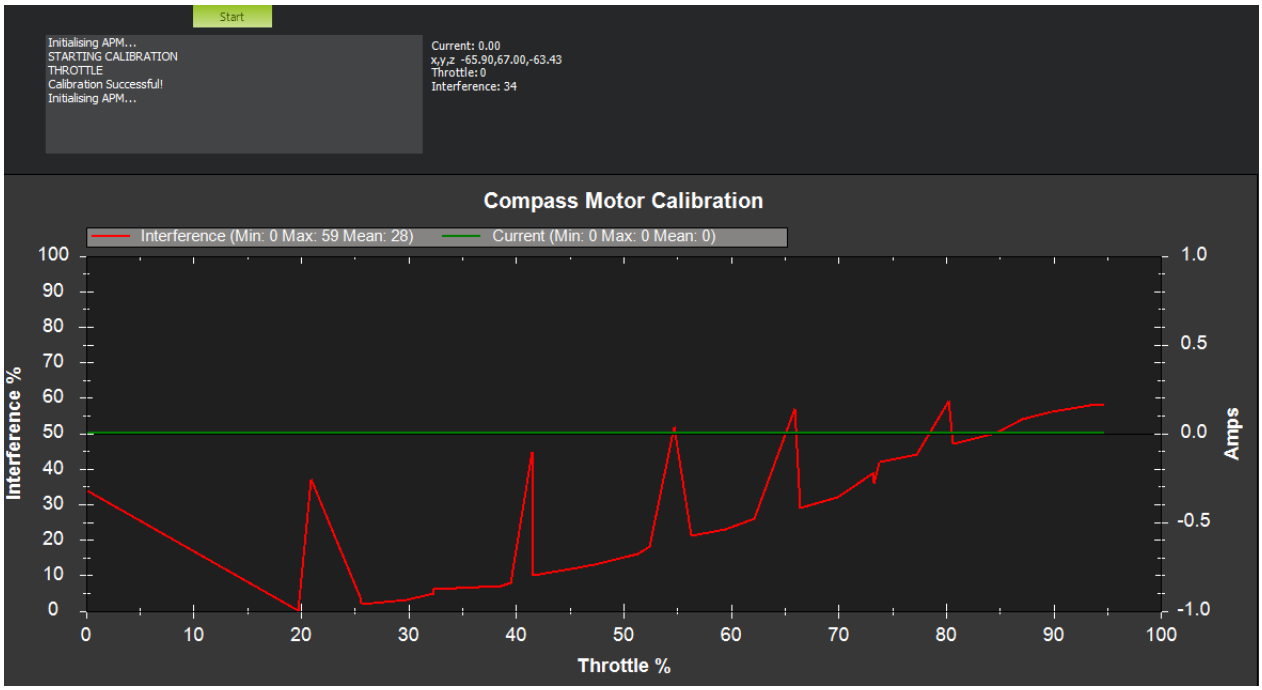


Figure 4.34: The interference when the internal compass is used without grounded conductor

