ABSTRACT

Title of dissertation: 3D FAST GEOMETRIC COLLISION AVOIDANCE ALGORITHM (FGA) AND DECISION-MAKING APPROACH BASED ON THE BALANCE OF SAFETY AND COST FOR UAS

Zijie Lin, Doctor of Philosophy, 2021

Dissertation directed by: Professor Huan Xu
Electrical Engineering

Unmanned Aircraft System (UAS) is a fast-growing industry with extensive economic implications and would be integrated into the national airspace system (NAS), which requires UAS to have the efficient sense and avoidance capability. This thesis develops a fast geometry-based algorithm FGA which shortens the collision avoidance computation time, path length and balances the probability of safety and energy cost by calculating and giving UAS proper selective avoidance starting time t_c, meaning the last possible point for the UAS to avoiding the potential threaten and it is based on the UAS kinematic, conflicts likelihood map, and navigation constraints. This operation enables the update path to be as close as possible to the UAVs resume designed path, decreasing the length of path variation and the corresponding time cost. In comparison to a current geometry method, the sampling-based method and the search algorithm, the FGA algorithm shows 40% to 90% of reduction in computational time and length of path for the same obstacle avoidance scenarios.

Quantitative analysis of the efficiency by different avoiding trigger times is also provided.

FGA with critical avoidance time t_c not only could improve the geometry methods, but also could be used for (1) research on the bounds of general geometry based collision avoidance, and (2) solving the multiple obstacles avoidance problem. For a scenarios with crowded obstacles which cannot be avoided at the same time, an applicable algorithm for obstacles classification is provided. It divides the obstacles into small groups with different urgent levels by their critical avoidance trigger time t_c, and then avoids them in sequence. Simulation validates the efficiency of this application.

Extremely difficult obstacle avoidance such as the UAV working under maneuver limitation and the obstacles are time-variant are discussed and solved in the following chapters. Monte Carlo simulation, statistical method and Machine learning algorithms especially the supervised logistic learning methods are implemented later to analyze the weight of the factors such as sensor detection distance, ratio of the speed, number of obstacles, which have impacts on the geometric based obstacle avoidance methods. Finally, flight missions in an aircraft simulator and the hardware fixed-wing aircraft experiments validate the algorithm effectiveness with successful results.

3D FAST GEOMETRIC COLLISION AVOIDANCE ALGORITHM (FGA) AND DECISION-MAKING APPROACH BASED ON THE BALANCE OF SAFETY AND COST FOR UAS

by

Zijie Lin

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Xu, Huan: *Chair*
Professor Qu, Gang: *Co-Chair*
Professor Bhattacharyya, Shuvra S.
Professor Babadi, Behtash
Professor Celi, Roberto
Professor Herrmann,Jeffrey *Dean's Representative*

© Copyright by
Zijie Lin
2021

Acknowledgments

Thanks for my advisor, Prof. Xu, Huan for the instruction, help and support in these years about my research. Thanks for Prof. Qu, Gang for the advice during proposal and defense. Thanks for all my committee members for the valuable time and efforts! Thanks Millennium Engineering Co. for their support of this project.

Table of Contents

Acknowledgements ii

List of Tables vi

List of Figures vii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 UAS Conflicts Overview . . . 3

2 Overview of the Collision Avoidance Algorithm 7
- 2.1 Sensor methods . . . 7
- 2.2 A* and D* . . . 8
- 2.3 Potential Field Methods and Virtual Force Field . . . 9
- 2.4 Sampling-Based Method and RRT . . . 11
- 2.5 Geometric Based Approach . . . 12

3 Fast Geometric Algorithm for Obstacle Avoidance 15
- 3.1 Algorithm Overview . . . 15
- 3.2 Model Assumption and Prerequisites . . . 17
- 3.3 FGA Working Mechanism . . . 19
- 3.4 Cost Function . . . 25
- 3.5 FGA Algorithm Design for Simulation and Coding . . . 27
- 3.6 Experiment Results of Time Cost . . . 29
- 3.7 Summary of FGA . . . 33

4 Geometry Collision Avoidance Algorithm Bounds 34
- 4.1 2D Avoidance Bounds . . . 34
- 4.2 3D Avoidance Bounds . . . 38

5 Class the Obstacles by Time 41
- 5.1 Simplification of Complicated Obstacle Avoidance Scenarios . . . 41
- 5.2 Simulation Results . . . 45

6 UAV Collision Avoidance Efficiency and Impact on the Neighborhood 47
6.1 Definition of Efficiency and Impacts . . . 48
6.2 Waypoints Missed . . . 49
6.3 Additional Path Length . . . 50
6.3.1 Shorter Path for Single Obstacle . . . 51
6.3.2 Shorter Path for Multiple Obstacles . . . 54
6.4 Smaller Space Occupation due to Avoidance Operation . . . 56
6.5 Impacts on the Environment . . . 57
6.6 Total Impacts and Reward Function . . . 58
6.7 Set Avoidance Operation Trigger Times . . . 59
6.8 Simulation Results and Analysis . . . 60
6.9 Conclusions . . . 71

7 Obstacle Avoidance with UAV Maneuver Limitation based on Geometry Methods 72
7.1 UAV Maneuver Limitation Overview . . . 72
7.2 Problem Formulation . . . 74
7.3 Geometry-based Collision Avoidance for UAV's with Maneuver Limitation . . . 75
7.3.1 Flight Models under Two Maneuver Limitations . . . 75
7.3.2 Control and States Update for the UAS . . . 77
7.4 Consequences of Maneuver Limitation on the UAVs Mission . . . 80
7.4.1 Solution Set of Collision Avoidance under Maneuver Limitation 81
7.4.2 Increased Cost of Collision Avoidance under Maneuver Limitation 82
7.4.3 Earlier Avoidance Trigger Time Needed to Avoid the Collisions 85
7.5 Decision Making Tree for UAV Collision Avoidance with Maneuver Limitation . . . 86
7.5.1 Anomaly Detection and Mitigation Node . . . 88
7.5.2 The Obstacle Detection and Avoidance Subtree . . . 89
7.5.3 The avoidance of high speed intruder node . . . 90
7.5.4 The Response to Environment Variation Subtree . . . 91
7.5.5 Setup of the Simulation Experiment . . . 93
7.6 Results and Analyses of the Simulation Experiment . . . 96
7.7 Conclusion . . . 107

8 Obstacle Avoidance of UAV for Time-Varying Obstacles based on Geometry Methods 108
8.1 Model of the Time Varying Obstacles . . . 108
8.2 The Safety Map . . . 109
8.3 The Cost Map . . . 111
8.4 Flexible Selection of the Obstacle Avoidance Solutions . . . 113
8.5 Conclusion . . . 114

9 Success Rates of Geometry Collision Avoidance Algorithm and the Failure Factor Analysis based on Machine Learning 116
9.1 Success Rates and the Experiment Results . . . 116
9.2 Failure Case Analysis using Machine Learning . . . 119

10 SITL Simulator and Hardware Experiment of Fixed-Wing Aircraft 125
10.1 Simulink Model and Test Environment Setup . . . 125
10.2 Software In The Loop (SITL) Simulator . . . 130
10.2.1 Ardupilot and DroneKit . . . 130
10.2.2 SITL Simulation Results . . . 132
10.3 Hardware Implementation . . . 134
10.4 Hardware Setup with Control Center . . . 140
10.5 Future Work . . . 141

A Using Matlab Machine Learning Tool to Get the Prediction Model and the Weight of the Factors 143

Bibliography 150

List of Tables

1.1 UAS conflicts reported to FAA on second quarter of 2020 3

3.1 Major variables used 20
3.2 Collision likelihood and action determination based on critical time . 24
3.3 Cost function to determine command direction in 3D avoidance maneuver 27
3.4 FGA obstacle avoidance time efficiency 30
3.5 Time cost of major collision avoidance methods 30
3.6 Use cost function to determine command direction in 3D 32

5.1 Cost function to determine command direction in 3D avoidance maneuver 45

6.1 Setup of the different avoidance trigger times 59
6.2 Avoidance operation cost by different trigger time 62
6.3 Monte carlo simulation statistic results of UAV avoiding obstacles by different trigger times 65

7.1 Collision avoidance strategy 74
7.2 Success rate of UAV with engine failure for static obstacles avoidance 98
7.3 Success rate of UAV with engine failure avoiding dynamic obstacles . 101
7.4 Success rate of UAV with maneuver limitation 2 for static obstacle . . 104
7.5 Success rates of UAV with maneuver limitation 2 for dynamic obstacles 105

8.1 Cost function to determine command direction in 2D space 112
8.2 Geometric based collision avoidance solutions 113
8.3 Possibility of successful collision avoidance 114
8.4 Cost of different collision avoidance solutions 114

List of Figures

2.1 Diagram of A* searching algorithm 8
2.2 Diagram of virtual force field collision avoidance algorithm 10
2.3 Diagram of RRT path planning algorithm 12
2.4 Diagram of geometry-based collision avoidance algorithm 14

3.1 Calculation of avoidance time 22
3.2 3D avoidance for static obstacles 31
3.3 3D avoidance for dynamic obstacles 32

4.1 2D collision avoidance geometry for Bound 2: UAV can turn clockwise or counterclockwise to avoid crossing protected area d_m in 2D space. 35
4.2 3D collision avoidance geometry for Bound 2. 39

5.1 Divide obstacles into smaller groups by critical avoidance trigger time 46

6.1 UAV misses some of waypoints when avoiding conflicts 50
6.2 L_1 and L_3 path comparison 52
6.3 Multiple obstacles: Obstacles overlap with initial path of the UAV . . 55
6.4 Multiple obstacles: No obstacles overlapped with initial path of the UAV 55
6.5 Collision avoidance action deviates the UAV from its original airway and may affect the space of other aircraft (schematic is not drawn to scale in airspace) 58
6.6 Impact of UAV avoiding multiple static obstacles by different avoidance trigger times 67
6.7 Success rate, total cost and rewards of collision avoidance by different avoidance trigger times for multiple static obstacles 68
6.8 Impact of UAV avoiding multiple dynamic obstacles by different avoidance trigger times 69
6.9 Success rate, total cost and rewards of collision avoidance by different avoidance trigger times for multiple static obstacles 70

7.1 diagram of maneuver limitation 80

7.2 The collision avoidance solution set regarding the yaw angle and the pitch angle under normal and abnormal conditions. A solution is represented by a point at (ψ, θ) as yaw angle and pitch angle combination . . . 82
7.3 Overall decision making tree for UAV collision avoidance with maneuver limitations . . . 87
7.4 The maneuver limitation subtree . . . 88
7.5 The collision avoidance subtree . . . 89
7.6 Sequence of commands issued to UAV to avoid high-speed obstacles . . . 91
7.7 The environment checking subtree . . . 91
7.8 Check and avoid the blocked weather area . . . 92
7.9 Simulink setup . . . 93
7.10 Collision avoidance sets for the UAV under healthy condition (whole area) and under maneuver limitation 1 (pink area only) . . . 97
7.11 Success rate of multiple static obstacle avoidance . . . 99
7.12 Avoidance success rate for multiple dynamic obstacles . . . 102
7.13 Collision avoidance solution sets when the UAV is in good condition (the whole area) and under maneuver limitation 2 (the area in pink) . . . 103
7.14 Collision avoidance for dynamic obstacles when the UAV is under maneuver limitation 1 (2D plot) . . . 106
7.15 Collision avoidance for dynamic obstacles when the UAV is under maneuver limitation 1 (3D plot) . . . 107

8.1 Safety map: possibility of avoiding the conflict successfully . . . 111
8.2 Cost distribution of Collision avoidance solutions . . . 112
8.3 UAV avoids a time-varying obstacle . . . 115

9.1 FGA static obstacle avoidance success rate . . . 117
9.2 FGA dynamic obstacle avoidance success rate with initial distance and obstacles' velocity. . . . 118
9.3 FGA dynamic obstacle avoidance success rate with obstacle average velocity and number of obstacles. . . . 118
9.4 Features have effect on success rate of UAV collision avoidance . . . 121
9.5 Train the prediction model . . . 122
9.6 Decide the weight of each feature . . . 124

10.1 High architecture of the robot's dynamic system . . . 126
10.2 Path planning module including collision avoidance in the simulink . . . 128
10.3 Dynamic obstacles avoidance function in the simulink . . . 129
10.4 SITL Architecture . . . 132
10.5 SITL simulator: Obstacle avoidance of static obstacle [1] . . . 133
10.6 SITL with FlightGear visualization [1] . . . 133
10.7 Critical avoidance SITL implementation: Flight paths and FGA avoidance, Google Earth view [1] . . . 134
10.8 Hardware Implementation Architecture . . . 135

10.9 Distance and Time trigger for physical implementation 138
10.10Hardware Implementation: Apprentice Aircraft 139
10.11Hardware Set up with the control center 140

A.1 Features have effect on success rate of UAV collision avoidance 144
A.2 Choose the data from workspace 145
A.3 Train the prediction model 146
A.4 Export the prediction model 147
A.5 Decide the weight of each feature 148
A.6 Decide the weight of each feature 149

Chapter 1: Introduction

1.1 Motivation

Unmanned Aerial Systems (UAS) have been utilized in a number of civil and military applications, including important public missions such as disaster relief, firefighting, search and rescue, as well as other applications in agriculture, infrastructure and scientific research [2]. UAS are a fast growing industry with extensive economic implications and will eventually be integrated into the national airspace system (NAS) [3] [4], which will require UAS to safely interact with other air vehicles. Standards for safe integration must be in place before a UAS can share the same airspace as manned aircraft, or when they operate in areas where they may inadvertently deviate into airspace designated to manned air vehicles. A UAS consists of an unmanned air vehicle (UAV) plus a ground control station, dedicated software, sensors, and telemetry, etc., all of which need research, development, testing and evaluation for safe integration.

One of the imperative technological requirements for safe integration of UAS into NAS is sense and avoid (SAA) capabilities. The purpose of SAA features is to prevent collisions with other traffic, human beings on the ground, natural birds, or collateral damage to property. Self-separation service needs to be provided by

SAA function, which would detect the potential collision and operate the necessary avoidance action and could therefore guarantee the safety of the commanded UAS and other air vehicles [5].

Technical improvement of the sensors, data communication and avoidance algorithms benefits in providing solutions for safe separation between UAV and other aircraft as well as avert collisions. Airspace restrictions should be considered when flying in a restricted zone such as airports, military operation areas, and prohibited areas. These airspace limitations would create potential scenarios where only a limited space is available for UAV to operate in.

Low altitude airspace UAS operations, such as agricultural and wildlife monitoring applications [6], may benefit from a collision avoidance mechanism which allows the UAV to stay on its intended trajectory as much as possible, while keeping a safe distance from any potential obstacles. FGA allows the UAV to target its initial route after conflict avoidance, while a selective separation distance from the obstacle is also allowed. Another application where FGA may be of high relevance is for UAV landing operations, where there may be more vehicles on the ground or in close proximity. Moreover, the new algorithm is compatible with current and future manned aircraft collision avoidance system e,g, TCASII/ACASX [7] and surveillance system e.g. automatic dependent surveillance-broadcast(ADS-B) as well as the air traffic control (ATC) [8] for separation management procedures in case being applicable for real world implantation.

1.2 UAS Conflicts Overview

According to the Federal Aviation Administration and its released report [9], over 400 UAVs were involved in the small incidents during the second season of 2020, as Table 1.1 shows.

Table 1.1: UAS conflicts reported to FAA on second quarter of 2020

Month	**UAS incident reported**
April	50
May	148
June	210

In addition to the small accidents, serious UAV involved incidents did happen. Due to covid-19, tourists and traffic including air traffic decreases significantly so the severe UAS incidents listed here are mainly from previous years. For instance, a civilian UAV collided with a Black Hawk helicopter in the evening over the eastern shore of Staten Island, New York City, United States at 9/21/2017 due to the wrong operation by the pilot of the UAV who was unaware of the helicopters' presence. At 8/10/2018, a hot air balloon carrying passengers was struck by a drone while flying in Driggs, Idaho, United States. The balloon finally landed safely with no injuries, but the drone fell to earth and was destroyed. At Canada, a UAV collided with a Beech King Air A100 of Skyjet Aviation collided on the Quebec City's airport and

increases tense.

Not only in the North American, but all around the world, severe UAV conflicts happened. Austria in August 2016, a UAV flew close to a rescue helicopter carrying the victim from a car crash to hospital, making the helicopter take evasive action to avoid it. Same year in France, a UAV came within 5m of an A320 flight near Charles de Gaulle Airport and the A320 took urgent avoidance action.

Thus, in this work, an algorithm called FGA for navigation is developed for the scenarios where self-separation was either not possible or precluded by occurrence of emergent events so that obstacles avoidance was necessary. FGA was designed in the way that flexible start times for the avoidance maneuver would be available to select for shorter paths and selective separation from obstacles. This allows the UAV to pass through busy airspace with a more time efficient and less cost way. This would be especially useful when the UAV is getting near natural boundaries and restricted airspace areas, such as in low altitude performance where the UAV might be more exposed to interacting with obstacles.

FGA entails collision avoidance with static obstacles such as electric power lines and buildings as well as avoidance for multiple dynamic intruders. A separation sphere for protection of radius d_m is created with the obstacle and a conflict-free path is generated.

The second contribution of FGA is it's useful to class the obstacles into several groups, changing the problem of avoiding all obstacles at the same time which would be impossible with a crowded-obstacles environment into the problem of avoiding them in sequence. The most urgent obstacles will be avoided first and the obstacles

with less urgency are treated then. The urgent levels of the obstacles are classed through different critical trigger times (neither distance nor speed), which is the final possible escaping time of UAV for each obstacle and could fully represent the emergency level of the obstacles.

The application boundaries and the success rate with different factors for general geometric methods, not only the FGA are analyzed. Complicated scenarios in the real world, such as UAV with maneuver limitations and the obstacles changing their velocity and speed are also considered. Corresponding strategy such as decision making algorithm with cost function and success possibility map is provided for the UAV to have a flexible choice.

In addition, Monte Carlo simulations with high success rates are provided to validate the utility of FGA , efficiency of different trigger times on the obstacle avoidance, the avoidance solution for UAV with maneuver limitation and obstacles with time-varying variables. Furthermore, implementation of FGA in an aircraft simulator (SITL) is demonstrated successfully as well as flights on a physical fixed-wing UAV avoiding obstacles. Simulator and flight test results validate the utility and feasibility of this work.

This thesis is organized as follows: Chapter 2 describes the current collision avoidance methods. Chapter 3 describes the working mechanism of the FGA algorithm. Chapter 4 describes the application bounds of geometric algorithms. Chapter 5 introduces using different collision avoidance trigger time to simplify multiple obstacles avoidance problems. Chapter 6 analyzes the path and space efficiency vs different avoidance operation starting time. Chapter 7 is about UAVs collision

avoidance with the maneuver limitation. Chapter 8 is the collision avoidance with variant speed and direction intruders. Chapter 9 is the analysis of failure cases of geometry methods based on machine learning algorithms. Software in the loop implement and hardware in the loop test is in Chapter 10.

Chapter 2: Overview of the Collision Avoidance Algorithm

After decades of technical development, numerous different types of obstacle avoidance algorithms are developed depending on the information such as sufficiency of the data available of the environment and obstacles, the time of avoidance actions consumed, the working mechanism to generate a new path. The method fits the situation most will be chosen for the obstacle avoidance simulation, hardware experiments and further real time application.

2.1 Sensor methods

Algorithms concentrating on the sensing and detection aspect, such as improvement of sensor sensitivity [10], [11] are called sensor-relative methods. Sensor technologies include active interrogation transponders [5], electro-optic [12], laser/LIDAR [8], onboard and ground based radar, ADS-B [13] and acoustic technologies [5].

The full name of ADS-B is Automatic Dependent Surveillance Broadcast. It is a surveillance technology for an aircraft to know its position through satellite and share it periodically, enabling it to be tracked and giving real time precise situation awareness for pilots, ground controllers or other aircraft.

Real-time ADS-B is preferred for air traffic control in the NAS as a surveillance

method since it has been validated that ADS-B application improves efficiency and safety on the aviation, lessens harmful effects on the environment and reduces costs. The fixed-wind model in hardware test for this thesis installed ADS-B.

2.2 A* and D*

Global planning techniques compute a collision free trajectory with a configuration that complies with the vehicle constraints. These methods generally consist of searching for a path in a graph, where the graph is computed from the constraints. Some of these methods include: Dijkstras algorithm, A*, B*, D*, D*Lite algorithms [14] [15]. The A* search algorithm is an extension of Dijkstras which reduces the state's parameters during the search processes by incorporating heuristics. Figure 2.1 presents the searching mechanism of the A* algorithm.

