UAS Flight Path Planning and Collision Avoidance Based on Markov Decision Process

Tong He

A Thesis

In

The Department

of

Mechanical, Industrial & Aerospace Engineering

Present in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

October 2020

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Tong He**

Entitled: **UAS Flight Path Planning and Collision Avoidance Based on Markov Decision Process**

And submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Mechanical Engineering)

Complies with the regulations of this University and meets the accepted standards with respect to originality and quality.
Signed by the Final Examining Committee:

_____ Chair
Dr. Youmin Zhang

_____ External Examiner
Dr. Rastko Selmic

_____ Examiner
Dr. Youmin Zhang

_____ Thesis Supervisor
Dr. Wenfang Xie and Dr. Iraj Mantegh

Approved by _____
Dr. Ali Dolatabadi Chair of Department of Graduate Program Director

20/10/2020 _____
Date of Defence Dr. Mourad Debbabi Dean, Faculty of Engineering and Computer *Science*

Abstract

UAS Flight Path Planning and Collision Avoidance Based on Markov Decision Process

Tong He, MASc.

Concordia University, 2020

The growing interest and trend for deploying unmanned aircraft systems (UAS) in civil applications require robust traffic management approaches that can safely integrate the unmanned platforms into the airspace. Although there have been significant advances in autonomous navigation, especially in the ground vehicles domain, there are still challenges to address for navigation in a dynamic 3D environment that airspace presents. An integrated approach that facilitates semi-autonomous operations in dynamic environments and also allows for operators to stay in the loop for intervention may provide a workable and practical solution for safe UAS integration in the airspace.

This thesis research proposes a new path planning method for UAS flying in a dynamic 3D environment shared by multiple aerial vehicles posing potential conflict risks. This capability is referred to as de-confliction in drone traffic management. It primarily targets applications such as UAM [1] where multiple flying manned and/or unmanned aircraft may be present. A new multi-staged algorithm is designed that combines AFP method and Harmonic functions with AKF and MDP for dynamic path planning. It starts with the prediction of aircraft traffic density in the area and then generates the UAS flight path in a way to minimize the risk of encounters and potential conflicts. Hardware-in-the-loop simulations of the algorithm in various scenarios are presented, with an RGB-D camera and Pixhawk Autopilot to track the target. Numerical simulations show satisfactory results in various scenarios for path planning that considerably reduces the risk of conflict with other static and dynamic obstacles. A comparison with the potential field is provided that illustrates the robust and fast of the MDP algorithm.

Acknowledgments

This thesis is submitted in partial fulfillment of the requirements for the degree of Master of Applied Science (MASc) at Concordia University (CU) under the financial support of National Research Council Canada's CivUAS program and IAM (Integrated Autonomous Mobility) initiative, and also the NSERC (Grant No. N00892). This thesis research has been conducted under the supervision of Prof. Wenfang Xie and Dr. Iraj Mantegh at the Department of Mechanical, Industrial & Aerospace Engineering, Concordia University, and Canada National Research Council (C-NRC) respectively.

My deepest gratitude and appreciation go to my supervisor, Prof. Wenfang Xie, and Dr. Iraj Mantegh, for offering me this valuable and unique opportunity to pursue my Master's degree under their supervision. Their serious attitude towards research has taught me quite a lot. Without their consistent support and a great help as well as valuable and insightful guidance, my research progress would probably be slowed and unsatisfactory. There are no words that can express my sincere gratitude for their outstanding supervision

I would like to thank Dr. Youmin Zhang and Dr. Rastko Selmic for joining my examination committee and providing brilliant feedback and insightful comments during my comprehensive exam and research proposal exam. I would also like to thank all my colleagues in my research group and collaborators who have helped me a lot during my three years of master's study. It is my great pleasure to do research and share opinions with them and learn from them.

Contents

List of Figures

List of Tables

GLOSSARY

Explanation of Terms

UAV Unmanned aerial vehicle

UTM Unmanned Traffic Management

ATM Air Traffic Management

VTOL Vertical take-off and landing

UAS Unmanned aerial system

NRC National Research Council

RRT Rapidly-Exploring Random Tree

PRM Probabilistic Roadmaps

PF Potential field

ROS Robotics operation system

QC QGround Control

SLAM Simultaneous Localization and Mapping

VO Visual Odometry

BA Bundle Adjustment

IMU Inertial Measurement Unit

ESC Electronic Speed Controllers

GNC Guidance, navigation, and control

GPS Global positioning system

PID Proportional-integral-derivative

RNN Recurrent neural network

RNNs Recurrent neural networks

HPF Harmonic Potential Field

MDP Markov Decision Process

RTAB-map Real-Time Appearance-Based Mapping

TF Transform

CNN Convolutional Neural Networks

KF Kalman Filter

AKF Adaptive Kalman Filter

PCD Point Cloud Data

Chapter 1 Introduction

1.1 Background

1.1.1 Unmanned Aerial Systems

Unmanned Aerial Systems (UAS) is an unmanned aircraft that operates by using radio remote control equipment and on-board control devices. They typically include unmanned VTOL (Vertical Take-Off and Landing) vehicles, fixed-wing aircraft, multi-rotor aircraft, unmanned airship, unmanned parachute aircraft. In a broad sense, it can also include adjacent space vehicles (20-100 km airspace), such as stratospheric airships, high-altitude balloons, and solar-powered drones. From a certain point of view, unmanned aerial vehicles (UAVs) can be regarded as "aerial robots" that can accomplish complex flight and various other load tasks under similar to the way a robot or automated machinery behaves.

According to different platform configurations, UAVs can be classified into three major platforms: fixed-wing UAVs, single rotor helicopters, and multi-rotor UAVs, while other small types of UAVs also include para-wing UAVs, flapping UAVs, and unmanned airships. Fixed-wing UAVs are the main platform of military and most civil UAVs. An unmanned helicopter is the most flexible UAV platform, which can take off and hover in situ vertically. Multi-rotor (multi-axis) UAVs are the preferred platform for consumers and some civil USES. They are flexible between fixed wings and helicopters (taking off and landing requires thrust) but are simple to operate and are known for a low cost.

1) Fixed-wing aircraft. As shown in Figure 1.1(a).
   - Unable to take off or land vertically, similar to an airplane.
   - It is more energy-efficient and takes longer to fly due to its lift wings.
   - Used for long-distance transport or detection missions.

- In addition to batteries, it can be powered by a fuel engine.

2) Single rotor helicopters. As shown in Figure 1.1(b).
   - It can take off and land vertically
   - It can carry a variety of cargo and fly for a long time
   - In addition to batteries, it can be powered by a fuel engine.
3) Multi-rotor UAVs. As shown in Figure 1.1(c).
   - Multi-rotor UAVs features high reliability and low maintenance cost.
   - It can take off and land vertically.
   - Its payload capacity and flight endurance are both compromised.
   - Be more flexible.

a b c

Figure 1.1 Three major platforms: fixed-wing UAVs (a), Single rotor helicopters (b) and multi-rotor UAVs (c)

According to different areas of applications, UAVs can be divided into three categories: military, civilian and consumer, with particular emphasis on the performance requirements of UAVs.

1) Military UAV has higher requirements for sensitivity, flight height, speed and intelligence, and it is equipped with the highest technical level, including reconnaissance, bait, electronic countermeasures, communication relay, target aircraft and unmanned fighter aircraft.
2) Civil UAV's general requirements for speed, distance and ceiling are low. However, for the personnel training, comprehensive cost has higher requirements, so mature industrial

chain is needed to provide cheap components and support services. Currently, the civil UAV's biggest market is that the government provides public services, such as police, fire and weather, about 70% of the total demand, and we think the future UAV 's biggest potential market may be in civil. For example, new market demand may appear in agricultural plant protection, goods speed delivery, air wireless network, data acquisition, etc.

3) Consumer UAV generally uses low-cost multi-rotor platforms for casual purposes such as aerial photography and games.

The payload that a UAS carries is determined by the project requirements and limits of the platform. Digital RGB cameras are the leading sensor type, ranging from inexpensive GoPro cameras to high-end digital SLR cameras with quality lenses. There are also specialized sensors for agriculture, wetlands analysis, and environmental monitoring. These are multi-or hyperspectral sensors that allow the analysis of vegetation or wetlands through the use of remote-sensing software. Many of these sensors are now small enough to be carried on either platform. An autopilot is a system used to control the trajectory of an aircraft, marine craft, or spacecraft without requiring constant manual control by a human operator. Autopilot does not replace human operators. Instead, autopilot assists the operator's control of the vehicle, allowing the operator to focus on broader aspects of operations (for example, monitoring the trajectory, weather, and on-board systems). When present, autopilot is often used in conjunction with an auto throttle, a system for controlling the power delivered by the engines.

Nowadays, UAV has been widely used in meteorological monitoring, land, and resources law enforcement, environmental protection, remote sensing aerial photography, earthquake relief, express delivery, and other fields. With the development of the Internet of Things, the application of UAV to the Internet of Things technology is increasing. In order to better control the flight of UAV, the application of various sensors plays a very important role. Building on its legacy of work in air traffic management for crewed aircraft, NASA is researching and developing a prototype UTM system that would provide airspace integration requirements for enabling safe,

efficient low-altitude operations. While incorporating lessons learned from today's well-established Air Traffic Management (ATM) system, which was a response that grew out of a mid-air collision over the Grand Canyon in the early days of commercial aviation, the UTM system would enable automated safe and efficient low-altitude airspace operations by providing services such as airspace design, corridors, dynamic geofencing, severe weather, and wind avoidance, congestion management, terrain avoidance, route planning and re-routing, separation management, sequencing and spacing, and contingency management..

One of the main attributes of the UTM system is that it would not require human operators to monitor every vehicle continuously, as in the traditional ATM system. The system provides to human managers the data necessary to make strategic decisions related to initiation, continuation, and termination of airspace operations. This approach would ensure that only authenticated UAS could operate in the airspace.

In Canada, Transport Canada has launched an RTM (RPAS Traffic Management) initiative and organized an action team (RTMAT) comprised of the regulator, Nav Canada, NRC, and various industry representatives. The RTMAT architecture and roadmap announced by TC in 2019 identifies key services and performance levels for the full RTM roll out and plan a set of trials for rural and urban flight operations as well as airport security. The trials that will be selected through a proposal and committee review process are aimed to investigate the potentials and gap in the existing technologies to accommodate the services and also support the development of operational procedures. NRC has actively participated and contributed to the RTMAT meetings and discussions, effectively taking lead for assessments and gap analysis of the candidate RTM technologies and also a source of technical advice for upcoming trials. In RTMT 11 meeting in February 2020, we provided a high-level classification of the candidate technologies as potential solutions for some of the RTM Foundational and Primary services. While this preliminary assessment has been helpful to gain a common understanding of the technologies by the RTMAT members, there is a need for more detailed and careful assessment that can establish the base for the trials.

.

1.2 Problem definition

This research will focus on the path planning of multi-rotor commercial UAS in a dynamic, multi-vehicle environment.

Air traffic control and zoning play a big role in commercial UAS, which can pre-process the complex traffic regulations before taking the next action on UAVs. Effective traffic management could do more than less.

In terms of pathing planning, there are currently a lot of classic and improved methods that have been implemented in the UAVs field, each method has its pros and cons according to the specific situation. Applying a feasible algorithm is prone to making the UAVs more efficiently. Therefore, it is vital to choose an appropriate methodology in a specific environment. In this research, the issues on path planning will be discussed under the dynamic, multi-vehicle environment, where there must be plenty of static and dynamic obstacles that cannot be pre-detected. This problem arises with some issues about the real-time obstacle avoidance and object detection and tracking of UAS.

1.3 Motivation and contribution of this thesis

In this research, tremendous efforts have been dedicated to the collision avoidance of UAVs and to improve the performance of autonomous navigation to further ensure the safety and reliability of UAVs. To address the above-mentioned problems, the research objectives are to implement a novel and adaptive UAV path planning algorithm for an urban environment that can handle multiple UAVs and unmolded obstacles in a dynamic environment and real-time collision avoidance using RGB-D sensor.

The main contributions of this dissertation can be summarized as follows:

1) AKF is developed for zoning the flight airspace with consideration of air traffic control. The developed method can predict the position and velocity of other intruders after several seconds before making the planning strategy. It can effectively improve the efficiency of strategic planning.
2) 3D HPF is proposed for global path planning with consideration of pre-mapping. HPF can avoid the local minimum problem that exists in the traditional PF method by using the Laplace equation and harmonic function. 3D HPF makes use of the relation between P and G potential to achieve following the synthetic route. It can satisfy the combination of external control and elimination of the local minimum in the global path planning。
3) MDP is proposed for the real-time obstacle avoidance without pre-information about the unknown intruders. This method makes full use of the reward system to get the max optimal reward action as the principle of the decision. It can effectively process the data from sensors and decide the best action to make collision avoidance.
4) This dissertation proposes an algorithm on the combination of UAV path planning and vision-based real-time obstacle avoidance, and validate the effectiveness of the algorithm through simulation of virtual urban environment based on ROS and PX4 development platform. Most of the simulation results in this thesis are based on ROS / Gazebo and simulation software, and there is no real flight test due to Covid-19.

1.4 Organization of this thesis

This thesis is structured in the following manner:

- Chapter 2 mainly introduces the literature survey about global path planning and local collision avoidance for quadrotors. Moreover, some other preliminary pieces of knowledge, such as SLAM and object detection and tracking are also presented in this chapter.

- Chapter 3 addresses the detailed description of the various hardware components and software used for the Multi-copter used in this thesis, such as ROS and GAZEBO. The movement principle is also illustrated in this chapter.

- Chapter 4 illustrates the methods that are developed for generating the path and making obstacle avoidance by combining the computer vision algorithm. These methods could make the object detection/tracking, airspace zoning, path planning, and avoid obstacles in real-time, which are based on ROS/Gazebo and MATLAB/SIMULINK during the flight test simulation.

- Chapter 5 presents the test and simulation result on each algorithm in different scenarios, and the combination of all algorithms for urban simulation test.

- Chapter 6 presents the conclusions of the conducted research works and summarizes several predominant ideas for the future development of the dissertation's outcomes.

Chapter 2 Literature Survey

2.1 Motion planning

Motion planning [2] is a computational problem to find a sequence of valid configurations that moves the object from the source to the destination. The goal of a general-purpose planning problem is to come up with a sequence of actions that accomplish a given goal. The Motion Planning Problem is more specifically concerned with coming up with plans to move a robot from one location to another. To get it from Point A to Point B. In many cases, this boils down to a kind of geometry problem. Where the goal is to guide the robot through a particular trajectory that avoids all the obstacles that we know about in the environment. This basic approach can be applied to a wide variety of robotic systems including the quadrotors system..

Over the last decade, several studies were reported concerning path planning in unknown environments with dynamic or unexpected obstacles. Various methodologies have been proposed for UAV path planning and collision avoidance [3]. Traditionally, path planning has been conducted at the two levels of (c.f. [4]: i) global and strategic collision avoidance, and ii) local or tactical collision avoidance. The former utilizes a priori known information and the map of the environment to produce a connected path from the start to the goal location in the environment. The latter is performed by using sensor data to adapt to the changes to the environment representation, e.g., due to new obstacles appearing or un-modeled aircraft presence in the area [5]. Common approaches in the literature for global path planning include search-based algorithms, e.g., Dijkstra or A* algorithm [6]-[7]. More recently, sampling-based algorithms such as Rapidly-exploring Random Trees (RRT) [8]-[9] and Probabilistic Roadmap Method (PRM) [10] have been widely used. RRT algorithm is a sampling-based path planning algorithm, which is commonly used in mobile robot path planning. It is suitable for solving path planning problems under high-dimensional space and complex constraints. The basic ideas include searching forward to the target point by a step in the way of generating random points, effectively avoiding obstacles, avoiding the path falling into a local minimum, and converging to the target point quickly. RRT is a fast algorithm but cannot guarantee asymptotic optimality [11]. Lee et al. Proposed the RRT * algorithm based on the spline curve (SRRT *) to solve the UAV path

planning problem with nonholonomic constraints in the 3D environment. The algorithm expands the random tree through the B-spline curve. It can realize dynamic programming and also can check the geometric collision of the robot in the tree expansion stage. The SRRT * algorithm can generate smooth and optimal cost paths for UAVs. The overall RRT does not need to model the system, and there is no need to geometrically divide the search area. The coverage in the search space is high, and the search range is wide. Also, and the unknown area can be rediscovered. But at the same time, there is also a problem that the computational load of the algorithm is too large to find the path in narrow space. To improve the efficiency of the RRT algorithm, various methods have been proposed including potential function planner [12], density avoided sampling [13], and variations of those [14]-[18]. Rapidly Exploring Random Tree Star (RRT*) [19] is one of the recent sampling-based algorithms proposed as an extension to RRT, which iteratively generates and optimizes the path as the number of sampling increases. The algorithms mentioned above have their advantages and disadvantages according to the different situations. In this thesis, the algorithm is capable of safely navigating the UAS platform in a dynamic environment while other UAVs are also sharing the airspace.

PRM is a graph-based search method [10]. It converts a continuous space into a discrete space and then uses a search algorithm such as A * to find a path on the route map to improve search efficiency. This method can find a solution with relatively few random sampling points. For most problems, relatively few samples are enough to cover most of the feasible space, and the probability of finding a path is 1 (As the number of samples increases, P (Find a path) Index tends to 1). When there are too few sampling points or the distribution is unreasonable, the PRM algorithm is incomplete, but with the increase of the number of points used, it can also reach completeness. Therefore, PRM is probabilistically complete and not optimal. PRM method can effectively solve the path planning problem in high-dimensional space and complex constraints, but this is based on enough sample points and reasonable step size, and as the dimension increases, the amount of calculation also increases.

A* is a path on the graph plane with multiple nodes to find the lowest passing cost [6]. It is also a heuristic search algorithm. Heuristic search is a search in the state space for each search position Evaluate to get the best position, and then search from this position to the target. This can omit a

large number of unnecessary search paths and improve efficiency. The A* could find an Optimal path, but it takes a long time to do the calculation of the cost function. Also, the number of nodes will take a great effect on the speed of the process. In some specific situations, 3D path planning is required, in terms of exponential increase when adding one more dimension, A* is more suitable for the 2D environment. Potential Field is to create the gravitational field and repulsive field [24], in which the target point produces gravitation on the object and guides the object toward its movement (this is somewhat similar to the heuristic function h in the A* algorithm). Obstacles produce a repulsive force on objects to prevent them from colliding. The resultant force of an object at each point on the path is equal to the sum of all repulsive and gravitational forces at this point. The computational load of this Potential Field is small, to improve the operational efficiency of trajectory planning. However, there is a regional minimum value in the implementation process of this method, which makes the UAV unable to judge the next trajectory. At the same time, when it is close to the obstacle, the repulsion force is very large, so it is impossible to judge the magnitude and direction of the attraction force.

2.2 Real-time obstacle avoidance

2.2.1 Artificial Potential Filed

The local path planning methods employ sensor readings to react to the environment changes, and they can also be effective for GPS-denied navigation, where the global reference to the positioning system may not be available. The most common sensors used for obstacle avoidance and detection are Electro-Optical (E/O), Laser, Li-DAR, and in some cases radar (e.g., Ecodyne radar [20] and TrueView radar [21]). Optical sensors such as 3D depth camera (or RGB-D sensors) have been recently used in several works for 3D environment mapping and localization. They are capable of producing RGB-D (Red Green Blue-Depth) image that is augmented by the depth data at each pixel and hence can be used for localization. Jia Hu et al. [22] use a RealSense^{TM} camera and Blob analysis to binarize and segment the image in order to obtain the foreground and background and then detect the connected area to get the Blob block, which performs well in simulation. In [23], the researchers also use the image-based visual servoing by comparing the

real-time obstacle definition windows to make avoidance. Stereo camera or RGB-D camera is used for getting the image features.

Khatib formulated the concept of Artificial Potential Field (APF) [24] for local obstacle avoidance in 1985. It combines attractive and repulsive potential fields that represent the goal and obstacles positions, respectively. APF methods are applied to local path planning due to the local minima problem which can render global path planning incomplete in a general environment [25], [26]. However, some variants of APF can be used for global planning. HPF, for example, as an alternative to model the force fields in AFP have been used in several research works, including the pioneering work of C. I. Connolly [27] in 1993, and more recently in [28]-[32] and [33]. As another example of APF variant, the advanced fuzzy potential field method (AFPFM) is proposed by Park et al. [34]. The authors combined Takagi–Sugen (TS) Fuzzy inference [35] with AFP to avoid the local minima problem.

HPF method is a novel method based on harmonic functions, to overcomes the limitations of potential field methods. The most important trait of HPF method is that they are free from local minima. HPF method uses harmonic functions and boundary value conditions to solve the local minima problem. To build an artificial potential, we use harmonic function, which should satisfy Laplace equation. It should not have local extrema in a space free from singularities, and it should have second order derivatives. The solution of Laplace equation is also known as the mobile robot velocity potential. A harmonic function should also satisfy principle of superposition and principle of maxima and minima. These principles indicate that the harmonic function has its extremes only on the boundary, so it does not have local maxima/minima inside the boundary. Hence, it is convenient for us to define boundary conditions for all obstacles and goal. In this research Newman boundary conditions have been used. The Newman boundary condition states that the boundary of all obstacles will be assigned with the maximum value in the region and the boundary of goal position has the minimum value in the region. By defining the boundary conditions in this format, the potential field is harmonic field with only global minimum represented by goal position.

2.2.2 Markov Decision Process

More recently, many researchers have considered the MDP [61] for solving the collision avoidance problem. For example, Temizier et al. [26] have formulated the problem in an MDP framework with the sensors that localization formulation for the intruder aircraft and provided a collision-avoidance policy that balances the original flight path deviations with the probability of a collision given the positional uncertainty of the intruder. MDP is a discrete-time stochastic control process. It provides a mathematical framework for modeling decision-making in situations where the outcomes are partly random and partly under the control of a decision-maker. MDPs are useful for studying optimization problems solved via dynamic programming and reinforcement learning. At each time step, the process is in some state, and the decision-maker may choose any action that is available in the state. The process responds at the next time step by randomly moving into a new state and giving the decision-maker a corresponding reward. The probability that the process moves into its new state is influenced by the chosen action. Specifically, it is given by the state transition function. Thus, the next state depends on the current state and the decision maker's action. In other words, the state transitions of an MDP satisfy the Markov property. MDP are an extension of Markov chains. The difference is the addition of actions (allowing choice) and rewards (giving motivation). Conversely, if only one action exists for each state (e.g. "wait") and all rewards are the same (e.g. "zero"), a MDP reduces to a Markov chain.

In this work we adapt a MDP formulation for dynamic path planning problem in urban environments.

2.3 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is an important area that transcends various fields, particularly for real-time obstacle avoidance and path planning. The concept of localization deals with the question of where the object is located, while the concept of mapping is to visualize the environment surrounding the object. There are some difficulties in finding ways or methods to solve [36], including the feasibility of locating an object in the unknown surroundings and the

capability of mapping the environment as well as simultaneously getting the location, which has been tried to solve [35] by many researchers.

Andrew Davison's Mono-SLAM [36] applies EKF-SLAM based on the Laser Ranger-Finder into Single camera SLAM in the traditional robotics field. The algorithm is more complete, and the key improvement lies in increasing the computing speed through replacing invariant Feature Matching. PTAM [38,39] separates tracking and mapping into two separate threads, which not only does not affect the real-time experience of tracking, but also makes it easy to use BA in the mapping thread to improve accuracy. The ORB-SLAM [40] algorithm basically follows the framework of PTAM and adds in all the modules that have been validated in recent years to make an all-purpose system with high stability and accuracy that can be used in various scenarios such as indoor/outdoor and small-scale/large-scale. When the traditional EKF-based SLAM [41] does IMU fusion, it is similar to the Mono-SLAM mentioned above. Generally speaking, the state vector at each moment will save the current pose, velocity, and 3D Map points coordinates, etc. (IMU bias is generally added in IMU fusion). And IMU is used to predict step, and then the observation error of 3D Map points is used in the image frame to do update step. The motivation of MSCKF [42] is that every update step of EKF is observed in a single-frame frame based on 3D Map points. It would be good if the observation effect could be based on its observation in multiple frames (a little similar to the idea of local Bundle adjustment). Therefore, the performance of MSCKF is improved by the steps given as follows: the predict step is similar to EKF, but the update step is delayed until a 3D Map point is observed in multiple frames for calculation. Before the update, each frame is received, and the state vector is simply expanded and added to the pose estimate of the current frame. The idea is similar to the local bundle adjustment (or Sliding Window Smoothing), but in update Step, it was equivalent to optimizing the pose and 3D map point based on multiple observations. This thesis will not focus on using the SLAM part as an auxiliary part in an unknown environment.

In order to meet some real-time limitations, a closed-loop detection only uses a limited number of labeled points, and can access the labeled points of the entire map when needed. When the number of positioning points in the map makes the time to find a positioning match exceed a certain threshold, RTAB-MAP transfers the positioning points that are unlikely to form a closed loop in WM (Working Memory) to LTM (Long-Term Memory). In this way, these transferred position points will not participate in the calculation of the next closed-loop detection. When a

closed loop is detected, its lead positioning point can be retrieved from the LTM and put into the WM for future closed loop detection.