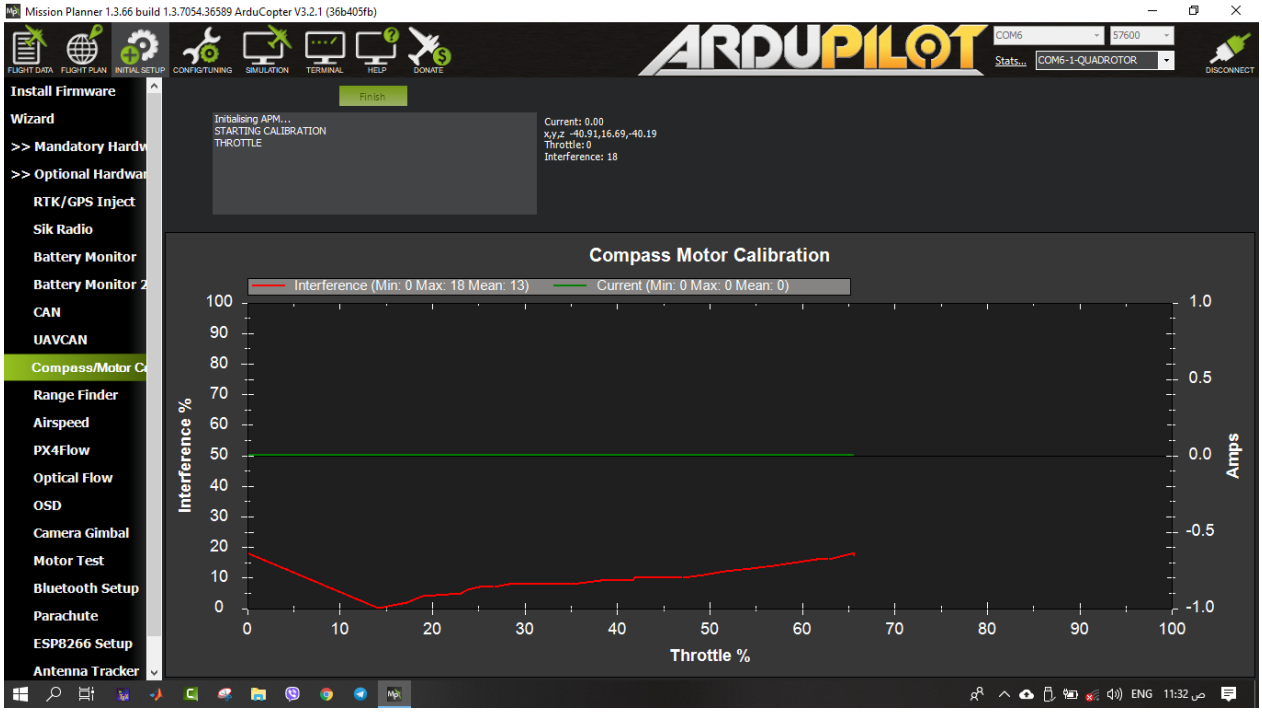


Figure 4.35: The interference when the internal compass is used with grounded

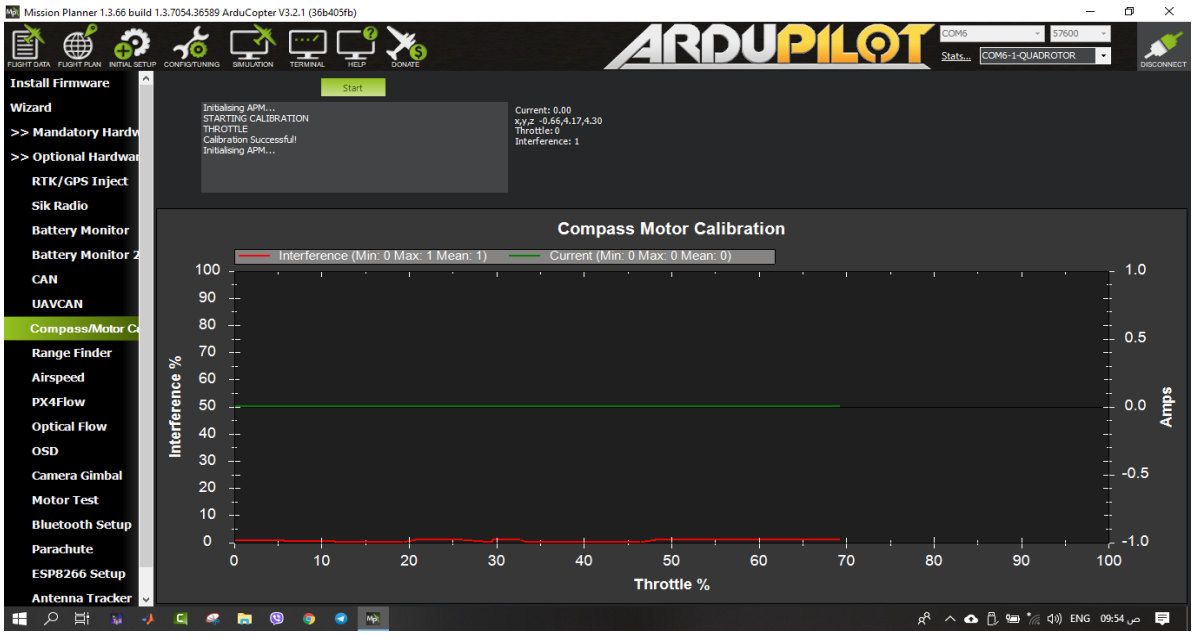


Figure 4.36: The interference when the external compass is used conductor

4.3 The Results of Object Detection

The model is trained based on the collected dataset. An accuracy of about 93% was obtained. It can be seen from the relationship shown in Figure 4.37 that the accuracy increases with the increase of iterations. On the other hand, testing error rate has an inverse relationship with the number of iterations as shown in Figure 4.38. Figure 4.39 shows an example of a case of a person who is detected using the camera installed on the drone.