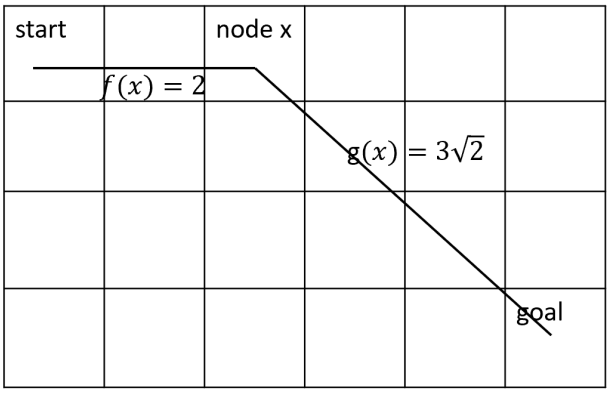


Figure 2.1: Diagram of A* searching algorithm

Suppose the current point evaluated is x. Then in A*, the total cost to choose

x as the next waypoint is

$$h(x) = f(x) + g(x) \tag{2.1}$$

Where $f(x)$ is the cost from start point to the point x and the $g(x)$ is the cost from x to the final destination point. Then for each available node x, push $h(x)$ into a priority queue which has the priority of minimum $h(x)$ in the top. In every iteration, the top node is popped from the queue and the nodes connected with it are evaluated until the goal is reached. The A* can be used for searching the shortest path but it consumes a long time.

B* is the best-first search algorithm in the application of finding the least cost path from a given initial node to any goal node. D* algorithm or called Dynamic A*, expands nodes backward from the goal node selectively with time-variant cost function [15]. All these algorithms typically require environment modeling [15], have a heavy computational cost, and thus are more applicable for static obstacles.

2.3 Potential Field Methods and Virtual Force Field

The force field approach [16], which based on the potential fields' concept, calculates the gradient or the potential of particular variables between obstacles and UAVs such as distance [17]. These reactive algorithms assume that obstacles generate an attraction and repulsion force, then find a path for UAV to avoid conflicts. However, local minima is a challenging issue in these methods, as well as oscillations when passing through the narrow passages [18] or the closely spaced obstacles. Another drawback of these methods is computational efficiency. When

the number of obstacles increases, the computational efficiency decreases.

Virtual Force Field (VFF) [19] is the combination of the attractive or repulsive force and potential field method. However, when the UAV is traveling among densely cluttered obstacles environments at increased speed, this algorithm has limitations.

The Vector Field Histogram (VFH) is an improvement of the VFF method and eliminates the disadvantages of VFF [20]. The VFH algorithm constructs polar histograms around the vehicle, then accordingly chooses an avoidance direction [21]. VFH is further improved, e.g. VFH+ [22], which develops binary histograms, and VFH*, which applies the A* algorithm to decrease the cost and heuristic functions. These methods also have local minima problems, which are similar to the shortcomings of the potential field methods. Figure 2.2 presents the working mechanism of the virtual force field algorithm.

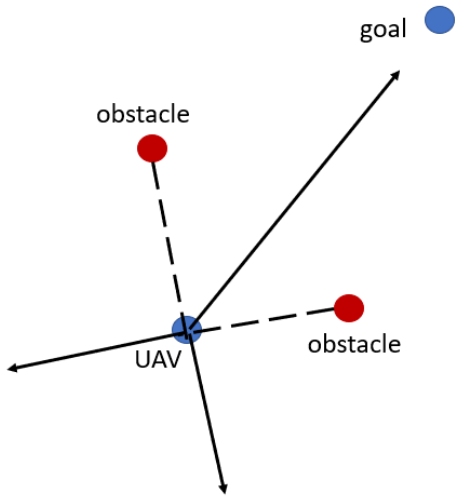


Figure 2.2: Diagram of virtual force field collision avoidance algorithm

In Figure 2.2, assume the goal has an attractive power to the UAV point.

Oppositely, the two obstacles (red dots) have the repulsive power to push away the UAV. The combination of those three powers will give the direction for UAV to move to the next step.

2.4 Sampling-Based Method and RRT

Sampling-based path planning algorithms are based on the knowledge of the coordination of the nodes in the grid. The rapidly-exploring random trees (RRT) [23] is one of the most representative method of them.

Figure 2.3 explains the exploring processes of the RRT algorithms. RRT strategies find the path from start to destination through expanding a tree by randomly sampling nodes (q_{rand}) and growing from a starting point until the tree reaches the target [24] or within a tolerant distance to the target. In each iteration a new node is added to its nearest point in the tree (q_{rand}), then one branch is generated for the tree ($\overline{q_{near}q_{rand}}$, such as the green line in Figure 2.3). The processes are finished when the goal is approached. However, the RRT algorithms always expand the whole feasible space before finding a solution to the target. In addition, if there are no solutions, the RRT algorithms are unable to provide a path or stop. Instead, they continue searching the white space, which slows down the simulation significantly and consumes computation memory.

Optimal path planning algorithms, optimal trajectory [25] [26] and waypoint generation methods [27] [28] are another three sampling based methods that have been studied. These methods develop obstacle-free paths but are particularly costly

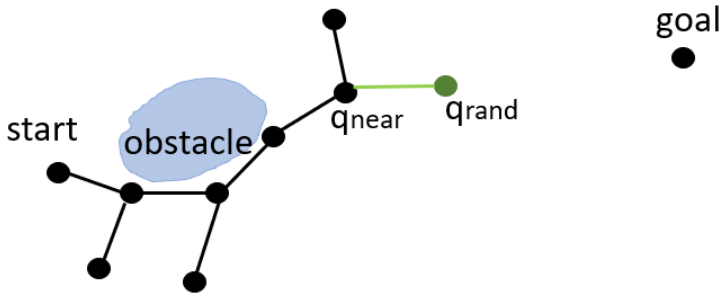


Figure 2.3: Diagram of RRT path planning algorithm

in computation space and time [29]. They test all available points nearby to select the next waypoint [27] [28], then generate an obstacles-free path between the current waypoint and the next waypoint, and repeat this step until an optimal path is found [29] [30]. Though they are successful at developing the conflicts-free trajectory for a UAV avoiding single or multiple obstacles, these methods are time-consuming and are not very practical for real time avoidance [31] [32].

In general, above methods don't fit for fast collision avoidance due to slowness caused by their complexity. Potential fields and associated methods have issues with local minima and their optimization of paths is done in an iterative way which is computationally expensive for multiple obstacles [33]. Sampling based methods are also computationally expensive as well since they generate new waypoints through exploring the environment sometimes even the whole space.

2.5 Geometric Based Approach

Geometric-based methods is another type of widely used obstacle avoidance algorithms, [5], [34], [35]. They use the geometric relationship between the obstacles

and the UAV to calculate a new proper speed, yaw angle, pitching angle which could avoid the conflicts for UAV and then generate the corresponding path. These algorithms require less processing power and are therefore more applicable for UAS onboard calculation due to their size, weight and power constraints. Geometry based collision avoidance algorithms can be roughly classified into three kinds: those which produce angle changes using geometry information such as the vehicle motion/location [5]; those which use velocity variation [35]; those which use combinations of these and those which also take other types of information into account, e.g. global [36], and probabilistic [37]. Changing the velocity will affect the time cost if arriving time is required; also,in some of the situations, it will exceed the maximum or minimum limit of speed [35]. Thus, when doing research on the geometry methods in this proposal, angle variation is chosen in priority than speed variation.

Different aspects of the collision avoidance problem have been considered in the literature. Guidance laws, controllers, tracking points, maneuver types, different vehicle types, single/multiple obstacle types, optimization, among other aspects, have been extensively explored. In the literature, geometric based algorithms have used linear and nonlinear guidance laws [33] to align heading and pitch angles to the desired spatial orientation. Minimum-effort guidance laws with collision cone approach were applied to guide the vehicle to a given collision-free waypoint [38]. Strategies for geometric based collision avoidance include keeping desired relative bearing and elevation during the collision avoidance encounter [39]. Multiple obstacle formulations have also been explored using velocity obstacle methods for cooperative conflict resolution [35] and vector resolution approach [34], which restricts avoidance

maneuvers to the miss distance vector, which is formed between the UAV and the closest point of approach (PCA). To minimize the miss distance, a cost function is used to find the optimal solution. Software in the loop (SITL) and hardware in the loop (HITL) tests have been conducted for collision avoidance using ADS-B transmitters for intruder aircraft [40].

Figure 2.4 presents the simplification working mechanism of geometry-based collision avoidance algorithm. The blue UAV will catch up with the moving obstacle(grey aircraft) and the conflict happens if the UAV follows its original speed and direction. Hence, to avoid the conflict, the UAV calculates the collision avoidance solution, changes its direction and follows the green trajectory to avoid entering the separation sphere of the grey aircraft.

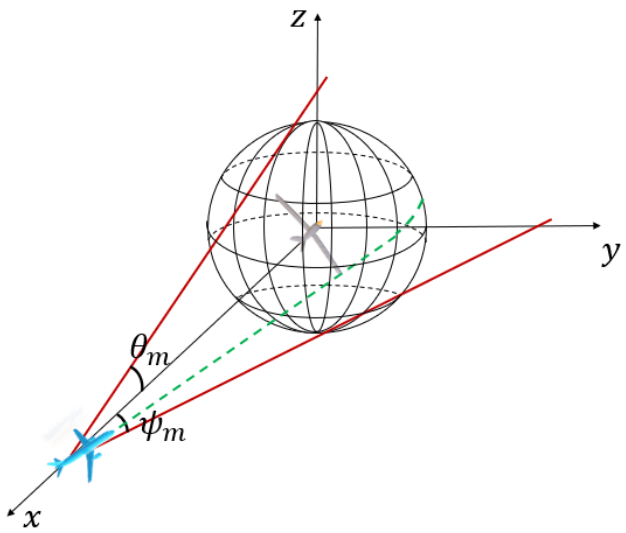


Figure 2.4: Diagram of geometry-based collision avoidance algorithm

Chapter 3: Fast Geometric Algorithm for Obstacle Avoidance

This chapter describes a novel geometric-based obstacle avoidance algorithm, the Fast Geometric Avoidance algorithm (FGA), which classifies obstacles, allows for the vehicle to selectively start the avoidance operation for time and path saving and return to its original navigation trajectory. The algorithm dependents on concepts presented by [41]. It is constructed using geometry relationship, which includes spatial location, angular displacements, relative velocities, as well as concept of collision cone [42] to find and determine probable conflicts [36].

3.1 Algorithm Overview

The algorithm builds on the fixed-wing vehicle. Data about obstacles' position and velocity is assumed accessible to the UAV. Firstly, FGA scans the environment around through on-board sensors and continuously determines if there are any obstacles that could potentially threaten the UAV and are closer than a safety surveillance distance R. After a potential obstacle is detected, it determines which particular avoidance procedure to apply depending on whether the obstacle has been determined to be static or dynamic, and whether there are single or multiple obstacles. Depending on the identified obstacle type and associated information,

FGA then automatically calculates the time to collision and the optimal time for the UAV to start the avoidance operation. The variables needed for the avoidance maneuver are calculated and locally stored for each potential obstacle.

The avoidance starting time in FGA will also be selected based on airspace limitations, and the crowded level of the local airspace. This adds functionality to typical avoidance procedures, where obstacle avoidance operations normally begin upon detection of a potential conflict. The FGA algorithm allows for selective separation with respect to the closest point of approach (CPA). FGA also allows for selective start times based on desired separation from the obstacle. In the event that separation from the obstacle needs to be minimal, as may be the case in certain specific or emergency scenarios, the UAV triggers the avoidance maneuver at t_c, the critical avoidance time. A detailed explanation of this calculation is presented in the next section. FGA then triggers the avoidance operation and allows it to operate during a time window of flexible duration until it is determined that a safe distance from the obstacle has been reached. After that, the algorithm searches for the next reachable way-point located on the original path and lets the UAV return to the original intended path shortly thereafter. This minimizes the length of the path which is generated to avoid the obstacles and reduces computation time in comparison to way-point based or optimal path planning methods, which typically recalculate full path lengths. Different collision avoidance methods were compared in simulation to FGA, and results showed that FGA is faster and yields shorter avoidance paths.

The algorithm's distinct features and contributions can be summarized as: (1)

obstacle classification after detection; (2) shorter computation time as compared to other algorithms; (3) selective avoidance maneuver starting time t_c; (4) least changing path (67% shorter in average than other geometry methods); (5) constant UAV speed if needed; and (6) the ability to deal with complex environments such as multiple moving obstacles.

3.2 Model Assumption and Prerequisites

The software simulation part of obstacle avoidance is performed in Matlab, Simulink and SITL while the hardware experiments are done on a shelf aircraft RC Apprentice 15e. Here are necessary assumptions:

(1) The fixed-wing UAV model has a velocity which ranges from 13 m/s to 19 m/s and its maximum bank angle is set to 30 degrees.

The turn radius ρ of an fixed wing aircraft is a property of the vehicle kinematics, generally a function in terms of its velocity and bank angle ϕ:

$$\rho = \frac{V_u^2}{gtan\phi}. \tag{3.1}$$

From Equation 3.1 we have the minimum turn radius $\rho = \rho_{min}$:

$$\rho_{min}(V_u) = \frac{V_u^2}{gtan\phi_{max}} \tag{3.2}$$

Where ϕ_{max} is usually set to be 30 degrees and V_u is set to be constant during the avoidance operation.

A pure yaw rate is then given by:

$$\dot{\psi} = \frac{V_u}{\rho} = \frac{gtan\phi}{V_u}, \tag{3.3}$$

with the aircraft kinematic model:

$$\dot{x} = V_x = V_u cos(\theta) cos(\psi), \tag{3.4}$$

$$\dot{y} = V_y = V_u cos(\theta) sin(\psi), \tag{3.5}$$

$$\dot{z} = V_z = V_u sin(\theta). \tag{3.6}$$

(2) Controller in FGA: The variables controlled in this algorithm is the UAV's current horizontal angle (yaw angle) ψ and the vertical angle (pitch angle, pitch angle used in this dissertation is for altitude control) θ. Suppose the commanded yaw and pitch angle are ψ_c and θ_c. Using the PID control:

$$\dot{\psi} = K_p(\psi_c - \psi) + K_i \int (\psi_c - \psi) + K_d \frac{d(\psi_c - \psi)}{dt}, \tag{3.7}$$

$$\dot{\theta} = K_p(\theta_c - \theta) + K_i \int (\theta_c - \theta) + K_d \frac{d(\theta_c - \theta)}{dt}. \tag{3.8}$$

Where K_p, K_i and K_d are the proportional, integral and differential parameters of the PID controller.

(3) The FGA algorithm does not require a specific type of onboard sensors. Any sensors onboard a UAV and provide accurate data are applicable, similar to [13].

(4)The search radius of the sensor onboard the UAV is set to be much larger than the minimum separation radius d_m and the potential danger radius R is set to be 10 times more than the minimum separation radius d_m. These parameters can be changed according to the different requirements.

(5) In the simulation, all of the obstacles are generated randomly by an obstacle-generation module to guarantee the general properties and test the UAV's capability to avoid different obstacles. The module generates 10-20 static or dynamic objects for each scenario and provides their information to the detection module. Usually about $20 - 30\%$ of objects become the actual threats for the UAV. The information generated for the obstacles includes e.g.velocity, initial location, direction.

- The velocity of the obstacle is randomly generated in three ranges: 15-20 m/s, 35-40 m/s, 55-60 m/s.
- The location of the obstacle is generated 360 degrees around the UAV and the initial distance from obstacle to UAV goes from 100 m to 500 m.
- The direction of each obstacle is between 0 and 360 degrees to the UAV.

(6) The radius d_m of the protected sphere depends on the obstacle characteristics. Obstacles with larger size and have a higher velocity have a larger safe zone, which is described by d_{ma}. Similarly, larger fixed-wing UAVs with a higher speed require a larger safe area d_{mu}. The total radius will be a combination of d_{ma} and d_{mu}, and additional factors depending on airspace limitations and distance to other obstacles.

Table 3.1 presents the frequent used parameters in the thesis.

3.3 FGA Working Mechanism

The FGA majorly includes the following steps.

Table 3.1: Major variables used

V_a, **Velocity of the obstacle**	Randomly generated between 0 and 60 m/s
V_u, **velocity of the UAV**	15 m/s
ρ, **min turning radius**	30 m
R, **potential danger radius**	300 m
$r_c(t_c)$, **critical avoidance starting distance (time)**	Dependent on different situations
$d_m(d_m, v, d_m, h)$, **safe separate radius**	30m, $R > r_c > d_m$
η_ψ, **coefficient of yaw angle deviation**	0.2
η_θ, **coefficient of pitch angle deviation**	0.5

- First, determine if the obstacles encountered are static or dynamic. Static obstacles include buildings, electric towers and trees. Dynamic obstacles include other aircraft or flying animals such as birds.

- Then, calculate the optimal avoidance starting time(point) for the UAV.

- Next, start the avoidance procedure for the UAV. The UAV can choose to pitch up, down, head left or right, acceleration, deceleration or a combination of these six major operations. The combination, which will yield a 3D turn, is based on the cost of the chosen yaw angle, pitch angle and speed (ψ_c, θ_c, v_c).

FGA combines relative velocity, distance and calculation of avoidance time. In FGA, the safe separation radius d_m is a very important parameter which is centered at the obstacle location with coordinates $A(A_x, A_y, A_z)$, and delimits a volume which the UAV should not enter. The estimation of d_m should include vehicle dimensions d_{ma}, vehicle velocity d_{mu}, and airspace limitations $d_{lim_{air}}$ as well as space limitations related to nearby obstacles $d_{lim_{obs}}$. In general, d_m can be formulated as:

$$d_m = c_1 d_{ma} + c_2 d_{mu} + c_3 d_{lim_{air}} + c_4 d_{lim_{obs}}. \tag{3.9}$$

Where the constants c_i are determined based on the environment and vehicles' mission constraints.

The critical avoidance trigger time t_c is calculated for each obstacle which is checked as a threat for the UAS.

$$t_c = \frac{r_0 cos\alpha_0 - \sqrt{(\rho + d_m)^2 - (\rho + r_0 sin\alpha_0)^2}}{||V_r||} \tag{3.10}$$

Where r_0 is the relative distance from the UAV to the obstacle, α_0 is the relative angular displacement and V_r is the relative velocity.

The geometry relation for critical avoidance time calculation is in Figure 3.1.

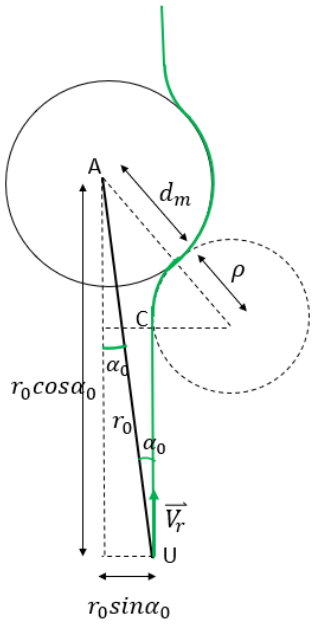


Figure 3.1: Calculation of avoidance time

The detection time t_0 is the time at which the UAV detects an actual obstacle, The critical avoidance time t_c is the time beyond which, the aircraft will trespass the protected area of radius d_m centered at the obstacle location. Therefore, the UAV can trigger its avoidance maneuver at any time between t_0 and t_c depending on mission goals and airspace limitations such as $d_{lim_{air}}$, and distance to obstacles $d_{lim_{obs}}$ as well as the safety level constraints C_s.

Mission goals include the preference for the shortest path to a goal location or the least deviation from the original trajectory. The avoidance trigger time encodes the features of the avoidance geometry. For the 2D avoidance maneuvers, the least

deviated path will be achieved using the minimum turn radius ρ_{min} and the critical triggered time t_c. The shortest path, as calculated in [43], can be executed by using the minimum turn radius ρ_{min} and a tangent to the protected disk of safe separation radius d_m. These two types of paths are part of FGA and are triggered depending on mission constraints. If the larger radial distance from the obstacle is required, then the calculation of turn radius ρ would take into account all the relevant constraints such as (Equation 3.9).

Table 3.2 shows a classification of threat level based on avoidance time calculations. The avoidance time calculation has three main functions:

(1) Determine the risk of intruding into the obstacle's protected distance.

(2) Determine the optimal time and location for the UAV to start the avoidance. Or during emergency procedures, require more agile vehicle maneuvers for UAV to avoid obstacles.

(3) Class large amounts of obstacles into small groups and avoid them in sequence.