Since the positioning points in the LTM do not participate in closed-loop detection, it is very important to choose which fixed points in the WM are transferred to the LTM. The idea of RTAB-map is shown as follows. It is assumed that the more frequently visited anchor points are easier to form a closed loop than other anchor points. The number of consecutive visits of such an anchor point can be used to measure its weight for easily forming a closed loop. When it is necessary to transfer the anchor point from the WM to the LTM, the anchor point with the lowest weight is preferentially selected. If there are more anchor points with the lowest weight, the one with the longest storage time is preferred. STM (Short-Term Memory) is used to observe the similarity of consecutive images in time, and update the weight of the anchor point accordingly. WM is used to detect the closed-loop hypothesis of the positioning point in space. The closed-loop detection of RTAB-MAP does not use the positioning points in the STM, because in most cases, the last obtained positioning point is mostly similar to its nearest positioning point. The storage size T of the STM depends on the speed of the robot and the frequency of acquisition of the positioning points. When the number of positioning points reaches t, the positioning points with the longest storage time in the STM are moved to the WM.

RTAB-Map uses discrete Bayesian filters to estimate the probability of forming a closed loop, and compares the new anchor point with the anchor point stored in WM. When it is found that there is a high probability of forming a closed loop between the new and old anchor points, a closed loop is detected, and the new and old anchor points are linked together. There are two key steps, one is "retrieving". For the anchor point with the highest probability of forming a closed loop, it takes leading anchor points that are not in the WM, and then takes it out of the LTM and put it back into the WM. The second step is called "transfer". When the processing time of the closed-loop detection exceeds the threshold, the positioning point with the lowest weight and the longest storage time will be transferred to the LTM. And the number of transferred positioning points depends on the number of anchor points stored by WM in the loop.

RTAB aims to provide a time - and scale-independent, appearance-based positioning and composition solution for online closed-loop detection in large environments. In order to meet some real-time constraints, closed-loop detection can only use a limited number of registration points, and can access the registration points of the whole map when needed. When the number

of registration points in the local map matches a certain threshold value, RTAB will transfer some registration points which have low probability to be a closed loop from WM to LTM.

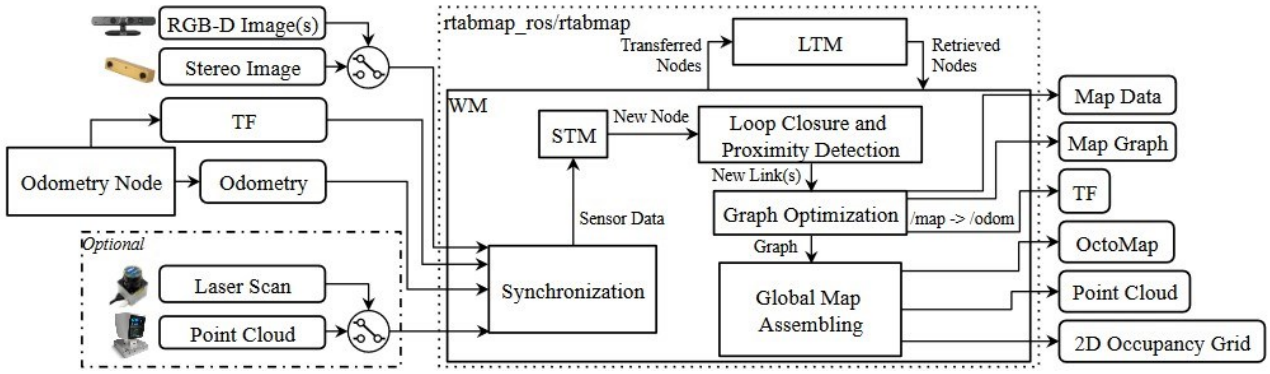

Figure 2.1 RTAB-Map ROS sensor placement.

The required inputs are: TF, which is used to define the position of the sensor relative to the robot base; an odometer from any source (can be 3DoF or 6DoF); one of the camera inputs (one or more RGB-D images, or binocular Stereo image) with corresponding calibration message. The optional inputs are: 2D laser radar scan, or 3D laser point cloud. Then, all messages from these inputs are synchronized and passed to the graph-SLAM algorithm.

The UAV's trajectory needs to be estimated and the correct map needs to be built. There are many ways to express the maps, such as feature point maps, grid maps, topological maps, etc. The map format we use is mainly a point cloud map. In the program, we splice the point cloud according to the optimized posture, and finally form a map. This approach is very simple, but has some obvious flaws:

- The map format is not compact.

  Point cloud maps are usually very large, so a PCD file will also be very large. A 640×480 image will produce 300,000 space points, requiring a lot of storage space. Even after some filtering, the PCD file is very large. The point cloud map provides a lot of unnecessary details. We are not particularly concerned about these folds and shadows on the carpet. Putting them on the map is a waste of space.
- The way to deal with overlap is not good enough.

  When constructing a point cloud, we directly put together according to the estimated pose. When there is an error in the pose, it will cause obvious overlap of the map. For example, a

computer screen has become two, and the original boundary has become a polygon. The treatment of overlapping areas should be better.

- Difficult to navigate

  The point cloud map cannot show whether the area is accessible or not, which cannot give a concrete information for the navigation.

Octo-map [43] is designed for this purpose, it can compress and update the map gracefully, and the resolution is adjustable. It stores maps in the form of an octree, which saves a lot of space compared to point clouds. The map created by Octo-map looks like this: (from left to right is a different resolution)

Figure 2.2 By limiting the depth of a query, multiple resolutions of the same map can be obtained at any time. Occupied voxels are displayed in resolutions 0.08m, 0.64m, and 1.28m.

Due to the octree, its ground image is composed of many small squares (much like mine-craft). When the resolution is high, the square is small; when the resolution is low, the square is large. Each square represents the probability that the square is occupied. Therefore, one can query a certain block or point "whether it can pass" to achieve different levels of navigation. In short, lower resolutions are used when the environment space is larger, while higher resolutions can be used for more elaborate navigation.

An octree, that is, a tree with eight child nodes:

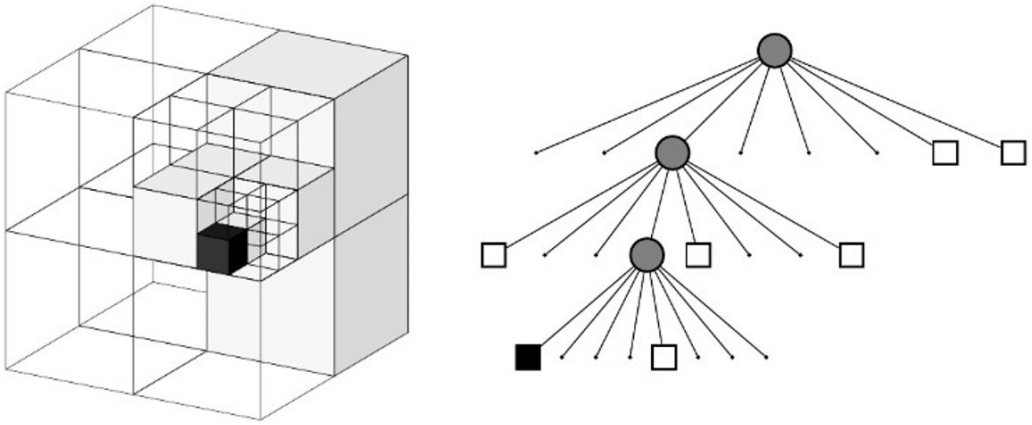

Figure 2.3 The model of resolution square (left) and the structure of the Octree (right)

The actual data structure is that a tree root continuously expands downwards and is divided into eight branches each time until the leaves. The leaf node represents the highest resolution. For example, if the resolution is set to 0.01m, then each leaf is a small square of 1cm square. Each small square has a number describing whether it is occupied. In the simplest case, it can be represented by two numbers from 0 to 1 (too simple to use). Usually a floating-point number between 0 and 1 is used to indicate its probability of being occupied. 0.5 means undetermined, the greater the probability of being occupied, and vice versa. Since it is an octree, all eight children of a node have a certain probability of being occupied or not occupied.

When using the benefits of the tree structure: when the child nodes of a node are "occupied" or "not occupied" or "undetermined", one can cut it off. In other words, if there is no need to further describe the finer structure (child node), one only need a thick square (parent node) information. This can save a lot of storage space. The occupancy probability of the father node in the octree can be calculated according to the value of the child node. The simple way is to take the average or maximum. If the octree is rendered according to the probability of occupancy, the uncertain squares are rendered transparent, and the occupied rendering is determined to be opaque,

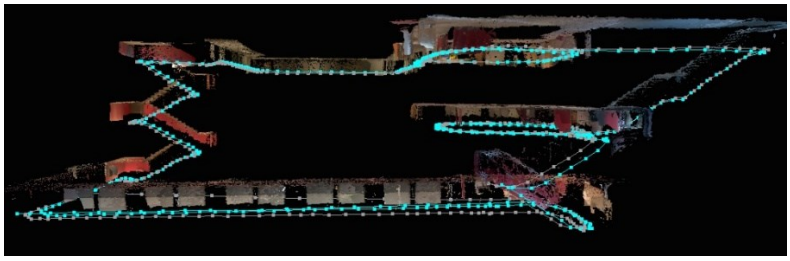

Figure 2.4 The Odometry and the virtual mapping of Octo-map

2.4 Object detection based on Convolutional Neural Networks (CNN)

Convolutional neural network is an efficient recognition method developed in recent years. In the 1960s, Hubel and Wiesel found that the unique network structure of the neurons used for local sensitivity and directional selection in the cat's cerebral cortex can effectively reduce the complexity of the feedback Neural Network, and then put forward the Convolutional Neural Networks (CNN). At present, CNN has become one of the research hotspots in many scientific fields, especially in the field of pattern classification. Since this network avoids the complicated pre-processing of images and it can directly input the original images, it has been widely used. The new recognizer proposed by K. Ukushima [44] in 1980 is the first implementation network of the convolutional neural network. Subsequently, more researchers improved the network. Among them, the representative research result is "improved cognitive machine" proposed by Alexander and Taylor, which combines the advantages of various improvement methods and avoids time-consuming error back propagation.

Generally speaking, the basic structure of CNN consists of two layers. The first is the feature extraction layer, in which the input of each neuron is connected to the local receiving domain of the previous layer and the local features are extracted. Once the local feature is extracted, the location relationship between it and other features is also determined. The second is the feature mapping layer. Each computing layer of the network is composed of multiple feature mappings, and each feature mapping is a plane on which the weights of all neurons are equal. The sigmoid function with small influence kernel is used as the activation function of the convolutional network in the feature map structure, which makes the feature map have displacement invariance. In addition, the number of network free parameters is reduced because the neurons on a mapping surface share weight. Each convolutional layer in a convolutional neural network is followed by a computing layer for local average and quadratic extraction. This unique two-time feature extraction structure reduces the feature resolution. CNN is mainly used to identify two-dimensional graphics with displacement, scaling and other forms of distortion invariance. Since the feature detection layer of CNN learns through training data, it avoids feature extraction when using CNN and learns implicitly from training data. Moreover, because the weights of neurons on the same feature mapping surface are the same, the network can learn in parallel, which is also a big advantage of convolutional network over the network with neurons connected to each other.

Convolution weights of neural network with its local Shared special structure in terms of speech recognition and image processing has its unique superiority, its layout is closer to real biological neural networks, a weight sharing reduces the complexity of the network, especially the multidimensional network input vector image can directly input this feature to avoid the data in the process of feature extraction and classification the complexity of the reconstruction.

Real-time object detection and tracking are important and challenging tasks in many computer vision applications such as video surveillance and UAV navigation. Object detection involves detecting the object in a sequence of videos and is the process of locating an object or multiple objects using either a static or dynamic camera. In [45], Qiang Ling et.al, developed a feedback-based object detection algorithm. It adopts a dual-layer updating model to update the background and segment the foreground with an adaptive threshold method and object tracking is treated as an object matching algorithm. A piece of structured labeling information in the partial least square analysis algorithm for simultaneous object tracking and segmentation was proposed in [46]. This algorithm allows for novel structured labeling constraints to be placed directly on the tracked object to provide useful contour constraint to alleviate the drawback of the online-learning-based tracking method is their sensitivity to drift, i.e, they gradually adapt to non-targets. In [47], Jianxin Wu et al proposed a real-time and accurate object detection framework called C 4 which detects the object based on their contour information using a cascade classifier and the CENTRIST visual descriptor. A major contribution of their work is to fast object detection. It involves no image preprocessing or feature vector normalization, and only requires O (1) steps to test an image patch.

At present, artificial intelligence is the core and focus of next-generation information technology, and computer vision plays an irreplaceable role. Especially in related fields such as obstacle avoidance and object tracking. Before 2014, target detection usually used a more traditional method, to find a way to generate some candidate boxes, then to extract the features of each box, and finally to confirm whether this box is the target object through a classifier. There are many ways to generate candidate frames. For example, different sizes of preselection boxes can be used to slide in the picture, or like the Selective Search [48] algorithm, some candidate boxes can be generated based on the texture of the picture itself. However, since 2014, with the development of deep learning related technologies, new models have emerged, which can achieve end-to-end training and detection networks, and the effect has been significantly improved

compared with traditional methods. The development of target detection is shown in the figure below:

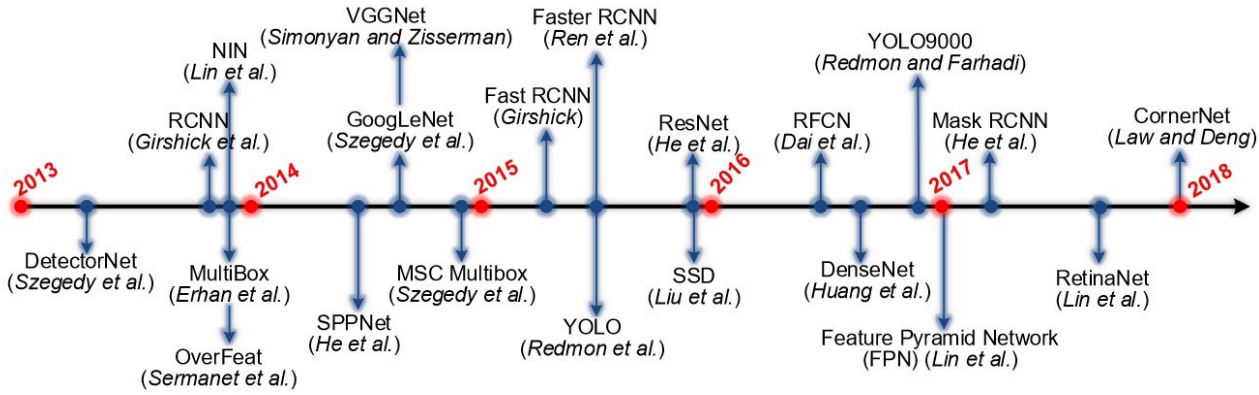

Figure 2.5 Milestones in generic object detection.

In 2015, Redmon J et al. proposed the YOLO network, which is characterized by combining the candidate box generation and classification regression into one step. The feature map is divided into 7x7 cells during prediction, and each cell is predicted, which greatly reduces the calculation complexity. It can accelerate the speed of target detection, frame rate up to 45 fps! The most common evaluation metric that is used in object recognition tasks is 'mAP', which stands for 'mean average precision'. It is a number from 0 to 100 and higher values are typically better, but its value is different from the accuracy metric in classification. After a lapse of one year, Redmon J once again proposed YOLOv2. Compared with the previous generation, the mAP on the VOC2007 test set is increased from 67.4% to 78.6%. However, because a cell is only responsible for predicting an object, facing the goal of overlap. The recognition result is not good enough. Finally, in April 2018, the author released the third version of YOLOv3.The mAP-50 on the COCO dataset was increased from 44.0% of YOLOv2 to 57.9%. Compared with RetinaNet [49] with 61.1% mAP, RetinaNet has an input size of 500. In the case of ×500, the detection speed is about 98 ms/frame, while YOLOv3 has a detection speed of 29 ms/frame when the input size is 416×416.

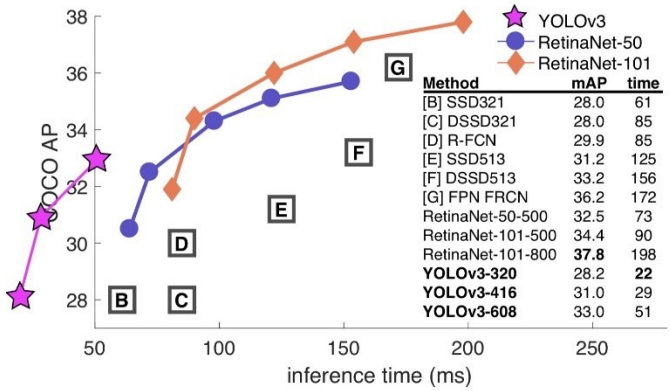

Figure 2.6 The mAP and cost time of different training Methods

Darknet-53 is comparable to the most advanced classifiers in accuracy, and it has fewer floating-point operations and the fastest calculation speed. Compared with ReseNet-101, the speed of Darknet-53 network is 1.5 of times that of the former; although ReseNet-152 and its performance are similar, but it takes more than 2 times of time. In this thesis, the UAV model has been trained based on Yolov3 Darknet53 weight. 100 databases have been trained for the detection.

2.5 Summary

In this chapter, a literature survey on global path planning and real-time collision avoidance combine with sensor fusion has been carried out. In the path planning section, plenty of global and local collision avoidance are been demonstrated, which has their cons and pros. Such as HPF can combine with the control, but the local minimum is the main issue in this method. Also, PRM and RRT have been mentioned, which have best optima and fast speed respectively, but they are all in sample-based, which means the branch and point are created randomly so that cannot combine with control. The computer vision part includes object detection tracking and SLAM, these algorithms could help collision avoidance more effectively.

Chapter 3 Integrate System Description

This chapter presents an overview of the different parts of the drone system used for the simulations of this research. This includes the multi-copter, the software and the necessary communication protocols.

3.1 The Multi-copter

3.1.1 Basic structure

For this thesis, a quadrotor simulation model was chosen. The quadrotor is a 3D Iris Quadrotor model, which has many virtual parameters, fixed-pith propellers, 850kv brush-less motors, SimonK Electronic Speed Controllers (ESC), aluminum arms, a power distribution board and GPS. The quadrotor is powered by a Hyperion 3s 4000mAh 25C battery. The quadrotor is shown in Figure 3.1.

Herein, the quadrotor is chosen to fly in Quadrotor X- configuration. The other alternative is +- configuration. These configurations are shown in Figure 3.2, where the blue and green arrows indicate rotor configuration.

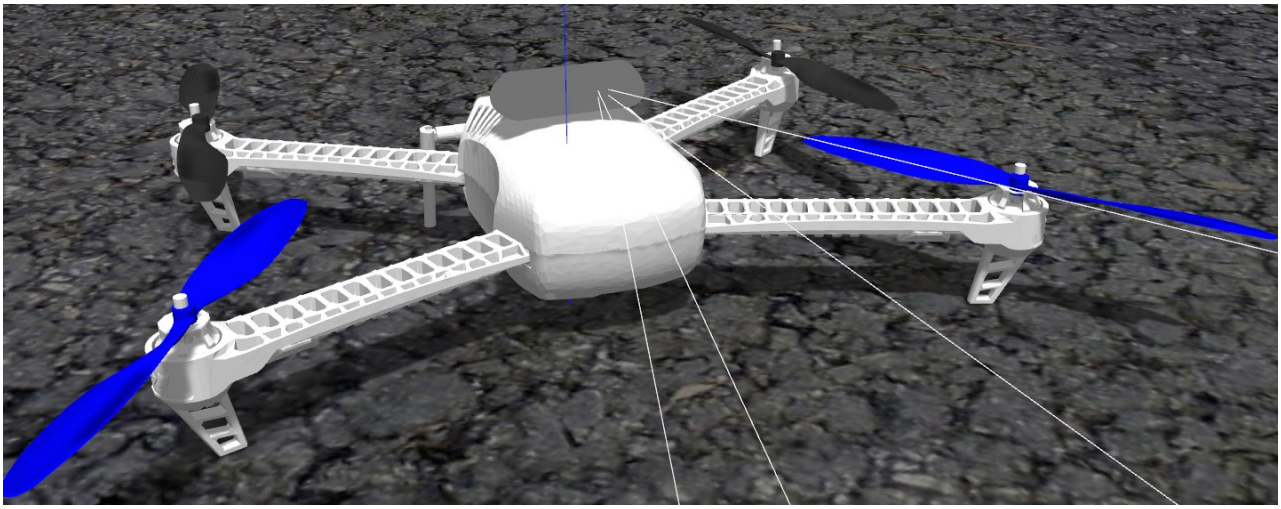

Figure 3.1 Quadrotor Firmware based on PX4 Platform

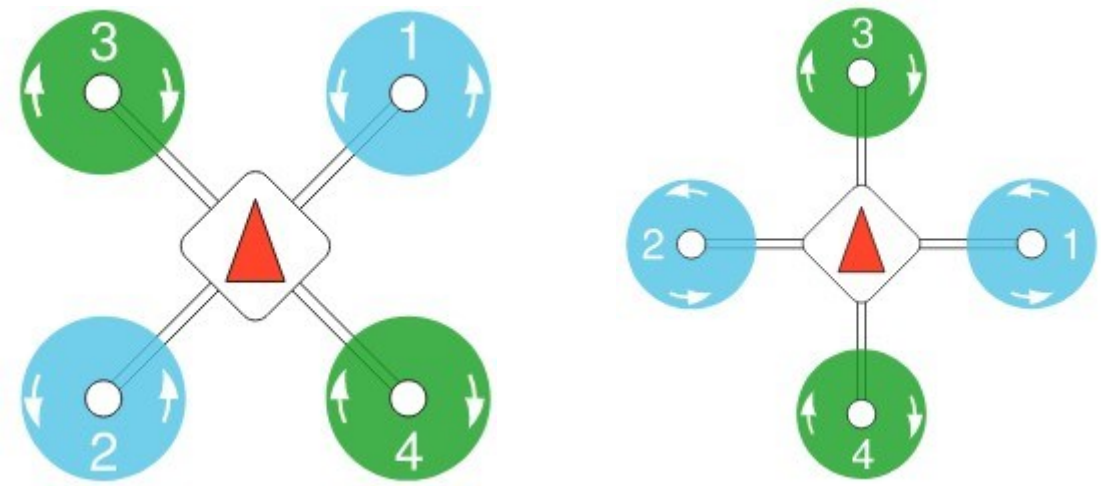

Figure 3.2 The two possible Multi-copter configurations, X-configuration to the left and +-configuration to the right. Green indicate clockwise direction of the rotors, while blue indicate counter-clockwise direction.

3.1.2 Movement principle

The X-shape quadrotor mathematical modeling of dynamical systems can be presented as:

$$\begin{cases} \ddot{x}=(\cos\phi\sin\theta\cos\varphi+\sin\phi\sin\varphi)\frac{1}{m}U_1-\frac{K_1}{m}\dot{x} \\ \ddot{y}=(\cos\phi\sin\theta\sin\varphi-\sin\phi\cos\varphi)\frac{1}{m}U_1-\frac{K_2}{m}\dot{y} \\ \ddot{z}=(\cos\phi\cos\theta)\frac{1}{m}U_1-g-\frac{K_3}{m}\dot{z} \\ \ddot{\phi}=\dot{\theta}\dot{\varphi}\left(\frac{I_y-I_z}{I_x}\right)-\frac{J_r}{I_x}\dot{\theta}\Omega+\frac{l}{I_x}U_2-\frac{lK_4}{m}\dot{\phi} \\ \ddot{\theta}=\dot{\phi}\dot{\varphi}\left(\frac{I_z-I_x}{I_y}\right)-\frac{J_r}{I_y}\dot{\phi}\Omega+\frac{l}{I_y}U_3-\frac{lK_5}{m}\dot{\theta} \\ \ddot{\varphi}=\dot{\phi}\dot{\theta}\left(\frac{I_x-I_y}{I_z}\right)+\frac{l}{I_z}U_4-\frac{lK_6}{m}\dot{\varphi} \end{cases} \quad (4\text{-}1)$$

Where ϕ, θ, φ are the rotation angle (counterclockwise) of the fuselage around the Y-axis, X-axis and Z-axis, I_x, I_y, I_z are the moment of inertia of the fuselage in three directions, J_r is the moment of inertia, $K_1, \dots K_6$ are the air resistance coefficient, l is the arm length from the motor to the center of mass, M is the mass of the body and g is the gravitational acceleration.

Then the X-shape control input can be defined as:

$$\begin{cases} U_1=b(\Omega_1^2+\Omega_2^2+\Omega_3^2+\Omega_4^2) \\ U_2=b(\Omega_1^2-\Omega_2^2-\Omega_3^2+\Omega_4^2) \\ U_3=b(-\Omega_1^2-\Omega_2^2+\Omega_3^2+\Omega_4^2) \\ U_4=d(-\Omega_1^2+\Omega_2^2+\Omega_3^2-\Omega_4^2) \end{cases} \quad (4\text{-}2)$$

Where $\Omega_1, \Omega_2, \Omega_3, \Omega_4$ are the speeds of the four motors respectively, b, d are the force to torque scaling factors respectively.