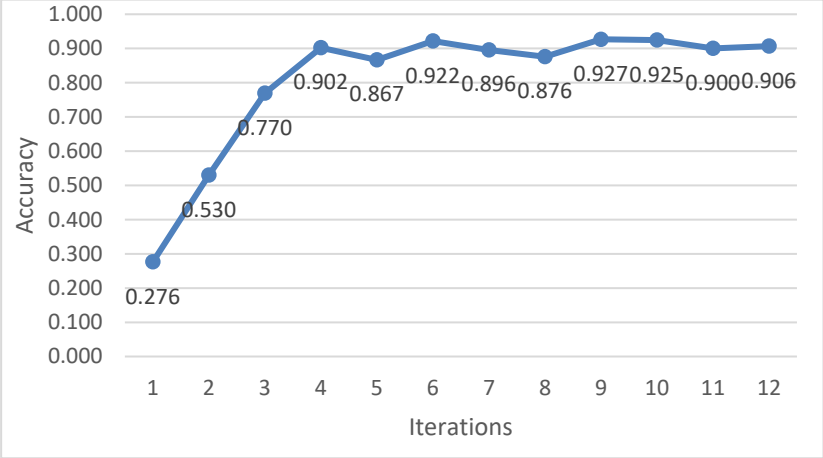


Figure 4.37: The testing Accuracy relation with iteration

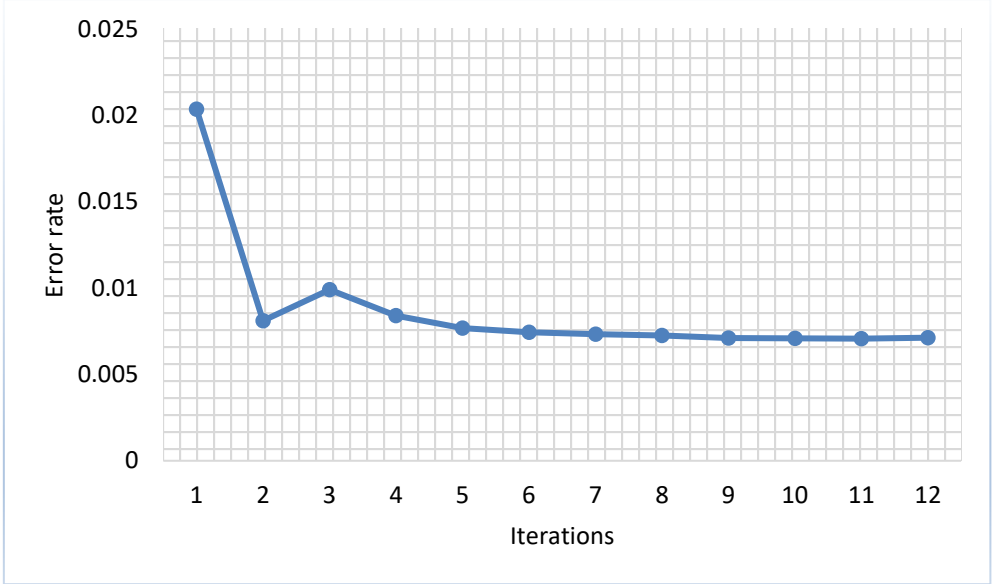


Figure 4.38: The testing Error rate relation with iteration

Figure 4.39: Object detection example

As shown in Figure 4.41, the relationship between the error rate and the number of iterations is an inverse relationship. The lowest error rate was obtained when the number of iterations exceeded 5. While Figure 4.42 shows the process of detecting a person.

4.4 Localization Results

The system responsible for determining the location depends on a number of factors, the most important of which is the accuracy of the camera's orientation, in addition to the optimal use of the gimbal to conduct a search in the area. The gimbal was directed with a certain movement to cover the area with sufficient accuracy and to obtain a comprehensive survey of it. The system managed to move the drone according to certain points and to obtain the direction of the target from several known locations and make the necessary calculations to determine the location as in the figure 4.40.