An obstacle which is known to be carrying passengers has the highest priority and will cause the FGA to trigger the avoidance maneuver much faster, e.g. at t_0. A safety score constant C_s is applied before avoidance time such that $t_c = C_s t_c$, where C_s is between 0 and 1. An obstacle with passengers would have $C_s = 0$ and a natural barrier would have $C_s = 1$ [1];

If $\rho = \rho_{min}$ in Equation 3.10, the last geometrically feasible time t_c can be

Table 3.2: Collision likelihood and action determination based on critical time

Avoidance Time	**Collision Likelihood**	**Avoidance**
$t_c <= 0$	Imminent	Start immediately
$t_c > 0$	Likely	Start depending on constraints

calculated as:

$$t_c = \frac{r_0 cos\alpha_0 - \sqrt{(\rho_{min} + d_m)^2 - (\rho_{min} + r_0 sin\alpha_0)^2}}{||V_r||}. \tag{3.11}$$

Let r_c corresponds to the distance between the UAV and the obstacle at t_c:

$$r_c = r_0 - ||V_r||t_c = r_0 - r_0 * cos\alpha_0 + \sqrt{(\rho_{min} + d_m)^2 - (\rho_{min} + r_0 sin\alpha_0)^2}. \tag{3.12}$$

If we use a geometric approximation which amounts to consider a buffer time t_b and the small system response time t_{res} due to physical constraints, we have that the critical avoidance time is:

$$t_c = t_c(theory) - t_b - t_{res} \tag{3.13}$$

Where $t_c(theory)$ is obtained from Equation 3.10. Or t_c be simplified to,

$$t_c = \frac{r_0 cos\alpha_0 - \rho_{min} - (d_m - r_0 sin\alpha_0)}{||V_r||} \tag{3.14}$$

The r_c corresponds to the distance between the UAV and the obstacle at time t_c:

$$r_c = r_0 - ||V_r||t_c = r_0 - r_0 * cos\alpha_0 + \rho + d_m - r_0 sin\alpha_0. \tag{3.15}$$

The avoidance operation will be successful if the UAV starts before the avoidance time t_c.

In the specific case where $\alpha_0 = 0$, the minimum in-theory distance r_c to avoid a collision with the obstacle is:

$$r_c(theory) = \sqrt{(\rho + d_m)^2 - \rho^2}, \tag{3.16}$$

or the critical distance in application becomes:

$$r_c = \rho + d_m, \tag{3.17}$$

where,

$$r_c \geq r_c(theory). \tag{3.18}$$

The critical time is then equal to:

$$t_c = \frac{r_0 - \rho_{min} - d_m}{||V_r||} = \frac{r_0 - r_c}{||V_r||}. \tag{3.19}$$

3.4 Cost Function

Obstacle avoidance maneuvers in FGA are executed in both horizontal plane (yaw maneuver) and vertical plane (pitch maneuver) [44] as well as the acceleration or deceleration. This is achieved by combining yaw and pitch maneuvers and is determined by a cost function. To reduce the time complexity, at the first loop UAV keeps its current speed constant and searches for the solution combined with new (ψ, θ). If under constant speed and there is no solution exists, then the UAV changes its velocity.

Suppose the original yaw and pitch direction of the UAV before encountering the obstacle is $(\psi_{ori}, \theta_{ori})$. The range of angles that the yaw angle ψ_{range} and pitch

angle θ_{range} can have are defined as follows:

$$-\psi_{range} < \psi_{ori} < \psi_{range} \tag{3.20}$$

$$-\theta_{range} < \theta_{ori} < \theta_{range} \tag{3.21}$$

If both $[-\psi_{range}, \psi_{range}]$ and $[-\theta_{range}, \theta_{range}]$ are divided into n units, there will be n by n pairs of (ψ, θ) values. For pair (ψ_i, θ_j), the cost function is defined as:

$$Cost(k, j) = \eta_\psi(\varphi_i - \varphi_{ori}) + \eta_\theta(\theta_j - \theta_{ori}) \tag{3.22}$$

Where η_ψ and η_θ are the coefficients of the horizontal and vertical direction deviation and they are positive numbers. Each one of the n by n pairs has its own associated cost. This is shown in Table 3.3. For different situations, there are different cost functions to choose, The cost function will:

(1) Search from these pairs and mark which pairs can drive the UAV to a safe distance away from the avoidance area.

(2) Changing a fixed-wing vehicle's pitch is less desirable due to its maximum angular deflection in pitch will be less than the one for yaw. Because of this limitation, the cost coefficient for pitch η_θ is set to be larger than η_ψ [5].

(3) From marked pairs and their calculated costs, a minimum cost $Cost_{min}$ is found. Its response angle pair $[\psi(cost_{min}), \theta(cost_{min})]$ is selected as the commanded yaw and pitch angle for the UAV to follow:

$$\psi_{command} = \psi(cost_{min}) \tag{3.23}$$

$$\theta_{command} = \theta(cost_{min}) \tag{3.24}$$

Table 3.3: Cost function to determine command direction in 3D avoidance maneuver

$Cost_{11}(\psi_1, \theta_1)$	$Cost_{12}(\psi_1, \theta_2)$	...	$Cost_{1n}(\psi_1, \theta_n)$
$Cost_{21}(\psi_2, \theta_1)$	$Cost_{22}(\psi_2, \theta_2)$	...	$Cost_{2n}(\psi_2, \theta_n)$
...	...	...	...
$Cost_{n1}(\psi_n, \theta_1)$	$Cost_{n2}(\psi_n, \theta_2)$	...	$Cost_{nn}(\psi_n, \theta_n)$

3.5 FGA Algorithm Design for Simulation and Coding

The FGA, Fast Geometric Avoidance algorithm is constituted by three sub-algorithms, detection (Algorithm 1), obstacle avoidance with optimal avoidance starting time (Algorithm 2) and the next waypoint search (Algorithm 3).

Algorithm 1 FGA algorithm: Detection

Precondition: necessary data is available for UAV

for $i \leftarrow 1$ to n **do**
  $\overrightarrow{r_0} \leftarrow \overrightarrow{A_0} - \overrightarrow{U_0}$ ▷ relative location vector
  $\overrightarrow{V_{r0}} \leftarrow \overrightarrow{V_{u0}} - \overrightarrow{V_a}$ ▷ relative velocity vector
  $\theta_{t0} \leftarrow f(\overrightarrow{r_0}, \overrightarrow{V_{r0}})$ ▷ angle bias
  $r_{PCA} \leftarrow r_0 \cdot sin\theta_{t0}$ ▷ closest distance in future
  if $r_{PCA} < d_m$ **then**
    $status \leftarrow dangerous$
    $M_1 \leftarrow$ obstacle information
  end if
end for
return $M_1, status$

Algorithm 2 FGA Algorithm: Obstacle Avoidance

for $i \leftarrow 1$ to m **do**

    $t_{ci} \leftarrow \dfrac{r_0 cos\theta_{t0} - \rho_{min} - (d_m - r_0 sin\theta_{t0})}{||V_r||}$

end for

$t_c \leftarrow min(t_{ci})$

if $t_c < 0$ **then**

    avoidance starts immediately

else

    avoidance starts at t_c

end if

    set ψ_{range} and θ_{range}

for $k \leftarrow 1$ to m_1 **do** ▷ m_1, number of sampling in ψ_{range}

    for $j \leftarrow 1$ to m_2 **do** ▷ m_2, number of sampling in θ_{range}

        $\psi \leftarrow -\psi_{range} + \psi_{range}/m_1 * k$

        $\theta \leftarrow -\theta_{range} + \theta_{range}/m_2 * j$

        $\overrightarrow{V_{ut}} \leftarrow (V_u cos\theta cos\psi, V_u cos\theta sin\psi, V_u sin\theta)$

        for $l \leftarrow 1$ to l_1 **do** ▷ l_1, number of obstacles

            $\overrightarrow{r_{ltc}} \leftarrow \overrightarrow{A_{ltc}}(A_{lx}, A_{ly}, A_{lz}) - \overrightarrow{U_{tc}}(U_x, U_y, U_z)$

            $d = min(d_l) \leftarrow |\overrightarrow{V_{ut}} \times \overrightarrow{r_{ltc}}|/|\overrightarrow{r_{ltc}}|$

        end for

        if $d > d_m$ **then**

            $Cost(k, j) \leftarrow \eta_\psi(\psi - \psi_{ori}) + \eta_\theta(\theta - \theta_{ori})$

        else

            $Cost(k, j) \leftarrow \infty$

        end if

    end for

end for $Cost_{min}, \psi_{com}, \theta_{com} = min(Cost(k, j)), \psi_k, \theta_j$;

return $\varphi_{com}, \theta_{com}, newpath$

Algorithm 3 FGA Algorithm: Next Waypoint Search

1: **for** $i \leftarrow 1$ to w **do**

2: $\quad \overrightarrow{r_{wi}} \leftarrow \overrightarrow{W_i}(W_{xi}, W_{yi}, W_{iz}) - \overrightarrow{U_t}(U_x, U_y, U_z)$

3: $\quad \theta_{wi} \leftarrow f(\overrightarrow{r_{wi}}, \overrightarrow{V_u})$ ▷ waypoints angle bias

4: $\quad$ **if** $\theta_{wi} < \pi$ **then** ▷ search the next waypoint

5: $\quad$ **end if**

6: **end for**

7: **return** W_i ▷ get the next waypoint

3.6 Experiment Results of Time Cost

Simulation results in Matlab and Simulink of the FGA are provided in this section. Variables in use are presented in Table 3.1. A set of 5000 to 20000 scenarios of UAV avoiding obstacles consists of Monte Carlo simulations. The obstacles in these cases are randomly generated by an obstacle generator module we designed, as described in Section 3.2

Monte Carlo simulation time about FGA avoiding static and dynamic obstacles is summarized at Table 3.4. In the table, the average time for the UAV to generate a new path to avoid multiple static obstacles and dynamic obstacles is 0.028s and 0.03s, which is very short and makes it applicable for real time hardware implementation.

Table 3.5 provides the comparison of computation time between FGA and other collision avoidance methods. It is validated that FGA consumes much less average computation time than others, such as sampling-based optimal methods and potential field methods.

Table 3.4: FGA obstacle avoidance time efficiency

Obstacle type	Num. of cases	Total computation time	Average time
Static obstacles	1413	39.564	0.028
Dynamic obstacles	3750	112.5	0.030
the number of obstacles in each case varies from 1-3			

Table 3.5: Time cost of major collision avoidance methods

Methods	average time each case	time saved by FGA	evaluation
Potential field [45]	0.082	63.41%	medium
Optimal 1 [31]	0.18	83.33%	slow
Optimal 2 [27]	0.3	90.00%	slow

Figure 3.2 shows that the UAV avoids a single obstacle in 3D space successfully with starting time $t_c \approx 3.0s$ and causes minimal deviations from its original heading.

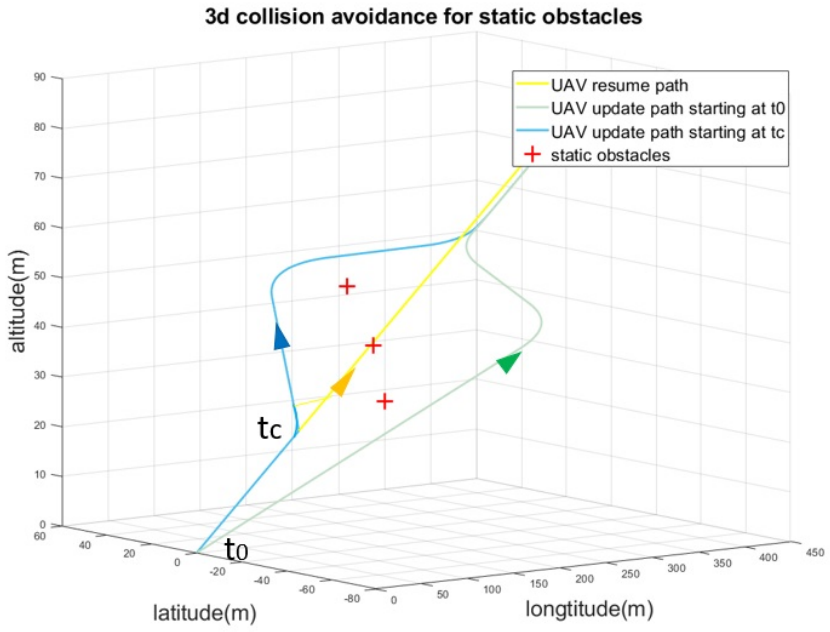


Figure 3.2: 3D avoidance for static obstacles

Figure 3.3 provides a successful result about UAV avoiding multiple dynamic obstacles in 3D space with starting time $t_c \approx 0.8s$ and returning to its original trajectory. Table 3.6 is the cost table of this scenario. It presents the cost of each potential collision avoidance solution, generated by the cost function. In this case, a pair (ψ, θ) with minimum cost of 0.1463 was selected as the final solution of UAV collision avoidance and this command was issued to the UAV execution part. Figure 3.3 presents this process.

Table 3.6: Use cost function to determine command direction in 3D

	θ_1	θ_2	...	θ_j	...	θ_n
ψ_1	0.4137	0.3987	...	0.1863	...	0.2313
ψ_2	0.3937	0.3787	...	0.1663	...	0.2113
...	...	...	...	...	...	...
ψ_i	0.3737	0.3587	...	0.1463	...	0.1913
...	...	...	...	...	...	...
ψ_n	0.4337	0.4187	...	0.2063	...	0.2513

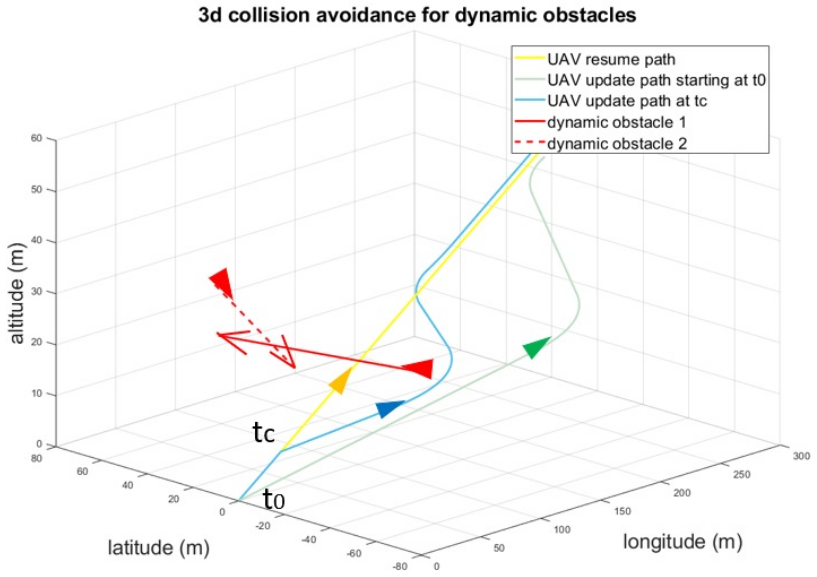


Figure 3.3: 3D avoidance for dynamic obstacles

3.7 Summary of FGA

In summary, FGA provides a faster and successful collision avoidance for the UAV while shortening computation time with consideration of the minimum turning radius and the nominal speed.

The contribution of FGA in determining the bounds of geometry based collision avoidance methods is provided at chapter 4. The contribution of FGA in determining the sequence of obstacles being avoided in case to manage crowded obstacles avoidance problems is presented at chapter 5. The contribution of FGA in shortening the length of the updated path, decreasing missed waypoints and the affected area will be proved at chapter 6.

Chapter 4: Geometry Collision Avoidance Algorithm Bounds

In this chapter, the bounds of the application for the geometric obstacles avoidance methods will be provided. They include the bounds of direction, speed and the limitation of obstacles' number in 2D and 3D space.

4.1 2D Avoidance Bounds

Bound 1: Boundary on the initial distance to guarantee $t_c > 0$

Based on the calculation of critical avoidance time in Eq. 3.14, a critical avoidance time which guarantee the possibility for UAV to escape from the conflicts exists if:

$$t_c = \frac{r_0 cos\alpha_0 - \rho_{min} - (d_m - r_0 sin\alpha_0)}{||V_r||} > 0, \tag{4.1}$$

that is:

$$r_0 cos\alpha_0 - \rho_{min} - (d_m - r_0 sin\alpha_0) > 0, \tag{4.2}$$

Rearrange it and get the initial detection radius bound:

$$r_0 > \frac{\rho_{min} + d_m}{cos\alpha_0 + sin\alpha_0}, \tag{4.3}$$

where α_0 is the angle between the relative velocity vector and relative location vector of the UAV and the obstacle.

For n obstacles, the UAV should satisfy the distance constraint for each obstacle:

$$r_{0i} > \frac{\rho_{min} + d_{mi}}{cos\alpha_{0i} + sin\alpha_{0i}}, i = 1, 2, ...n. \tag{4.4}$$

Bound 2: Boundary on the UAV velocity and obstacles velocity

In Figure 4.1, the initial location of the UAV is set at U(0). The initial dynamic obstacle location is $A(A_x, A_y)$. The velocity of the UAV is $\overrightarrow{V_u}$, pointing in the positive x axis for simplicity. The velocity of the obstacle is $\overrightarrow{V_a}$. The distance between the UAV and the obstacle detected is r_0.

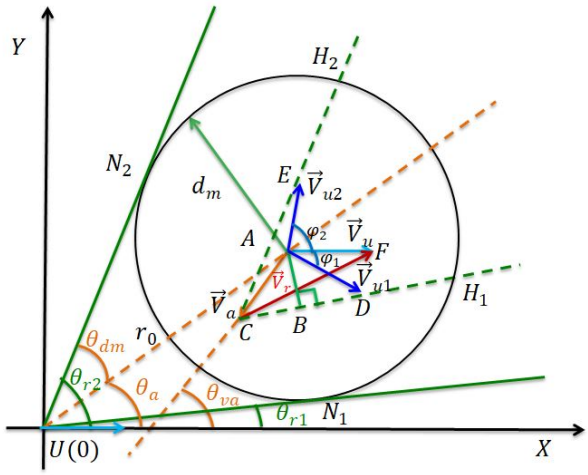


Figure 4.1: 2D collision avoidance geometry for Bound 2: UAV can turn clockwise or counterclockwise to avoid crossing protected area d_m in 2D space.

The safety cone is defined by tangent line UN_1 and UN_2 in Figure 4.1. If the relative velocity $\overrightarrow{V_r}$ remains outside the cone defined by UN_1 and UN_2, the collision avoidance operation is successful.

Let's now move the velocity vector of UAV to the obstacle center A such that $\overrightarrow{AF} = \overrightarrow{V_u}$, and $|AF| = |\overrightarrow{V_u}|$. The velocity of the obstacle, $\overrightarrow{V_a}$ is generated in random

directions in this obstacle avoidance simulation and has the absolute value $|\overrightarrow{AC}| = \overrightarrow{V_a}$. Draw a line segment $CH_1//UN_1$. In order to avoid a collision, the relative velocity is rotated clockwise or counterclockwise. Consider the clockwise solution first, the relative velocity needs to be at least in the direction of CH_1 or outside line CH_1. This means the UAV needs to rotate φ_1 radians to $\overrightarrow{V_{u1}}$ ($|\overrightarrow{V_u}| = |\overrightarrow{V_{u1}}|$) and have at least one intersection point with line CH_1, or point D. Hence, there is $\overrightarrow{AD} = \overrightarrow{V_{u1}}$. This constraint is equivalent to letting triangle $\triangle ACD$ exist, which requires $AD \geq AB$. Where $|AB|$ is perpendicular to line CD. Thus,

$$V_u = |\overrightarrow{V_u}| = |\overrightarrow{AD}| \geq |AB|, \tag{4.5}$$

so

$$V_u \geq |AB|. \tag{4.6}$$

In the triangle $\triangle ACD$,

$$|AB| = |AC||sin\angle ACD| \tag{4.7}$$

Where,

$$\angle ACD = \theta_{va} - \theta_{r1} \tag{4.8}$$

and,

$$|AC| = |\overrightarrow{V_a}| = V_a \tag{4.9}$$

That is:

$$|AB| = |AC||sin\angle ACD| = Va|sin(\theta_{va} - \theta_{r1})| \tag{4.10}$$

and since $V_u \geq |AB|$, we now have:

$$V_u \geq V_a|sin(\theta_{va} - \theta_{r1})| \tag{4.11}$$

or,

$$V_a \leq V_u/|sin(\theta_{va} - \theta_{r1})|. \tag{4.12}$$

Where θ_{va} is the direction of the obstacle velocity and $\theta_{va} = tan^{-1}(\frac{V_{ay}}{V_{ax}})$. θ_{r1} is the angle between UN_1 and the x-axis as Figure 4.1 shows. hence:

$$\theta_{r1} = \theta_a - \theta_{dm} = tan^{-1}(\frac{A_y}{A_x}) - sin^{-1}(\frac{d_m}{r_0}). \tag{4.13}$$

Where θ_a is the direction of the obstacle location and $\theta_a = tan^{-1}(\frac{A_y}{A_x})$. Therefore, Equation 4.12 becomes:

$$V_a \leq V_u/|sin[tan^{-1}(\frac{V_{ay}}{V_{ax}}) - tan^{-1}(\frac{A_y}{A_x}) + sin^{-1}(\frac{d_m}{r_0})]|. \tag{4.14}$$

So for a UAV flying at constant speed, the speed of the obstacle and the UAV must satisfy Equation 4.14 for a successful obstacle avoidance operation using geometry-based methods.