Hovering

As each motor rotates with its propeller, it generates an upward lift force and a counter-torque force in the opposite direction. When the counter-torque force generated by the two diagonal shafts (motor 1+2 VS motor 3+4) is equal to each other, the system stability is guaranteed. At the same time, the combined lift from the four motors is just enough to cancel out the plane's own gravity, and the plane hovers.

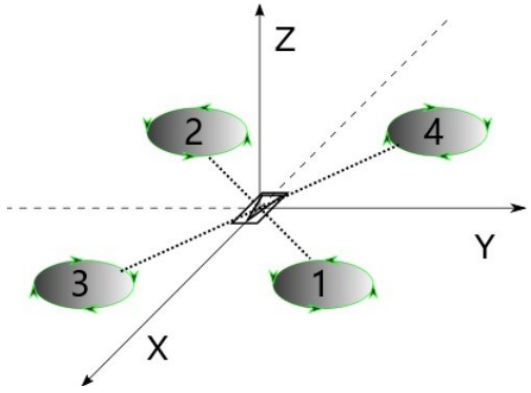

Figure 3.3 The rotation state of the four motors when hovering

Vertical motion

It is ensured that the reversing torques cancel each other and the total lifting force is increased so that it is greater than gravity, and the body can rise vertically. If the total lifting force is decreased to the level less than gravity, the body can fall vertically.

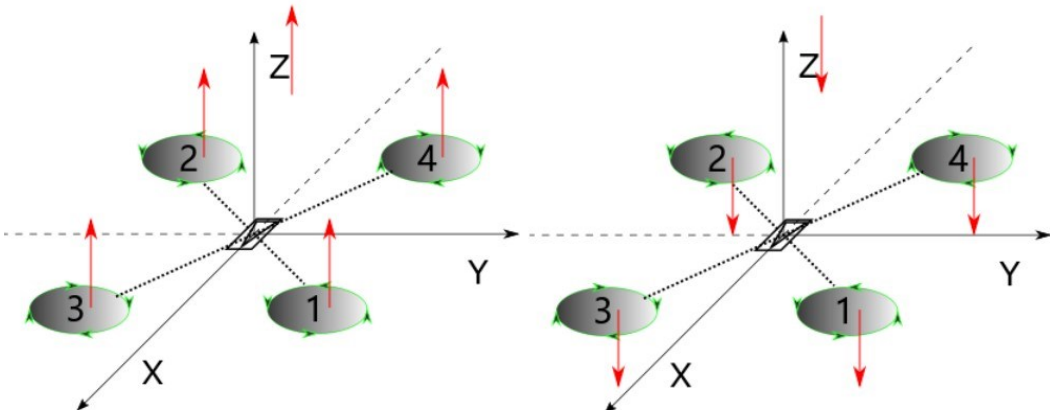

Figure 3.4 The rotation state of the four motors when doing Vertical movement

Pitch motion (forward and backward motion)

At the same time, the speeds of motors 1 and 3 are reduced and, the speeds of motors 2 and 4 are increased, so the aircraft will be bent forward. The total lift in the forward position is not vertical, but forward with the plane. This will produce a component going forward in the horizontal direction. In this position, the plane moves forward with this horizontal force. Similarly, if we increase the motors speed of 2 and 4 and decrease the motor speeds of 1 and 3, the plane will lean back. The total lifting force in the case of rearward also rearward, creating a component of the horizontal rearward force. In this position, the plane will move backward with this horizontal force.

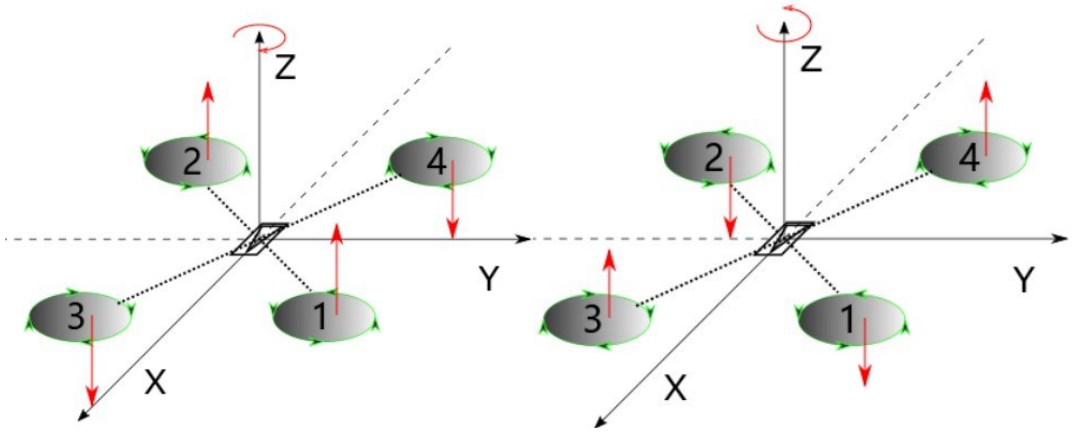

Figure 3.5 The rotation state of the four motors when doing Pitch motion

Rolling motion (side motion)

The principle is similar to pitching motion. If one increases the speeds of motors 1 and 4 and decreases the speeds of motors 2 and 3, and the plane will roll to the right. If one leans to the right,

the plane will move to the right. If one increases the speeds of motors 2 and 3 and decreases the speed of motors 1 and 4, and the plane will roll to the left. If one leans to the left, the plane will move to the left.

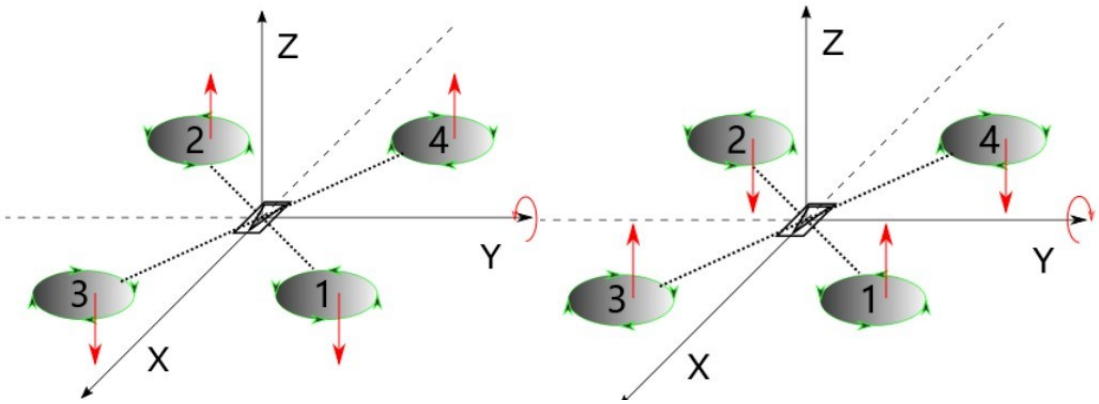

Figure 3.6 The rotation state of the four motors when doing Rolling motion

The flight control and method

Pixhawk [50] is the world's most famous open-source flight control hardware launched by manufacturer 3DR. It is a UAV control system, running PX4 and APM environment. Pixhawk is known for its powerful functions and reliable performance. The open-source of its hardware has enabled many hardware manufacturers to join the ranks of manufacturing Pixhawk. The Pixhawk has an Atmel ATMEGA2560 chip for processing, and an Atmel ATMEGA32U-2 chip for USB-functions.

The Pixhawk includes an Inertial Measurement Unit (IMU) with the following:

- InvenSense MPU-6000, 3-axis Gyro / 3-axis Accelerometer
- Honeywell HMC5883L-TR 3-axis Digital Compass
- Measurement Specialties MS5611-01BA03 Barometric Pressure Sensor

Figure 3.7 Pixhawk hardware

An inertial navigation module composed of a 3-axis gyroscope and a 3-axis acceleration sensor is installed on the four axes. The flight control has the information of the current attitude, acceleration, and angular velocity of the aircraft according to the data returned by these sensors. The flight control calculates and evaluates the deviation between the current attitude and the target attitude through algorithms, and then outputs the action correction of the four motors through these deviations.

In other words, the pilot simply tells the flight control what he wants to do with the four-axis motion, and the flight control combines the sensor information to split the pilot's commands into four motors.

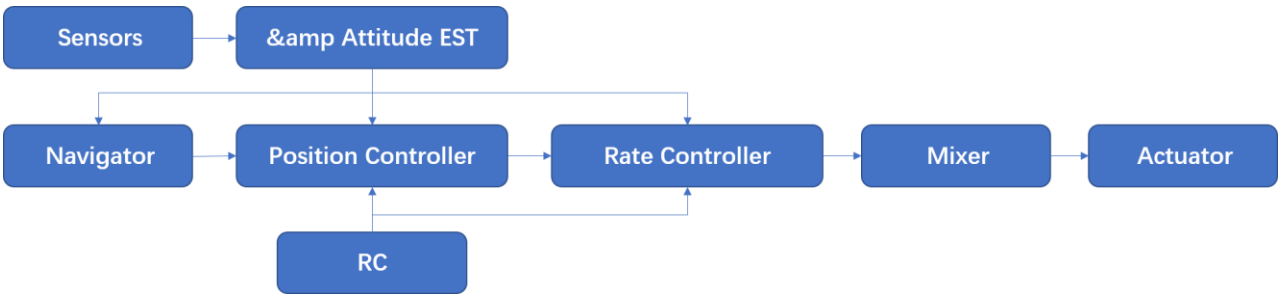

Figure 3.8 Flight control and execution process

3.2 Payload

The payload is the carrying capacity of a quadrotor. This could be upgrading the camera to a dual thermal and RGB imaging system, adding LiDAR technology, sticking on a GPS system, or increasing the number of sensors in order to process more data simultaneously. In this dissertation, the payload is about 3000g.

3.2.1 Raspberry Pi

Raspberry Pi 4 Model B [51] is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. Some of its most important properties are listed in Table3.1

Table 3.1 The configuration of Raspberry Pi

Overview	Raspberry Pi 4
CPU Processor Type RAM	1.5GHz Quad Core Cortex-A72 (ARM v8) 4GB - CPU
Physical Attributes	
Size Weight	8.5cm × 5.6cm × 3cm 82g
Interfacing	
Display Port Power Storage Ethernet	HDMI 5V DC via USB-C SD Card - Ethernet Wi-Fi RJ45 and Wi-Fi
Environment	
Operating Temperature Humidity	Operating temperature 0–50ºC N/A

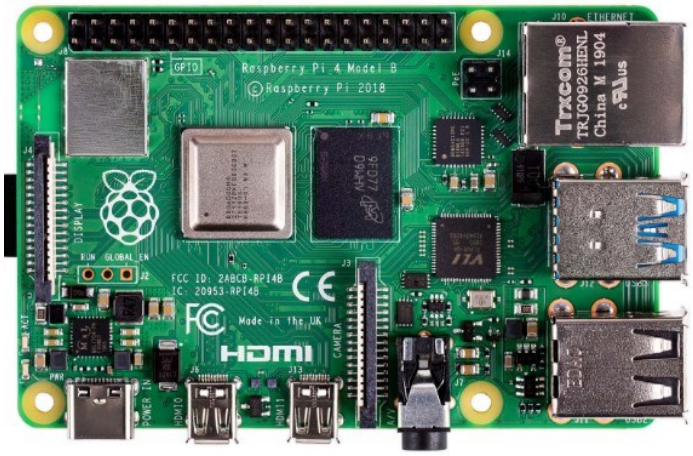

Figure 3.9 Raspberry Pi board

3.2.2 Realsense D435i

The Intel® RealSense™ D435i [52] places an IMU into our cutting-edge stereo depth camera. With an Intel module and vision processor in a small form factor, the D435i is a powerful complete package which can be paired with customizable software for a depth camera that is capable of understanding its own movement. which gives a field of view of 86° × 57° (±3°) with a frame rate of Up to 90 fps. This camera can be seen in Figure 3.10

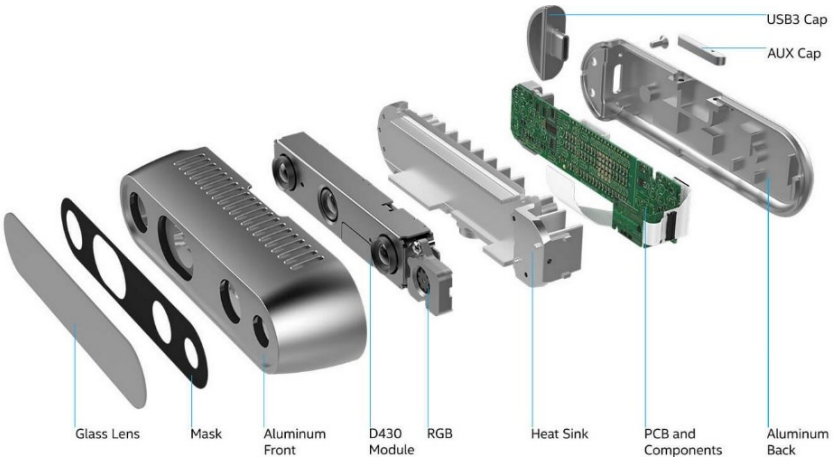

Figure 3.10 RealSense D435i Camera

3.3 Software

3.3.1 PX4

The PX4 is a complete open source autopilot system capable of controlling angular rotations and altitude and performing programmed GPS missions with waypoints. It is an open source flight control software for drones and other unmanned vehicles. The project provides a flexible set of tools for the drone developers to share technologies to create tailored solutions for drone applications. PX4 provides a standard to deliver drone hardware support and software stack, allowing an ecosystem to build and maintain hardware and software in a scalable way.

3.3.2 Ubuntu

Ubuntu is an open source operating system based on the Linux kernel. It is one of the most popular Linux distributions, and therefore it has the advantage of being frequently updated and having support for many different USB-components such as frame grabbers and different GPS devices. Ubuntu is also widely used for platforms such as the ROS, resulting in a variety of distributions that are optimized with respect to the ARM architecture. All of these factors make
Ubuntu an ideal operation system for the ROS.

3.3.3 Computer vision

OpenCV
Intel Open Computer Vision Library, referred to as OpenCV, is a function library mainly aimed at real-time computer vision. The library is open-source and cross-platform, and receives frequent updates from its large user base. In this thesis OpenCV can be implemented in object detection and object tracking.
Yolo
YOLO (You Only Look Once) is a state-of-the-art (2019) technique to detect objects within images. One of its advantages is that it's extremely fast compared to other techniques, which makes it suitable for using it with video feeds at high frame rates (with a fast Nvidia GPU). YOLO

applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

3.3.4 ROS GAZEBO RVIZ

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. In this project, there are two simulation software (GAZEBO and RVIZ) based on ROS being implemented, GAZEBO is a 3D environment simulation software, which can load the sim-env and the flight model. RVIZ is a topic publishing software, which can also show the model in vision by using the urdf file.

3.3.5 MATLAB/SIMULINK

Simulink is a simulation and model-based design environment for dynamic and embedded systems, integrated with MATLAB. Simulink, also developed by MathWorks, is a data flow graphical programming language tool for modelling, simulating and analyzing multi-domain dynamic systems. It is basically a graphical block diagramming tool with customizable set of block libraries. It allows you to incorporate MATLAB algorithms into models as well as export the simulation results into MATLAB for further analysis. In this thesis, the construction of PID controller is based on the SIMULIINK and also the trajectory generation source from MATLAB calculation.

3.4 Communication Protocols

MAVlink [53] protocol is a higher-level open source communication protocol based on serial port communication. It is mainly used in the communication of micro aerial vehicle. MAVlink provides rules for sending and receiving data frequently used in communications between small

aircraft and ground stations (or other aircraft) and adds a checksum function. It is used for communication between an PX4, and the Ground Control Station. Figure 3.11 describes the communication between PX4 on SILT with QC and API by using MAVlink.

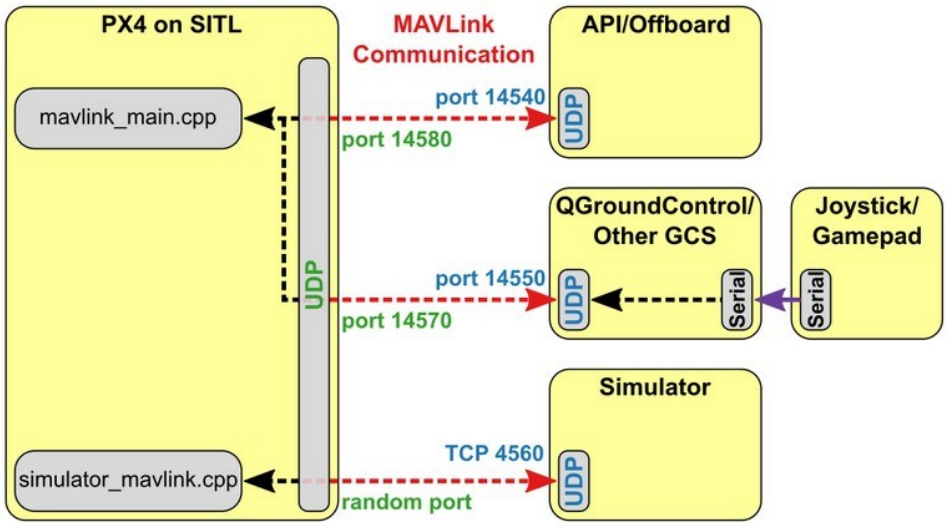

Figure 3.11 MAVlink Communication Protocols

3. 5 Summary

This chapter mainly describes the design structure of the quadrotor, X-type and the four Rotor to achieve operational guidelines through positive and negative rotating speeds, and meanwhile introduces in detail the rotor system, flight control, Raspberry Pi, Realsense D435i camera and other hardware systems, simulation platform, Computer Vision software and communication protocol needed for simulation.

Chapter 4 Research methodologies

In this chapter, we will present the developed methodologies on AKF, HPF and MDP. Figure 4.1 shows the whole process consisting of each algorithm that plays a role in the autonomous navigation.

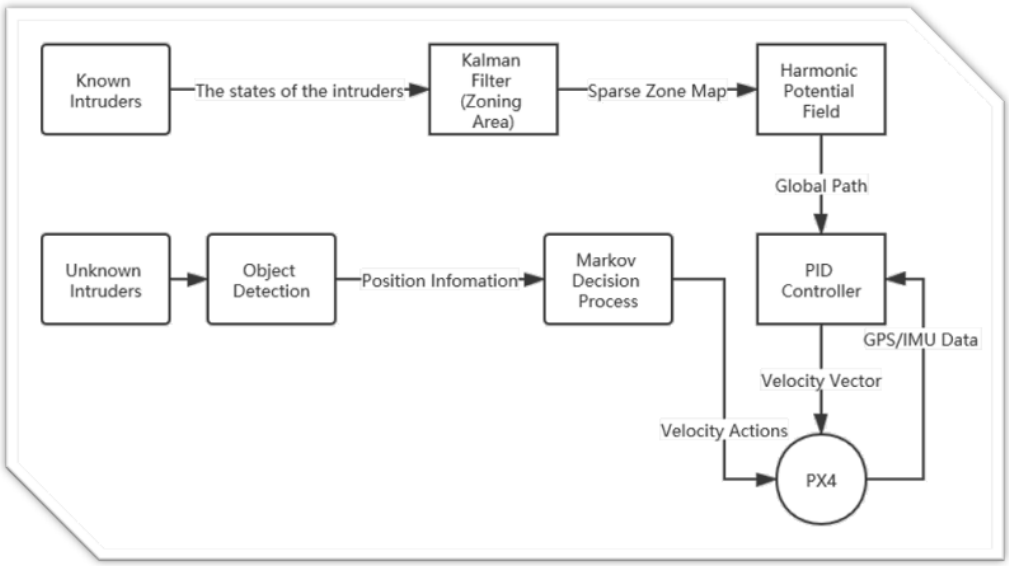

Figure 4.1 Overview of Control Algorithm

4.1 Path planning and collision avoidance

A new method is proposed for path planning in a dynamically changing environment. It particularly targets safe integration of UAS in an airspace that is shared by other flying aircraft amid ground-based static and dynamic obstacles. Such an environment is common in an urban air mobility (UAM) application where several UAS platforms and other manned aircraft may also operate. The proposed method on the one hand adopts HPF with APF for global planning, and on the other hand, applies AKF prediction scheme with MDP for collision avoidance. The harmonic field formulation helps avoid the local minimum issue, and the MDP governs the flight path and direction selection to minimize the probability of encountering dynamic flying obstacles. We have conducted hardware-in-the-loop simulations of the proposed algorithm using MATLAB/Simulink and ROS gazebo environment Pixhawk and D435i hardware.

The following sub-sections explain the adopted methods of each section of the algorithm in greater details.

4.1.1 Adaptive Kalman Filter

The AKF is a recursive filter that can estimate the internal states of a linear system from the noisy measurements. In this thesis, the AKF is used to predict the discrete states of the UAVs and facilitate dynamic path planning [54]-[57]. By predicting the location of existing UAVs at several future instances, we can segment the environment based on the level of traffic density expected at each segment. This step will attempt to reduce the likelihood of an encounter with other UAVs in the area and the time and effort required to execute local collision avoidance.

Considering that UAV in a 3Dspace, the state vector x_t of a UAV can be expressed as follows:

$$x_t = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{4-1}$$

Accordingly, for the relationship between position and velocity, one has

$$p_t = p_{t-1} + v_{t-1} \times \Delta t + \frac{1}{2} u_t \times \Delta t^2 \tag{4-2}$$

$$v_t = v_{t-1} + u_t \times \Delta \tag{4-3}$$

Where u_t is the acceleration of the UAV.

We can write (4-6) and (4-7) in the following matrix format:

$$\begin{bmatrix} p_t \\ v_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta t^2/2 \\ \Delta t \end{bmatrix} u_t \tag{4-4}$$

and define F_t as the state transition matrix and B_t the control matrix, as below

$$F_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{4-5}$$

$$B_t = \begin{bmatrix} \Delta t^2/2 \\ \Delta t \end{bmatrix} \tag{4-6}$$

The prediction equation of AKF can now be written as:

$$\hat{x}_{t|t-1}^{-} = F_t \hat{x}_{t-1|\mathrm{t}-1} + B_t u_{t-1} \tag{4-7}$$

Considering the uncertainty in the observations, the state prediction equation will be:

$$\Sigma_{t|t-1}^{-} = F \Sigma_{t-1|\mathrm{t}-1} F^T + B \bar{Q}_{t-1} B^T \tag{4-8}$$

$$\Sigma_{t|t-1}^{-} = F \Sigma_{t-1|\mathrm{t}-1} F^T + Q \tag{4-9}$$

Where Σ_{t-1} is covariance matrix at $t-1$ moment, and Q is the noise covariance matrix

Then by combining two Gaussian distributions, Kalman gain K_t can be set:

$$K_t = \Sigma_{t|t-1}^{-} H^T \left(H\Sigma_{t|t-1}^{-} H^T + \bar{R}_{t-1}\right)^{-1} \tag{4-10}$$

$$K_t = \Sigma_{t|t-1}^{-} H^T \left(H\Sigma_{t|t-1}^{-} H^T + R\right)^{-1} \tag{4-11}$$

Where H is the transfer matrix and R is the observation covariance matrix.

The relation between observations and predictions is presented as:

$$Y_t = H\hat{x}_{t|\mathrm{t-1}} + v \tag{4-12}$$

$$\varepsilon_t = Y_t - H\hat{x}_{t|\mathrm{t-1}} \tag{4-13}$$

Where Y_t is the observation value from sensor and v is the observation noise.

Finally, the covariance and state variables are updated as:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1}^{-} + K_t\varepsilon_t \tag{4-14}$$

$$\Sigma_{t|t} = (I - K_t H)\Sigma_{t_t-1}^{-} \tag{4-15}$$

The locations of the UAVs in the area at a number of future steps can be predicted by the AKF. Here the updates of R and Q matrix are shown:

$$\bar{Q}_t = (1-d_{t-1})\bar{Q}_{t-1} + d_{t-1}\left(K_t\varepsilon_t{\varepsilon_t}^T{K_t}^T + \Sigma_{t|t} - F_t\Sigma_{t-1|t-1}{F_t}^T\right) \tag{4-16}$$

$$\bar{R}_t = (1-d_{t-1})\bar{R}_{t-1} + d_{t-1}\left(\varepsilon_t{\varepsilon_t}^T - H\Sigma_{t|t-1}H^T\right) \tag{4-17}$$

Where $d_t = \frac{1-b}{1-b^{t+1}}$ $0 < b < 1$.

Assuming that there are k number of intruders in the environment, the positions of them are defined as:

$$\hat{x}_{t(m)} = \hat{x}_{t(m)}^{-} + K_{t(m)}\left(Y_{t(m)} - H\hat{x}_{t(m)}^{-}\right) \tag{4-18}$$

For the simulations of this thesis and without loss of generality the number of future steps is set at 15 and the transfer matrix H is defined accordingly. Using this prediction method, we can define multiple zones inside the environment according to the different number of UAVs predicted to occupy each at any given instant.