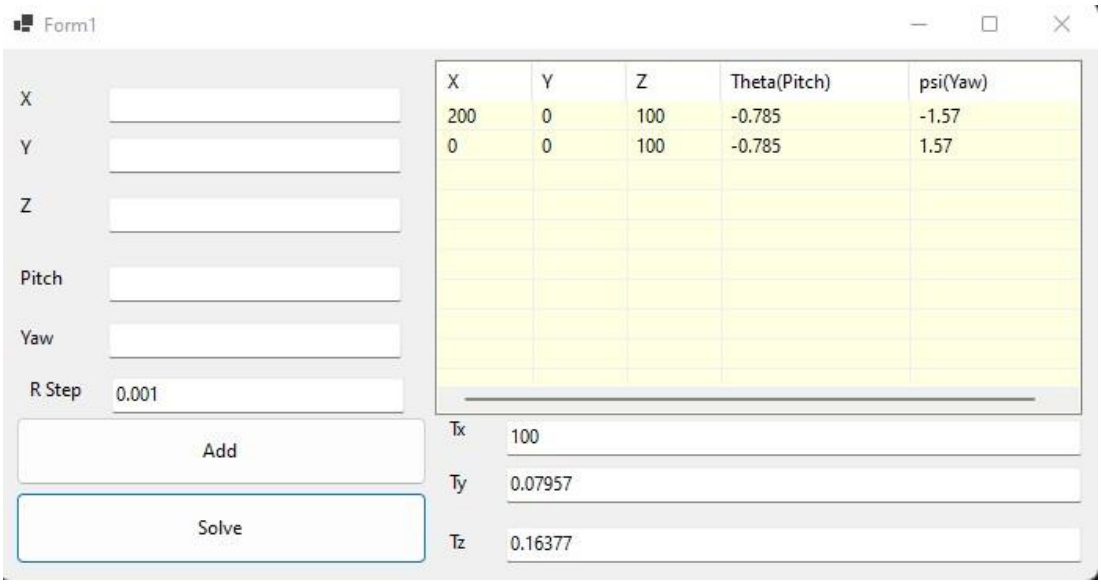


Figure 4.40: Localization case

4.5 Wireless Evaluation

The system was used in Figure 4.41 (2.4 Ghz Outdoor Long-Range Wi-Fi) to communicate with the drone via Wi-Fi. The performance of the

connection was tested and the results appeared as shown in Table 4.1. The used frequency is 2.4 GHZ while the Bandwidth is 40Mbps. The throughput is 1Mbps

Table 4.1 Distance relation with latency

Distance	**latency**
25m	2 ms
50m	5 ms
100m	15 ms
150m	23 ms
200m	35 ms

Figure 4.41: 2.4Ghz Outdoor Long-Range Wi-Fi

CHAPTER FIVE

CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

The main conclusions of the proposed system demonstrated as follows.

- The trained YOLO detector makes a trade-off between speed and precision in object recognition and localization of people.
- The proposed system can distinguish objects in UAV images effectively and consistently in real-time at a detection speed of 60 frames per second.
- The proposed n-visibility tree algorithm can find the shortest path compared to all algorithms that compared with it.
- The use of a spherical building unit to construct bigger obstacles leads to the possibility of obtaining boundaries of obstacles from the perspective of the quad rotor. This process is successfully accomplished by using the equations derived in this thesis.
- Using mathematical equations to find the boundaries of any obstacle instead of using numerical methods leads to an increase in the speed of completion in addition to reducing complexity and increasing accuracy.
- There are three parameters affect on the performance of the n visibility tree algorithm and tangent tree algorithm which are n,r, and R.
- The great circle principle is used to find the optimal curve around the obstacle in both n-visibility tree algorithm and tangent tree algorithm.
- The resulting path of the n-visibility tree algorithm is smooth, regardless of collision angles.

- There is no need to provide any type of communication with the quad rotor when its work in the path tracker mode.
- The quad rotor is implemented and tested in all modes. it can track the path generated by any of the proposed algorithms.
- The calibrations are very essential to get smooth flying. It must be done for ESC, Gyroscope, compass, and Radio transmitter.

5.2 Future Works

- Develop a new modeling algorithm to model all 3D shapes of obstacles based on spheres only. This algorithm can be compatible with the proposed algorithm in this thesis.
- Develop a nonlinear control system and optimize it with the Gray wolf optimization (GWO) algorithm in order to increase the accuracy of low-level path tracking.
- Re-produce the proposed algorithm in the thesis with different basic unit (like using cubic instead of sphere). This will increase accuracy in many environments. And use dynamic environment.
- Develop a cooperation system using multiple drone to detect a target location faster.

References

[1] I. A. Hameed, A. la Cour-Harbo, and O. L. Osen, “Side-to-side 3d coverage path planning approach for agricultural robots to minimize skip/overlap areas between swaths”, Robotics and Autonomous Systems, vol. 76, pp. 36–45, 2016, doi: 10.1016/j.robot.2015.11.009.

[2] V. F. Vidal, L. M. Honório, M. F. Santos, M. F. Silva, A. S. Cerqueira, and E. J. Oliveira, “UAV vision aided positioning system for location and landing,” 2017 18th Int. Carpathian Control Conf. ICCC 2017, pp. 228–233, 2017, doi: 10.1109/CarpathianCC.2017.7970402.

[3] Y. Singh, S. Sharma, D. Hatton, and R. Sutton, “Optimal path planning of unmanned surface vehicles”, Artic. Indian J. Geo-Marine Sci, vol. 47, no. 07, pp. 1325–1334, 2018.