Similarly, if we look at the counterclockwise solution, we have that:

$$V_u \geq V_a|sin(\theta_{r2} - \theta_{va})| \tag{4.15}$$

or,

$$V_a \leq V_u/|sin(\theta_{r2} - \theta_{va})| \tag{4.16}$$

Then, the analogous solution becomes:

$$V_a \leq V_u/|sin[tan^{-1}(\frac{A_y}{A_x}) + sin^{-1}(\frac{d_m}{r_0}) - tan^{-1}(\frac{V_{ay}}{V_{ax}})]|. \tag{4.17}$$

Thus, to guarantee at least one solution, the UAV velocity needs to satisfy either the clockwise or counterclockwise conditions:

$$V_u \geq min(V_a|sin(\theta_{va} - \theta_{r1})|, V_a|sin(\theta_{r2} - \theta_{va})|) \tag{4.18}$$

If there are n obstacles, the velocity of the UAV should satisfy the constraint for every obstacle:

$$V_u \geq min(V_{ai}|sin(\theta_{vai} - \theta_{r1i})|, V_{ai}|sin(\theta_{r2i} - \theta_{vai})|), i = 1, 2, ...n \quad (4.19)$$

Bound 3: Boundary on the UAV angles:

From our previous discussion in chapter 3, the UAV needs to rotate at least clockwise by ψ_1 or counterclockwise by ψ_2 to avoid a collision. Hence, the direction variation of the UAV is: $\psi \leq -\psi_1$ or $\psi \geq \psi_2$. Written as a uniform set:

$$\psi \subseteq \Psi_i. \quad (4.20)$$

For n obstacles:

$$\psi \subseteq (\Psi_1 \cap \Psi_2 \cap ...\Psi_n,) \quad (4.21)$$

Therefore, to let the solution exist:

$$(\Psi_1 \cap \Psi_2 \cap ...\Psi_n) \neq \emptyset. \quad (4.22)$$

4.2 3D Avoidance Bounds

The bounds in the 2D plane can be extended into the 3D space.

Bound 1: Extending Equation 3.14 to 3D space, the analogous margins are applied. In this scenario, $\overrightarrow{r_0}$ and $\overrightarrow{V_r}$ are the three-dimension vectors and α_0 is a 3D solid angle.

Bound 2: the extension of the 2D collision cone to the one in 3D space is shown in Figure 4.2. Inside this cone, another cone K which starts at the point C is

defined. The axis of this smaller cone is parallel to line UA and the apex angle of the two cones is the same.

If the avoidance solution exists, the sphere centered at the obstacle A with the radius V_u must at least have a tangent point with the cone K. To find this point, cut the cone K by a plane passing through AC and the cone's axis. In this cut-plane, there will be two angles; one between the line AC and the cone edge and another one between the cone edge and the axis. Figure 4.2 presents the 3D cones with the smaller angle produced by the plane-cut, symbolized by σ. Therefore, as proved in the 2D case, if a solution exists:

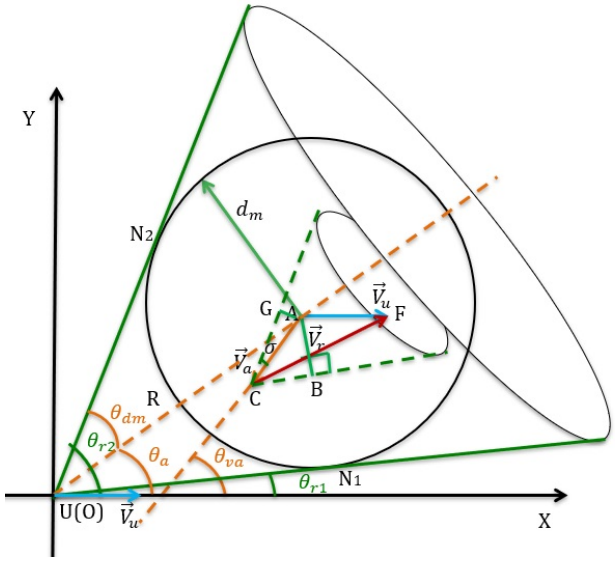


Figure 4.2: 3D collision avoidance geometry for Bound 2.

$$V_u \geq V_a|sin\sigma|. \tag{4.23}$$

For n obstacles, the solution exist if:

$$V_u \geq V_{ai}|sin\sigma_i|, i = 1, 2, ...n. \tag{4.24}$$

Bound 3: Suppose the direction of the solid angle that the UAV needs to be changed for avoiding the i_{th} obstacle is:

$$\Omega \subseteq \Omega_i. \tag{4.25}$$

Then for n obstacles:

$$\Omega \subseteq (\Omega_1 \cap \Omega_2 \cap ...\Omega_n), \tag{4.26}$$

Thus, the solution exists only if:

$$(\Omega_1 \cap \Omega_2 \cap ...\Omega_n) \neq \emptyset. \tag{4.27}$$

In 3D space, the UAV can rotate by a full 4π solid angle, which gives the UAV much more free space to avoid the collisions than in the 2D plane.

In summary, the bounds of UAV and obstacles' speed, direction and the number of obstacles are the basic prerequisites for a successful collision avoidance action using geometry-based algorithms.

Chapter 5: Class the Obstacles by Time

The critical avoidance trigger time not only could be used for determining collision operation starting time, but also benefits in simplifying the complicated avoidance scenario through classing the urgent level of the obstacles in quantitative. Section 5.1 describes the working mechanism of FGA in classification of the obstacles while section 5.2 validates the theory by simulation results.

5.1 Simplification of Complicated Obstacle Avoidance Scenarios

From chapter 4, if a geometric method has a collision avoidance solution for a single dynamic obstacle, the following equations of initial detection distance r_0, speed V_u and direction changing Ω should be satisfied:

$$t_c = \frac{r_0 cos\alpha_0 - \rho_{min} - (d_m - r_0 sin\alpha_0)}{||V_r||} > 0, \tag{5.1}$$

$$r_0 > \frac{\rho_{min} + d_{mi}}{cos\alpha_0 + sin\alpha_0}, i = 1, 2, ...n. \tag{5.2}$$

$$V_u \geq V_a |sin(\theta_{va} - \theta_r)| \tag{5.3}$$

$$\Omega \subseteq \Omega_{range}. \tag{5.4}$$

where Ω_{range} is the maximum range of the UAV yaw(pitch)angle variation at this moment.

For n obstacles at the same time, the UAV should satisfy the following equations for each obstacle:

for distance:

$$r_{0i} > \frac{\rho_{min} + d_{mi}}{cos\alpha_{0i} + sin\alpha_{0i}}, i = 1, 2, ...n. \tag{5.5}$$

for speed:

$$V_u \geq max(V_{ai}|sin(\theta_{vai} - \theta_{ri})|)i = 1, 2, ...n. \tag{5.6}$$

for direction(yaw and pitch angle):

$$\Omega_i \subseteq \Omega_{range}. \tag{5.7}$$

$$\Omega \subseteq (\Omega_1 \cap \Omega_2 \cap ...\Omega_i...\Omega_n) \neq \emptyset. \tag{5.8}$$

However, as the number of the obstacles increases, the UAV might not be able to satisfy the above equation at the same time. Thus, to deal with this situation, it is necessary to determine the sequence of obstacles being avoided. Depending on the different classification of obstacles, there could be several algorithms.

First, avoiding the obstacles by their relative distance to the UAV from closest to the most faraway. This algorithm is direct and uses the data from the UAV onboard sensors. However, obstacles faraway from the UAV could have a higher velocity, which makes them to approach the UAV quickly.

Oppositely, the second algorithm is avoiding the obstacles from highest velocity to lowest. The similar problem is that the obstacles with lower speed but are closer to the UAV have the possibility to approach the collision point faster. Therefore this method is not applicable, neither.

Third one is to calculate the time to collision for each obstacle. This method is much more reasonable than the above two methods but still has a remaining issue. That time to the collision is not equal to the time for UAV to escape, which is the time determining if a collision avoidance action is possible or not.

Hence, in this work an algorithm based on FGA critical avoidance trigger time (the final time that the UAV could escape from the conflicts) is developed to decide the sequence of obstacles going to be avoided. Here are the major steps:

(1)For each obstacle, calculate the critical avoidance starting time,

$$t_{ci} = \frac{r_{0i}cos\alpha_{0i} - \rho_{min} - (d_m - r_{0i}sin\alpha_{0i})}{||V_{ri}||}, \tag{5.9}$$

(2)Sort the t_{ci}.

(3)Pick up the n/2 obstacles with smaller t_{ci} and calculate the solution. If the solution exists, go to (4), else repeat (3).

(4)Let UAV follow the solution to avoid the first half obstacles. Calculate the collision avoidance solution for the remaining half obstacles. Repeat the steps until all obstacles are avoided.

Algorithm 4 describes the simulation setup and the coding process in the simulation environment.

Algorithm 4 Avoid the obstacles in sequence by critical avoidance time t_c

Precondition: necessary data is accessible for UAV

1: **for** $i \leftarrow 1$ to n **do**

2: $t_{ci} \leftarrow \dfrac{r_0 cos\theta_{t0} - \rho_{min} - (d_m - r_0 sin\theta_{t0})}{||V_r||}$ ▷ critical avoidance time

3: **end for**

4: sort t_{ci} from smallest to largest

5: calculate the solution for half of the current examined obstacles with smaller t_{ci}.

6: **if** solution **then**

go to line 9

7: **else**

repeat line 5

8: **end if**

9: follow the solution to avoid the collisions.

10: calculate the solution for remain half of the obstacles based on current positions, speed, direction as well as the new t_{ci}.

11: repeat above steps until all obstacles are avoided.

5.2 Simulation Results

Table 5.1 presents some scenarios that need the obstacles classification. In those scenarios, avoiding all obstacles at the same time is impossible for the UAS after computation of their geometry relationships. Thus, avoiding them in a time sequence is necessary. Table 5.1 presents the UAV obstacle avoidance by sequence based on their critical avoidance trigger time and provides the successful results.

Table 5.1: Cost function to determine command direction in 3D avoidance maneuver

scenarios	Num. of obstacle	Group classification	Avoidance result
scenario 1	2	1,1	success
scenario 2	5	2,3	success
scenario 3	3	1,2	success
scenario 4	7	2,2,3	success
scenario 5	7	3,4	success

The Figure 5.1 shows the first scenario from the table. The blue path is the trajectory for UAV while the two red paths are the trajectories for the moving obstacles. The UAV encounters two obstacles in $t = 10s$ and $t = 15s$. The critical avoidance trigger times for them are $t = 7s$ and $t = 11s$. For the first obstacle, the UAV needs to turn left more than 1.6 rad or turn right more than 0.3 rad to avoid the obstacle. For the second obstacle, the UAV needs to turn left more than 0.5 rad or turn right more than 1.57 rad to avoid the obstacle. However, for this UAV model in this situation, the turning angle should within the range$[\psi - 0.5\pi, \psi + 0.5\pi]$, thus the final solution set of avoiding the two obstacles at

the same time is $\Omega \subseteq (\Omega_1 \cap \Omega_2 \cap ...\Omega_i...\Omega_n) = \emptyset$.

In this condition, class the obstacles into two groups with one each. Let UAV to avoid the one with smaller critical trigger time $t = 7s$ first and then go to the second one with the critical avoidance time $t = 11s$. The UAV finally finished this collision avoidance task.

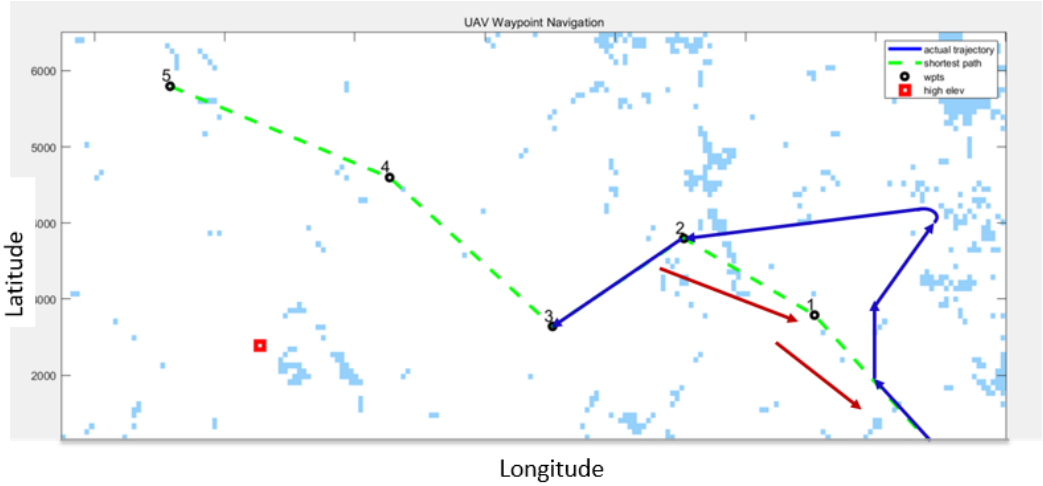


Figure 5.1: Divide obstacles into smaller groups by critical avoidance trigger time

In summary, class multiple obstacles into smaller groups and avoid them by sequence based on the critical time is valid to be efficient for those complicated collision avoidance scenarios which have lots of dynamic obstacles.

Chapter 6: UAV Collision Avoidance Efficiency and Impact on the Neighborhood

This chapter discussed about the selection for the different collision avoidance trigger times t_c would have different results on UAV's safety and avoidance efficiency as well as finds the optimal trigger time t_t for scenarios. The trigger time is defined on chapter 3, indicating when a collision avoidance maneuver begins. The earlier avoidance operation trigger time is safer for conflicts avoidance due to the more response space but results in the UAV to deviate faraway from its intended trajectory. Oppositely, a delayed avoidance trigger performance reduces overall path deviation but it has a higher risk of collision. Therefore, the balance between collision avoidance efficiency and safety is important. Simulations for different mission scenarios prove that by selecting proper avoidance action trigger times, the impacts caused by collision avoidance such as missed waypoints, length of updated path and impacted space due the avoidance maneuver could be reduced by over 40%, while keeping an acceptable safety level. Hence, the proper choice of the collision avoidance trigger time can improve the UAV collision avoidance performance and make it applicable for real-time situations.

6.1 Definition of Efficiency and Impacts

Very few studies research on the effects of different avoidance starting times and ending times have on algorithm outcomes, including efficiency of conflict resolutions.

However, after conducting twenty thousand of simulations, it is found that even the same collision avoidance approach is applied to the same scenario but with different avoidance action trigger times, the results such as the collision avoidance success possibility and the time cost are significantly different.

In [1] and chapter 3, the critical avoidance time for the UAV to escape from the collision at the last minute is calculated as well as the proof of the shortest renewal path. However, the corresponding quantitative analysis of the cost and efficiency for this collision avoidance action is not provided. Also, the variations of the obstacle avoidance success rate when UAV starts its escaping maneuver at t_t, which is the time point between t=0 and critical trigger time t_c are not explored.

In this chapter, we define several "impacts" variables that are the opposite of the efficiency and relative to the UAV collision avoidance performance. And the smaller the impact, the higher the efficiency will be.

(1) P_{succ}, rate of collision avoidance success, that is the percentage the UAV avoids the collisions successfully with the avoidance trigger time t_t when tested with the randomly generated scenarios.

(2) E_W, waypoint-related impact: the portion of waypoints that UAV misses due to collision avoidance operation.

(3) E_L, path-related impact: the length of the new path generated compared

to the original path [46] due to the collision avoidance action.

(4) E_S, space-related impact: ratio of the space affected by collision avoidance operation.

(5) E_T, time-related impact: ratio of the time that the UAV deviates from its original airway.

(6) E_{total}, average of the above variables, E_W, E_L, E_S and E_T,

The following sections would analyze these variables in sequence and validate the advantages of the proper selection of the collision avoidance trigger times.

6.2 Waypoints Missed

Figure 6.1 presents the scenario that the waypoints located inside the collision zone would be missed since the UAV will avoid this unsafe area due to collision avoidance performance. As Figure 6.1 shows, the UAV (located at U) detects the obstacles at $t = 0$. If it starts to avoid the obstacles immediately (the blue trajectory), the 1st, 2nd 3rd and 4th waypoints W_1 W_2 W_3 and W_4 will be missed. Oppositely, if starting later at time t_c (green trajectory), the 3rd and 4th waypoints W_3 and W_4 will not be followed by the UAV while 1st, 2nd and 5th waypoints are tracked. The ratio of missed waypoints by the total impact, E_w.

Suppose the number of total waypoints in the UAV's intended path is N. The position of UAV when it detects the obstacles is U, the conflict position is C, and the number of waypoints between U and C is M as Figure 6.1 presents.

Assume the critical time at which the UAV can escape the collides with obstacles

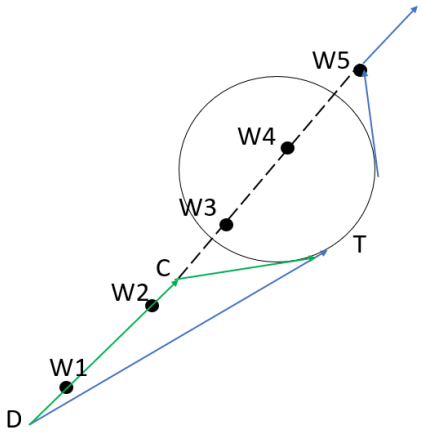


Figure 6.1: UAV misses some of waypoints when avoiding conflicts

is t_c, the time of obstacles detection is t_0, the actual time UAV starts the avoidance is t_t. Hence, the approximate number of waypoints missed M_w is

$$M_w = M\frac{t_c - t_t}{t_c - t_0} \tag{6.1}$$

Where $t_c > t_t \geq t_0$. When t_t increases from t_0 to t_c, the number of missed waypoint M_w decreases. The impact associated with the missed waypoints is defined,

$$E_w = \frac{M_w}{N} = \frac{M}{N}\frac{t_c - t_t}{t_c - t_0} \tag{6.2}$$

6.3 Additional Path Length

Study [47] points out that different trigger times result in a different path length. In Figure 6.1, line $\widehat{CTW_5}$ represents the UAV's path where the avoidance action starts at $t = t_c$, and the blue line $\widehat{DTW_5}$ represents the UAV's path where the avoidance action starts at $t = 0$. From the work [1], it's concluded that the $\widehat{CTW_5}$

is shorter than the $\widehat{DTW_5}$ in almost all situations and the detailed proof for this conclusion is provided on section 6.3.1.

The ratio of the additional length over the UAV original trajectory's length is considered as the impact of path, E_L. Assume the length of UAV original path from start to the destination is L_0, and the additional path due to collision avoidance action has length L. Then the impact corresponding to the UAV's path is defined,

$$E_L = \frac{L}{L_0} \tag{6.3}$$

6.3.1 Shorter Path for Single Obstacle

This section provides a comparison of the length of the updated path, starting at different times to perform collision avoidance. The proof shows that FGA, which starts at time t_c, has the shorter updated path as compared to the path starting from t_0.

It is proved that the deviation of the UAV when starting the avoidance process from t_0 (the time at which the obstacle is detected), will be much longer than UAV starting from t_c (the critical avoidance time). Where the deviation is classified as the total time spent away from the initial heading after the avoidance action, with a approximate constant velocity. This is equivalent to path length away from the initial heading. That is: $L_3 > L_1$, these paths are shown in Figure 6.2.

ρ is the minimum turning radius for UAV. $\alpha = \angle BAD$, where point B is the tangent point of the path L_1 and the d_m is the required separation radius between obstacle A and the UAV. $\beta = \angle H_2AD$, where point H_2 is the tangent point of the

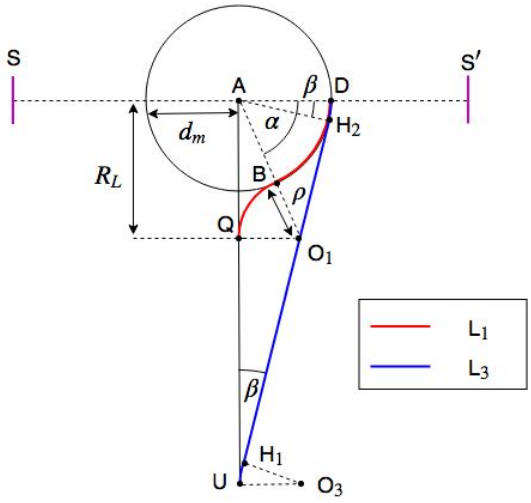


Figure 6.2: L_1 and L_3 path comparison

path L_3 and circle A.

Theorem 1: For any trigger time $0 \leq t_t \leq t_c$ satisfying the Equation 6.7, the updated path starting from t_t is shorter than the updated path starting from t_0.