To simplify the simulation, we divide the environment map into four equal area zones $Zone_n$ ($n = 1，2，3，4$) on the 2D plane with the four coordinates of (x_ln,y_ln,x_hn,y_hn), where x_ln is the lower coordinate of the lower vertices of the zone n and x_hn is the high

coordinate of the higher vertices of zone n.

We track the position of intruders $\hat{x}_{t(m)}$ inside the environment and zones to identity sparse and dense zones according to the following criteria. We designate zones with low number of intruder drones (N = 0,1) as sparse and then zones with higher numbers as dense.

$$\begin{cases} Zone_{sparse} & (N = 1) \\ Zone_{dense} & (N > 1) \end{cases} \tag{4-19}$$

Where N is the number of the drone in the $Zones$ after the set number of future steps (15s).

4.1.2 Harmonic Potential Field

HPF is used for global path planning. Mathematically, they are the solutions to the Laplace's equation (below) that can be used to avoid the local minima problem of APF method.

$$\nabla^2 f = 0 \tag{4-20}$$

Where ∇^2 is called Laplace operator applied to a potential filed f. In the context of fluid low, for example, f can be understood as representing a potential that is inviscid, incompressible and irrotational. The HPF must satisfy the Laplace's equation, below,

$$\nabla^2 P = \frac{\partial^2 P}{\partial X^2} + \frac{\partial^2 P}{\partial Y^2} + \frac{\partial^2 P}{\partial Z^2} \tag{4-21}$$

$$P = P(X, Y, Z) \tag{4-22}$$

in a 3D environment with x, y, z axes.

In the thesis, since our boundary conditions are approximately spherical, we convert rectangular coordinates to spherical coordinates.

$$\begin{cases} X = r \sin\varphi \cos\theta \\ Y = r \sin\varphi \sin\theta \\ Z = r \cos\varphi \end{cases} \tag{4-23}$$

Where r is the Radius of the spherical boundary and θ, φ are the angle rotated by Z axis in X-Y plane and angle with Z axis respectively. Substituting formula (4-27) into formula (4-25) yields:

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial P}{\partial r}\right) + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial P}{\partial \theta}\right) + \frac{1}{r^2 \sin\theta^2}\frac{\partial^2 P}{\partial \varphi^2} = 0 \tag{4-24}$$

In order to solve the Laplace formula, we use the method of separating variables to solve and

(4-26) can be expressed as in the form:

$$P(r,\theta,\varphi) = R(r)Y(\theta,\varphi) \tag{4-25}$$

Then putting (4-29) into (4-28), we get two equations

$$r^2\frac{d^2R}{dr} + 2r\frac{dR}{dr} - l(l+1)R = 0 \tag{4-26}$$

$$\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial Y}{\partial\theta}\right) + \frac{1}{\sin\theta^2}\frac{\partial^2 Y}{\partial\varphi^2} + l(l+1)Y = 0 \tag{4-27}$$

Where l is a constant $\in N$.

Regarding R is an Euler equation, one needs to replace the variables to solve the equation. Hence, a new equation is acquired.

$$r = e^t, t = lnr \tag{4-28}$$

$$\frac{d^2R}{dt^2} + \frac{dR}{dt} - l(l+1)R = 0 \tag{4-29}$$

$$R(r) = Cr^l + D\frac{1}{r^{l+1}} \tag{4-30}$$

Where C and D are constants $\in R$. The parameter separation of Y can be solved as,

$$Y(\theta,\varphi) = \Psi(\varphi)\Theta(\theta) \tag{4-31}$$

Combining (4-31) with (4-25), we get

$$\Psi(\varphi) = A\cos n\varphi + B\sin n\varphi \tag{4-32}$$

Where A, B, n are constants $\in R$. According to Legendre Polynomial, the other equation can be defined as,

$$p_l(x) = \frac{1}{2^1 l!}\left(\frac{d}{dx}\right)^l (x^2-1)^l \tag{4-33}$$

$$\Theta(\theta) = (1-x^2)^{\frac{m}{2}} p_l^m(x) \tag{4-34}$$

Where x $= \cos\theta$ and m $\in N$. In the end, putting $R(r), \Psi(\varphi), \Theta(\theta)$ into (4-24), one has

$$P(r,\varphi,\theta) = R(r), \Psi(\varphi), \Theta(\theta) \tag{4-35}$$

$$P(r,\varphi,\theta) = \sum_{m=0}^{\infty}\sum_{l=m}^{m} r^i [A_l^m \cos n\varphi + B_l^m \sin n\varphi]\, p_l^m(\cos\theta) + \sum_{m=0}^{\infty}\sum_{l=m}^{m} \frac{1}{r^{i+1}} [C_l^m \cos n\varphi + D_l^m \sin n\varphi]\, p_l^m(\cos\theta) \tag{4-36}$$

To guide the UAV towards the goal and avoid obstacle collision, a potential field algorithm is

used. The goal is to use a standard attractive force that varies linearly with distance. The linear relationship ends at a threshold distance. The value of g_{thresh} is set by user heuristically based on the tolerable limit for getting close to the obstacle. Beyond this threshold, the attractive force stays constant in magnitude to prevent high attractive forces. Together, the attractive force $\boldsymbol{G}$ is:

$$G = \begin{cases} \xi\left(s_{goal} - s_{start}\right) & if \left\| s_{goal} - s_{start} \right\| \le g_{thresh} \\ \xi g_{thresh} \frac{s_{goal} - s_{start}}{\left\| s_{goal} - s_{start} \right\|} & if \left\| s_{goal} - s_{start} \right\| > g_{thresh} \end{cases} \tag{4-37}$$

Where $\xi > 0$ is a scaling constant, $s_{start} \in \mathfrak{R}^3$ is the current location and s_{goal} is the goal location.

Implement Newman boundary conditions

Newman boundary conditions is applied in this method. Given a continuous function f on a smooth closed surface Γ, look for the function $P(r, \varphi, \theta)$: In the interior of Γ, Ω is a harmonic function, which is continuous on (Γ+ Ω), and the normal guide number exists at any point on Γ and is equal to the known function F, which is,

$$\frac{\partial u}{\partial n}|_{\Gamma} = F \tag{4-38}$$

Numerically, the 3-D Laplace's equation can be solved by using the Separation of Variables, and the Newman boundary condition:

$$\begin{cases} \nabla^2 P = 0 \\ \frac{\partial P}{\partial n}|_{\Gamma} = 0 \\ \frac{\partial G}{\partial n}|_{\Gamma} = 0 \end{cases} \tag{4-39}$$

For boundary conditions the gradient of the obstacles and the border of the map are equal to zero. At the start point, the boundary value is equal to 1, and the end point is 0.

4.1.3 Markov Decision Process

MDP is applied for obstacle avoidance and relies on an onboard depth sensor (RealSense^{TM} camera is used for simulations of this work). When the 3D camera receives the depth information from an intruder aircraft, then the relative position away from own ship is computed. The algorithm could detect the intruder whether go through into the safety area.

There are five parameters in MDP $\langle S \quad A \quad P \quad R \quad \gamma \rangle$ representing state, action, state transfer probability, reward, and discount, respectively. P and R are corresponding with specific action:

$$P_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a] \tag{4-40}$$

$$R_s^a = E[R_{t+1}|S_t = s, A_t = a] \tag{4-41}$$

Where $P_{ss'}^a$ and R_s^a represent the probability from s to s' and the reward after taking action a respectively.

Then we need to define a policy to determine which action should be taken.

$$\pi(a|s) = P(A_t = a|S_t = s) \tag{4-42}$$

So, the $\pi(a|s)$ is the probability of action at state s.

When given a MDP and a policy $\langle S \quad P^\pi \quad R^\pi \quad A \quad \gamma \rangle$, one makes the reward satisfying two equations:

$$P_{ss'}^\pi = \sum_{a \in A} \pi(a|s)\, P_{ss'}^a \tag{4-43}$$

$$R_s^\pi = \sum_{a \in A} \pi(a|s)\, R_s^a \tag{4-44}$$

Then the state value function and action value function can be defined as:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s'} P_{ss'}^a\, v_\pi(s') \right) \tag{4-45}$$

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s')\, q_\pi(s', a') \tag{4-46}$$

One can define the optimal function with (4-49) and (4-50) as following:

$$v_*(s) = max_a R_s^a + \gamma \sum_{s'} P_{ss'}^a\, v_*(s') \tag{4-47}$$

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a\, max_{a'} q_\pi(s', a') \tag{4-48}$$

The next state of the drone from KF can be obtained as:

$$p_{t+1} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4-49}$$

Then we set the obstacle reward R_{obs} a negative value and goal reward R_{goal} a positive value.

$$R = \begin{cases} R_{obs} = Negative \\ R_{goal} = Positive \\ R_{free} = Zero \end{cases} \tag{4-50}$$

The action set (18 direction) can be defined as below:

$$A = \{a_n \quad a_s \quad a_e \quad a_w \quad a_{n-e} \quad a_{n-w} \quad a_{s-e} \quad a_{s-w} \ldots\} \tag{4-51}$$

Where n, s, e, w represent geographical North, South, East, and West respectively.

When the collision is predicted to occur at t_{k+1}, MDP is applied to select the next action at time t_k.

To apply the MDP for the obstacle avoidance, first a local small grid map $n \times n$, is considered around the UAS current location and the reward R is associated with the grid points. Then

according to equations (4-44) and (4-45), one can get a set of state reward value matrix ($n \times n$).

$$V_{ij} = \begin{pmatrix} v_{ij} & \cdots & v_{in} \\ \vdots & \ddots & \vdots \\ v_{nj} & \cdots & v_{nn} \end{pmatrix} \tag{4-52}$$

According to equation (4-46), the maximum of V_{ij} will determine the policy $\pi(A|p)$, and the policy is to decide which direction the drone should take.

$$\pi(A|p) = \begin{pmatrix} \pi_{ij} & \cdots & \pi_{in} \\ \vdots & \ddots & \vdots \\ \pi_{nj} & \cdots & \pi_{nn} \end{pmatrix} \tag{4-53}$$

Then one transfers the policy to the state, and the drone will make an avoidance. This method also fits for multi-drone scenario.

Below, the pseudocode of the algorithm is presented:

Table 4.1 Run AKF to make airspace zoning

```
1: Initialize map and intruder location
2: For each intruder
3:      Get the state (Velocity and Position) information
4:      Predict the location in next several seconds
5: Return the prediction
6: Create a count node
7:      If count node > threshold
8:           Dense Zone
9:      Else Sparse Zone
```

Table 4.2 Run HPF to make path planning

```
1: Initialize G field
2:      Start =1, goal = 0, grad_n = 0 at all boundaries
3: Initialize P field
4:      Obstacle boundaries = 1, map boundaries = 0
5: While follow the path
6: If G = 1 Change the direction
7:      If gradP * gradG >0
8:      break
9: If P < P_thresh
10:     Follow contour of G
11: Else
12:     Follow contour of P
```

Table 4.3 Run MDP to avoid obstacles in real time

1: Initialized the RGB-D camera
2: Compute the relative position with the intruders
3: Define the discount factor and reward rule
4: While intruder invade safety range
5: Divide the grid map
6: Calculate the sum of rewards and the policy
7: Decide the action

4.2 Architecture and hardware in the loop simulation

4.2.1 Hardware loop

As illustrated in Figure 4.8, the simulation software interfaces with a Pixhawk autopilot, RealSense^{TM} D435i camera and Raspberry Pi 4 via MAVLINK, and is also integrated with the PX4 Firmware dynamic model and platform.

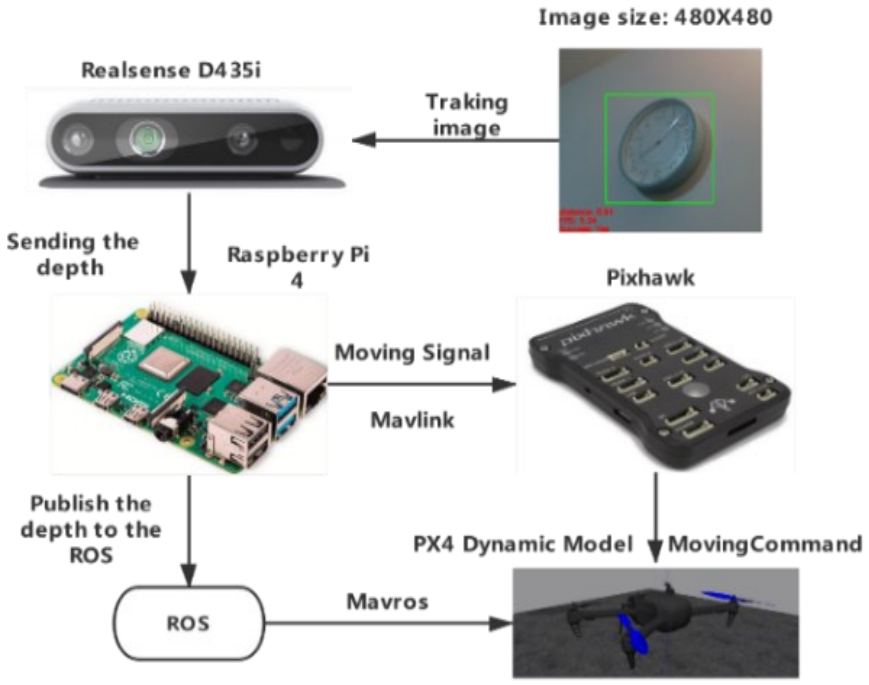

Figure 4.2 Hardware-in-the loop simulation

Further, OpenCV median tracker [58] is implemented for tracking the object in this research. For the planned simulations the object is first detected and tracked by using the RealSense^{TM} 435i depth camera, then Raspberry Pi transfers the depth signal to the motion command to the Robot Operating System (ROS) and the Pixhawk, and then the simulation's codes are generated and rendered in Gazebo.

4.2.2 Real-time PID control loop

In this research, plenty of virtual simulations of flight tests are based on the PID controller, as shown in Figure 4.3 Firstly, the path planning algorithm is used to generate the path. Once the path has been created, we set up a stack protection mechanism for the UAV, which compares the GPS information with the path information in real-time. When the position error is larger than the built-in threshold of the system, the control system carries out stack protection and stops supplying the value of the next target point until the threshold value is less than the built-in threshold of the system. Then the PID is used to control the x, y, and z positions of the drone. PID is a model-free control, while an adaptive control method that does not need a process model. It is capable of effectively controlling the position of the drone, and the effective parameters of the PID (P)=0.5，Integral (I)=0.3，Derivative (D)=0.3 are tuned by using the trial and error method.

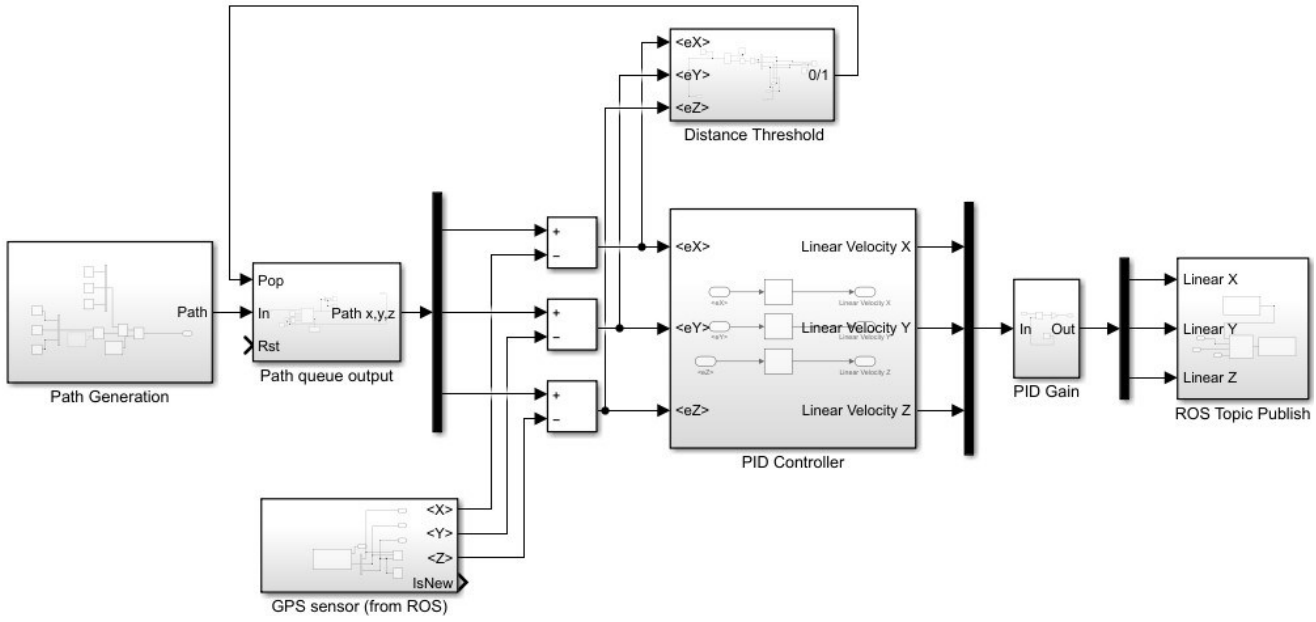

Figure 4.3 PID controller

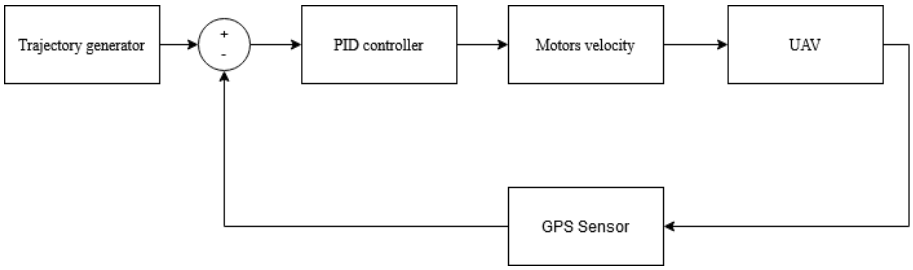

Figure 4.4 simple block diagram of the PID control system

Here there is a simple test to validate the PID controller:

In Figure 4.4, the drone will fly around the cube, the red arrow represents the direction and the simplified path of the drone,

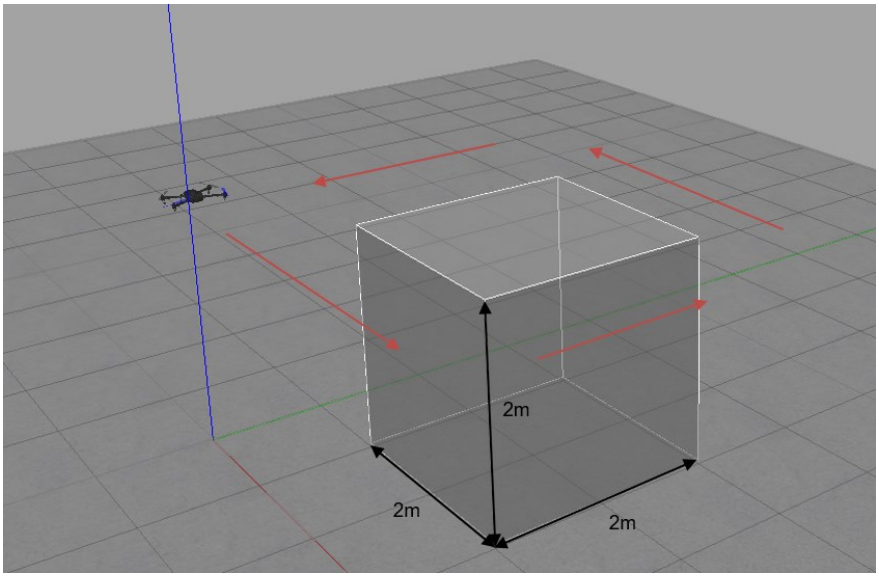

Figure 4.5 The reference simulation flying a square path

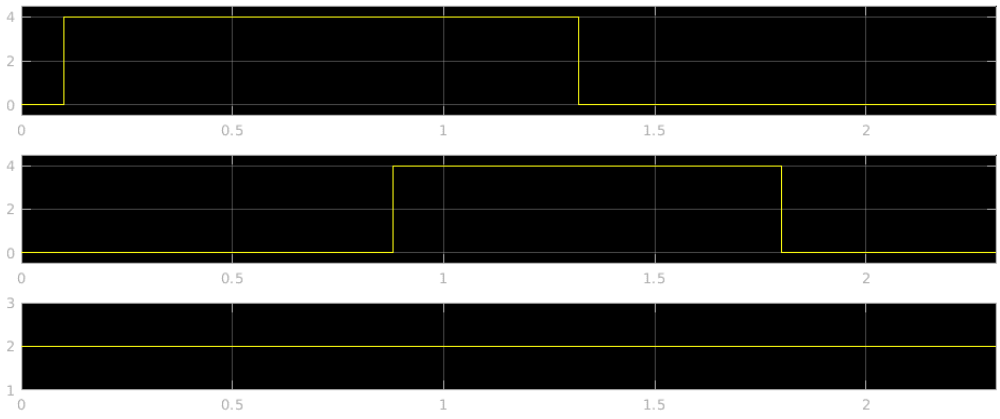

Figure 4.6 The reference path

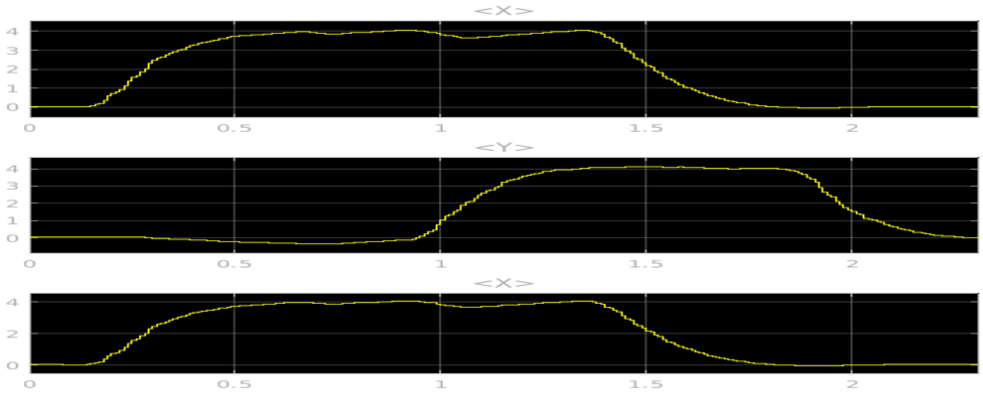

Figure 4.7 The output path processed by PID controller

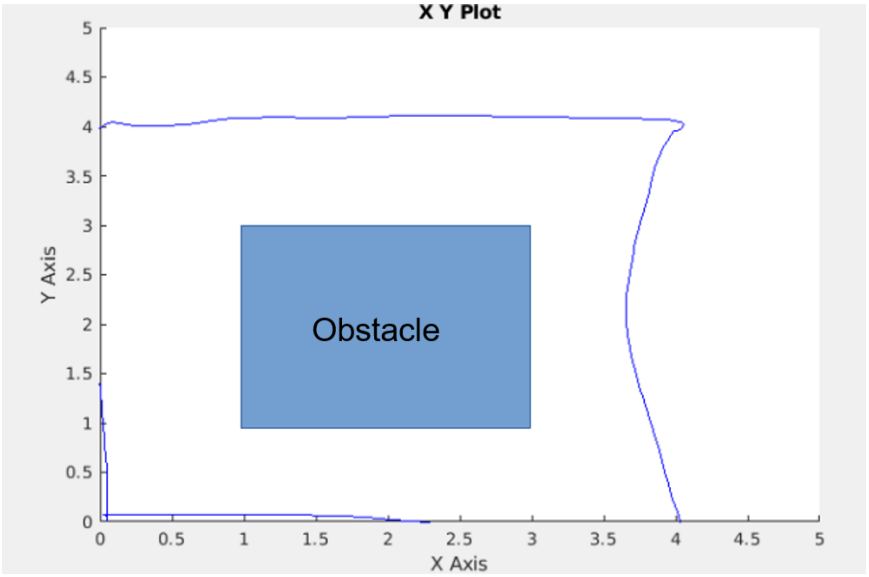

Figure 4.8 The output path processed by PID controller in 2D

Figure 4.5 describes the reference path that the drone should follow, and Figure 4.6 show the raw path in X, Y, Z direction respectively after implementing the PID Controller, also Figure 4.7 gives a clear view of the path.

4.3 Summary

This chapter illustrates how to achieve autonomous navigation by using mathematical methods. Moreover, the function of each approach in the whole navigation system is described, such as the AKF estimates the states of the known intruders, 3D HPF generates the desired trajectory, and MDP makes real-time collision avoidance. Besides, the block diagram and some instructions of the simulation of Hardware-in-the-loop and PID controller loop are provided.

Chapter 5 Simulations and Experiments

The proposed motion planning algorithm is illustrated and evaluated through a series of test cases by simulations. First, one test case is shown for one stage of the algorithm separately, and then a combined case is presented. The fourth and last scenario integrates the three stages of the algorithms i.e., AKF, HPF, and MDP in a case that resembles the UAM application. For a better demonstration of tracking the performance of the algorithm, the simulation scenarios are arranged to follow the pseudocode sections, and hence we will first start with the AKF section, the HPF, and finally the MDP. As mentioned before, the last scenario is a combined case that tests the entire algorithm.