[4] Gall, Juergen, Nima Razavi, and Luc Van Gool. "An introduction to random forests for multi-class object detection." Outdoor and large-scale real-world scene analysis. Springer, Berlin, Heidelberg, 2012. 243-263, doi: 10.1007/978-3-642-34091-8_11.

[5] M. R. Hsieh, Y. L. Lin, and W. H. Hsu, “Drone-Based Object Counting by Spatially Regularized Regional Proposal Network,” Proc. IEEE Int. Conf. Comput. Vis., vol. 2017-Octob, pp. 4165–4173, 2017, doi: 10.1109/ICCV.2017.446.

[6] Y. Lu, Z. Wang, Z. Tang, and T. Javidi, “Target Localization with Drones using Mobile CNNs,” IEEE Int. Conf. Intell. Robot. Syst., pp. 2566–2573, 2018, doi: 10.1109/IROS.2018.8594163.

[7] P. J. Baeck, N. Lewyckyj, B. Beusen, W. Horsten, and K. Pauly, “Drone based near real-time human detection with geographic localization,” Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch., vol. 42, no. 3/W8, pp. 49–53, 2019, doi: 10.5194/isprs-archives-XLII-3-W8-49-2019.

[8] H. Zhang, Z. Lei, G. Wang, and J. N. Hwang, “Eye in the sky: Drone-based object tracking and 3D localization,” MM 2019 - Proc. 27th ACM Int. Conf. Multimed., no. 1, pp. 899–907, 2019, doi: 10.1145/3343031.3350933.

[9] J. Barraquand, B. Langlois, and J.-C. Latombe, “Numerical potential field techniques for robot path planning”, IEEE transactions on systems, vol. 22, no. 2, pp. 224–241, 1992, doi: 10.1109/21.148426.

[10] Y. Wang and G. S. Chirikjian, “A new potential field method for robot path planning”, in Proceedings 2000 ICRA. Millennium Conference. IEEE International

Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), IEEE, vol. 2, 2000, pp. 977– 982.

[11] Y. b. Chen, G.-c. Luo, Y. s. Mei, J.-q. Yu, and X. l. Su, "Uav path planning using artificial potential field method updated by optimal control theory", International Journal of Systems Science, vol. 47, no. 6, pp. 1407–1420, 2016, doi: 10.1080/00207721.2014.929191.

[12] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning", IEEE Transactions on Robotics and Automation, vol. 8, no. 1, pp. 23–32, 1992

[13] M. J. Mohammed, M. T. Rashid, and A. A. Ali, "Design optimal pid controller for quad rotor system", International Journal of Computer Applications, vol. 106, no. 3, pp. 15–20, 2014

[14] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing", in Sixth International Conference on Intelligent Systems Design and Applications, IEEE, vol. 2, 2006, pp. 622–627, doi: 10.1109/ISDA.2006.253908.

[15] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996, DOI: 10.1109/70.508439.

[16] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning", Iowa State University, Ames, Iowa, USA, 1998.

[17] S. Choudhury, S. Scherer, and S. Singh, "Rrt*: Sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter", in 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 3947–3952.

[18] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods", in 49th IEEE conference on decision and control (CDC), IEEE, 2010, pp. 7681–7687, DOI: 10.1109/CDC.2010.5717430.

[19] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of robot 3d path planning algorithms", Journal of Control Science and Engineering, vol. 2016, p. 5, 2016, doi: 10.1155/2016/7426913.

[20] H Tabealhojeh and A Ghanbarzadeh, "Two steps optimization path planning algorithm for robot manipulators using imperialist competitive algorithm", in 2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM), IEEE, 2014, pp. 801– 806, DOI: 10.1109/ICRoM.2014.6991002.

[21] Y. Chen, J. Yu, Y. Mei, Y.Wang, and X. Su, “Modified central force optimization (mcfo) algorithm for 3d uav path planning”, Neurocomputing, vol. 171, pp. 878–888, 2016, doi: 10.1016/j.neucom.2015.07.044.

[22] M. Achtelik, T. Zhang, K. Kuhnlenz, and M. Buss, “Visual tracking and control of a quadcopter using a stereo camera system and inertial sensors”, in 2009 International Conference on Mechatronics and Automation, IEEE, 2009, pp. 2863–2869, DOI: 10.1109/ICMA.2009.5246421.

[23] M. Achtelik, T. Bierling, J.Wang, L. Höcht, and F. Holzapfel, “Adaptive control of a quadcopter in the presence of large/complete parameter uncertainties”, in Infotecha Aerospace 2011, 2011, p. 1485, doi: 10.2514/6.2011-1485.