Proof. $L_1 : Q \rightarrow B \rightarrow D$,

$L_3 : U \rightarrow H_1 \rightarrow H_2 \rightarrow D$.

$L_1 = \int dl = LB + BD = \alpha\rho + \alpha d_m$

$L_3 = \int dl = UH_1 + H_1H_2 + H_2D$

$\alpha = arcos\frac{\rho}{\rho + d_m}$

Let $AU = r_0$, where r_0 is the initial distance between the UAV and obstacle when the obstacle is first detected.

Since the minimum distance between two points is the straight line, $L_3 = UH_1 + H_1H_2 + H_2D > UD$, where UD is the straight line from point U to point D. Let $L_{30} = UD$. Thus, $L_3 > L_{30}$ and the following will compare the length between

L_1 and L_{30}.

Let $\rho = k_1 d_m, r_0 = k_2 d_m$, we have that:

$$\alpha = arcos(\frac{\rho}{\rho + d_m}) = arcos(\frac{k_1}{k_1 + 1}) \tag{6.4}$$

$$L_{30} = \sqrt{r_0^2 + d_m^2} = d_m(\sqrt{k_2^2 + 1}) \tag{6.5}$$

$$L_1 = (d_m + \rho)\alpha = d_m(k_1 + 1)\alpha \tag{6.6}$$

where $0 < \alpha < \frac{\pi}{2}$,

While α is a function of k_1, which decreases, let $\alpha = \frac{\pi}{2}$ and $L_1^* = d_m(k_1 + 1)\frac{\pi}{2}$ for brevity of this proof. Then, here is $L_1^* > L_1$. Thus, if L_3 is bigger than L_1^* then it can also be extended to L_1. Let's calculate the necessary lower bound for k_2.

$L_1^* = d_m(k_1 + 1)\frac{\pi}{2}$

Let $L_{30} > L_1^*$, then:

$d_m(\sqrt{k_2^2 + 1}) > d_m(1 + k_1)\frac{\pi}{2}$, thus we have:

$$k_2 > \sqrt{\frac{\pi}{4}(1 + k_1)^2 - 1} \tag{6.7}$$

□

The Equation 6.7 is the lower bound of k_2. For any k_2 satisfies this equation, $L_{30} > L_1^*$. From the precondition of the FGA where $k_2 >> k_1$, eq 6.7 is always satisfied in real world scenarios since detection radius r_0 is always much larger than the minimum turn radius ρ. Since $L_3 > L_{30}$, $L_{30} > L_1^*$ and $L_1^* > L_1$, it is concluded that $L_3 > L_1$.

6.3.2 Shorter Path for Multiple Obstacles

For multiple obstacle cases, each moving obstacle has its own critical avoidance trigger time t_{ci} associated with it. Therefore, for n multiple obstacles, the overall critical time t_c, as calculated in Equation 3.13, is the minimum of all calculated t_{ci}:

$$t_c(theory) = min(t_{ci}), i = 1, 2, 3...n. \tag{6.8}$$

After taking vehicle response time into account:

$$t_c = t_c(theory) - t_{res} = min(t_{ci}) - t_{res}; i = 1, 2, 3...n. \tag{6.9}$$

For a multiple obstacles avoidance scenario, it can be roughly classified into two cases:

Case 1. More than one obstacles have the overlaps with the initial path of the UAV, as shown in Figure 6.3(a).

Case 2. Only one obstacle has overlap with the initial path of the UAV, as shown in Figure 6.4(a).

In Case 1, the area occupied by the multiple obstacles can be equivalent into a single obstacle with larger safe distance d_e, as shown in Figure 6.3(b). Thus, proof of $L_1 < L_3$ in this scenario is changed to the proof under single obstacle situation, which has been provided before.

In Case 2, there is more than one obstacle in the future path of the UAV, but only one obstacle is in direct conflict with the UAV's initial path. Thus, other obstacles can be ignored. The task to prove $L_1 < L_3$ is equal to the proof in a single obstacle situation, which has been provided, as shown in Figure 6.4(b).

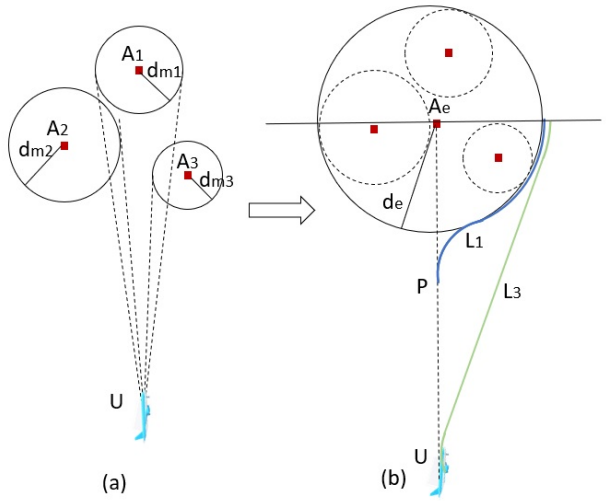


Figure 6.3: Multiple obstacles: Obstacles overlap with initial path of the UAV

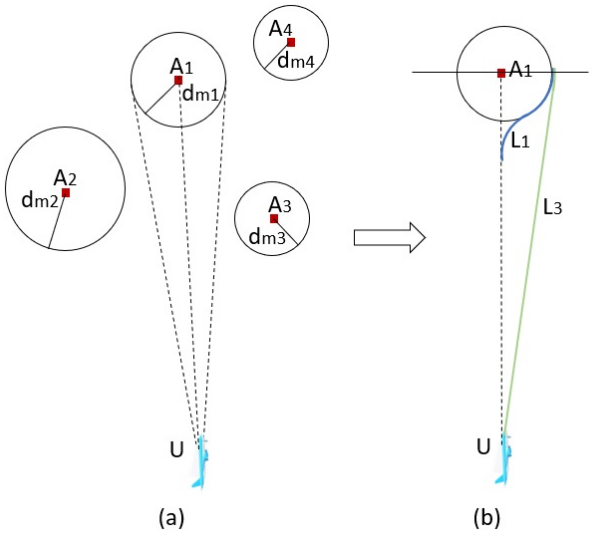


Figure 6.4: Multiple obstacles: No obstacles overlapped with initial path of the UAV

6.4 Smaller Space Occupation due to Avoidance Operation

The areas affected by the UAV collision avoidance action is defined as avoidance operation space S. Though the avoidance action is in 3-Dimension with yaw angle, pitch angle and speed variation, for easy calculation, the space is claimed as the surface instead of the volume, which is computed as when UAV starts its obstacle avoidance action, the planar space surrounded by the original path, and the new path. It might be a 3D curved surface or a planar and the Figure 6.1 shows a example of it. In the Figure 6.1, if UAV starts avoidance at the time $t = t_0$ (the time UAV first detects the obstacles), the corresponding operation space is S_1, that is ΔDTW_5. Oppositely, if the UAV starts the avoidance action at t_c, the corresponding affected space is S_2, that is ΔCTW_5. S_1 and S_2 have different size. Note: in the Figure 6.1 , for clear explanation, the resumed path of UAV is drawn as a straight line. However, in simulation or real experiments, the initial waypoints are generated randomly or by nature marks, which results in the curved path, but the definition of impacted space remains similarly.

The length of the original path from start to the destination is L_0. Since the UAV has separation distance d_m to keep safe, the space used during the flight is,

$$S_0 = \int_0^{L_0} 2d_m dl \tag{6.10}$$

The avoidance impacted space is S. Thus, the impact of space occupied is defined as,

$$E_S = \frac{S}{S_0} = \frac{\int_c^g (d(t) + d_m) dl}{2d_m L_0} \tag{6.11}$$

Where $d(t)$ is the distance for UAV to its initial path, varying with time. c is the point UAV starts collision avoidance and g is the point the UAV comebacks to its initial path. The quantitative comparison of the avoidance space size due to different trigger times is provided in the simulation section.

6.5 Impacts on the Environment

Different impacts on the environment (other air vehicles' airways) are found by the different collision avoidance trigger times. As shown in Figure 6.5, initially the UAV travels in $Airway_1$ with yellow line. The width of the $Airway_1$ is $2d_m$. To avoid obstacles located in its future path, the UAV deviates from its original path. The light blue trajectory represents that the UAV deviates at time t_0 while the green trajectory means it deviates at time t_t. The time during which the UAV travels outside the $Airway_1$(green trajectory or blue trajectory beyond the $Airway_1$) is defined as effect time T_{eff}, which has the potential to affect another aircraft nearby. The percentage of the effect time over the original time consumed to finished flight task is E_T.

If T is the total time taken by the UAV to complete its mission with obstacles-free path, then the E_T is defined as the following ratio:

$$E_T = \frac{T_{eff}}{T} \tag{6.12}$$



Figure 6.5: Collision avoidance action deviates the UAV from its original airway and may affect the space of other aircraft (schematic is not drawn to scale in airspace)

6.6 Total Impacts and Reward Function

The total cost function for a UAV's flight mission and conflict resolution can be defined in many ways, as previously mentioned [48] [49] and [50], depending on flight plan goals and overall scenario. In this paper, to class the internal cost of UAV when it follow the certain command angles and the outside cost such as waypoints missed and additional path, we name the outside cost to be "impact" since they might have impacts on the environment and airway of other aircraft. The overall impact function is defined as,

$$E_{total} = (E_w + E_L + E_S + E_T)/4 \tag{6.13}$$

The reward function is,

$$Reward = P_{succ}/E_{total} \tag{6.14}$$

Considering for the clear reading when plotting the overall impact, success rate and the reward in the same figure, the final reward calculated would be,

$$Reward = 10P_{succ}/E_{total} \tag{6.15}$$

6.7 Set Avoidance Operation Trigger Times

After calculating the critical avoidance time t_c, the next step is to set the trigger time. The UAV could choose to trigger the collision avoidance anytime between 0 and t_c. In our experiments, the test times will be chosen in a uniform distribution from 0 to t_c, as presented in Table 6.1.

Table 6.1: Setup of the different avoidance trigger times

trigger time t_t/t_c	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0

For easier visualization of results, results of $t_t = 0$, $t_t = 0.2t_c$, $t_t = 0.5t_c$, $t_t = 0.8t_c$, $t_t = t_c$ will be presented in the tables and figures in section 6.8, including missed waypoints, path length, space occupation, total impact and the total reward. The Algorithm 5 explains the processes involved in the trigger time selection and reward calculation.

Algorithm 5 Trigger time selection and reward

Precondition: necessary data is available for UAV

1: **for** $i \leftarrow 1$ to n **do**

2: calculate critical trigger time t_{ci} for avoidance of each obstacle using Equation 3.10.

3: **end for**

4: Get the overall trigger time $t_c = min(t_{ci}) - t_{res}$

5: Set the sequence of avoidance trigger time in an uniform distribution, $t_t = 0.1t_c, 0.2t_c, ...t_c$

6: Run simulations, get the E_{total} and the *Reward* using Equation 6.13 and 6.15.

7: Pick up the time t_{opt} which has the highest *Reward* as the optimal collision avoidance trigger time.

6.8 Simulation Results and Analysis

Table 6.2 displays two scenarios of collision avoidance, case 1 for three static obstacles and case 2 for two dynamic obstacles.

For case 1, the UAV detects static obstacles from 70-100 meters away and calculates the critical avoidance time $t_c = 3s$. (1) If the UAV starts the avoidance operation right away, that is $t_t = 0$, it can avoid the obstacles successfully with 30% missed waypoints. The path length changed by 37m, the planar air space impacted by the collision avoidance operation was 432 m^2 and the UAV flew beyond its airway for 9.49 seconds. (2) If the UAV starts the avoidance operation at $t_t = 0.2t_c$, $t_t = 0.5t_c$, $t_t = 0.8t_c$, $t_t = t_c$, percentage of missed waypoints are 20%, 20%, 10%, 10%. The

path lengths changed by 33m, 29m, 26m and 23m. The planar airspace impacted was $405m^2$, $370m^2$, $319m^2$ and $290m^2$, respectively. And the UAV traveled beyond its air corridor by 8.23 seconds, 5.95 seconds, 5.84 seconds and 5.12 seconds. All of the variables have smaller values (smaller cost) as compared to when the collision avoidance stating at time $t = 0$ and the avoidance action remains successful.

For case 2, the UAV detects dynamic obstacles from 70-100 meters away and calculates the critical avoidance time $t_c = 1.5s$. (1) If the UAV starts the avoidance operation right away, that is $t_t = 0$, it can avoid the obstacles successfully with 30% missed waypoints. The path length changed by 35m, the planar air space impacted by the collision avoidance operation was 450 m^2 and the UAV flew beyond its airway for 11.05 seconds. (2) If the UAV starts the avoidance operation at $t_t = 0.2t_c$, $t_t = 0.5t_c$, $t_t = 0.8t_c$, $t_t = t_c$, percentage of missed waypoints are 20%, 10%, 10%, 10%. The path lengths changed by 32m, 27m, 25m and 20m. The planar airspace impacted was $337m^2$, $325m^2$, $287m^2$ and $256m^2$, respectively. And the UAV traveled beyond its air corridor by 8.87 seconds, 7.49 seconds, 7.47 seconds and 5.95 seconds. All of the variables have smaller values (less cost) as compared to when the collision avoidance starting at time $t = 0$ and the avoidance action remains successful.

It is clear that as the trigger time t_t gets closer to the critical avoidance time t_c, the overall impact, including waypoints missed, additional path, occupied space and the time flying outside the original lane is decreased. This is further indicated by the Monte-Carlo simulation results in Table 6.3, which summarizes the results for the success rate of avoiding static and dynamic obstacles and their corresponding cost(impact). From the results, similar to Table 6.2, as the trigger time t_t gets closer

Table 6.2: Avoidance operation cost by different trigger time

cases	case 1	case 2
Number and type of obstacles	3 static	2 dynamic
Initial distance	70-100	70-100
Initial altitude	550-650	550-650
Speed m/s	0	15-20
Total number of waypoints	10	10
Collision avoidance starts at $t = 0$ **Avoidance success?**	yes	yes
Waypoints missed num	3	3
Waypoint missed percentage	30%	30%
Additional path length /m	37	35
Space occupied $/m^2$	432	350
Time outside airway /s	9.49	11.05
Collision avoidance starts at $0.2t_c$ **Avoidance success?**	yes	yes
Waypoints missed num	2	2
Waypoint missed percentage	20%	20%
Additional path length /m	33	32
Space occupied $/m^2$	405	337
Time outside airway /s	8.23	8.87
Collision avoidance starts at $0.5t_c$ **Avoidance success?**	yes	yes
Waypoints missed num	2	1
Waypoint missed percentage	20%	10%
Additional path length /m	29	27
Space occupied $/m^2$	370	325
Time outside airway /s	5.95	7.49

Collision avoidance starts at $0.8t_c$		
Avoidance success?	yes	yes
Waypoints missed num	1	1
Waypoint missed percentage	10%	10%
Additional path length /m	26	25
Space occupied $/m^2$	319	287
Time outside airway /s	5.84	7.47
Collision avoidance starts at t_c		
Avoidance success?	yes	yes
Waypoints missed num	1	1
Waypoint missed percentage	10%	10%
Additional path length /m	23	20
Space occupied $/m^2$	290	256
Time outside airway /s	5.12	5.95

to the critical avoidance time t_c, the overall impacts, including waypoints missed, length of additional path, occupied space and the time outside its airway decrease.

In Table 6.3, the second column depicts the UAV avoiding a single static obstacle. Single static obstacle avoidance is a simple scenario, thus under the proper initial detection distance between the UAV and the obstacle, avoidance success rate remains approximately 100%. But the impacts of the avoidance operation vary. As trigger time t_t changes from 0, $0.2t_c$, $0.5t_c$, $0.8t_c$ to t_c, the impacts, including E_w, E_L, E_S and E_T decrease monotonously. As a result, the total reward gained from Equation 6.15 as $10P_{succ}/E_{total}$, increases from 28.65, 32.79, 37.45, 44.84 to 52.63, which means choosing a later collision avoidance trigger time would be better.

Now let's take a look at the fourth column of Table 6.3, the UAV avoiding a single dynamic obstacle. The single obstacle avoidance is not a tough task if the initial detection distance between the UAV and the obstacle is not a very small margin. And the success rate remains approximately the same as the trigger time varies. But the impacts vary. As the trigger time t_t changes from 0, $0.2t_c$, $0.5t_c$ to $0.8t_c$, t_c, the impacts including E_w, E_L, E_S and E_T decreases monotonously from 0.286, 0.278, 0.239 to 0.223 and 0.207. As a result, the total reward calculated from Equation 6.15 increases oppositely from 35.03, 35.97, 41.84 to 44.84 and 48.31, which means choosing a later collision avoidance trigger time is better for the UAV.

The analysis of situations for multiple static obstacles is more complex. From Table 6.3 third column, it is clear that the safety level (avoidance success rate) of static obstacles decreases as t_t increases, which means the failure rate increases. The total impact E_{total}, composed of missed waypoint $Cost_w$, additional path impact E_L,

Table 6.3: Monte carlo simulation statistic results of UAV avoiding obstacles by different trigger times

Num of cases	5000	5000	5000	5000
Num of obs each case	single	multi	single	multi
Obstacle type	static	static	dyn	dyn
Initial distance /m	100-200	100-200	200-300	200-300
Initial altitude /m	550-650	550-650	550-650	550-650
Average speed m/s	0	0	20-25	20-25
Avoidance starts at t=0				
Avoid succ rate P_{succ}	100%	99.90%	100%	90.00%
Waypoint missed E_w	0.511	0.519	0.316	0.395
Additional path E_L	0.710	0.810	0.440	0.540
Space occupied E_S	0.149	0.227	0.093	0.308
Deviation time E_T	0.025	0.633	0.293	0.733
Total impact E_{total}	0.349	0.547	0.286	0.495
Total reward $Reward$	28.65	18.26	35.03	18.18
Avoid starts at $0.2t_c$				
Avoid succ rate P_{succ}	100%	99.05%	100%	87.17%
Waypoint missed E_w	0.432	0.444	0.298	0.363
Additional path E_L	0.633	0.707	0.443	0.510
Space occupied E_S	0.130	0.192	0.101	0.296
Deviation time E_T	0.024	0.549	0.271	0.592
Total impact E_{total}	0.305	0.473	0.278	0.441
Total reward $Reward$	32.79	20.94	35.97	19.77

Avoid starts at $0.5t_c$ **Avoid succ rate** P_{succ}	100%	97.79%	100%	86.53%
Waypoint missed E_w	0.345	0.353	0.288	0.319
Additional path E_L	0.570	0.593	0.447	0.480
Space occupied E_S	0.108	0.148	0.098	0.256
Deviation time E_T	0.044	0.397	0.122	0.499
Total impact E_{total}	0.267	0.372	0.239	0.392
Total reward $Reward$	37.45	26.29	41.84	22.07
Avoid starts at $0.8t_c$ **Avoid succ rate** P_{succ}	100%	95.65%	100%	81.11%
Waypoint missed E_w	0.271	0.269	0.255	0.299
Additional path E_L	0.493	0.487	0.443	0.487
Space occupied E_S	0.084	0.105	0.096	0.266
Deviation time E_T	0.043	0.389	0.098	0.498
Total impact E_{total}	0.223	0.312	0.223	0.389
Total reward $Reward$	44.84	30.66	44.84	20.85
Avoid starts at t_c **Avoid succ rate** P_{succ}	100%	92.30%	100%	70.15%
Waypoint missed E_w	0.225	0.226	0.231	0.252
Additional path E_L	0.427	0.430	0.443	0.487
Space occupied E_S	0.072	0.086	0.094	0.229
Deviation time E_T	0.037	0.342	0.060	0.463
Total impact E_{total}	0.190	0.286	0.207	0.356
Total reward $Reward$	52.63	32.27	48.31	19.71

space occupied impact E_S and deviation time E_T decreases.

Figure 6.6 presents the Monte-Carlo simulation results of Table 6.3 third column of UAV avoiding multiple static obstacles. As the trigger time varies from $t_t = 0$, $0.2t_c$, $0.5t_c$, $0.8t_c$ to t_c, the ratio of waypoints missed decrease from 0.519, 0.444, 0.353, 0.269 to 0.226, reducing around 60%. The average path increasing proportion varies from 0.810 to 0.427. The space proportion decreases from 0.227 to 0.086 while the deviation time proportion decreases from 0.633 to 0.427. Time, path, space saved are around 30% to 70%. The overall impact calculated as an average of above impacts decreases from 0.547 to 0.286 while the avoidance success rate also decreases from 99.90% to 92.30% (failure rate increases from 0.1% to 7.7%). In a word, as impact decreases, the safety level also decreases while increasing the safety level increases the impact.