5.1 Case 1: Computer vision

5.1.1 Object Detection

Object detection and tracking are vital for UAVs to make obstacle avoidance, especially working in an unknown environment. In this case, the simulation about the computer vision is all based on a PX4 platform equipped with a RealSense D435i camera. Also, the Yolov3 Darknet 53. Conv .74 pre-trained weights on 100 databases are implemented. GPU OPENCV and CUDA have been chosen for accelerating the training speed. The configuration and the parameters are shown here:

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 5.1 The Pre-trained weight model of Yolov3 Darknet 53. Conv .74

Table 5.1 The configurations of pre-set up value

Parameters	**Value**
Input Dimension	416*416*3
momentum	0.9
saturation	1.5
Learning rate	0.001
batch	64
subdivisions	16
activation	linear

Eventually, through 1000 times iterations and 64000 images for training, the training loss makes regression. The training batch and subdivisions outputs have been illustrated in the following table.

Table 5.2 The training batch output

Batch Output	
Iteration times	1000
Total Loss	0.127639
Average Loss	0.110730
Learning rate	0.001
Batch time	9.428663
Total Training Images	**64000**

Table 5.3 The subdivisions batch output

Subdivisions output	
Region Avg IOU	56.98%
Class	0.999018
Obj	0.989786
No. Obj	0.000793
Count	**14**

In the above tables, the average loss reaches at 0.110730, which is good for 100 databases training, also the value of class arrives at 0.999019 close to 1, which means this training model could recognize the test image and classify the character almost 100 percent.

In Figure 5.2, there are a truck and a plant in front of own ship. At this time, the yolo weights are applied to detect the object. Yolo weights are trained by Darknet firmware which could recognize 80 classes in the real world.

Figure 5.2 Object detection by using standard yolo weights

Figure 5.3 illustrates that own VOC datasets are trained based on Yolov3 Darknet 53. Conv .74 shown in Figure 5.1. During this training, 100 images about PX4 SILT are treated as the input, then the input-goes through the Yolov3 weights, generating a new weight in the end. The result can be shown in Figure 5.3.

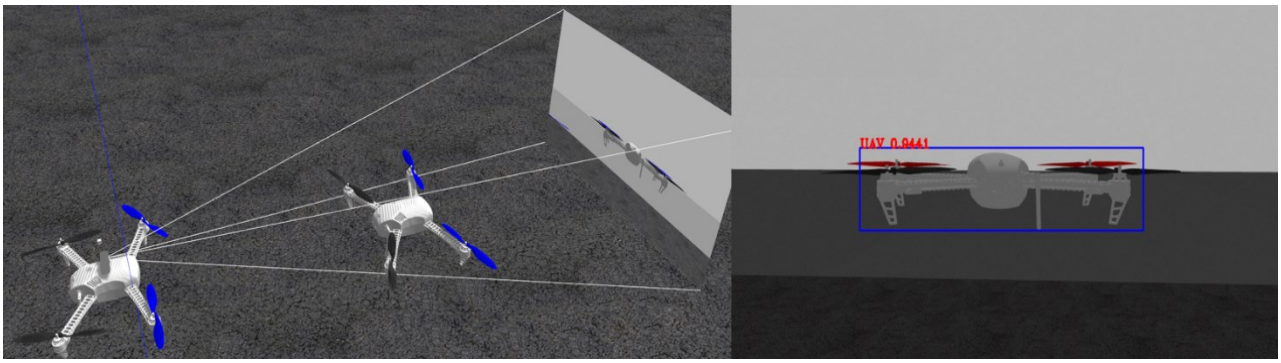

Figure 5.3 Object detection by using own VOC weights

5.1.2 Object tracking

OPENCV has plenty of object tracking algorithms, and each tracking method has own pros and cons, the following table show the characteristic of each approach.

Table 5.4 OpenCV trackers description/Pros and cons

Algorithms	Description/Pros and Cons
BOOSTING Tracker	**B**ased on the same algorithm used to power the machine learning behind Haar cascades (AdaBoost), but like Haar cascades, is over a decade old. This tracker is slow and doesn't work very well. Interesting only for legacy reasons and comparing other algorithms.
MIL Tracker	**B**etter accuracy than BOOSTING tracker but does a poor job of reporting failure.
KCF Tracker	**K**ernelized Correlation Filters. Faster than BOOSTING and MIL. Similar to MIL and KCF, does not handle full occlusion well
CSRT Tracker	**D**iscriminative Correlation Filter (with Channel and Spatial Reliability). Tends to be more accurate than KCF but slightly slower.
MedianFlow Tracker	**D**oes a nice job reporting failures; however, if there is too large of a jump in motion, such as fast-moving objects, or objects that change quickly in their appearance, the model will fail.
TLD Tracker	**T**he TLD tracker was incredibly prone to false-positives.
MOSSE Tracker	**V**ery, very fast. Not as accurate as CSRT or KCF but a good choice if you need pure speed.
GOTURN Tracker	**T**he only deep learning-based object detector included in OpenCV. It requires additional model files to run (will not be covered in this post). My initial experiments showed it was a bit of a pain to use even though it reportedly handles viewing changes well

According to the comparison of the tracking test from the real camera and the simulation cam, it turns out MedianFlow Tracker performs better than other trackers in the virtual environment. During the real-world tracking test, the CSRT tracker works better than others do. In this thesis, the MedianFlow Tracker and CSRT tracker are implemented in the simulation.

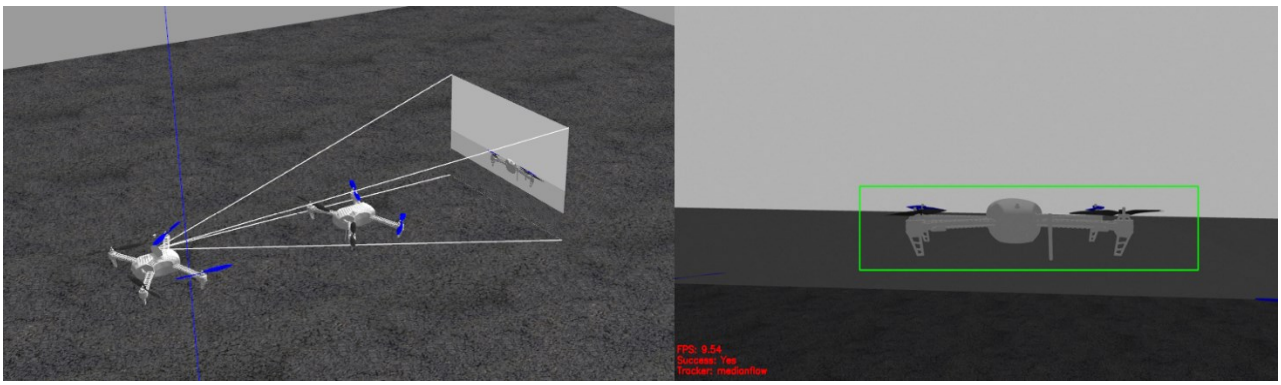

Figure 5.4 Object tracking by using MedianFlow Tracker (FPS: 9.54)

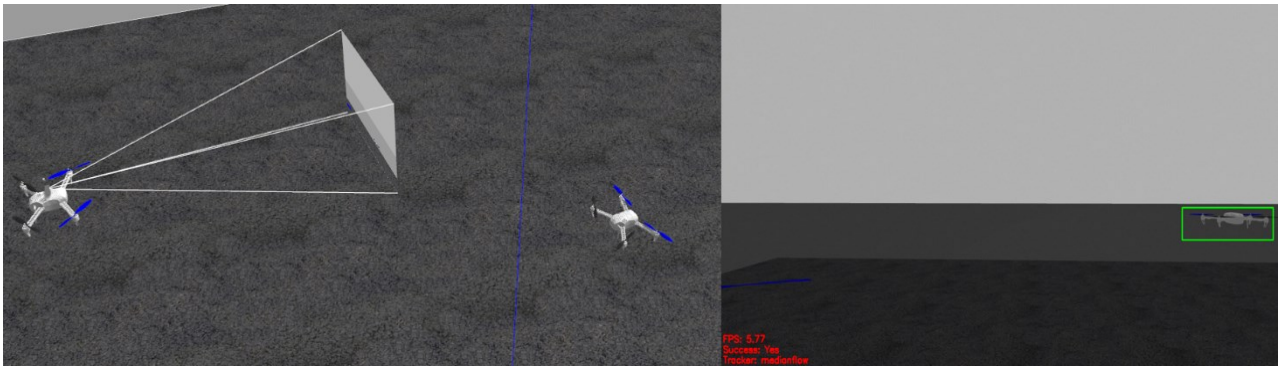

Figure 5.5 Object tracking by using MedianFlow Tracker (FPS: 5.77)

Figure 5.4 and Figure 5.5 show the tracking results when the intruder is moving. It is noticed that, the tracking is stable and fast. One drawback is that FPS may drop when it enables high speed.

5.2 Implementing RTAB-Octomap to build the environment

Using RTAB-map's powerful feature extraction function and Odometry, the combination with Octo-map can simplify the environment, reduce the amount of calculation of the point cloud map, speed up the drone's scanning speed of the map, thus help the drone to complete the task more efficiently. Figure 5.6 shows the drone turning on the camera at the starting position to model the environment

Table 5.5 RTAB-Map Default parameters

GFTT/MinDistance	3 pixels	RGBD/OptimizeMaxError	1
GFTT/QualityLevel	0.001	RGBD/ProximityMaxGraphDepth	50 nodes
Kp/MaxFeatures	500 features	Rtabmap/DetectionRate	2 Hz
Odom/KeyFrameThr (F2M)	0.3	Rtabmap/TimeThr	0 ms
Odom/KeyFrameThr (F2F)	0.6	Rtabmap/MemoryThr	0 nodes
Odom/ScanKeyFrameThr	0.9	Rtabmap/LoopThr	0.11
OdomF2M/MaxSize	2000 features	Vis/CorNNDR	0.6
OdomF2M/ScanMaxSize	10000 points	Vis/CorGuessWinSize	20 pixels
OdomF2M/ScanSubtractRadius	0.05	Vis/MaxFeatures	1000 features
Mem/RehearsalSimilarity	0.2	Vis/MinInliers	20
Mem/STMSize	30 nodes		

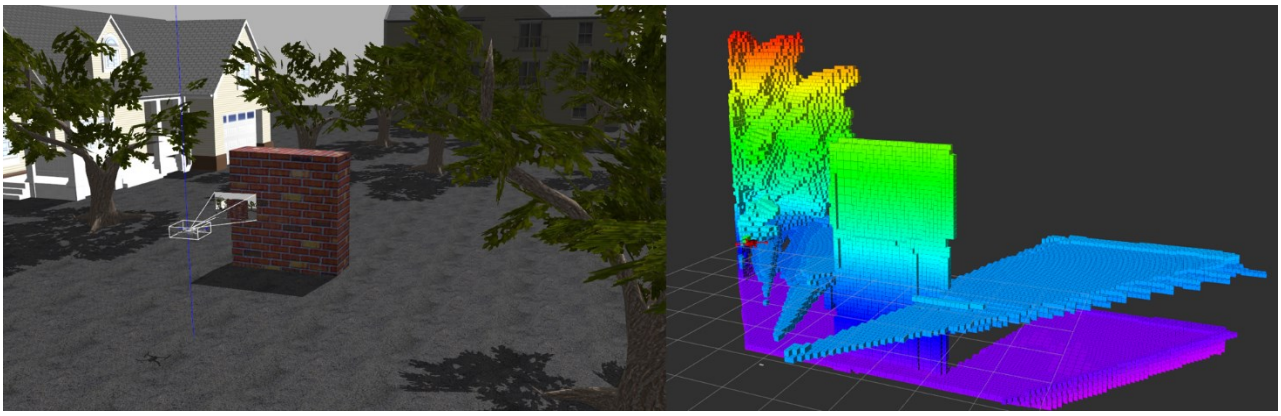

Figure 5.6 The partial Octo-mapping of the virtual world

In Figure 5.7, the left image is the 3D Octo-map and the right are the Odometry Feature image by using the RTAB algorithm.

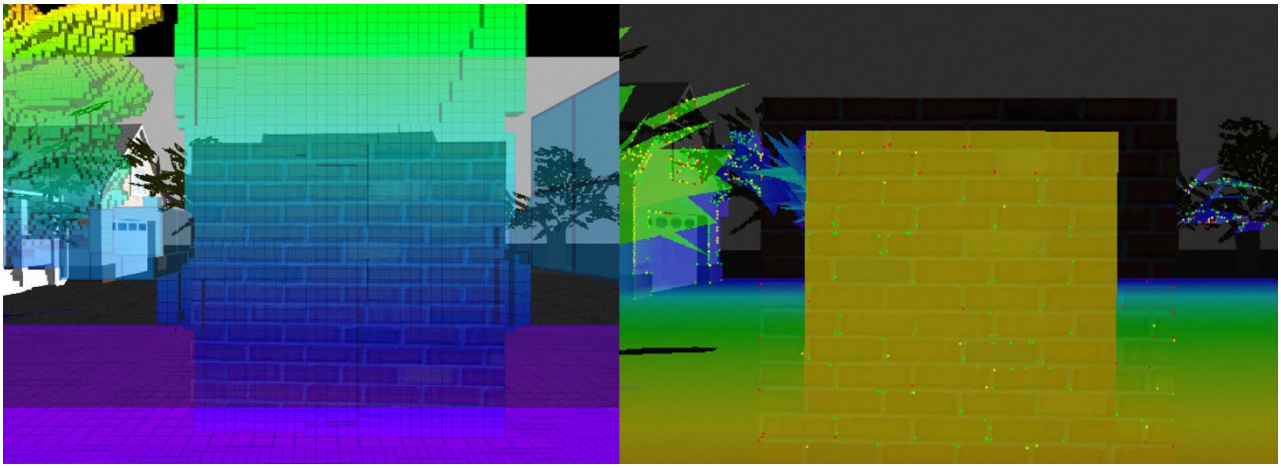

Figure 5.7 Octo-mapping image (left), Odometry Feature extraction image (right) at beginning point

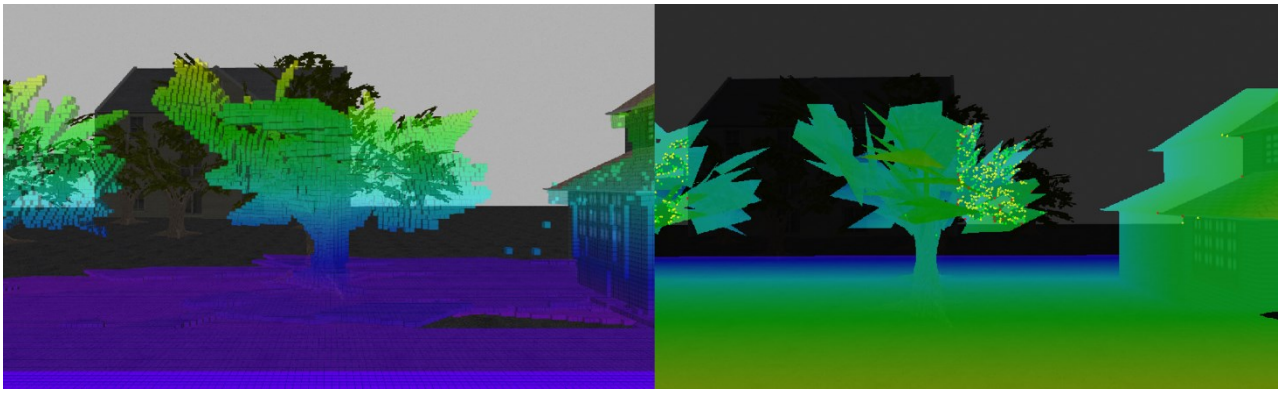

Figure 5.8 Octo-mapping image (left), Odometry Feature extraction image (right) at end point

Figure 5.9 describes the whole map construction after the RTAB-Octo-map algorithm. which shows a very clear and precise view about the surroundings.

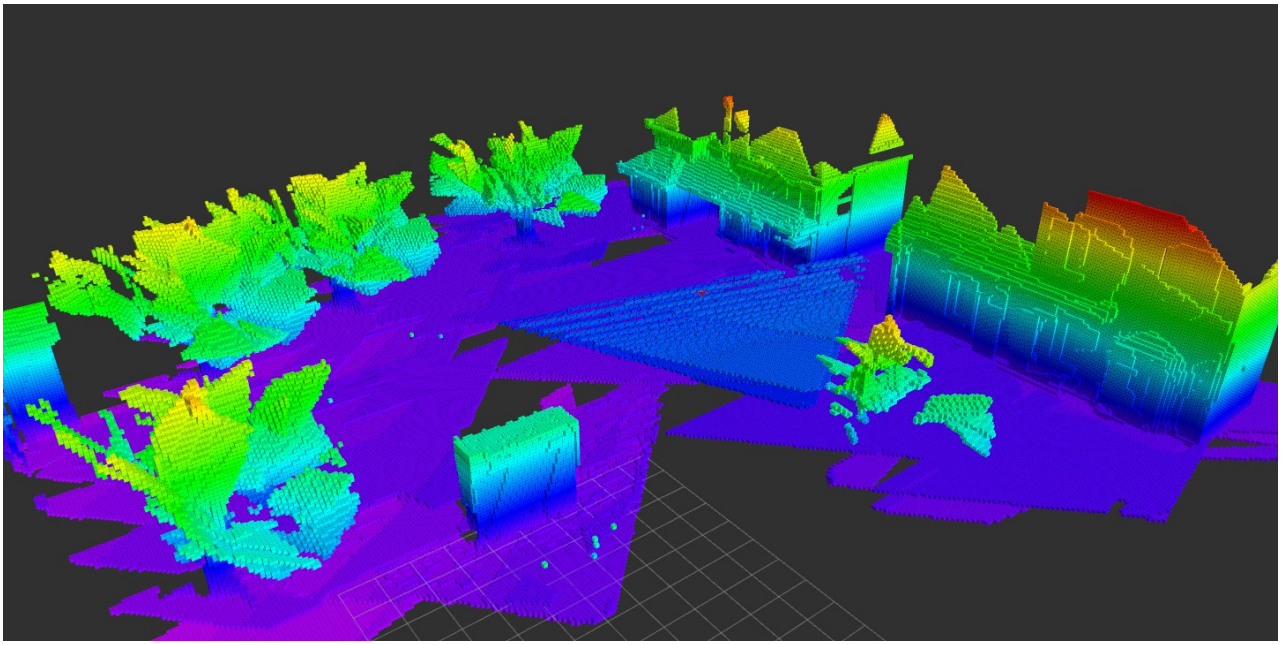

Figure 5.9 The whole view of RTAB-Octo-map world

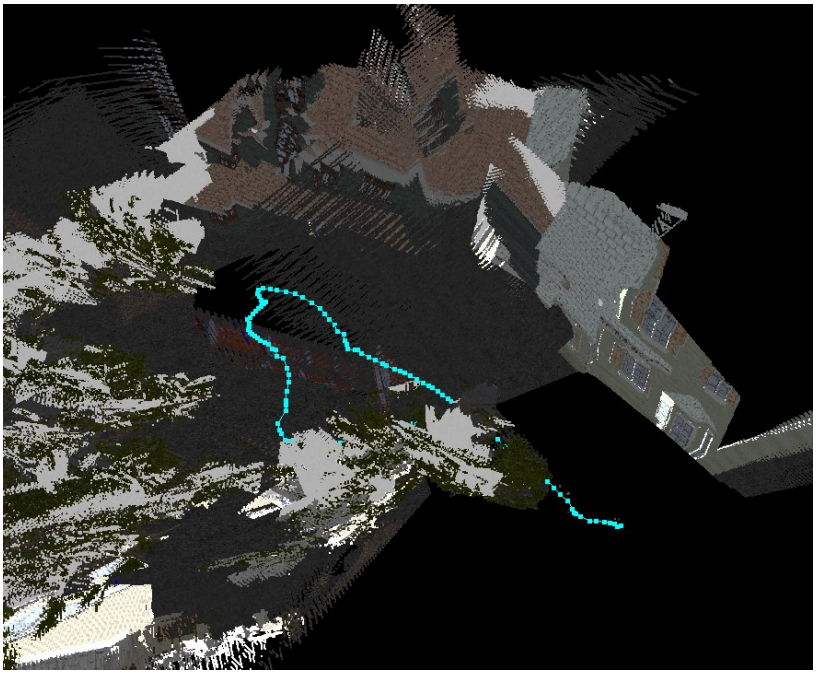

Figure 5.10 The whole view of RTAB-Octo-map world

Figure 5.10 is the image that is combined with the Odometry information based on the points clouds of the image feature. Through the above simulation, we can conclude that when the UAV

maintains a certain flight speed, this algorithm can quickly and effectively sketch the map of its environment and retain Odometry information at the same time, so as to achieve SLAM better.

5.3 Adaptive Kalman Filter

AKF is an efficient autoregressive method. It can estimate the state of the dynamic system in many uncertain situations. It is a powerful and versatile tool to filter out the noise and estimate the states. Before moving forward on to the next step, there are some simple position and velocity prediction simulations to test the AKF Assuming there is a drone with Uniform speed and uniform acceleration respectively, through the prediction of the AKF, the following results are obtained.

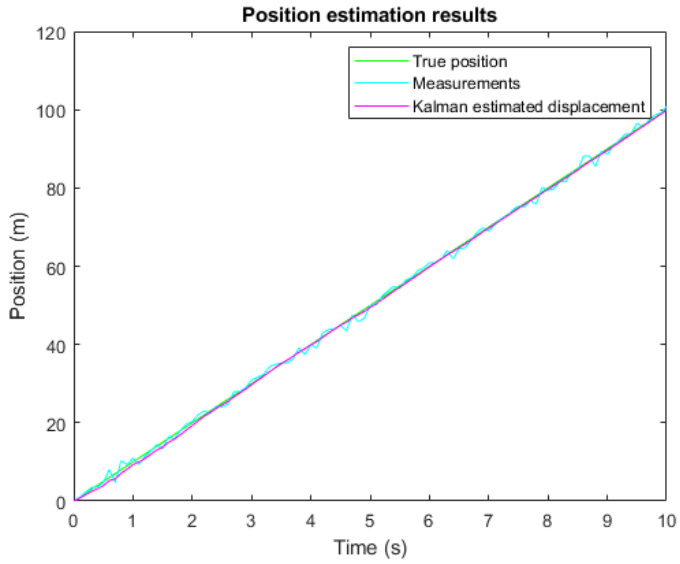

Figure 5.11 The position estimation results uniformly motion

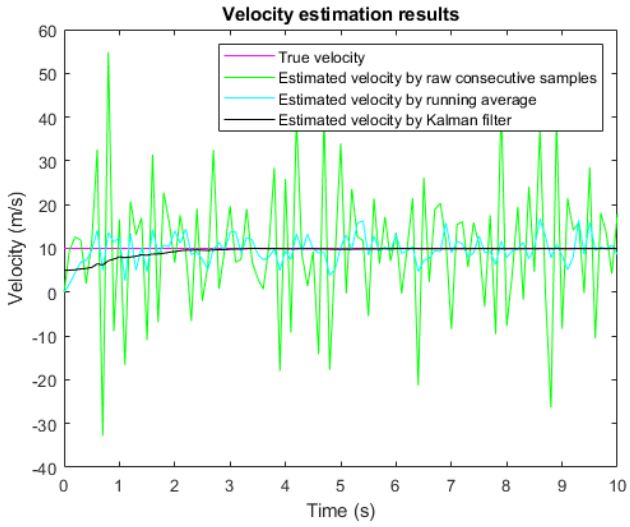

Figure 5.12 The velocity estimation results uniformly motion

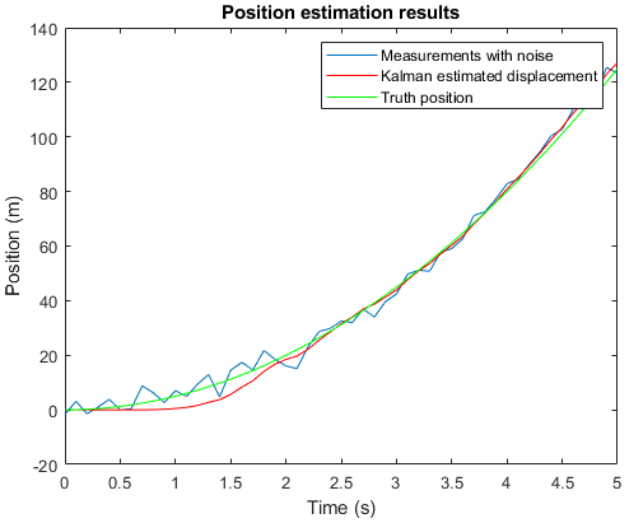

Figure 5.13 The position estimation results in uniformly accelerated motion

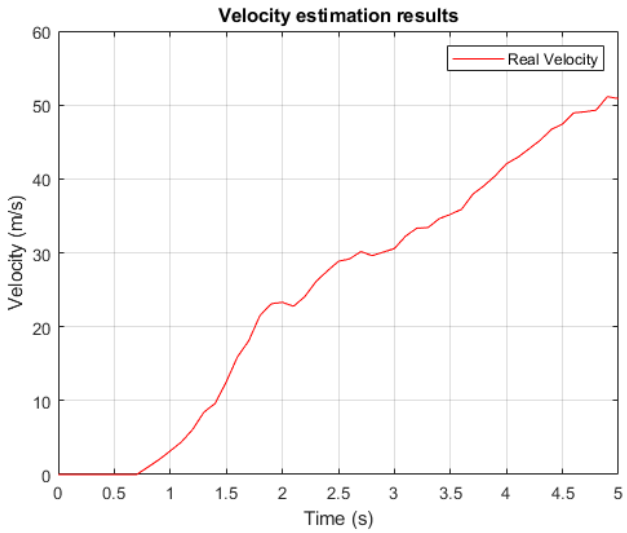

Figure 5.14 The velocity estimation results in uniformly accelerated motion

From the above figures, it can be seen that whether the motion is uniform motion or uniformly accelerated motion, the AKF has some increased deviation at the beginning of prediction. When time goes by, the accuracy of the model improves and the original can be well fitted in the case of noise interference.