[24] I. Sa and P. Corke, “Estimation and control for an open-source quadcopter”, in Proceedings of the Australasian Conference on Robotics and Automation 2011, 2011.

[25] Inkyu Sa , Peter Corke, “System identification, estimation and control for a cost effective open-source quadcopter”, in 2012 IEEE International Conference on Robotics and Automation, IEEE, 2012, pp. 2202–2209, DOI: 10.1109/ICRA.2012.6224896.

[26] B. T. M. Leong, S. M. Low, and M. P.-L. Ooi, “Low-cost microcontroller based hover control design of a quadcopter”, Procedia Engineering, vol. 41, pp. 458–464, 2012, doi: 10.1016/j.proeng.2012.07.198.

[27] M. A. Ma Sum, G. Jati, M. K. Arrofi, A.Wibowo, P. Mursanto, and W. Jatmiko, “Autonomous quadcopter swarm robots for object localization and tracking”, in MHS2013, IEEE, 2013, pp. 1–6, DOI: 10.1109/MHS.2013.6710447.

[28] D. Hanafi, M. Qetkeaw, R. Ghazali, M. N. M. Than,W. M. Utomo, and R. Omar, “Simple gui wireless controller of quadcopter”, International Journal of Communications, Network and System Sciences, vol. 6, no. 01, p. 52, 2013, DOI:10.4236/ijcns.2013.61006.

[29] M. Y. Chen, D. H. Edwards, E. L. Boehmer, N. M. Eller, J. T. Slack, C. R. Speck, S. M. Brown, H. G. Williams, S. H. Wilson, C. S. Gillum, et al., “Designing a spatially aware and autonomous quadcopter”, in 2013 IEEE Systems and Information Engineering Design Symposium, IEEE, 2013, pp. 213–218, DOI: 10.1109/SIEDS.2013.6549521.

[30] M. Fatan, B. L. Sefidgari, and A. V. Barenji, “An adaptive neuro pid for controlling the altitude of quadcopter robot”, in 2013 18th International Conference on Methods&Models in Automation&Robotics (MMAR), IEEE, 2013, pp. 662–665, DOI: 10.1109/MMAR.2013.6669989.

[31] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He, “Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface”, Journal of neural engineering, vol. 10, no. 4, p. 046 003, 2013.

[32] T. G. Carreira, “Quadcopter automatic landing on a docking station”, Instituto Superior Tecnico, 2013.

[33] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths”, IEEE transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968, DOI: 10.1109/TSSC.1968.300136.

[34] F. A. Cosío and M. P. Castañeda, “Autonomous robot navigation using adaptive potential fields”, Mathematical and computer modelling, vol. 40, no. 9-10, pp. 1141–1156, 2004, doi: 10.1016/j.mcm.2004.05.001.

[35] I. Châari, A. Koubaa, H. Bennaceur, S. Trigui, and K. Al-Shalfan, “Smartpath: A hybrid aco-ga algorithm for robot path planning”, in 2012 IEEE congress on evolutionary computation, IEEE, 2012, pp. 1– 8, DOI: 10.1109/CEC.2012.6256142.

[36] T. Saito and Y. Kuroda, “Mobile robot localization by gps and sequential appearance-based place recognition”, in Proceedings of the 2013 IEEE/SICE International Symposium on System Integration, IEEE, 2013, pp. 25–30, DOI: 10.1109/SII.2013.6776624.

[37] D. Forouher, M. G. Besselmann, and E. Maehle, “Sensor fusion of depth camera and ultrasound data for obstacle detection and robot navigation”, in 2016 14th international conference on control, automation, robotics and vision (ICARCV), IEEE, 2016, pp. 1–6, DOI: 10.1109/ICARCV.2016.7838832.

[38] Robin R. Murphy, Introduction to AI Robotics, Computer Science, Robotics & Agents, ISBN: 9780262038485, October 2019.

[39] Steven M. LaValle, Planning Algorithms, University of Illinois, UrbanaChampaign, Online ISBN:9780511546877, August 2009.

[40] A. Koubâa, H. Bennaceur, I. Chaari, S. Trigui, A. Ammar, M.-F. Sriti, M. Alajlan, O. Cheikhrouhou, and Y. Javed, Robot Path Planning and Cooperation. Springer, vol. 772. ISBN 978-3-319-77042-0. 2018.

[41] J. H. Reif, “Complexity of the generalized mover’s problem.”, harvard univ cambridge ma aiken computation lab, Tech. Rep., 1985.

[42] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths”, IEEE transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968, DOI: 10.1109/TSSC.1968.300136.

[43] M. Šeda, “Roadmap methods vs. cell decomposition in robot motion planning”, in Proceedings of the 6th WSEAS international conference on signal processing, robotics and automation, World Scientific, Engineering Academy, and Society (WSEAS), 2007, pp. 127–132.