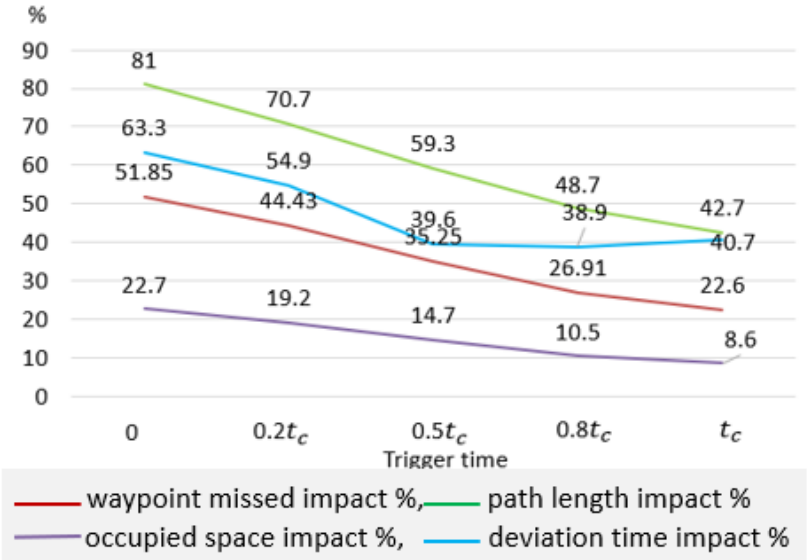


Figure 6.6: Impact of UAV avoiding multiple static obstacles by different avoidance trigger times

Hence, finding a point which balances the cost and safety is important. After applying the reward function $Reward = 10P_{succ}/E_{total}$, we got Figure 6.7, which presents the final reward for UAV avoiding multiple static obstacles with varying trigger time t_t. As t_t gets closer to the critical trigger time t_c, the total reward increases from 18.26 to 32.27 monotonously, reaches its max R_{max} at t_c, which means $t_t = t_c$ is the optimal avoidance trigger time for the UAV. That is because the collision avoidance success rate P_{succ} for the static obstacles is decreasing slower than the rate at which the impact decreases. Thus, the reward is a monotonous increasing function of t_t. Therefore, t_c is the right point for the UAV to start collision avoidance for static obstacles.

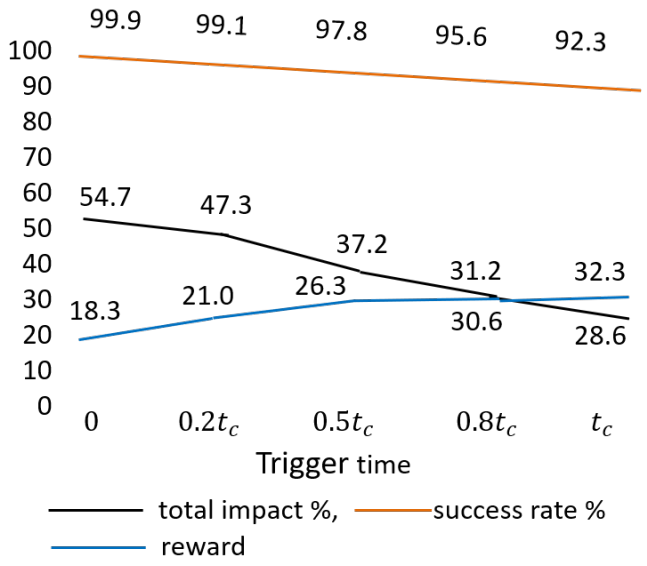


Figure 6.7: Success rate, total cost and rewards of collision avoidance by different avoidance trigger times for multiple static obstacles

In Table 6.3, the fifth column describes the collision avoidance experiments for multiple moving obstacles. The safety level (avoidance success rate) of moving

obstacles decreases a lot as t_t increases. The total impact E_{total}, composed of waypoint missed E_w, additional path impact E_L, space occupied impact E_S and impact time impact E_T decreased.

Figure 6.8 shows the Monte-Carlo simulation results of UAV avoiding multiple dynamic obstacles (Table 6.3 last column). As the trigger time gets from $t_t = 0$, $0.2t_c$, $0.5t_c$, $0.8t_c$ to t_c, the proportion of missed waypoints varies from 0.395 to 0.252, reducing about 40%. The average proportion of additional path decreases from 0.541 to 0.482, around 11%. The impacted space proportion drops from 0.309 to 0.229 (reducing 36.2%) while the deviation time impact drops from 0.736 to 0.462 (reducing 27.4%). The overall impact decreases from 0.495 to 0.356. However, the success rate also decreases from 90.00% to 70.15%, meaning that the failure rate increases from 10.00% to 29.85%.

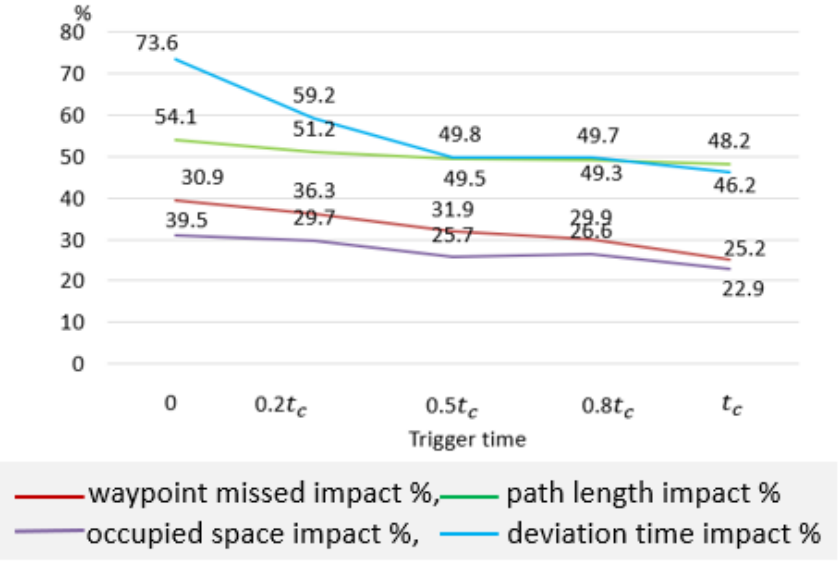


Figure 6.8: Impact of UAV avoiding multiple dynamic obstacles by different avoidance trigger times

After applying the reward function $Reward = 10P_{succ}/E_{total}$, we got Figure 6.9. However, different from the Figure 6.7, as shown in Figure 6.9 for the dynamic obstacles, the shape of reward trajectory is totally different. The reward, composed of the collision avoidance success rate and the impacts, are not a monotonous function, it is similar to a parabola, which goes up from 18.18 to 19.77, reaches its peak at 22.07, then decreases to 20.85 and 19.71. That is because both collision avoidance success rate and the impact decreases as the t_t gets closer to t_c. After $t_t = 0.5t_c$, the success rate has a higher decreasing speed than at which the impact decreases. So the reward function reaches its peak when avoidance action starts at time $t_t = 0.5t_c$. Thus, $0.5t_c$ is the optimal time point for the UAV to start avoidance action for multiple dynamic obstacles if 85% to 90% collision avoidance success rate is acceptable.

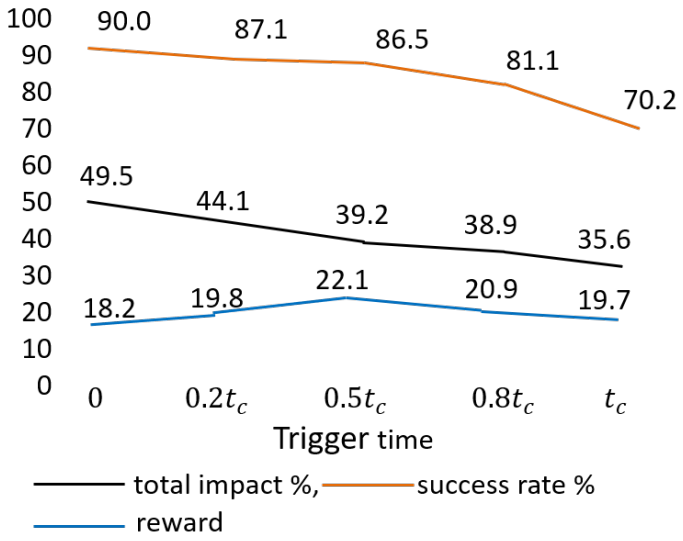


Figure 6.9: Success rate, total cost and rewards of collision avoidance by different avoidance trigger times for multiple static obstacles

6.9 Conclusions

In summary, this chapter provides the analysis of the impact for UAV collision avoidance performance and the success rate, as well as the selection of the time at which the UAV should trigger the avoidance maneuver to get the most efficient results. When selecting properly, about 40% to 60% time, length, space and other relative variables are decreased while keeping the high collision avoidance success rate, which is applicable for the real time flight task. In addition, large amount of Monte-carlo simulation results can be used for UAV future machine learning training to determine the different weight of the factors affecting the obstacles avoidance success.

As mentioned in chapter 5, besides to manage the tasks for multiple UAVs selection of the different trigger times can also be used to help UAV classify the hazard level of the obstacles, breaking the large amount of obstacles into several smaller groups according to the critical avoidance time, and then avoid them in sequence, which solves the difficult simultaneous collision avoidance problem.

Chapter 7: Obstacle Avoidance with UAV Maneuver Limitation based on Geometry Methods

Maneuver limitations may occur for unmanned aerial vehicles (UAV) during flight due to controller problems, electric or mechanical errors. These limitations will damage the capability for UAVs in collision avoidance, new path searching and waypoints tracking. Therefore, collision avoidance of UAV under maneuver limitations is needed to be studied. This chapter addresses the influence of UAV maneuver limitation on obstacles avoidance solutions and provides the corresponding strategy to improve the avoidance success rate based on decision making algorithm and cost function from my previous work [51]. Simulation studies show that the combination of decision making algorithm and cost function have improved the collision avoidance success rate under maneuver limitations [51].

7.1 UAV Maneuver Limitation Overview

A relevant study [52] showed that, in addition to human errors, electronic and mechanical errors are other contributors to UAV accidents.

There are many factors causing an UAV to malfunction, resulting in partial or complete loss of maneuverability and putting the UAV and other flying vehicles in

danger. These factors include limitation in visibility, sudden system failure, pilot errors, electronic and mechanical failures, sensor failure, engine failure, problems in actuator, wing damage, stabilizer faults and so on. We can classify the factors into the following categories: (1) pilot error in decision making, (2) aircraft body abnormality, (3) environmental obstacles and (4) flaws in flight plan.

This chapter focuses on failures in UAV physical components and the ultimate maneuver limitations and uses yaw angle problems, pitch angle problems and engine faults as the study examples.

A decision making tree is developed here to formulate the collision avoidance processes with the UAV maneuver limitations due to many complicated choices needing to be made here. To achieve the better results of the collision avoidance, the UAV will be first classified as in good condition or not. Then, the sensor will detect if there are obstacles so that the UAV will resume flight or need to do the collision avoidance. Then, according to the different types of the obstacles, different collision avoidance algorithms and path planning algorithms are chosen.

Thus, a decision making algorithm combining with the cost function is developed to handle the obstacle avoidance processes and flight plan, helping UAV to determine the best action in the complex situation.

There are two mainly benefits brought by this research. First, according to the different maneuver limitation of the UAV, provides the collision avoidance solution set for UAV based on cost function so that UAV can choose the optimal solution. Quantitatively compares and analyzes the solutions under normal and maneuver limitations, therefore figure out the impact of these maneuver limitations on UAV

collision avoidance and path selection.

Second, provides an efficient decision tree to handle UAV collision avoidance process under maneuver limitations/complex situations.

7.2 Problem Formulation

The problem formulation has several aspects: (1) decide whether the UAV needs to avoid collision under maneuver limitations using decision making tree, and (2) determine preconditions such as available information and the random obstacles production.

Two boolean parameters f_1 and f_2 are defined to represent the status of the UAV and the obstacles. f_1 indicates the maneuver limitation status, true for the UAV has fault modes or maneuver limitations. Otherwise it is false. f_2 indicates if there are obstacles located in the UAV's initial path [51].

Table 7.1: Collision avoidance strategy

	$f_1(true)$	$f_1(false)$
$f_2(true)$	**Collision avoidance with maneuver limitations**	**Normal collision avoidance**
$f_2(false)$	**Resume flight with maneuver limitations**	**resume flight**

Table 7.1 summarizes the collision avoidance strategy determined by the two boolean variables (f_1 and f_2). For example, if both f_1 and f_2 are TRUE, the UAV

needs to perform collision avoidance under maneuver limitation, which is difficult and has a smaller solution set than in normal condition. Therefore, our decision tree algorithm will select the best strategy for the UAV to avoid collision under this complicated conditions with higher probability.

7.3 Geometry-based Collision Avoidance for UAV's with Maneuver Limitation

Avoiding multiple obstacles no matter static or dynamic under maneuver limitation can be very complicated. The decision making tree algorithm will execute based on the following steps: (1) from the collected information, determine whether the UAV is in good condition or anomaly condition(2) detect obstacles and perform mitigation. (3) detect environmental hazard and conduct mitigation such as avoiding the bad weather area.

The step(2) mainly has follow aspects: find the potential collision avoidance solution set under normal condition based on geometry method, where the solution set means the combination of yaw angle, pitching angle and speed; apply the flight model, find the reduced solution set under maneuver limitation; and find the best solution from the reduced solution set and allowing the UAV to follow that path.

7.3.1 Flight Models under Two Maneuver Limitations

The two maneuver limitations are studied in this chapter.

(1) The first scenario: the UAV has engine failure, resulting in no thrust and

no acceleration, but yaw, roll and pitch angle can still be managed. The UAV cannot change speed in this condition. If the UAV confronts obstacles or moves toward a non-enter zone and needs to avoid collision by changing velocity, the corresponding collision avoidance ability will be substantially reduced. However, application of a suitable obstacle avoidance strategy will significantly improve the probability of collision avoidance.

(2) The second scenario: the UAV has rudder failure, aileron failure and engine failure but the UAV can still change the pitch angle. The yaw angle represents the UAV direction, the roll angle represents the UAVs turning. These problems are caused by rudder stuck and aileron failure. These problems will make the UAVs lose the ability to change speed, horizon direction and to turn circles. In this case, we can only manage the pitching angle. Since the UAV has lost thrust due to the engine error, the collision avoidance solution set will be substantially reduced. Similarly, application of a suitable obstacle avoidance strategy will improve the probability of collision avoidance.

In either case, due to engine failure, the normal aircraft equation system will be replaced by the aircraft glider equation system.

The complete equations [53] of the UAV under normal condition are:

$$m\frac{dV_u}{dt} = -mgsin\theta - D + Tcos\theta \tag{7.1}$$

$$mgcos\theta sin\phi = mV_u\dot{\psi}cos\theta cos\phi \tag{7.2}$$

$$mgcos\theta cos\phi - L - Tcos\theta = -mV_u\dot{\psi}cos\theta sin\phi \tag{7.3}$$

Where T is the thrust, D is the drag, m is mass of the UAV, g is the gravity acceleration, θ is the pitch angle, ψ is the yaw angle and ϕ is the roll angle.

(1) For scenario 1, engine failure will result in the loss of thrust, changing the dynamic mode into the gliding mode, meaning that $T = 0$. Also, the speed of UAV remains constant or decreases for a small amount in a short duration of time. That is $\dot{V_u} = 0$. Therefore, equations 7.1,7.2 and 7.3 will be revised as

$$mgsin\theta + D = 0 \tag{7.4}$$

$$\omega = \dot{\psi} = \frac{gtan\phi}{V_u} \tag{7.5}$$

$$mgcos\phi + mV_u\dot{\psi}sin\phi - L = 0 \tag{7.6}$$

(2) For scenario 2, the engine failure accompanied by the rudder and aileron problem will cause $T = 0$, $\dot{V_u} = 0$ and $\dot{\psi} = 0$. Therefore, equations 7.1,7.2 and 7.3 will be simplified as

$$mgsin\theta + D = 0 \tag{7.7}$$

$$\omega = \dot{\psi} = 0, \phi = 0 \tag{7.8}$$

$$mgcos\theta - L = 0 \tag{7.9}$$

7.3.2 Control and States Update for the UAS

Among the potential collision avoidance solutions, suppose (ψ_c, θ_c) is chosen as the final solution and is sent to the UAV as its new yaw and pitch angle.

Recall that the yaw angle of the UAV currently is ψ and the pitching angle is θ. Denote the proportional, the integral and the derivative parameters of the controller by K_p, K_i and K_d, respectively. Use equations 3.7 and 3.8, We now have

$$\dot{\psi} = K_p(\psi_c - \psi) + K_i \int (\psi_c - \psi) + K_d \int (\psi_c - \psi) \tag{7.10}$$

$$\dot{\theta} = K_p(\theta_c - \theta) + K_i \int (\theta_c - \theta) + K_d \int (\theta_c - \theta) \tag{7.11}$$

Consider scenario (1) and case (2) separately when time is incremented by Δt.

(1) For scenario 1, engine failure, the yaw angle at Δt is updated to:

$$\psi(\Delta t) = \psi + \dot{\psi}\Delta t \tag{7.12}$$

The pitch angle at Δt is updated to:

$$\theta(\Delta t) = \theta + \dot{\theta}\Delta t \tag{7.13}$$

The roll angle at Δt is updated to:

$$\phi(\Delta t) = tan^{-1}(\frac{\dot{\psi}V_u}{g}) \tag{7.14}$$

The velocity and lift at Δt are updated to:

$$V_u(\Delta t) = V_u \tag{7.15}$$

$$L(\Delta_t) = mgcos\phi + mV_u\dot{\psi}sin\phi \tag{7.16}$$

(2) For scenario 2, rudder and aileron stuck accompanied by engine failure, which is much more severe to UAV than that in scenario 1. The yaw angle at Δt is:

$$\psi(\Delta t) = \psi + \dot{\psi}\Delta t = \psi \tag{7.17}$$

The pitch angle at Δt is:

$$\theta(\Delta t) = \theta + \dot{\theta}\Delta t \tag{7.18}$$

The roll angle reminds 0:

$$\phi = 0 \tag{7.19}$$

The velocity and lift approximately remain constants,

$$V_u(\Delta t) = V_u \tag{7.20}$$

$$L(\Delta_t) = mgcos\theta \tag{7.21}$$

Therefore, in both scenario the portion speed of the UAV for x, y and z at Δt can be calculated:

$$V_{ux} = \dot{U_x} = V_u cos(\theta(\Delta t))cos(\psi(\Delta t)) \tag{7.22}$$

$$V_{uy} = \dot{U_y} = V_u cos(\theta(\Delta t))sin(\psi(\Delta t)) \tag{7.23}$$

$$V_{uz} = \dot{U_z} = V_u sin(\theta(\Delta t)) \tag{7.24}$$

Then the corresponding locations of the UAV is,

$$U_x = U_x + \dot{U_x}\Delta t \tag{7.25}$$

$$U_y = U_y + \dot{U_y}\Delta t \tag{7.26}$$

$$U_z = U_z + \dot{U_z}\Delta t \tag{7.27}$$

Through the above equations, the states of UAV, including location, velocity and heading angles for every time step are known.

7.4 Consequences of Maneuver Limitation on the UAVs Mission

This section focuses on the impacts of maneuver limitation on UAV collision avoidance. Figure 7.1 sketches the possible causes of maneuver limitation. The fault and the maneuver limitation has a complicated relationship such as the different faults may lead to the same maneuver limitation while one fault may generate multiple consequences [51]. For instance, the fault mode of engine failure will result in the loss of thrust followed by the loss in altitude and the slow reduction of speed. Loss action of the rudder and loss of the ability to change for the yaw angle may be caused by either physical failures of the actuator or the problems of the controller. Hence, to guarantee a good performance of the UAV, the anomaly mitigation process is result-based, that is the algorithm mitigates the anomaly and issues collision avoidance commands depending on the presence of maneuver limitation, not on the causes of maneuver limitation.

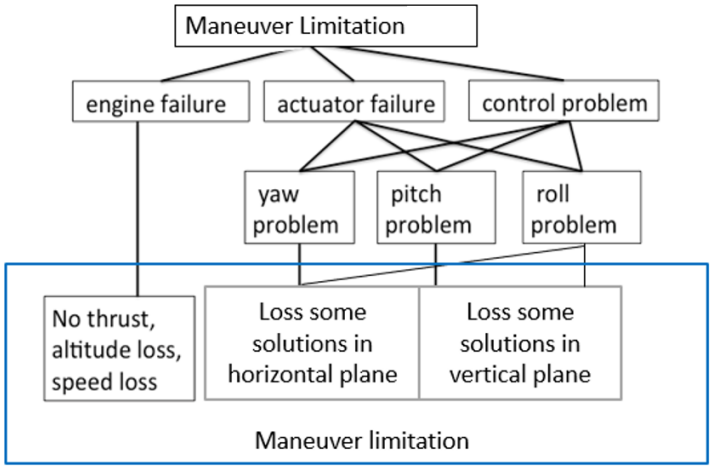


Figure 7.1: diagram of maneuver limitation

7.4.1 Solution Set of Collision Avoidance under Maneuver Limitation

Maneuver limitation of the UAV will decrease the number or range of the possible obstacle-avoidance solutions [51].