In this research, the AKF is implemented to zone the airspace. Moreover, the real states of the intruders will be published from a database platform and by using an AKF to zone the sparse space for their own ship. This stage is concerned with efficient path planning based on a prior known representation of the environment. The AKF is employed to predict the pose of other flying UAVs sharing the same airspace. Those UAVs are termed intruders here and the objective is to minimize a conflict between the multiple UAVs (the subject of the path planning). The estimated position (yellow circle) of each intruder after 15s is shown in Fig. 5.15 and using this data the airspace can be segmented into different zones that exhibit sparse, and dense traffic presence, as illustrated in the figure. According to the density of intruders, two kinds of zones are designated (the red square is the Dense zone and the blue is the sparse area in Fig. 5.15). In the Dense zone, the number of intruders equal or are more than 2. The output of this algorithm, in particular, will also be used for planning path across the environment and zones, as it will be illustrated in Scenarios 3 and 4.

As demonstrated here, the KF formulation allows us to define different zones that will guide global path planning in a way to avoid potentially denser traffic, and hence higher chances of separation.

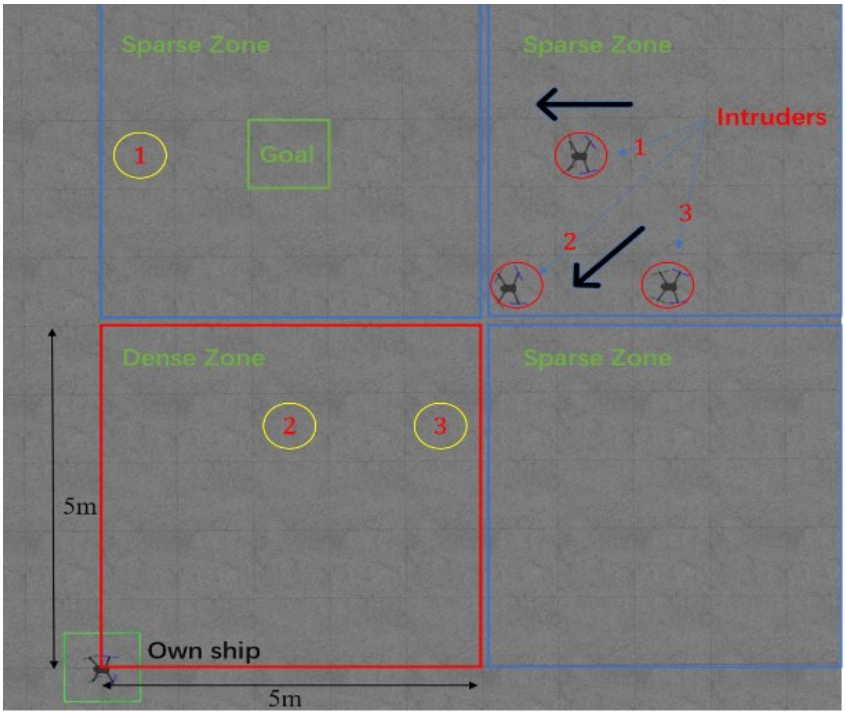

Figure 5.15 Implementing strategy planning through AKF

5.4 Harmonic Potential Field

This section uses the HPF to represent the environment. As explained in Section II. A HPF with Newman boundary condition is developed for the environment with the known location of start and goal points and the obstacle boundaries. The 3-D Laplace equation can be solved with the separation parameter method to generate a continuous field over the designated environment. Following the potential field gradient, a trajectory can be generated from the starting point (high potential field) along the edge of the obstacle to the endpoint (low potential field).

However, calculating a single Harmonic field for the entire environment may pose numerical challenges and complexities. To avoid this, the researchers [33] utilize separate Harmonic functions to represent obstacles individually. In this way, each object can be added or subtracted separately to the environment with a minimum effort. It was demonstrated that by following a direct trajectory towards the goal point while it can generate a connected path and avoid obstacles while following the iso-potential elliptical contours around each obstacle, where a distance threshold to the objects is passed.

Figure 5.16 demonstrates the iso-potential contours in 3-D around each obstacle in the environment and Figure 5.17 illustrates the connected path from a start point to the goal.

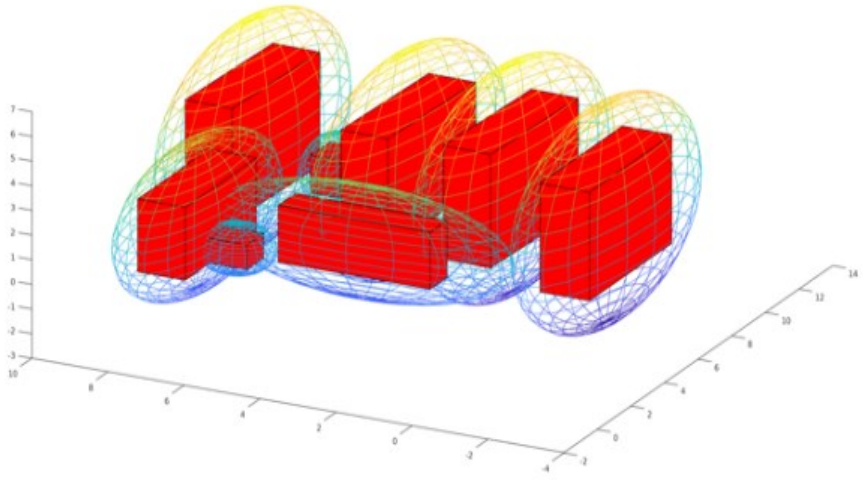

Figure 5.16 The contour of obstacle

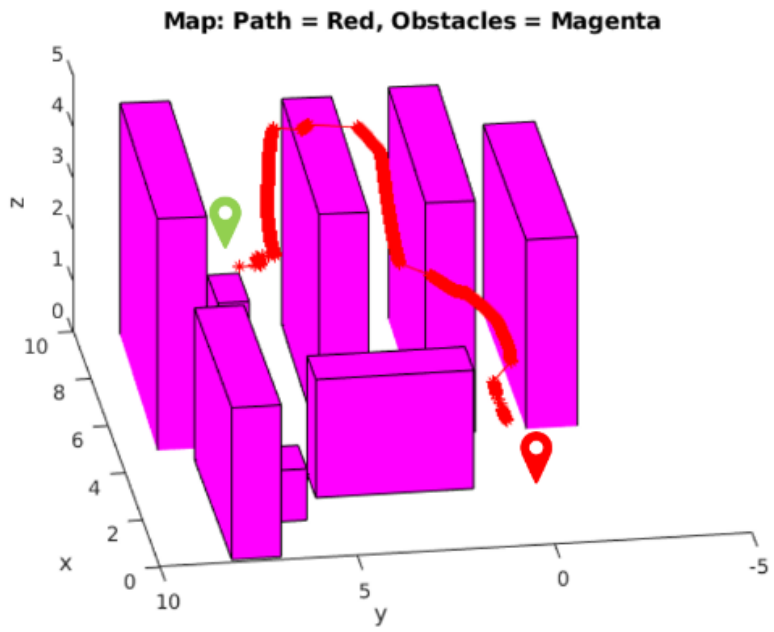

Figure 5.17 The 3D trajectory based on HPF representation in MATLAB

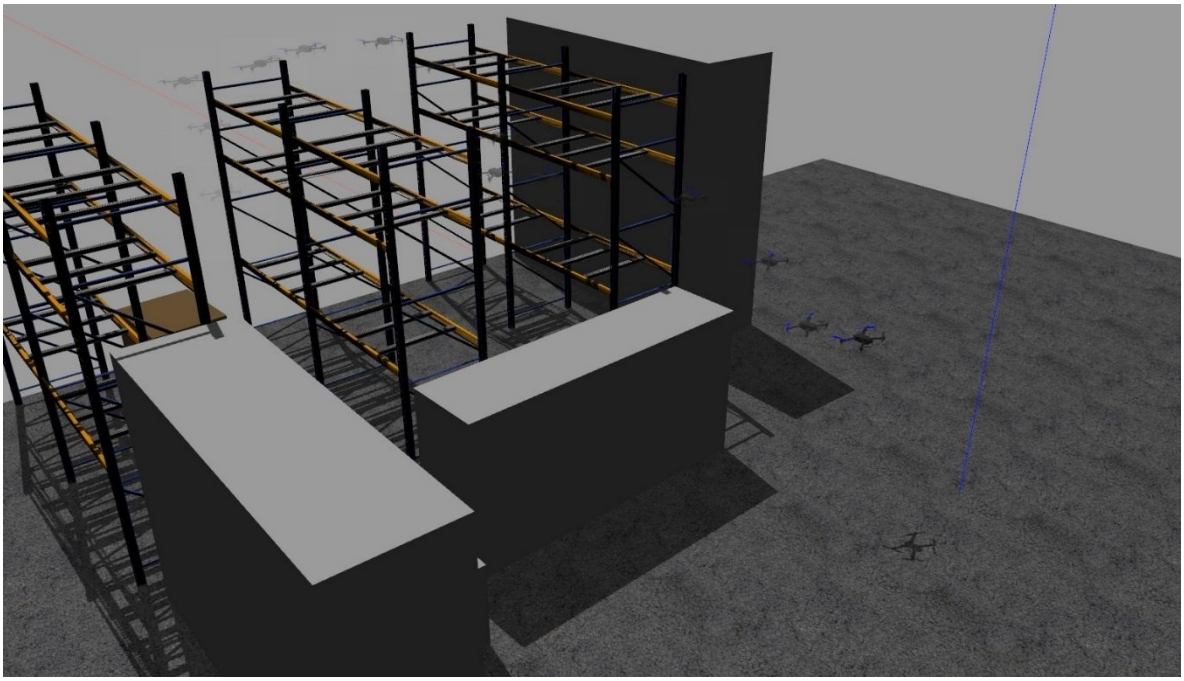

Figure 5.18 The simulation of flight path under HPF in Gazebo

The second case will show the partial paths in each individual point in Figure 5.20, in this case the ability of continuous path generating and planning can be fully illustrated.

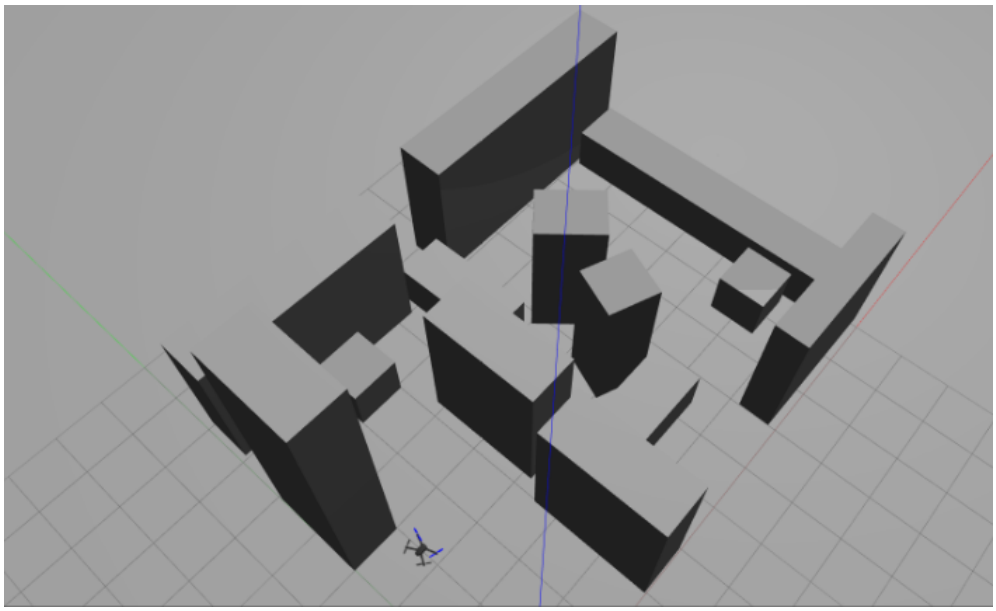

Figure 5.19 The distribution of obstacles in Gazebo

In Figure 5.20, the contour function is used to build the contour of the obstacles. At the same time, ten 3D points are located on the map. In this step, the HPF is implemented in two points. Thus, 9 partial paths can be generated in Figures 5.20-5.23. The simulation of the flight path under HPF in Gazebo is shown in Figure 5.24.

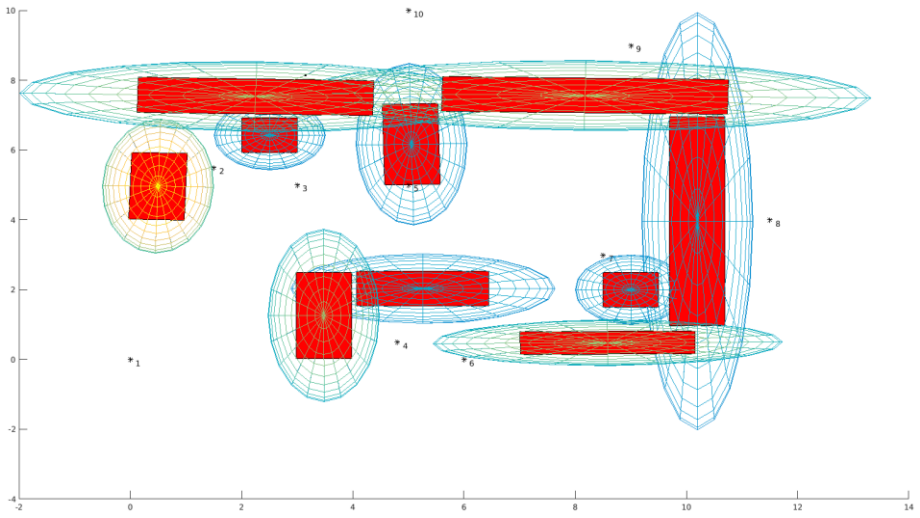

Figure 5.20 The contour map and sample point in MATLAB

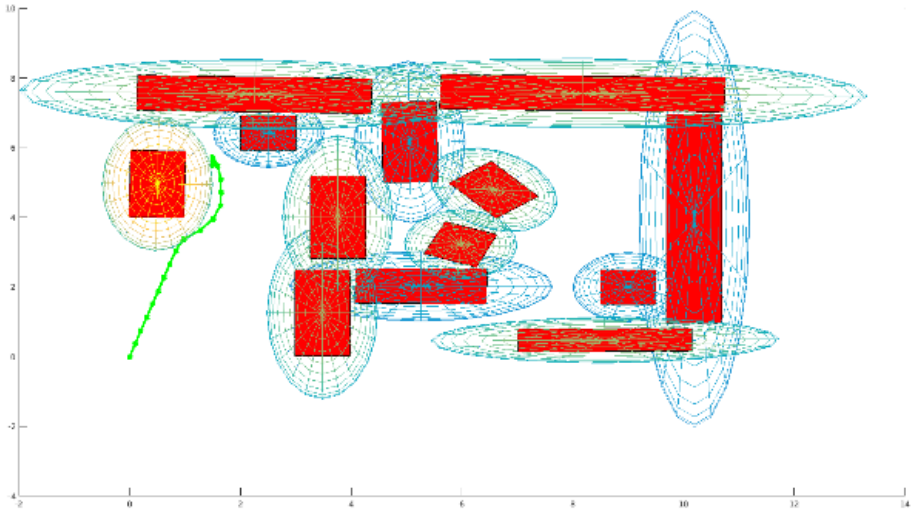

Figure 5.21 The path of first two points after applying HPF in MATLAB

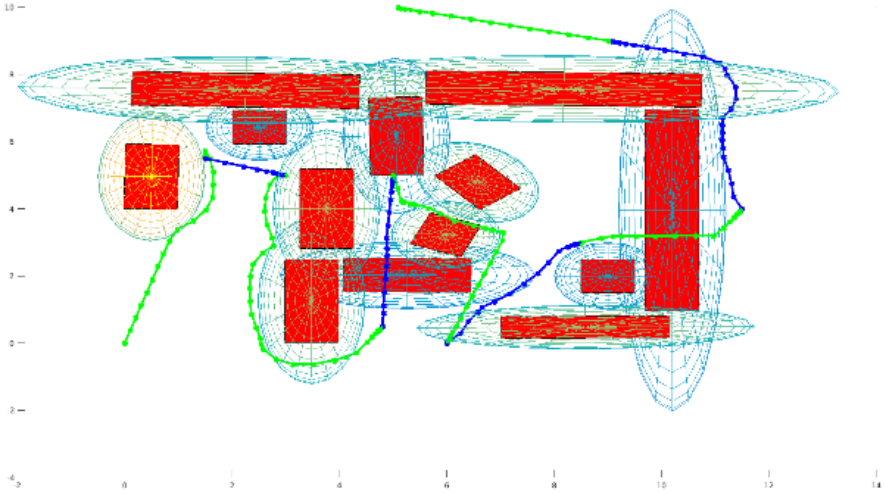

Figure 5.22 The full path after applying HPF (2D view) in MATLAB

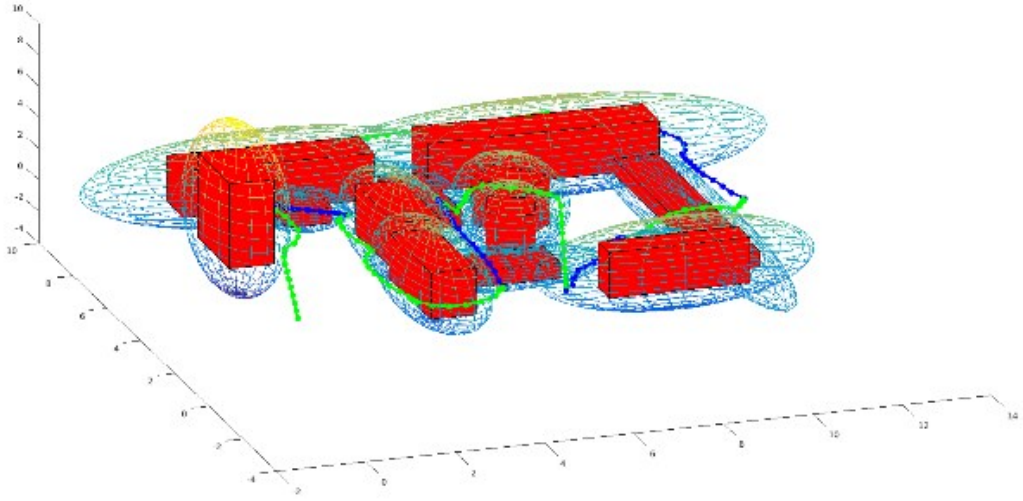

Figure 5.23 The full path after applying HPF (3D view) in MATLAB

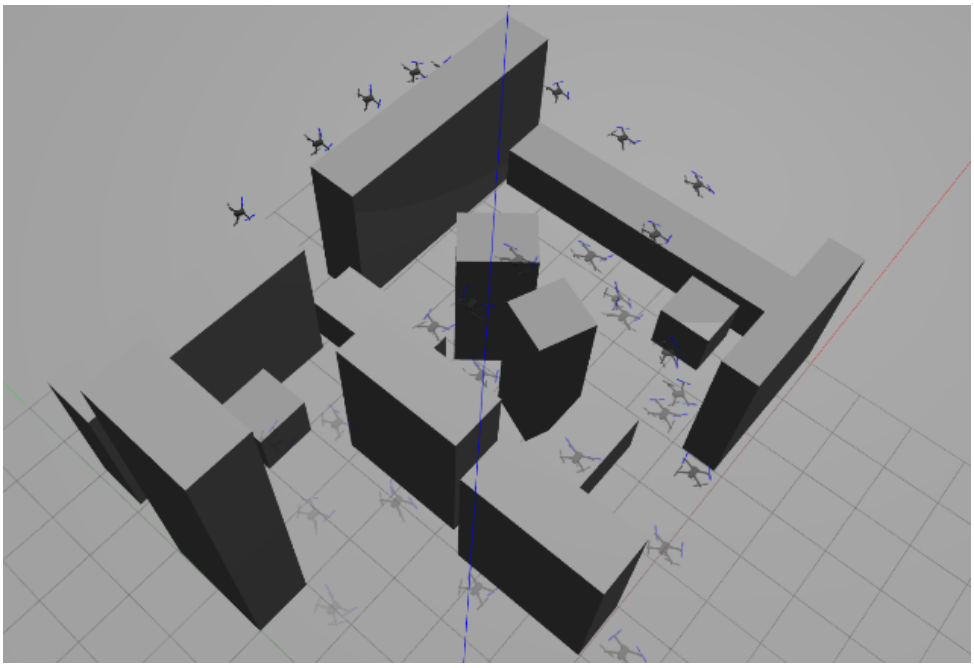

Figure 5.24 The simulation of flight path under HPF in Gazebo

5.5 MDP

Real-Time avoidance is carried out based on MDP. Scenarios 1 and 2 are the simulations of known environments and obstacles. MDP can perform obstacle avoidance for unknown intruders which can let the policy to decide which action should be taken when a new obstacle is encountered and sensed. the camera is used to locate the relevant position between own ship and the intruders. For the simulations, as explained in Chapter 4, 18 states (directions) are considered. And also, the safety distance is 1.2m. To simplify the simulation, the center of one sparse area is selected as the target point randomly and the known intruder No.1 in Figure 5.25 is set as an unknown intruder in this case. In Figure 5.25, our ship (green box) reaches the center of the sparse area after the initial planning in Figure 5.26. At the same time, Intruder No.1 is approaching the own ship from east to west. When the RGB-D camera detects that the distance between the intruder and own ship is less than, the MDP policy will give a command, which represents moving X=Y=Z=1meter position relative to the own ship coordinate. Figure 5.25 shows that the own ship is heading Southeast to avoid the intruder No.1 and then returns to the goal point. Figure 5.26 represents MDP from another perspective in the 3D environment. As demonstrated in this stage, MDP is suitable for obstacle avoidance using an onboard sensor such as the RGB-D camera to obtain the position information of a dynamic unknown intruder relative to the own ship.

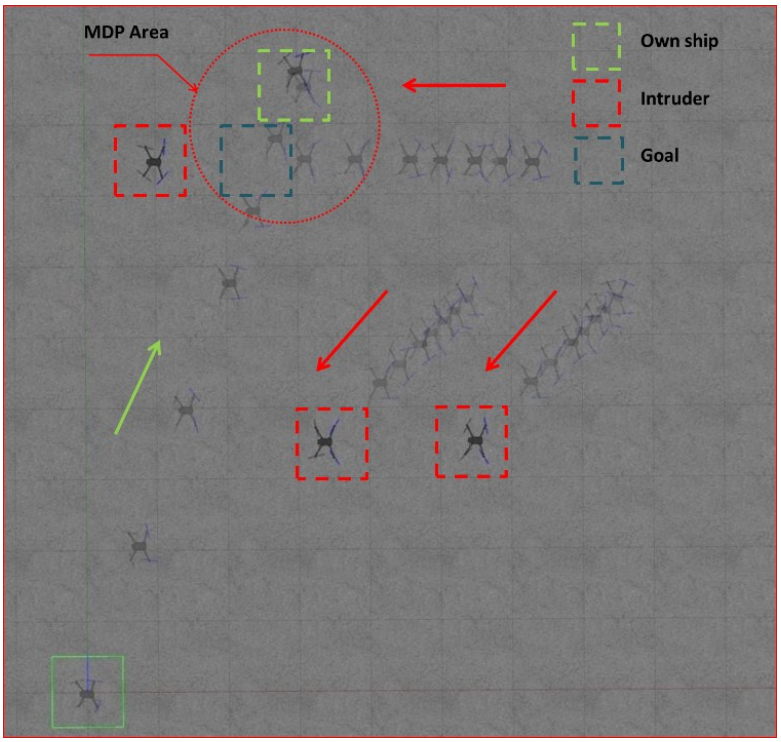

Figure 5.25 The simulation of MDP scenario in GAZEBO

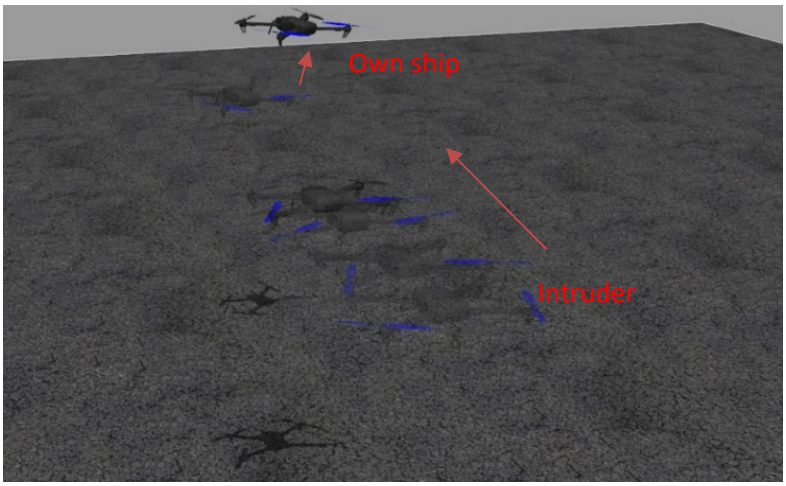

Figure 5.26 The moving state simulation of own ship in GAZEBO

In this thesis, a comparison group between MDP and APF will be shown in the following table and figures. In this case, the simulations will be tested by 12 groups respectively, among which 6 groups will be tested for a single intruder, 3 groups will be tested for two intruders, and the final 3

groups will be tested for three intruders. The safety distance can be defined as 1.5m. The intruder's location information is random in the 3D coordinate. The intruders will head to the own ship at a different speed. The simulations test the decision time and execution time of both MDP and AFP. Table 5.6 illustrates the result when there is one intruder towards to own ship.