[44] Z. Yan, N. Jouandeau, and A. A. Cherif, “Acs-prm: Adaptive cross sampling based probabilistic roadmap for multi-robot motion planning”, in Intelligent Autonomous Systems 12, Springer, 2013, pp. 843– 851, doi: 10.1007/978-3-642-33926-4_81.

[45] A. N. Nazif, A. Davoodi, and P. Pasquier, “Multi-agent area coverage using a single query roadmap: A swarm intelligence approach”, in Advances in practical multi-agent systems, Springer, 2010, pp. 95–112, doi: 10.1007/978-3-642-16098-1_7.

[46] F. Ducho? n, A. Babinec, M. Kajan, P. Be? no, M. Florek, T. Fico, and L. Jurišica, “Path planning with modified a star algorithm for a mobile robot”, Procedia Engineering, vol. 96, pp. 59–69, 2014, doi: 10.1016/j.proeng.2014.12.098.

[47] Zou, Zhengxia, et al. "Object detection in 20 years: A survey." arXiv preprint arXiv:1905.05055 (2019).

[47] Zhao, Zhong-Qiu, et al. "Object detection with deep learning: A review." IEEE transactions on neural networks and learning systems 30.11 (2019): 3212-3232.

[49] Padilla, Rafael, Sergio L. Netto, and Eduardo AB Da Silva. "A survey on performance metrics for object-detection algorithms." 2020 international conference on systems, signals and image processing (IWSSIP). IEEE, 2020, DOI: 10.1109/IWSSIP48289.2020.9145130.

[50] Tan, Mingxing, Ruoming Pang, and Quoc V. Le. "Efficientdet: Scalable and efficient object detection." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[51] Carion, Nicolas, et al. "End-to-end object detection with transformers." European conference on computer vision. Springer, Cham, 2020.

[52] Xu, Shuyuan, et al. "Computer vision techniques in construction: a critical review." Archives of Computational Methods in Engineering 28.5 (2021): 3383-3397, doi : 10.1007/s11831-020-09504-3.

[53] Esteva, Andre, et al. "Deep learning-enabled medical computer vision." NPJ digital medicine 4.1 (2021): 1-9.

[54] Dong, Chuan-Zhi, and F. Necati Catbas. "A review of computer vision–based structural health monitoring at local and global levels." Structural Health Monitoring 20.2 (2021): 692-743,doi: 10.1177/1475921720935585.

[55] O'Mahony, Niall, et al. "Deep learning vs. traditional computer vision." Science and information conference. Springer, Cham, 2019.

[56] Yin, Dong, et al. "A fourier perspective on model robustness in computer vision." Advances in Neural Information Processing Systems 32 (2019).

[57] Jiang, Peiyuan, et al. "A Review of Yolo algorithm developments." Procedia Computer Science 199 (2022): 1066-1073, doi: 10.1016/j.procs.2022.01.135.

[58] He, Xiaowei, et al. "Small object detection in traffic scenes based on YOLO-MXANet." Sensors 21.21 (2021): 7422, doi: 10.3390/s21217422.

[59] Sahib, FI Abd-AL, H. B. Taher, and R. F. Ghani. "Detection of the autonomous car robot using Yolo." Journal of Physics: Conference Series. Vol. 1879. No. 3. IOP Publishing, 2021.

[60] Zhang, Yu-Ming, et al. "CSL-YOLO: A new lightweight object detection system for edge computing." arXiv preprint arXiv:2107.04829 (2021), doi: 10.48550/arXiv.2107.04829 Focus to learn more.

[61] Lee, Sang-Hyun. "A Study on Fruit Quality Identification Using YOLO V2 Algorithm." International Journal of Advanced Culture Technology 9.1 (2021): 190-195,doi: 10.17703/IJACT.2021.9.1.190.

[62] Huang, Rui, et al. "A rapid recognition method for electronic components based on the improved YOLO-V3 network." Electronics 8.8 (2019): 825, doi: 10.3390/electronics8080825.

[63] Rahman, Ferdousi, et al. "An assistive model for visually impaired people using YOLO and MTCNN." Proceedings of the 3rd International Conference on Cryptography, Security and Privacy. 2019, doi: 10.1145/3309074.3309114.

[64] Dazlee, Nurin Mirza Afiqah Andrie, et al. "Object Detection for Autonomous Vehicles with Sensor-based Technology Using YOLO." International Journal of Intelligent Systems and Applications in Engineering 10.1 (2022): 129-134, doi: 10.18201/ijisae.2022.276.