Figure 7.2 sketches the differences in solutions of collision avoidance under normal and abnormal conditions, including variation in the ranges of yaw and pitch angles of the UAV [51] (From section 3.2, the UAV speed approximately remains constant under the two types of maneuver limitations studied in this dissertation, thus variable of the speed isn't shown on the Figure 7.2). In Figure 7.2, the horizontal axis represents the yaw angle and the vertical axis represents the pitch angle and a solution is represented by a point at (ψ, θ) in the coordinate. Every point of the coordinate inside the blue oval satisfies the collision avoidance requirement in this situation. Figure 7.2 (a) represents the avoidance solutions for UAV under the healthy condition. Figure 7.2 (b) sketches a situation that if the elevator is partially or completely stuck so that the pitch angle cannot vary to its original maximum range $[-\theta_{max}, \theta_{max}]$ and results in the smaller solution set. Similarly, Figure 7.2(c) represents that if the yaw angle is affected and the corresponding solutions are decreased.

The maneuver limitations on the UAV will change the solution set. Suppose that there is a potential collision scenario. After the UAV detecting the possible collision, avoidance solutions are calculated and all of the solutions make up to the solution set, denoted by S_0. However, if the UAV has maneuver limitation such as elevator stuck, followed by pitching angle affected, then the new collision avoidance

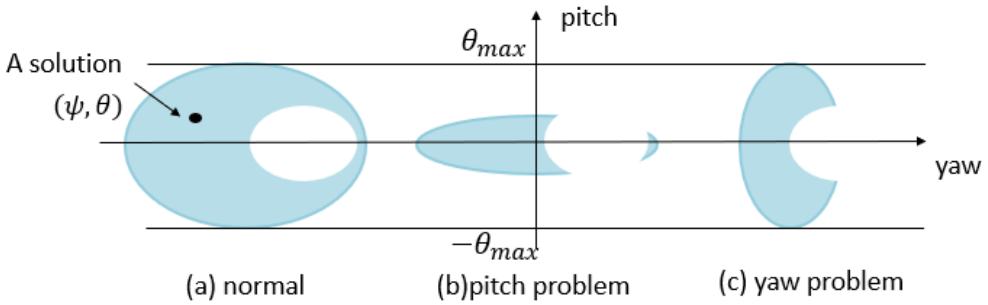


Figure 7.2: The collision avoidance solution set regarding the yaw angle and the pitch angle under normal and abnormal conditions. A solution is represented by a point at (ψ, θ) as yaw angle and pitch angle combination

solution $(V_u, \psi_{new}, \theta_{new})$ will be different from the solution (V_u, ψ, θ) under normal condition. All of the new solutions make up a new solution set, denoted by S_1. Alternatively, if the engine is out of order, then the collision avoidance solution changes to $(V'_{new}, \psi'_{new}, \theta'_{new})$ and these solutions make up a solution set denoted by S_2. Solution sets S_1 and S_2 are different from S_0 and we know that $S_1, S_2 \leq S_0$.

7.4.2 Increased Cost of Collision Avoidance under Maneuver Limitation

A cost function is used in the 3D collision avoidance to determine the yaw and pitch angles of the UAV [1]. Let the original yaw and pitch angles be $(\psi_{original}, \theta_{original})$ and the ranges of the two angles be ψ_{range} and θ_{range}. Let us partition each of $[-\psi_{range}, \psi_{range}]$ and $[-\theta_{range}, \theta_{range}]$ into n segments and thus we will have an $n * n$ segment pairs between the two. For each pair (ψ_i, θ_j), assume that the cost is

proportional to the deviation of the current angle from the original angle, the cost function is defined as

$$Cost(k,j) = \eta_\psi(\psi_i - \psi_{original}) + \eta_\theta(\theta_j - \theta_{original}) \tag{7.28}$$

where

$$\psi_{ori} - \psi_{range} < \psi_i < \psi_{ori} + \psi_{range} \tag{7.29}$$

$$-\theta_{range} < \theta_{ori} < \theta_{range}. \tag{7.30}$$

η_ψ and η_θ are the coefficients of the horizontal and vertical variations and both are positive numbers. From [5] and chapter 3, changing vertically is often more difficult and costs more than changing horizontally, thus it is always set that $\eta_\theta > \eta_\psi$.

Finding the solutions and calculating the costs under the abnormal condition is similar to those under the normal condition except that the extra constraints described in sections 3.2 and 3.3 are added. The additional constraints for fault mode 1 (equations 7.12 to 7.15) and the additional constraints for fault mode 2 (equations 7.18- 7.20) are summarized in equation 7.31.

$$f(\psi_i, \theta_i, V_{ui}) = 0 \tag{7.31}$$

The inequality constraints presented in equations 7.29 and 7.30 are summarized by equation 7.32.

$$g(\psi_i, \theta_i, V_{ui}) \leq 0 \tag{7.32}$$

If the angles do not satisfy the above constraints, the cost is set to infinity.

$$Cost(\psi_i, \theta_j) = \infty \tag{7.33}$$

After collecting the array of costs for candidate yaw and pitch angles, pick the minimum cost $Cost'_{min}$. The command yaw and pitch angles are given below,

$$\psi_{command} = \psi(cost'_{min}) \tag{7.34}$$

$$\theta_{command} = \theta(cost'_{min}) \tag{7.35}$$

The collision avoidance solution set under abnormal conditions is reduced compared to that under normal conditions. As a consequence, some low cost solutions under normal conditions are removed since they do not satisfy the constraints when the UAV is under maneuver limitation. From section 4.1, the UAVs collision avoidance solution set under normal conditions is S_0 with the minimum cost $Cost_{min}$. Under anomaly conditions, however, the solution set becomes S_1. Let

$$S_0 - S_1 = S_{11} \tag{7.36}$$

where S_{11} is the complementary set of S_1 and both S_1 and S_{11} are subsets of S_0.

Assume the minimum costs for solution sets S_0, S_1 and S_{11} be $Cost_{min}$, $Cost'_{min}$ and $Cost''_{min}$, respectively. We have

$$Cost_{min} = min(Cost'_{min}, Cost''_{min}) \tag{7.37}$$

and thus

$$Cost'_{min} \geq Cost_{min} \tag{7.38}$$

Verbally, we say that the minimum cost for collision avoidance under abnormal conditions ($Cost'_{min}$) is greater than the cost under normal condition ($Cost_{min}$).

7.4.3 Earlier Avoidance Trigger Time Needed to Avoid the Collisions

From Equation 3.13, the critical avoidance starting time t_c for the UAV to avoid collision under healthy conditions is,

$$t_c = \frac{r_0 cos\alpha_0 - \rho_{min} - (d_m - r_0 sin\alpha_0)}{||V_r||} \tag{7.39}$$

It is clear that the critical avoidance starting time t_c has a negative relationship to the minimum turning radius ρ_{min} of the UAV.

However, when the UAV is under maneuver limitation, the ability to manage collision avoidance will be compromised, and thus the probability of successful collision avoidance will be reduced. That is, the critical time point (the last possible time) for the UAV to escape from collision under the healthy condition is no longer proper for UAV with maneuver limitations. E.g., for fault mode 2, due to the problems of rudder and aileron, the maximum roll angle will be reduced and can even be zero. As a result, the minimum turning radius ρ_{min} is expanded to ρ'_{min}. In other words, $\rho'_{min} > \rho_{min}$ where ρ'_{min} is the anomaly minimum turning radius. From equation 7.39, a larger ρ_{min} will lead to a smaller avoidance trigger time t_c.

Therefore, for safety consideration, the UAV should start the collision avoidance process early. From equation 7.39, we have

$$t_{mc} < t_c; \tag{7.40}$$

where t_{mc} is the critical avoidance trigger time under the anomaly condition and t_c is the original avoidance trigger time.

Understanding the new solution set and choosing the proper solution from the

solution set will improve (1) the probability of collision avoidance under maneuver limitation, (2) the safety level of the UAV itself and (3) accurate prediction and assessment of the consequences or severity of the maneuver limitation.

Based on the different maneuver limitations, the safety priority, the constraints and the cost, an optimal collision avoidance solution is obtained. For instance, if the engine is partially out of order, the speed and the altitude will be slowly out of control and the UAV will choose to descend to avoid conflicts instead of doing turning, pitching up and other energy consuming operations, which is different from the choice with controller lost problem. Therefore, a decision making tree is developed here for UAV to handle the above activity in order to find the optimal solution to avoid collision under maneuver limitation.

7.5 Decision Making Tree for UAV Collision Avoidance with Maneuver Limitation

To improve the avoidance success rate for UAV under maneuver limitation, two methods are combined used here. One is based on cost function and geometry relation, setting $Cost(\psi_i, \theta_j) = \infty$ if (ψ_i, θ_j) doesn't satisfy the dynamic model of UAV under maneuver limitation and obstacles avoidance requirements. Another method is use the decision making system to class the situation and choose the proper action for the UAV at each step.

Figure 7.3 [51] shows the high level decision making processes. The decision tree first determines whether or not the UAV is under maneuver limitation. If yes,

go to the collision avoidance mode under the anomaly subtree. Otherwise, go to the collision avoidance mode under the normal condition subtree. Under the anomaly subtree, the algorithm then determines whether the obstacles are static or moving and, if moving, what the speed is. For a moving obstacle, if the speed is greater than the speed of the UAV, go to the leaf node that issues a sequence of command solutions for collision avoidance or lands in emergency. If the speed is lower than that, go to the node that issues one command of collision avoidance solution.

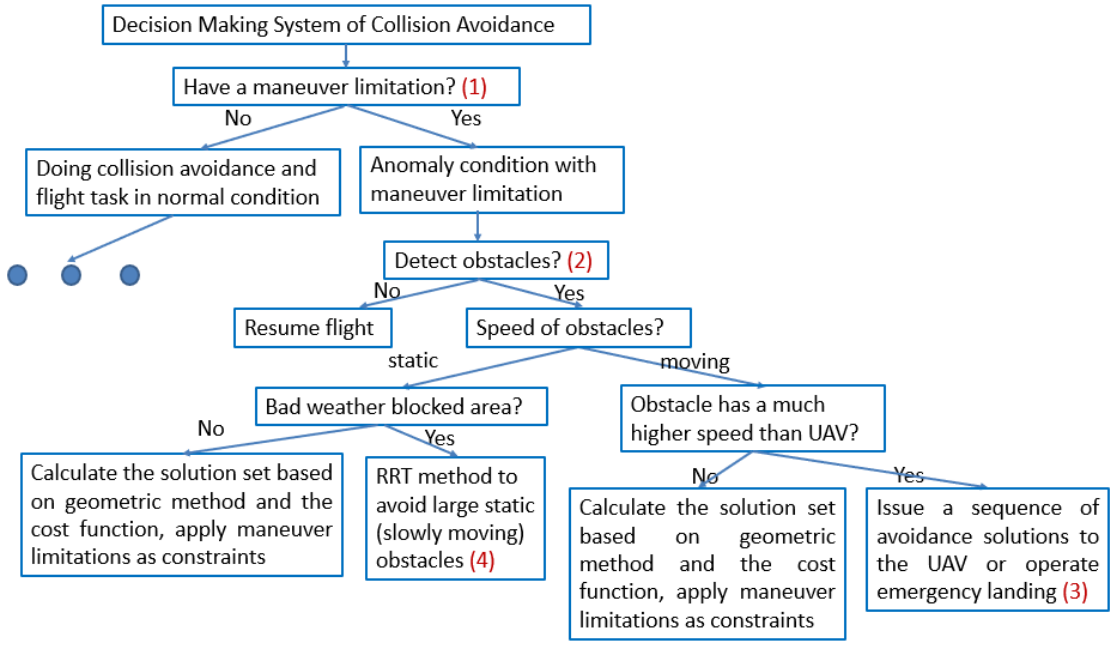


Figure 7.3: Overall decision making tree for UAV collision avoidance with maneuver limitations

The following sub-trees or nodes will be introduced: (1) the detection of maneuver limitation and mitigation node, (2) the detection of obstacles and avoidance sub-tree, (3) avoidance node for high speed intruders and (4) the environment response sub-tree (avoidance for large static obstacles) as shown in the Figure 7.3.

7.5.1 Anomaly Detection and Mitigation Node

Figure 7.4 describes some of the possible maneuver limitations. The anomaly is detected, classified, verified and then a corresponding mitigation is performed. If there are obstacles nearby, go to the collision detection and avoidance subtree. If the anomaly is cleared, the normal collision avoidance is activated, otherwise, the collision avoidance is executed under maneuver limitation. Note that when a fault is triggered at a time point, it will be permanent for the rest of the flight if it is not cleared since it cannot disappear by itself.

Two typical maneuver limitations are studied in this chapter: engine failure; and engine failure with aileron and rudder problem, which causes limitation of yaw angle and speed variation. In both of the two maneuver limitations, the general aircraft equations will change into glider equations due to the loss of thrust (engine failure).

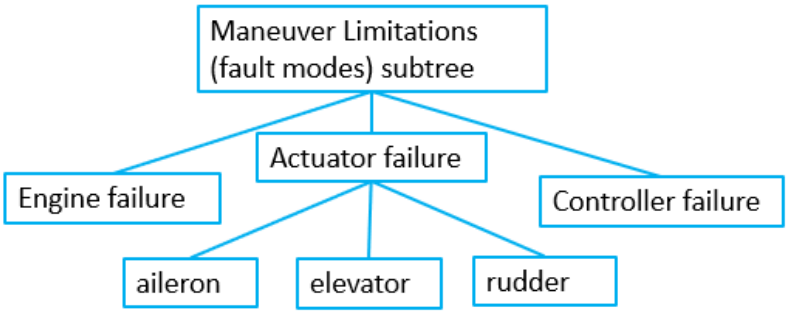


Figure 7.4: The maneuver limitation subtree

7.5.2 The Obstacle Detection and Avoidance Subtree

If an obstacle is detected, then collision avoidance is triggered. If the UAV is in good condition, go to the normal collision avoidance subtree, otherwise, go to the anomaly collision avoidance sub-tree.

Figure 7.5 shows the flow chart of the decision making algorithm that uses different strategies to avoid collisions. Firstly, based on FGA method [1], check if the t_c for obstacles is greater than 0. If not, trigger emergency landing. If yes, check the speeds of the obstacles and classify them into static or moving categories. For moving obstacles, determine if their speeds are much higher than the speed of the UAV. If yes for example, go to the leaf node for high speed obstacles avoidance or emergency landing.

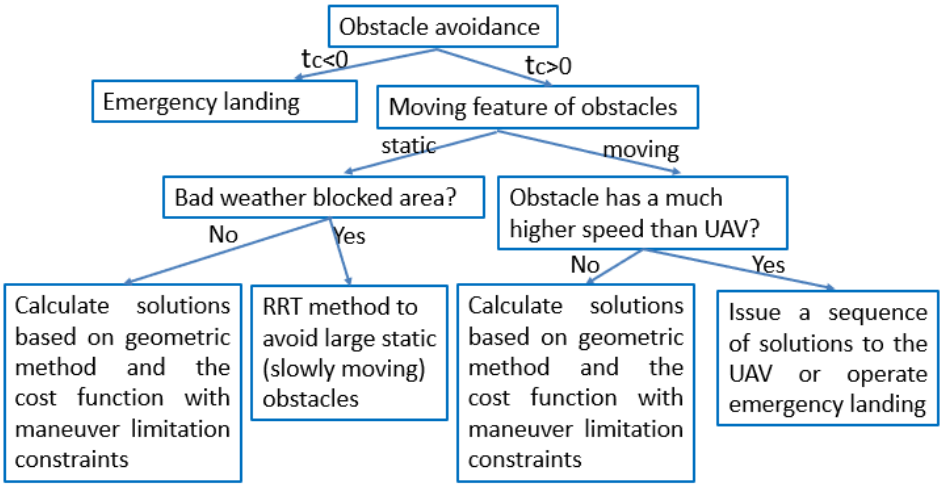


Figure 7.5: The collision avoidance subtree

7.5.3 The avoidance of high speed intruder node

This part focuses on multiple moving obstacles whose speeds are much higher than that of the UAV. In this case, condition of velocity that satisfy collision avoidance as Equation 4.13 and [1] provide will be violated:

$$V_u \geq V_a|sin(\theta_{va} - \theta_a + \theta_{dm})|. \tag{7.41}$$

Where $\theta_{dm} = sin_{-1}\frac{d_m}{r_0}$, which is the angle of safe separation. θ_a is the angle of the obstacle location vector $\overrightarrow{A}$ and θ_{va} is the angle of the obstacle velocity vector $\overrightarrow{V_a}$. To deal with the high speed intruders, the one-time collision avoidance algorithm for static obstacles or normal-speed moving obstacles is modified accordingly.

For every checking time $\triangle T$, one command is calculated for the UAV to follow in the entire duration of $\triangle T$. Check to see whether this command can let UAV have the closest point of approach $CPA \geq d_m$. If so, the UAV will search for the resumed path and fly back. Otherwise, continue to calculate the next solution based on the current location and velocity of the UAV. This process is illustrated in Figure 7.6 where points (1) to (6) represent the sequence of commands issued to UAV to complete the collision avoidance task. The algorithm has been proved to improve the probability of obstacle avoidance success of the UAV facing high speed obstacles.

In addition, if currently the UAV is unable to keep safe, including but not limited to: loss capability to perform any collision avoidance actions, the conflicts will happen within checking time $\triangle T$; start the emergency landing for the UAV immediately, such as searching for low population area to land.

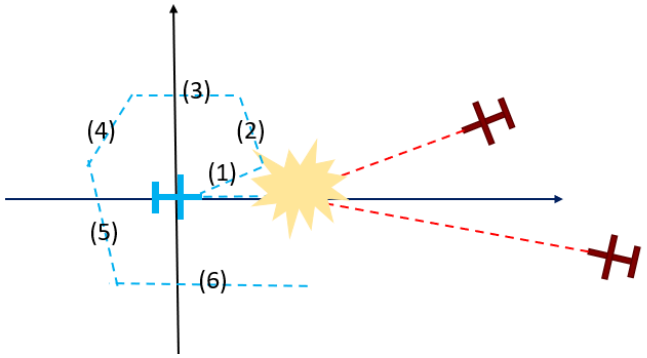


Figure 7.6: Sequence of commands issued to UAV to avoid high-speed obstacles

7.5.4 The Response to Environment Variation Subtree

This part focuses on avoiding bad weather areas such as the rain, storm and heavy cloud, which are always modeled as the large static or slowing moving obstacles. Thus, instead of using a geometry algorithm, sampling-based algorithm is applied here with the critical avoidance time t_c to avoid the obstacles and plan the new path, as illustrated in Figure 7.7.

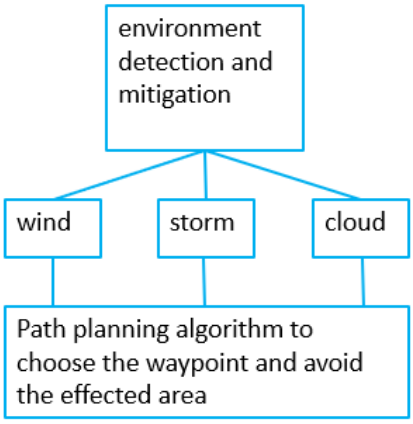


Figure 7.7: The environment checking subtree

The upon part of Figure 7.8 shows the UAV avoiding the bad weather areas. The dark blue line describes the trajectory of the UAV. The light colorful areas mean the blocked weather areas. The red and yellow blocks mean they must or would be better for UAV to avoid. The light green and blue blocks mean that the UAV could pass the outer part or directly pass through these areas. Hence, the UAV will decide to pass through or to detour away from the areas, depending on the information received. Please note that the base of y axis is much smaller than the base of x axis, so the trajectory of the UAV looks like having many sharp peaks while in fact the trajectory is pretty smooth. The bottom of the 7.8 is the zoom in of the UAV's trajectory.