Table 5.6 MDP and APF response times in the case of an intruder

Intruder = 1	*Start Point*	*Execution Time*	*Decision Time*	*Result*
MDP	**(3,0,3)**	**0.1275s**	**0.1275s**	No collision
	(4,0,1)	**0.2057s**	**0.2057s**	No collision
	(3,3,3)	**0.1865s**	**0.1865s**	No collision
	(4,3,5)	**0.2358s**	**0.2358s**	No collision
	(-5,4,3)	**0.2854s**	**0.2854s**	No collision
	(-7,6,5)	**0.2965s**	**0.2965s**	No collision
APF	**(3,0,3)**	**0.3328s**	**0.8759s**	No collision
	(4,0,1)	**0.3564s**	**1.0564s**	No collision
	(3,3,3)	**0.3512s**	**1.1258s**	No collision
	(4,3,5)	**0.3421s**	**1.3544s**	No collision
	(-5,4,3)	**0.3358s**	**1.2593s**	No collision
	(-7,6,5)	**0.3258s**	**1.1456s**	No collision

Figure 5.27 MDP and APF response times in the case of an intruder

From Table 5.6 and the Figure 5.27, the collision avoidance speed of MDP is about 3 times faster than APF especially in decision time. In this case, there is no any collision happened when

the number of intruders is equal to one. The next table shows the outcome under the situation of two intruders.

Table 5.7 MDP and APF response times in the case of two intruders

Intruder = 2	*Start Points*	*Execution Time*	*Decision Time*	*Result*
MDP	**(3,0,3) (4,0,1)**	**0.2157s**	**0.2517s**	No collision
	(5, -3,4) (6,2,2)	**0.2578s**	**0.2859s**	No collision
	(-5, -3,4) (6,4,2)	**0.2474s**	**0.2958s**	No collision
APF	**(3,0,3) (4,0,1)**	**0.3247s**	**1.5894s**	No collision
	(5, -3,4) (6,2,2)	**0.3689s**	**1.8642s**	collision
	(-5, -3,4) (6,4,2)	**0.3358s**	**1.6841s**	No collision

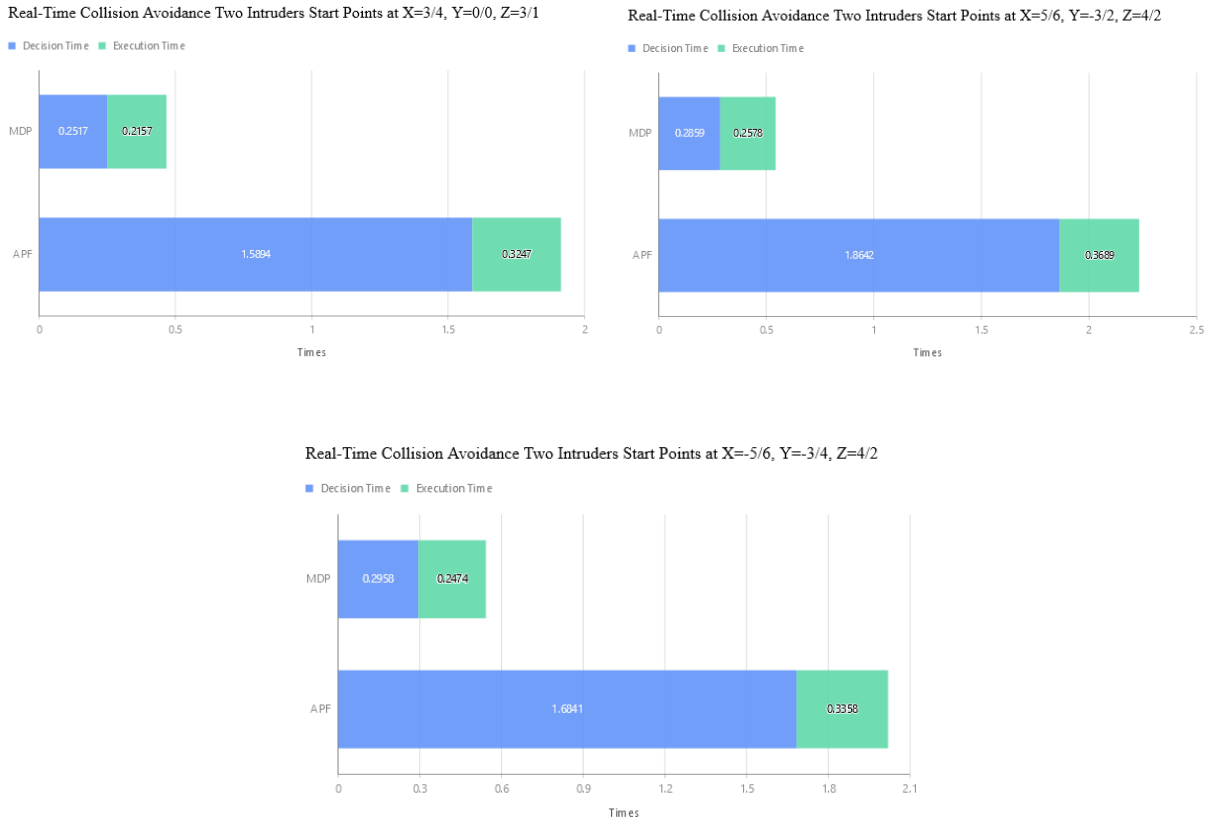

Figure 5.28 MDP and APF response times in the case of two intruders

The Figure 5.28 shows that the speed of MDP still keeps faster than APF, which fluctuates around 0.5s, however, there is one case that the collision avoidance happened in the result, also the time consumes around 2s.

Table 5.8 MDP and APF response times in the case of three intruders

Intruder = 3	*Start Points*	*Execution Time*	*Decision Time*	*Result*
MDP	**(3,0,3) (4,0,1) (7,6,2)**	**0.3158s**	**0.3587s**	No collision
	(5, -3,4) (6,2,2) (5,0,7)	**0.2878s**	**0.3721s**	No collision
	(-5, -4,4)(6,1,2) (8,1,4)	**0.2157s**	**0.3025s**	No collision
APF	**(3,0,3) (4,0,1) (7,6,2)**	**0.3687s**	**1.9265s**	No collision
	(5, -3,4) (6,2,2) (5,0,7)	**0.3457s**	**2.1054s**	collision
	(-5, -4,4)(6,1,2) (8,1,4)	**0.3358s**	**2.0258s**	collision

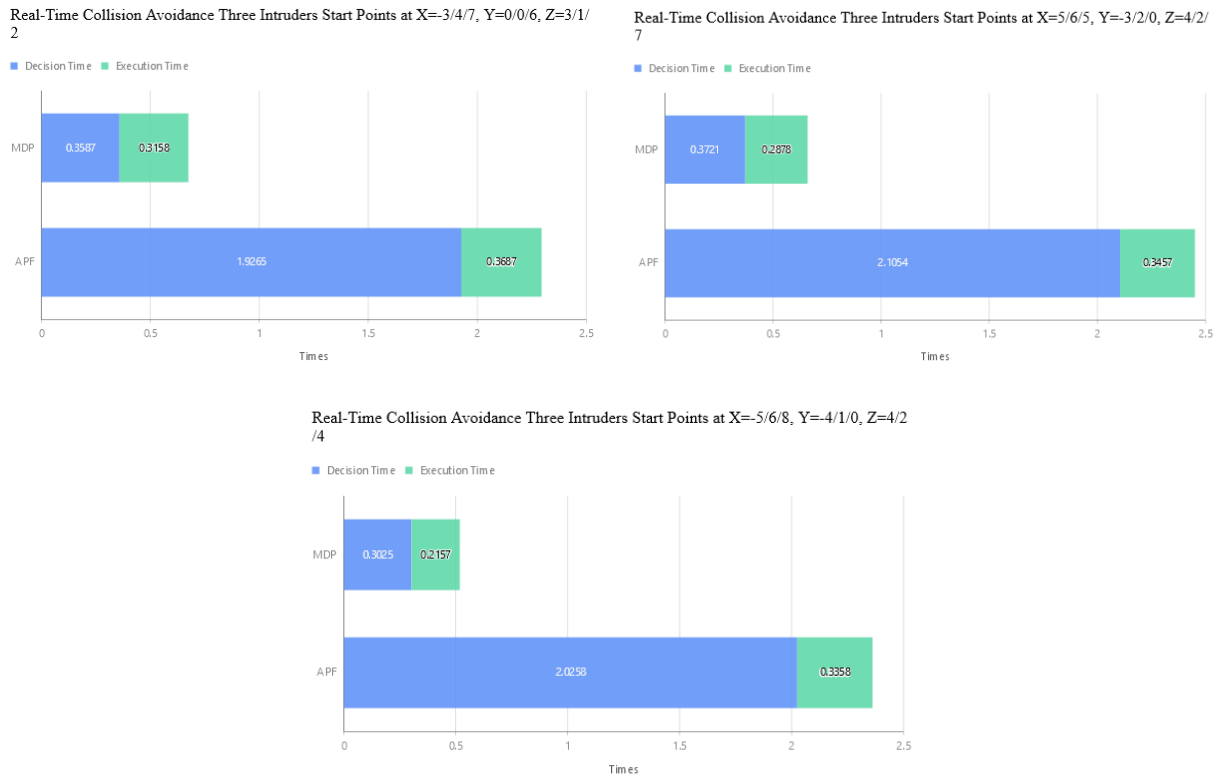

Figure 5.29 MDP and APF response times in the case of three intruders

When the APF is faced with three intruders or more intruders, APF performs not well. On the one hand, the computing speed is reduced, and on the other hand, there is no time to dodge nearby intruders, thus causing collisions.

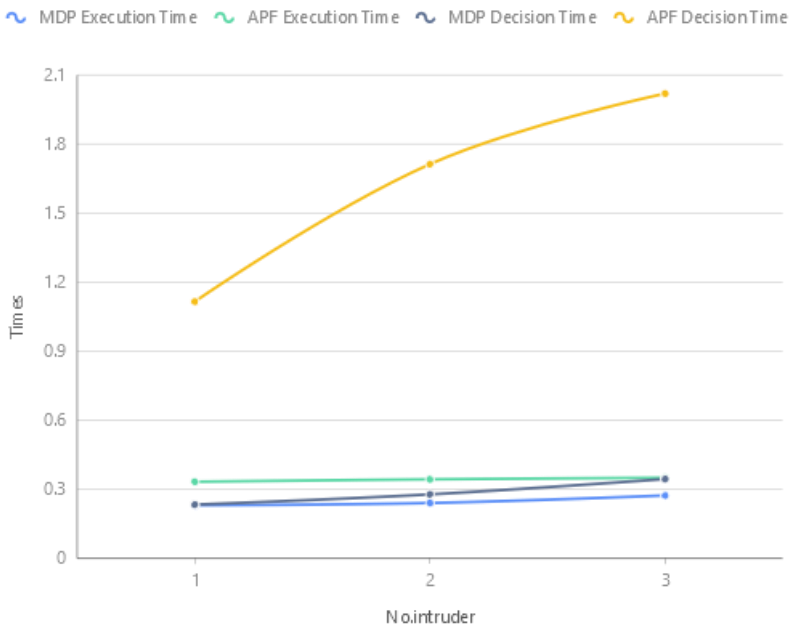

Figure 5.30 The average ET and DT of MDP and APF when increases with the number of intruders

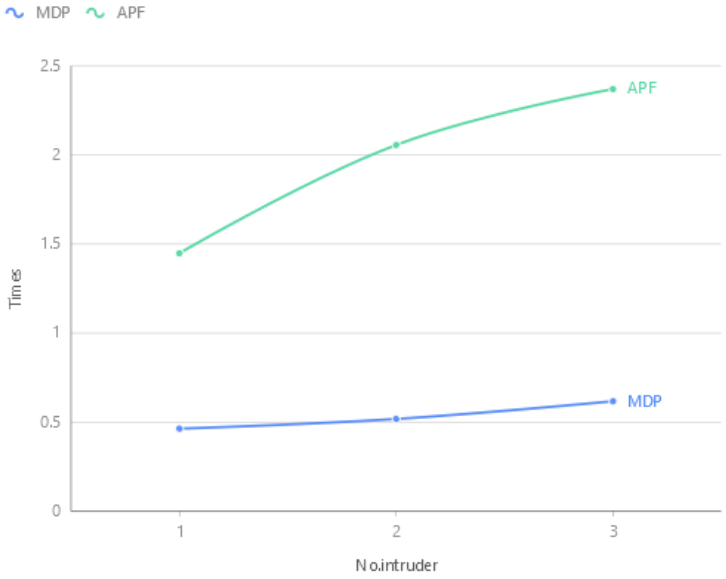

Figure 5.31 The average ET and DT of reaction time when increases with the number of intruders

From the above simulation result, MDP responds two to three times faster than APF, and the obstacle avoidance speed of MDP changes slightly as the number of invaders increases without collisions.

5.6 Integrated simulation in urban area

This simulation utilizes the combination of avoidance and path planning system in urban area which can be divided into 3 steps.

Step 1: Adaptive Kalman Filter estimation

Citysim [59] GAZEBO model and Iris [60] model are setup here for this simulation. The yellow circle represents own ship and the red are other UAVs in the area (known as intruders). According to Figure 5.32, AKF is used to estimate the position of the intruder. As shown in Figure 5.33, the urban streets are divided into two types. The blue block is Sparse zone, and the red one is Dense zone.

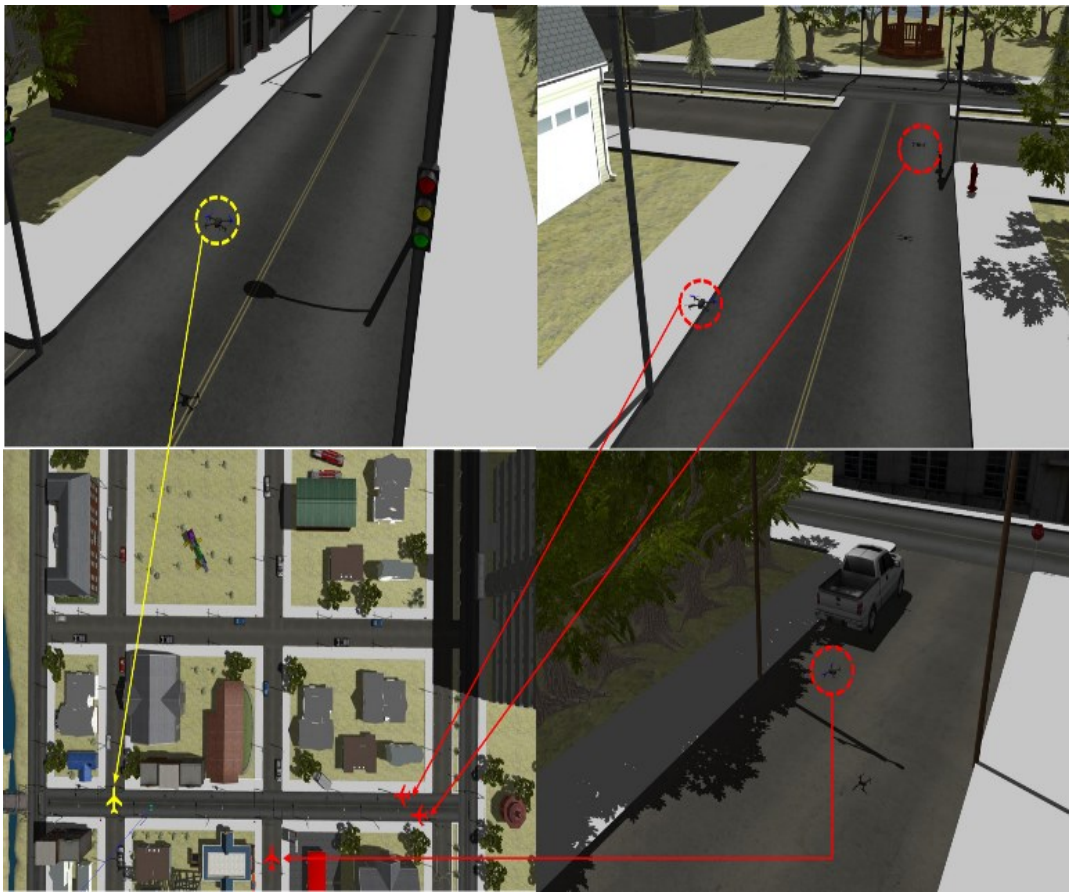

Figure 5.32 The distribution of known intruders in the urban map

Figure 5.33 The distribution of sparse and dense zones

Step 2: Global path planning based on HPF

A point is chosen as goal (yellow point in Figure 5.38) in the urban map. Then the method referring to Scenario 2 makes global path planning by recognizing the Dense street (red zone) as the same with the other obstacle (e.g. house).

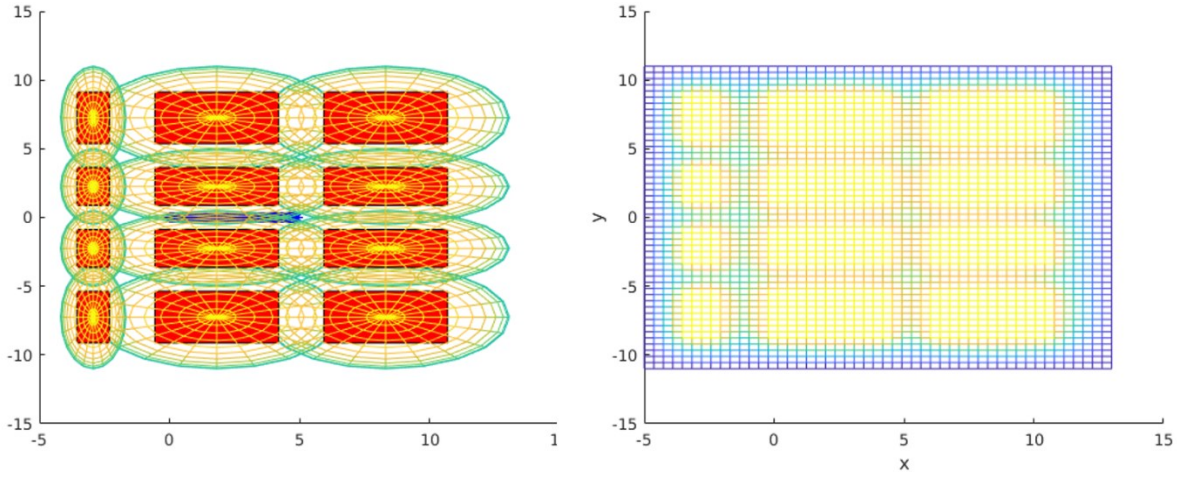

Figure 5.34 (left) The contour construction of urban city in MATLAB, (right) The boundary field from 0 to 1 (top view)

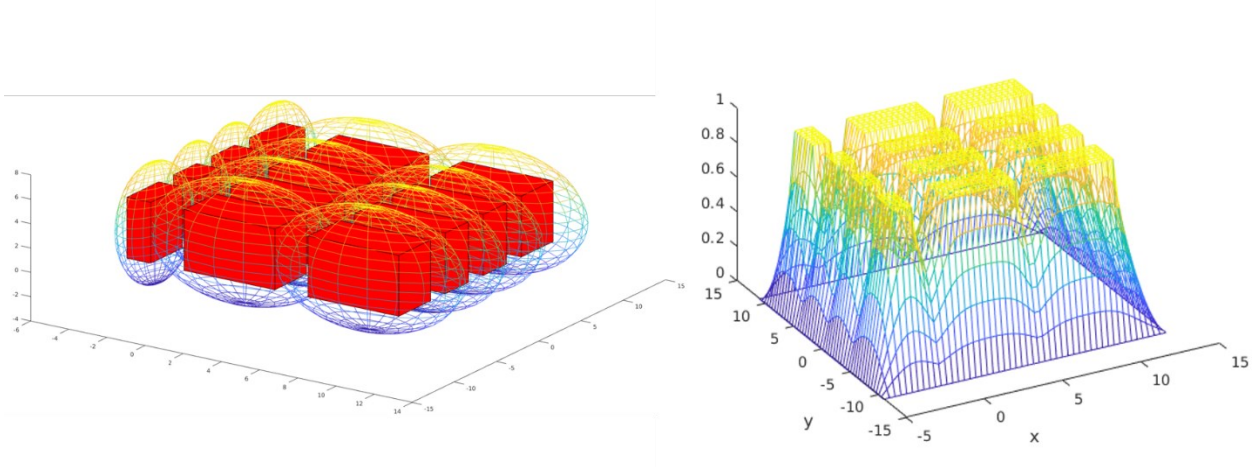

Figure 5.35 (left) The contour construction of urban city in MATLAB, (right) The boundary field from 0 to 1 (3D view)

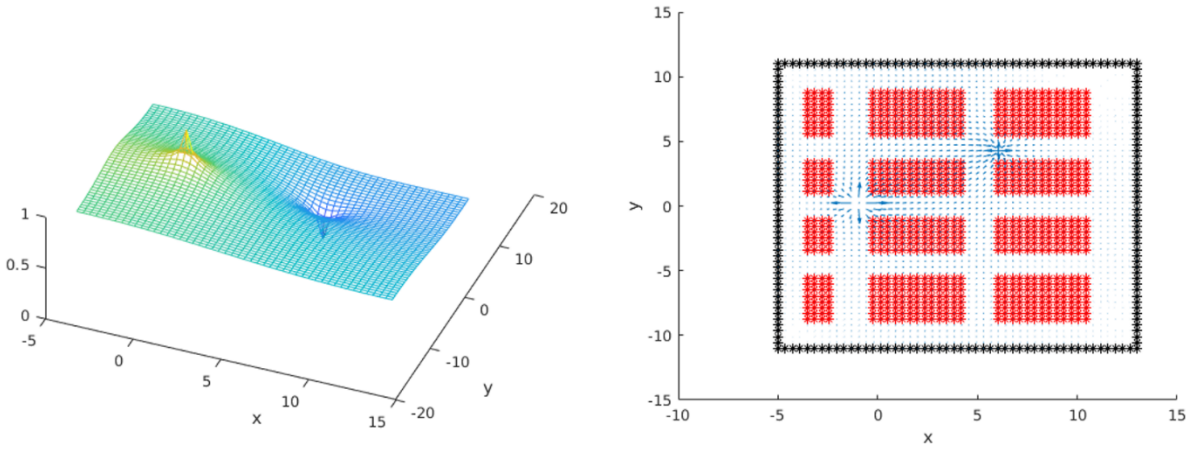

Figure 5.36 Setting up the potential field of start point and end point

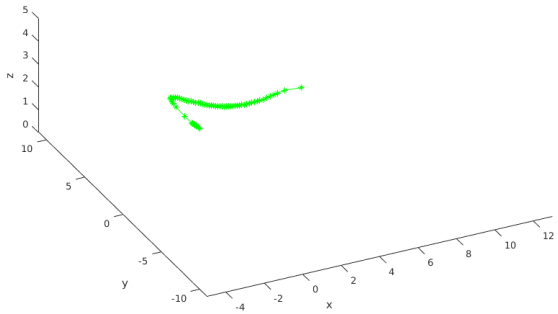

Figure 5.37 Path generation in 3D

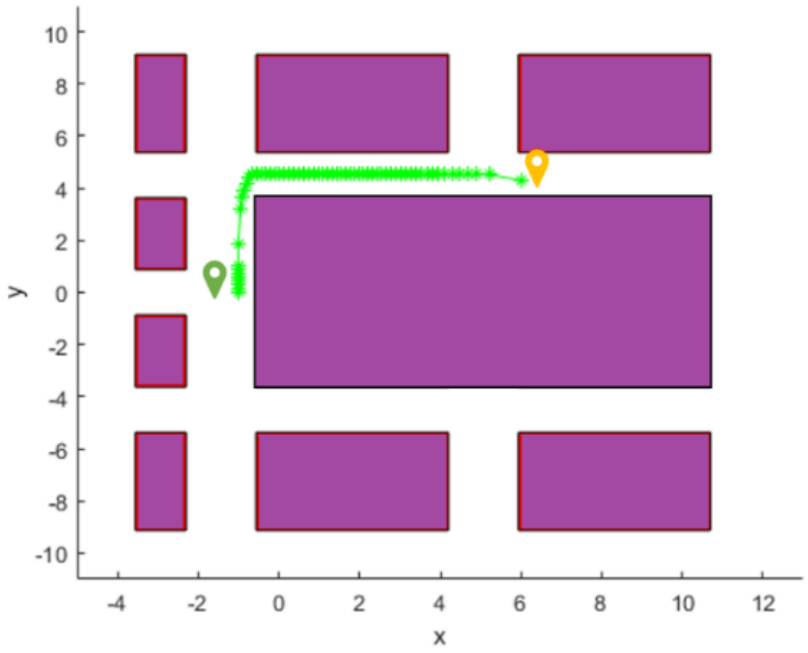

Figure 5.38 The trajectory based on HPF

Step 3: Obstacle avoidance based on MDP

For this step, an unknown intruder has been added in the figure which is located in center of intersection (to simplify the simulation, the intruder is in take-off state and stays at $\boldsymbol{X} = 45\boldsymbol{m}, \boldsymbol{Y} = 45\boldsymbol{m}, \boldsymbol{Z} = 2\boldsymbol{m}$ which is shown as purple plane in Figure 5.39).