[65] Li, Min, et al. "Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Comparison of faster R-CNN, YOLO v3 and SSD." Sensors 20.17 (2020): 4938, doi: 10.3390/s20174938.

[66] Liu, Yang, et al. "Real-Time Object Detection for the Running Train Based on the Improved YOLO V4 Neural Network." Journal of Advanced Transportation 2022 (2022), doi: 10.1155/2022/4377953.

[67] Kafadar, Özkan. "RaspMI: Raspberry Pi assisted embedded system for monitoring and recording of seismic ambient noise." IEEE Sensors Journal 21.5 (2020): 6306-6313, DOI: 10.1109/JSEN.2020.3043753.

[68] Shehova, Daniela, Slavi Lyubomirov, and Katya Asparuhova. "Hardware and Software for Learning Analog-toDigital Converters in Engineering Education." 2019 X National Conference with International Participation (ELECTRONICA). IEEE, 2019, 10.1109/ELECTRONICA.2019.8825619.

[69] Kondaveeti, Hari Kishan, et al. "A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations." Computer Science Review 40 (2021): 100364, doi: 10.1016/j.cosrev.2021.100364.

[70] Olmedo, Gonzalo, et al. "Performance analysis of a novel TCP protocol algorithm adapted to wireless networks." Future Internet 12.6 (2020): 101.

[71] Sari, Indah Clara, Muhammad Zarlis, and T. Tulus. "Optimization of data encryption modeling using RSA cryptography algorithm as security e-mail data." Journal of Physics: Conference Series. Vol. 1471. No. 1. IOP Publishing, 2020.

الخلاصة

ازدادت تطبيقات الطائرات المسيرة في الفترة الاخيرة و بدأت تلعب دور اساس في مختلف المجالات. اصبح من الضروري تطوير انظمة ملائمة وقادرة على اداء وضيفة معينة كالكشف و تحديد موقع جسم معين بدقة.

هناك العديد من المشاكل التي سيتم تناولها في هذا العمل وهي: البيئات المعقدة ، والاستقرار مع وجود عوامل الاضطرابات ، وصعوبات التحليق فوق بعض الأهداف ، والكشف في الوقت الحقيقي والاتصال بين الطائرات بدون طيار والمحطة الأساسية ، وتحديد موقع الهدف المعروف.

يحقق النظام المقترح اربع اهداف اساسية لاجل القيام بوظيفته بدقة عالية. الهدف الاول هو ايجاد مسار امن ضمن بيئة متعددة العوائق. الهدف الثاني هو كشف هدف معين. الهدف الثالث هو تحديد موقع هدف معين وتحديد احداثياته. الهدف الرابع هو بناء اتصال امن لغرض ارسال الاحداثيات من الطائرة بدون طيار الى المحطة الارضية في الوقت الحقيقي.

حُقق الهدف الاول من خلال بناء خوارزمية تخطيط مسارات قادرة على ايجاد المسار الافضل ضمن بيئة ثلاثية الابعاد ومتعددة العوائق. تحقق الهدف الثاني من خلال التعلم العميق باستخدام خوارزمية YOLO. تم تحقيق الهدف الثالث من خلال بناء خوارزمية قادرة على تحديد موقع الهدف المحدد من خلال موقع الطائرة ومتجه الكاميرا فقط. حُقق الهدف الرابع من خلال انشاء شبكة تواصل امنه باستخدام ال TCP و خوارزمية تشفير RSA

بُنيت طائرة رباعية المراوح وحُققت جميع الطرق المذكورة انفا عليها. تمكنت الطائرة من التحليق باستقرار. حيث تم تَتبع المسار المحدد وكشف الهدف بدقة تصل الى ٩٣ % في كشف الكائنات و تم تحسين التعقيدات الحسابية لخوارزمية تخطيط المسار المقترحة بحوالي 27٪ من أقرب خوارزمية وهي خوارزمية مجال محتملة. تمت محاكاة خوارزمية الكشف عن الموقع المستهدف وزيادة الدقة مع زيادة عدد نقاط الكشف.احتسبت الاحداثيات حسب الطريقة المذكورة.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنلوجيا المعلومات
قسم شبكات المعلومات

نظام الكشف عن الأجسام في الوقت الفعلي باستخدام الطائرات بدون طيار والتعلم العميق

الرسالة

مقدمة الى مجلس كلية تكنلوجيا المعلومات للدراسات العليا بجامعة بابل في استيفاء جزئي لمتطلبات درجة الماجستير في تكنلوجيا المعلومات - شبكات المعلومات

مصطفى فاهم ابراهيم محمد

باشراف

ا.م.د مهدي عبادي مانع مهدي

1444 هـ

2022 م