Figure 5.39 Global path planning and strategy planning

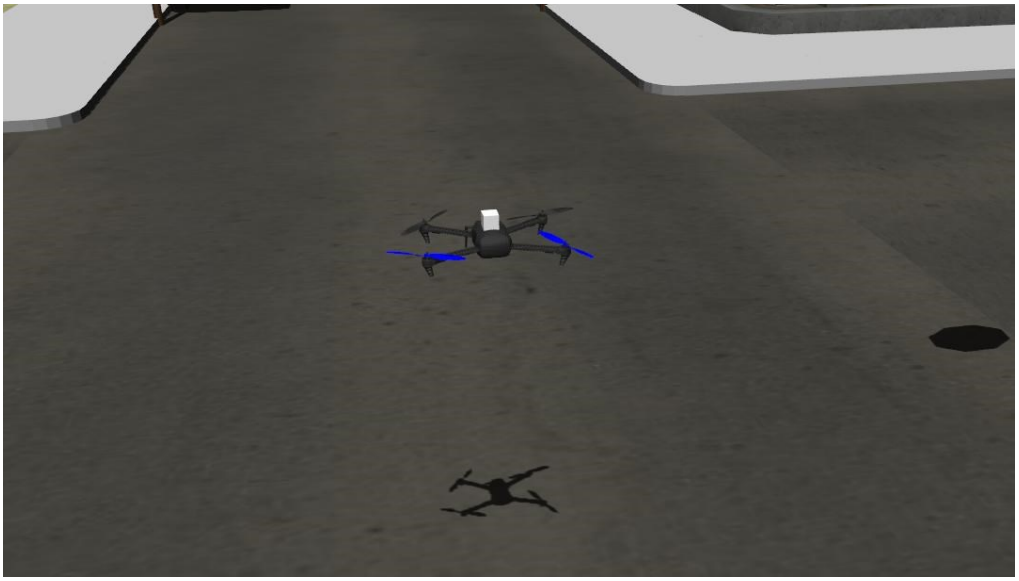

Figure 5.40 Iris model with RGB-D camera

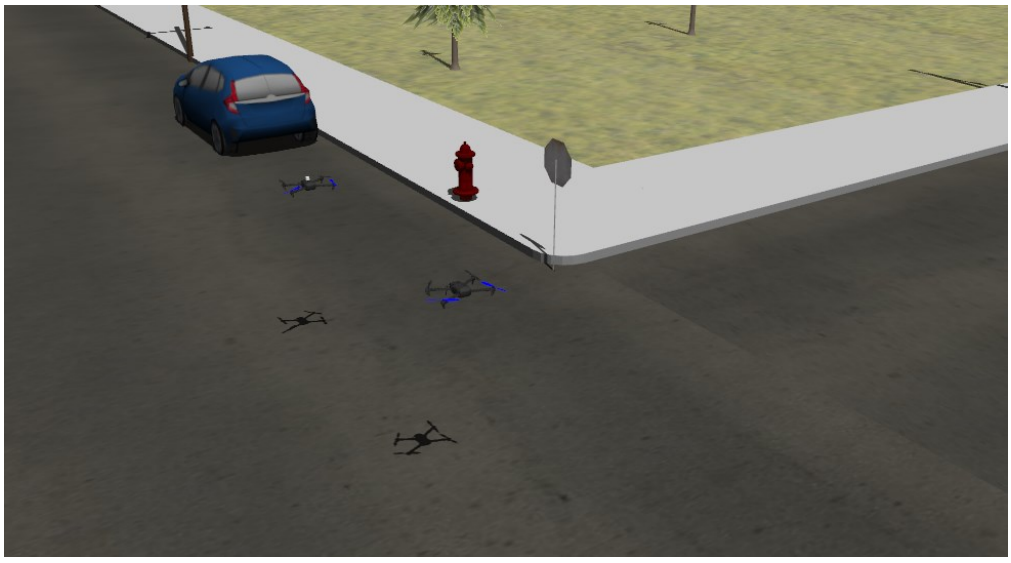

Figure 5.41 Own ship detects the unknown intruder

Figure 5.42 Depth map and object tracking from own ship

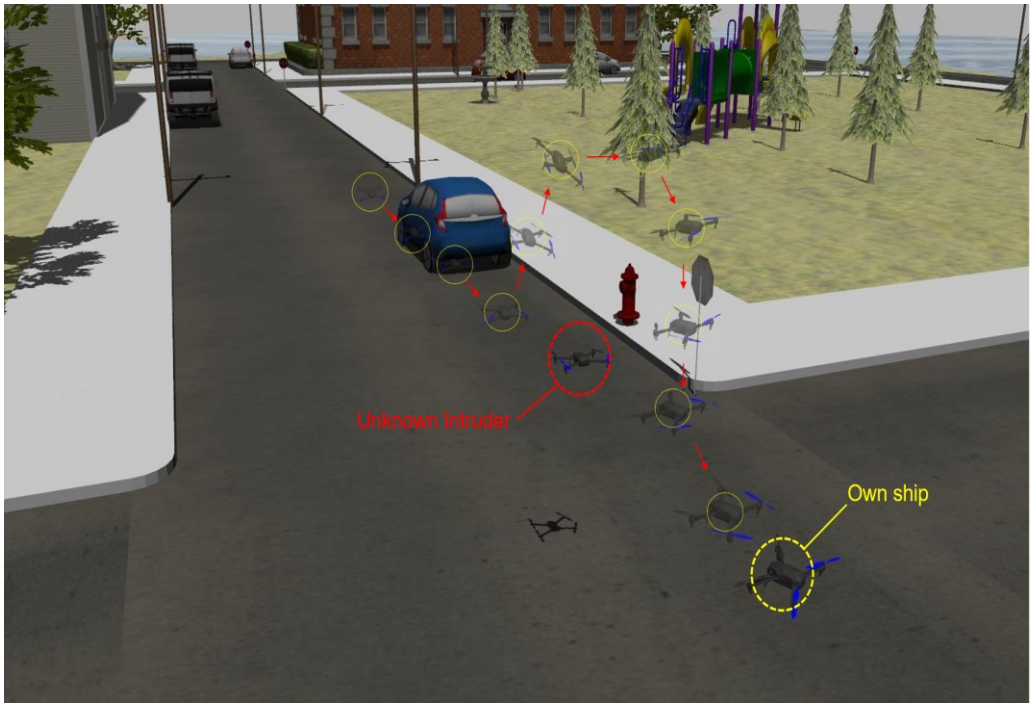

Figure 5.43 Real-Time avoidance in Urban

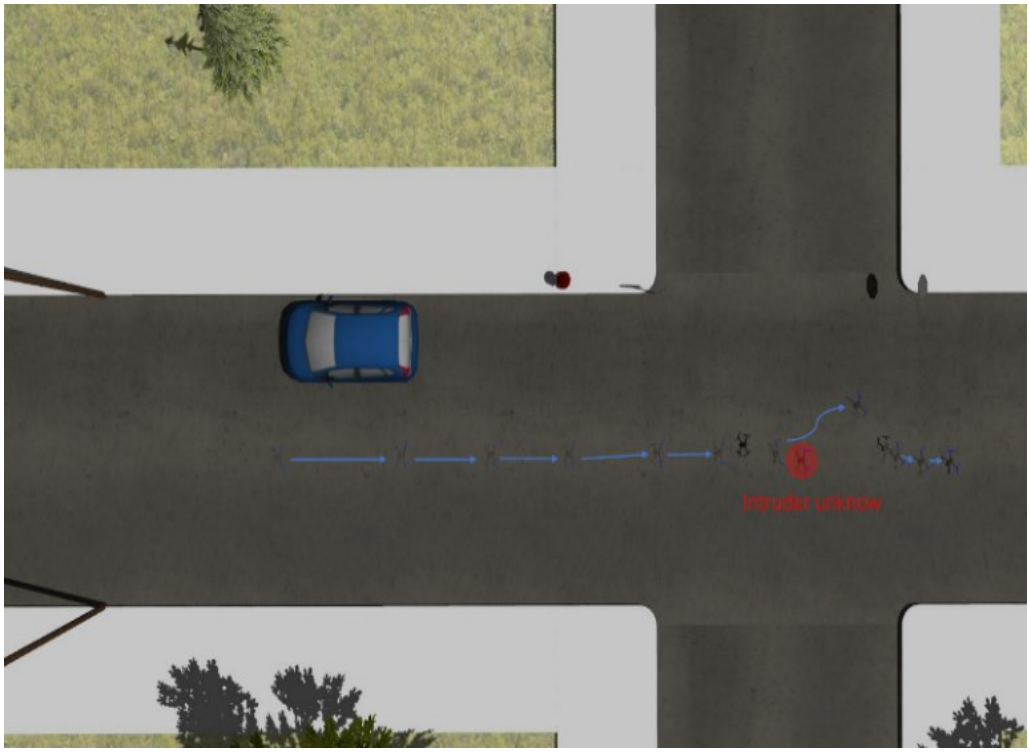

Figure 5.44 Real-Time avoidance in Urban (top view)

5.7 Discussion

As shown in the simulation above, in the first scenario by using the AKF, the map is well divided into different areas, which greatly helps the drone to identify the busyness of the area. The second scenario shows that HPF can be very flexible to generate trajectories, and it is easy to implement the underlying control. Also, the MDP scenario is combined with the sensor to avoid dynamic obstacles. In the final scenario, the first step we use the AKF algorithm to predict the location of the city where the known intruder is located. Next, we used the HPF algorithm in the second step to model the entire block and make global path planning. Finally, we used the Markov decision algorithm to avoid obstacles in real-time for unknown intruders.

Chapter 6 Conclusions and future works

6.1 Conclusions

In this dissertation, several efficient-critical issues on reliable navigation of UAV are well studied, which include:

- A comprehensive literature review on UAVs as well as their global path planning and real-time collision avoidance approaches are provided.
- AKF is proposed to sperate the sparse and dense zone for safe traffic regulations. Before taking off, the proposed AKF can zone the accessible areas according to predicting the number of intruders after several seconds. Compared with the traditional Kalman filter, AKP could update Prediction Covariance and Measurement Covariance adaptively, which could make the prediction more precise than the traditional method. The computation is running in the data platform. Once the GPS data from other intruders have been collected, the data platform will implement the AKF to zone the map and send the map to the global planner. The proposed scheme is validated under different faulty scenarios. The simulation results show that the proposed AKF achieves good estimation performance in zoning airspace.
- Global path planning based on a 3D HPF is proposed to obtain a goal-attaining path that can combine with external control factors. HPF solves the local minimum problem that exists in the traditional Potential field method. Meanwhile, it can realize to build the potential fields artificially and combine with the Newman boundary condition. Moreover, the method in this thesis implements the relation between P and G field to achieve HPF in 3D. The simulation results from a quadrotor demonstrate the effectiveness and applicability of the proposed global path strategy.
- A real-time collision avoidance strategy based on MDP combined with sensor fusion is proposed to accommodate the unknown intruders and danger. With the help of the Realsense camera, the quadrotor could obtain the relative position with the unknown intruders which makes full use of the reward system to get the max optimal reward action as the principle of the decision and also can process the data from sensors and decide the best action to make

collision avoidance. MDP is utilized as a decision-making tool to determine the next step for the UAS to avoid the obstacle. The demonstrated simulation shows the effectiveness and superiority of the MDP in local obstacle avoidance.

- As demonstrated in the simulations, the algorithm is capable of safely navigating UAS platform in a dynamic environment while other UAVs are also sharing the airspace.

6.2 Future works

Following the current research in this dissertation, the following future directions are outlined:

- Due to the COVID-19 and the limitation of calculation in the Raspberry board, the real flight test has not been implemented. The work can be developed by using NVIDIA TX2 powerful board.
- The proposed AKP strategy in this dissertation has better performance in linear and uniformed movement than in nonlinear movement. To adapt the complex curves, the nonlinear predict model should be developed in the future works.
- The proposed AKP strategy in this dissertation has better performance in linear and uniformed movement than in nonlinear movement. To adapt the complex curves, the nonlinear predict model should be developed in the future works.
- The speed of object detection and tracking which are implemented on Raspberry PI-Board is very essential to do the real flight test, but the model that is trained in this thesis only have 5-10 FPS value in the image. This can be improved by changing the convolutional frame to simplify image processing.

Bibliography

[1] Shamiyeh M., Rothfeld R. and Hornung M. (Sep 14, 2018). "A Performance Benchmark of Recent Personal Air Vehicle Concepts for Urban Air Mobility". icas.org. Retrieved Aug 20, 2019.

[2] Latombe, Claude J., (1991) Robot Motion Planning The Springer International Series in Engineering and Computer Science.

[3] Nakamura H., Harada K. and Oura Y., (2018) "UTM concept demonstrations in Fukushima; overview of demonstration and lesson learnt for operation of multiple UAS in the same airspace," 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, pp.222-228, Doi: 10.1109/ICUAS.2018.8453425.

[4] Albaker, B.M., Rahim, N.A., (Dec. 2009) "A survey of collision avoidance approaches for unmanned aerial vehicles," in Technical Postgraduates (TECHPOS), 2009 Int. Conf. for, vol., no., pp.1-7, 14-15.

[5] Gupta D.K.，Jaiswal A.K., (May 2013) "Path Planning with Real Time Obstacle Avoidance" Int, J. of Computer Applications (0975 –8887) Volume 70–No.5.

[6] Li A., Luo P., (2012) "Introduction to A* From Amit's Thoughts on Pathfinding". theory. stanford.edu.

[7] Xue Q. and Chen Y. (1995) "Determining the path search graph and finding a collision-free path by the modified A* algorithm for a 5-link closed chain Applied Artificial Intelligence", vol. 92.

[8] Lavalle, S.M. (1998). "Rapidly-exploring random trees: A new tool for path planning", Computer Science Dept, Iowa State University, Tech. Rep. TR: 98–11. Retrieved 2008-06-30. S.

[9] Kala R., Warwick K., (Sep. 2011)" Planning of Multiple Autonomous Vehicles using RRT", IEEE Int'l Conf. on Cybernetic Intelligent Systems (CIS), pp.20-25.

[10] Dale L., Amato N. (2001) "Probabilistic roadmaps – Putting it all together", Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1940–1947.

[11] Karaman, S. and Frazzoli, E. (2011) "Sampling-based Algorithms for Optimal Motion Planning", Int. Journal of Robotics Research, vol. 30, no. 7, pp. 846–894.

[12] Garcia, I., How, J. P. (2005) "Improving the efficiency of Rapidly- exploring Random Trees Using a Potential Function Planner", Proc. of 44th IEEE Conf. on Decision and Control, and the European Control Conference, pp. 7965–7970.

[13] Khanmohammadi S., Mahdizadeh A. (2008) "Density Avoided Sampling: An Intelligent Sampling Technique for Rapidly-Exploring Random Trees", Eight Int. Conf. on Hybrid Intelligent Systems, HIS, pp.672–677.

[14] Kuffner, J., LaValle, S. M. (Apr. 2000) "An efficient approach to single-query path planning", in Proc. of IEEE Intl. Conf. on Robotics and Automation, pp. 995–1001.

[15] Lydia E.K., Petr S., Jean-Claude L., Mark H.O. (Aug. 1996) "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE Trans. on Robotics and Automation, vol. 12, pp. 566–580.

[16] Hsu, D., Latombe, J.-C., and Motwani, R. (1999) "Path planning in expansive configuration spaces", Int. J. of Computational Geometry and Applications, vol. 9, no. 4/5, pp. 495–512.

[17] LaValle, S. M. and Kuffner, J. J. (2001) "Randomized kinodynamic planning" Intl. J. of Robotics Research, vol. 17, no. 5, pp. 378–400.

[18] Şucan, I. and Kavraki, L. E. (2012) "A sampling-based tree planner for systems with complex dynamics", IEEE Trans. on Robotics, vol. 28, no. 1, pp.116–131.

[19] Wang X., Li X., Guan Y., Song J. and Wang R., (2019)"Bidirectional Potential Guided RRT* for Motion Planning," in IEEE Access, vol. 7, pp. 95046-95057.

[20] Ecodyne rada,weblink: https://echodyne.com/

[21] TrueView, weblink: https://fortemtech.com/products/trueview-radar/

[22] Hu J., Niu Y., Wang Z., (2017) "Obstacle Avoidance Methods for Rotor UAVs Using RealSense Camera", Chinese Automation Congress (CAC), pp.7151-7155.

[23] Beyeler A., Zufferey J.C., Floreano D, (April, 2007) "3D vision-based navigation for indoor microflyers". In Proc of the 2007 IEEE Int. Conf. on Robotics and Automation, Roma, Italy, 10–14 pp. 1336–1341.

[24] Khatib, (1985) "Real-time obstacle avoidance for manipulators and mobile robots, " in Proc of the IEEE Int. Conf. on Robotics and Automation (ICRA '85), vol. 2, pp. 500–505.

[25] Park M.G., Lee M.C. (2002) "Experimental Evaluation of Robot Path Planning by Artificial Potential Field Approach with Simulated Annealing", SICE 2002. Aug.5-7, Osaka.

[26] Temizer S., Mykel J., Leslie P.K., James K., (Aug. 2010) "Collision avoidance for unmanned aircraft using Markov decision processes", Proc of the AIAA Guidance Navigation and Control Conf.

[27] Barraquand J., Langlois J., and Latombe J. C. (1989) "Numerical potential field techniques for robot path planning". Technical Report ST.4N-CS-89-1285, Stanford University.

[28] Mantegh, M.R.M. Jenkin A.A. Goldenberg, (1998) "A modular and less complex environment representation algorithm [for mobile robots]", Industrial Electronics 1998. Proc. ISIE '98. IEEE Int. Symposium on, vol. 2, pp. 668-673 vol.2.

[29] Mantegh, M. R. M. Jenkin and A. A. Goldenberg, (1997) "Solving the find-path problem: a complete and less complex approach using the BIE methodology," Proc 1997 IEEE Int. Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation', Monterey, CA, USA, 1997, pp. 115-121.

[30] Ahmad A.M., (July, 2010)"Motion planning with gamma-harmonic potential fields" 2010 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics pp.297-302.

[31] Motonakah K., Watanabe1 K., Maeyama1 S., (2014) "3-Dimensional Kino-dynamic Motion Planning for an X4-Flyer Using 2-Dimensional Harmonic Potential Fields " 2014 14th Int. Conf. on Control, Automation and Systems (ICCAS 2014) pp.1181-1184.

[32] Panagiotis V., Constantinos V., Charalampos P. B., Kostas J. K., (2019) "Orientation-Aware Motion Planning in Complex Workspaces using Adaptive Harmonic Potential Fields," 2019 Int. Conf. on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 8592-8598.

[33] Mantegh I., Liao J., and He J., (2020) "Integrated Collision Avoidance for Unmanned Aircraft Systems with Harmonic Potential Field and Haptic Input". Proc. of the IEEE-Int. Symposium of System Integration 2020, Honolulu, Hawaii.

[34] Park J.W., Kwak H.J., Kang Y., and Dong W. K., (2016) "Advanced Fuzzy Potential Field Method for Mobile Robot Obstacle Avoidance" Computational Intelligence and Neuroscience, vol. 2016, Article ID 6047906, 13 pages, 2016.

[35] Takagi T. and Sugeno M., (1985) "Fuzzy identification of systems and its applications to modeling and control" IEEE Trans on Systems, Man, and Cybernetics, vol. 15, no. 1, pp. 116–132.

[36] Davison A., Reid I., Molton N., and Stasse O,. (2007) MonoSLAM: Real-time single camera SLAM. TPAMI 2007.

[37] Klein G. and Murray D., (2007) "Parallel Tracking and Mapping for Small AR Workspaces". ISMAR 2007.

[38] Klein G. and Murray D., (2008) "Improving the Agility of Keyframe-based SLAM". ECCV 2008.

[39] Klein G. and Murray D., (2009) "Parallel Tracking and Mapping on a Camera Phone". ISMAR 2009.

[40] Raul M., Montiel J. M. M., and Juan D. T. (2015) "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". IEEE Transactions on Robotics 2015.

[41] LiM., Mourikis A. (2013) "High-Precision, Consistent EKF-based Visual-Inertial Odometry". International Journal of Robotics Research 2013 CO, pp. 221–228.

[42] Ma X., Yao X., and Ding R., (April, 2020) "Influence of IMU's quality on VIO: based on MSCKF method", Proc. SPIE 11455, Sixth Symposium on Novel Optoelectronic Detection Technology and Applications.

[43] Hornu A., Kai M.W., Bennewitz W., (2013) "OctoMap: an efficient probabilistic 3D mapping framework based on octrees" Autonomous Robots volume 34, pp.189–206.

[44] Fukushima K., Murakami S., Matsushima J., Kato M. (1980) "Vestibular Responses and Branching of Interstitiospinal Neurons" Brain Res pp.131-145.

[45] Ling Q., Yan J., Li F., and Zhang Y., (2014) "A background modeling and foreground segmentation approach based on the feedback of moving objects in traffic surveillance systems", Journal of Neuro Computing, Elsevier. pp.1158–1179.

[46] Zhong B., Yuan X., Ji R., Yan Y., Cui Z., Hong X., Chen Y., Wang T., Chen D., and Yu J., (2014) "Structured Partial Least Squares for Simultaneous Object Tracking", Journal of Neurocomputing, Elsevier.

[47] Wu J., Liu N., Geyer C. and James M. R., (2013) "C4: A Real-time Object Detection Framework", IEEE Transaction on Image Processing.

[48] Jasper R.R. T. Gevers, A., (2012) "Selective Search for Object Recognition" International Journal of Computer Vision 104(2):154-171.

[49] Lin T ., Goyal P., Girshick R., He K ., Dollár P., (Oct. 2017) "Focal Loss for Dense Object Detection" 2017 IEEE International Conference on Computer Vision (ICCV), 22-29.

[50] https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk.html

[51] https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[52] https://www.intelrealsense.com/zh-hans/depth-camera-d435i/

[53] https://mavlink.io/en

[54] Paul Z., Howard M., (2000). "Fundamentals of Kalman Filtering: A Practical Approach. American Institute of Aeronautics and Astronautics, Incorporated". ISBN 978-1-56347-455-2.

[55] Hesam K.F., Faria S., Bak C., (2016)"A performance comparison between extended Kalman Filter and unscented Kalman Filter in power system dynamic state estimation" 51st International Universities Power Engineering Conference (UPEC).

[56] Hargrave P.J. (Feb. 1989) "A tutorial introduction to Kalman filtering" IEE Colloquium on Kalman Filters: Introduction, Applications and Future Developments.

[57] He Q., Wei C., Xu Y., (2017) "An improved adaptive Kalman filtering algorithm for balancing vehicle" 2017 Chinese Automation Congress (CAC) pp. 5721-5725.

[58] David S., Ross j., (2010) "Visual Object Tracking using Adaptive Correlation Filters". In CVPR.

[59] Citysim, a GAZEBO model for urban area, Weblink:https://www.wilselby.com/2019/05/ouster-os-1-ros-gazebo-simulation- in-mcity-and-citysim/

[60] Iris, a GAZEBO model for Iris drone: Weblink:https://dev.px4.io/master/en/simulation/

[61] Yu X., Zhou X., Zhang Y., (2019) "Collision-Free Trajectory Generation and Tracking for UAVs Using Markov Decision Process in a Cluttered Environment". J Intell Robot Syst 93, 17–32.