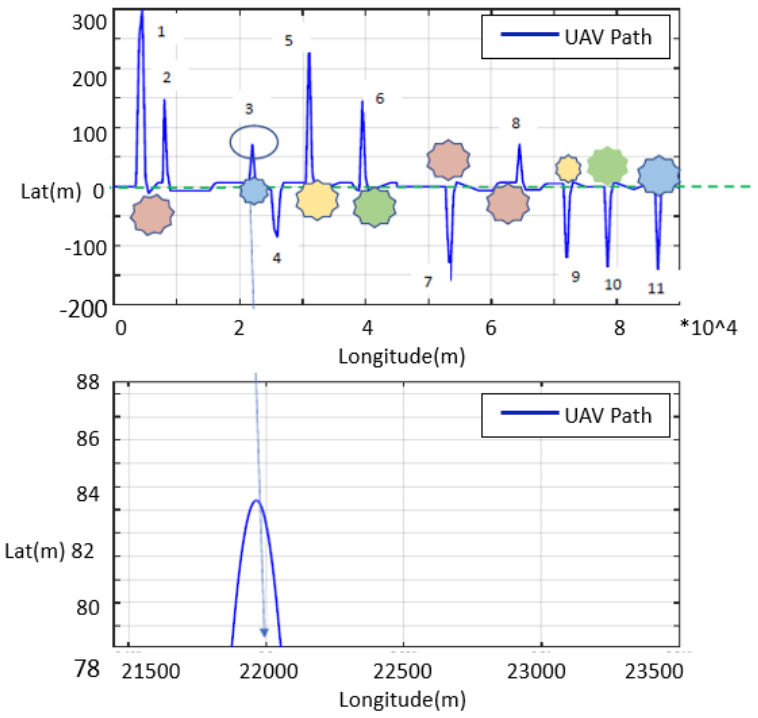


Figure 7.8: Check and avoid the blocked weather area

7.5.5 Setup of the Simulation Experiment

This section focuses on how to set up the decision making tree in Simulink, including detection, path planning, control, actuator operation and plane execution, see Figure 7.9.

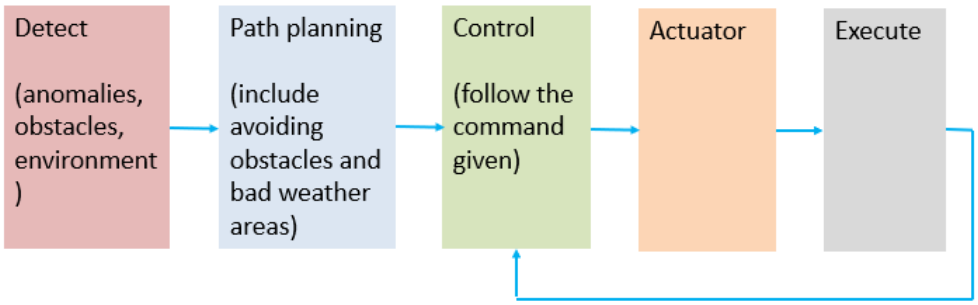


Figure 7.9: Simulink setup

- Detection: detects for anomalies, obstacles and information of environment
- Path planning: avoids obstacles, plans and updates the path
- Control: controls the UAV to follow the commanded values
- Actuator: drives machine to operate
- Execute: action part

Algorithm 6 manages the obstacle detection part of the decision making process. Algorithm 7 is about the obstacle avoidance part of the decision making process. Algorithm 8 controls the waypoint searching part of the decision making process.

Algorithm 6 the obstacle detection part of the decision making process

Precondition: necessary data is available for UAV

1: **for** $i \leftarrow 1$ to n **do**

2: $\overrightarrow{r_0} \leftarrow \overrightarrow{A_0} - \overrightarrow{U_0}$ ▷ relative location vector

3: $\overrightarrow{V_{r0}} \leftarrow \overrightarrow{V_{u0}} - \overrightarrow{V_a}$ ▷ relative velocity vector

4: $\alpha_0 \leftarrow \frac{\overrightarrow{r_0}\overrightarrow{V_{r0}}}{|\overrightarrow{r_0}||\overrightarrow{V_{r0}}|}$ ▷ angle of two vectors

5: $r_{PCA} \leftarrow r_0 \cdot sin\alpha_0$ ▷ closest point of approach

6: **if** $r_{PCA} < d_m$ **then**

7: $status \leftarrow dangerous$

8: $M_1 \leftarrow$ obstacle information

9: $f_2 = 1$, collision avoidance action need to trigger

10: **end if**

11: **end for**

12: **return** obstacles' information

Algorithm 7 the obstacle avoidance part of the decision making process [51]

Precondition: necessary data is available for UAV

calculate the collision avoidance solution set S_0
if UAV under normal condition [1] **then**
    calculate the cost for each solution in S_0 and pick up the least cost solution
else ▷ UAV under anomaly condition
    apply the constraints to S_0
    for $k \leftarrow 1$ to k_1 **do**
        for $j \leftarrow 1$ to j_1 **do**
            $\varphi \leftarrow -\varphi_{range} + \varphi_{range} k/k_1$
            $\theta \leftarrow -\theta_{range} + \theta_{range} j/j_1$
            $\overrightarrow{V_{ut}} \leftarrow (V_u cos\theta cos\varphi, V_u cos\theta sin\varphi, V_u sin\theta)$
            if Equation 7.31 and Equation 7.32 are not satisfied **then**
                break
            else
                for $l \leftarrow 1$ to l_1 **do**
                    $\overrightarrow{r_l} \leftarrow \overrightarrow{A_l}(A_{lx}, A_{ly}, A_{lz}) - \overrightarrow{U}(U_x, U_y, U_z)$
                    $d = min(d_l) \leftarrow |\overrightarrow{V_{ut}} \times \overrightarrow{r_{ltc}}|/|\overrightarrow{r_{ltc}}|$
                end for
                if $d \geq d_m$ **then**
                    add (φ, θ, V_{ut}) into solution set S_1
                end if
            end if
            $Cost(k, j) \leftarrow \eta_\psi(\varphi - \varphi_{ori}) + \eta_\theta(\theta - \theta_{ori})$
        end for
    end for
end if
from the solution set S_1, find the solution with least cost $Cost_{min}, \varphi_{com}, \theta_{com} = min(Cost(k, j)), \varphi_k, \theta_j$;
return $\varphi_{com}, \theta_{com}, newpath$
UAV follows the commanded solution

Algorithm 8 the UAV waypoint tracking part of decision making process

1: **for** $i \leftarrow 1$ to w **do**

2: $\quad \overrightarrow{r_{wi}} \leftarrow \overrightarrow{W_i}(W_{xi}, W_{yi}, W_{iz}) - \overrightarrow{U_t}(U_x, U_y, U_z)$

3: $\quad \theta_{wi} \leftarrow f(\overrightarrow{r_{wi}}, \overrightarrow{V_u})$ ▷ waypoints angle bias

4: $\quad$ **if** $\theta_{wi} < \pi$ **then** ▷ search the next waypoint

5: $\quad$ **end if**

6: **end for**

7: **return** W_i ▷ get the next waypoint

7.6 Results and Analyses of the Simulation Experiment

This section shows the comparison of responses to the appearance of obstacles of the UAV in normal conditions and abnormal conditions from my work [51].

If the UAV is under maneuver limitation due to the presence of faults but the limitation has not been taken into account when planning the obstacles avoidance, the solution chosen may not be able to satisfy the constraints enforced by the limitation, where the constraints include the yaw angle, the pitch angle, the turning radius and the critical avoidance starting time limits. Results of the simulation study are based on 5000 different scenarios of the simulation experiments. Below are the comparisons.

1 The collision avoidance success rate P_s and the solution set in good condition.

2 The collision avoidance success rate P_{sb} and the solution set under fault mode but the maneuver limitation is not taken into account when performing avoidance.

3 The collision avoidance success rate P_{sa} and the solution set with fault mode and the maneuver limitation has been fully considered when issuing the solution.

Maneuver limitation 1: engine failure, no thrust

Figure 7.10 shows one scenario of collision avoidance solution impacted by maneuver limitation 1. The UAV confronts three static obstacles in the path with the initial distance 150-200 m and the speed of the UAV is 15m/s. The UAV detects the obstacles and chooses the anomaly node first,then chooses the static obstacle avoidance node to trigger the obstacle avoidance action. The whole area (both blue and pink areas) represents the collision avoidance solution set for UAV in good condition. The pink area represents the solution set for UAV in anomaly condition. The pink area is about 60% smaller than the total area, meaning the solutions reduce by 60% due to maneuver limitation. For solution cost, lighter pink links with less cost. So the UAV chooses ($\psi_c = 0.2rad, \theta_c = -0.05rad$), the least cost yaw-pitch angle as a new direction to avoid obstacles.

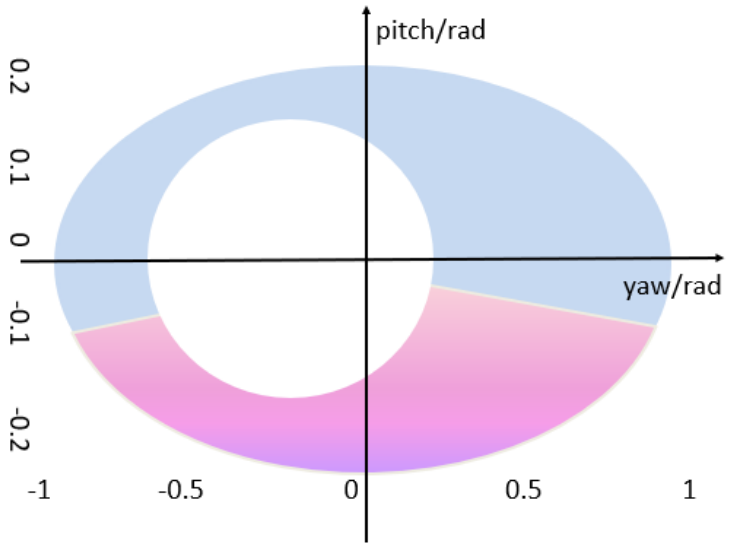


Figure 7.10: Collision avoidance sets for the UAV under healthy condition (whole area) and under maneuver limitation 1 (pink area only)

Table 7.2 summarizes the comparison of Monte Carlo simulation in Matlab regarding the collision avoidance success rates and the costs confronting static obstacles between the UAV is in healthy condition and under maneuver limitation 1. The collision avoidance success rate reaches the highest value when the UAV is in good condition. Under maneuver limitation 1, if no applying decision making process, the success rate of avoiding multiple obstacles declines from 97.79% to 83.38% from the initial place of detection 150-200m. After applying DMP (decision making process) by choosing the anomaly condition, the static obstacle avoidance node and the corresponding actions, the avoidance success rate recovers up to 95.45%, which is close to the success rate when the UAV is in good condition.

Table 7.2: Success rate of UAV with engine failure for static obstacles avoidance

Parameter	Single obstacle	Multiple obstacles	Average r_0
P_s	100%	95.83%	100-150m
P_{sb}	72.11%	70.06%	
P_{sa}	100%	94.29%	
$Cost_{min}$	0.0981	0.0841	
$Cost_{minb}$	0.0981	0.0841	
$Cost_{mina}$	0.1433	0.1656	
P_s	100%	97.79%	150-200m
P_{sb}	87.30%	83.78%	
P_{sa}	100%	95.45%	
$Cost_{min}$	0.1062	0.1095	
$Cost_{minb}$	0.1062	0.1095	
$Cost_{mina}$	0.1345	0.1582	

Figure 7.11 illustrates the trends presented in Table 4 for the static obstacle collision avoidance success rate against the initial detection distances under healthy condition and under maneuver limitation 1. As the initial detection distance increases, the collision avoidance success rate decreases. At the same initial detection distances, the success rate of the UAV in good condition (the blue curve) is higher than the success rates in all other situations. When the UAV is in fault mode, the success rate declines (the red curved). When the UAV is in the fault mode and it has been taken into account in decision making (anomaly condition is chosen, static obstacle avoidance node is set, and actions corresponding to them are taken), the success rate has recovered (see the green curve below the blue one). Therefore, the fault mode is not always a hazard to the UAV for collision avoidance, provided that proper solutions are chosen and corresponding actions are taken. A high success rate is still guaranteed with the new decision making algorithm.

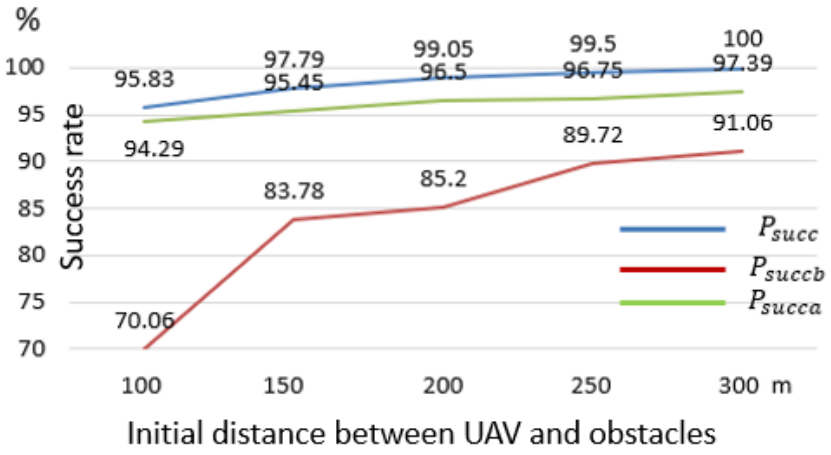


Figure 7.11: Success rate of multiple static obstacle avoidance

Table 7.3 summarizes the collision avoidance success rate and the cost for avoiding moving obstacles under healthy condition and maneuver limitation 1. When the UAV is in good condition the UAV has the highest collision avoidance success rate. If the UAV has maneuver limitations but is ignored (unrecognized) when issuing the collision avoidance command, the avoidance success rate decreased from 93.35% to 71.74% for avoidance of moving obstacles with speed 5-15 m/s. However, after applying DMP (decision making process), the avoidance success rate has recovered to 91.99%, which is close to that when the UAV is in healthy condition. By applying DMP, we mean that the maneuver limitation has been properly classified, the obstacles are classified into the moving obstacle avoidance node and corresponding actions are taken.

Figure 7.12 compares the collision avoidance success rates for multiple dynamic obstacles when the UAV is in healthy condition and under maneuver limitation 1. As the speed of the obstacles increase, the success rate decreases. The success rate is always at the highest level (the blue curve) when the UAV is in good health P_{succ}. When the UAV is in a fault mode but does not apply DMP, the success rate is dramatically decreased (see the red curve). When the UAV is in a fault mode and applies DMP when calculating the collision avoidance solution, the success rate has recovered (the green curve below the blue curve). By applying DMP, we mean that the maneuver limitation has been properly classified, the obstacles are classified into the dynamic obstacle avoidance node and corresponding actions are taken. We conclude that the fault mode is not always a hazard for collision avoidance of the UAV if a proper solution has been selected.

Table 7.3: Success rate of UAV with engine failure avoiding dynamic obstacles

Parameter	Single obstacle	Multiple obstacles	Average speed
P_s	100%	93.35%	5-15m/s
P_{sb}	69.74%	71.74%	
P_{sa}	95.15%	91.99%	
$Cost_{min}$	0.1500	0.1496	
$Cost_{minb}$	0.1500	0.1496	
$Cost_{mina}$	0.2263	0.2508	
P_s	100%	85.53%	15-25m/s
P_{sb}	59.60%	61.00%	
P_{sa}	90.42%	81.44%	
$Cost_{min}$	0.1500	0.3463	
$Cost_{minb}$	0.1500	0.3463	
$Cost_{mina}$	0.2429	0.4301	
P_s	100%	59.85%	35-45m/s
P_{sb}	50.92%	39.79%	
P_{sa}	77.47%	56.23%	
$Cost_{min}$	0.1500	0.3257	
$Cost_{minb}$	0.1500	0.3257	
$Cost_{mina}$	0.2427	0.4253	

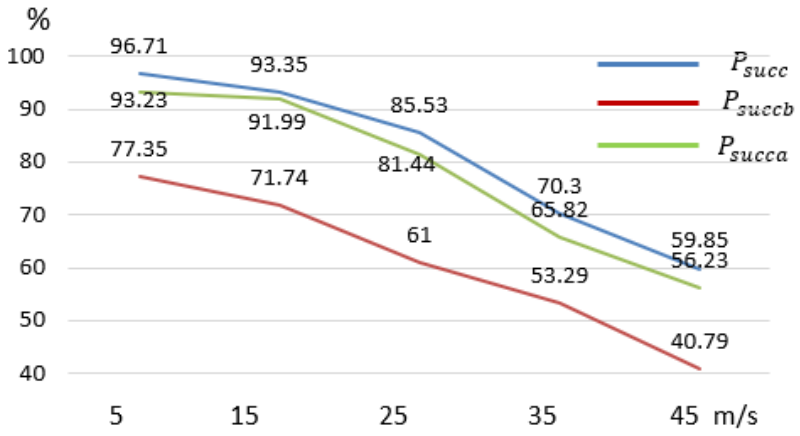


Figure 7.12: Avoidance success rate for multiple dynamic obstacles

Maneuver Limitation 2: Engine is out. Aileron and rudder have problems, limiting the range of yaw angle variation.

Figure 7.13 shows one scenario of collision avoidance impacted by maneuver limitation 2. The initial detection distance between the UAV and the obstacle is 150-200 m and the UAVs speed is 15m/s. The UAV detects a single dynamic obstacle, then goes to the anomaly branch and further goes to the dynamic obstacle avoidance branch. Next, the UAV chooses the node that deals with a normal speed intruder and triggers the avoidance action. The whole area (the blue and pink areas) is the solution set for UAV in good health. The pink area is the solution set for UAV in anomaly condition, which is about 28% of the total area, implying that under maneuver limitation, the collision avoidance solution is reduced by 72%. For solution cost, lighter pink associates with less cost so the UAV chooses ($\psi_c = 0.25rad, \theta_c = 0rad$), the least cost yaw-pitch angle as a new direction to avoid obstacles [51].

Table 7.4 summarizes the Monte Carlo simulation results regarding the collision

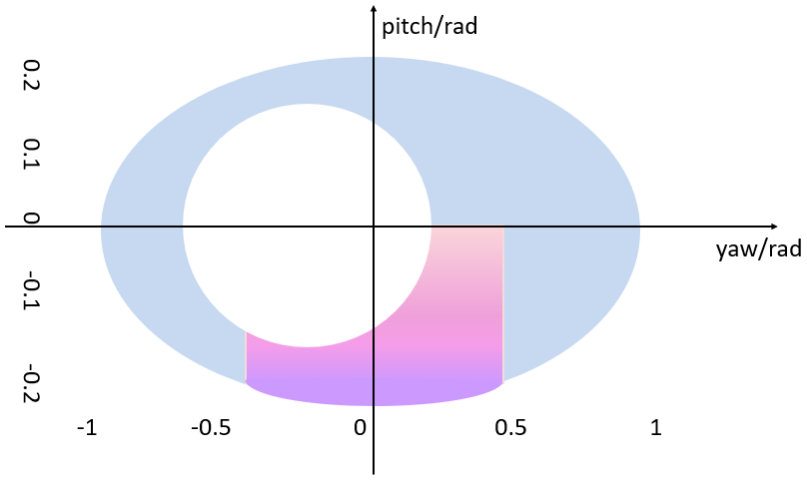


Figure 7.13: Collision avoidance solution sets when the UAV is in good condition (the whole area) and under maneuver limitation 2 (the area in pink)

avoidance (static obstacles) success rates and the costs when UAV in different health condition. When the UAV is healthy, the collision avoidance has the highest success rate. If the UAV is under maneuver limitation 2 and does not apply DMP, the success rate decreases from 97.79% to 24.12% for a detection distance of 100-150 m. But after applying DMP, the avoidance success rate recovers to 44.01%.

Table 7.5 summarizes the collision avoidance (for dynamic obstacles) success rates and the costs for the UAV in different health condition. When the UAV is healthy, it has the highest obstacle avoidance success rate. If the UAV is under maneuver limitation 2 and does not apply DMP when issuing the avoidance solution command, the success rate decreased dramatically from 93.35% to 25.33% for dynamic obstacles at speed 5-15 m/s. However, after applying the DMP, the success rate recovers to 89.83%, very close to 93.35%, the rate when the UAV is healthy.

Figures 7.14 and 7.15 show the result for collision avoidance from Simulink for

Table 7.4: Success rate of UAV with maneuver limitation 2 for static obstacle

Parameter	Single obstacle	Multiple obstacle	Average r_0
P_s	100%	95.83%	100-150m
P_{sb}	47.99%	21.66%	
P_{sa}	68.04%	40.46%	
$Cost_{min}$	0.0964	0.0872	
$Cost_{minb}$	0.0964	0.0872	
$Cost_{mina}$	0.1500	0.1500	
P_s	100%	97.79%	150-200m
P_{sb}	49.24%	24.12%	
P_{sa}	68.68%	44.01%	
$Cost_{min}$	0.1062	0.1095	
$Cost_{minb}$	0.1062	0.1095	
$Cost_{mina}$	0.1501	0.1523	

Table 7.5: Success rates of UAV with maneuver limitation 2 for dynamic obstacles

Parameter	Single obstacle	Multiple obstacles	Average speed of obstacles
P_s	100%	93.35%	5-15m/s
P_{sb}	57.08%	25.33%	
P_{sa}	94.22%	89.83%	
$Cost_{min}$	0.1500	0.1496	
$Cost_{minb}$	0.1500	0.1496	
$Cost_{mina}$	0.2301	0.2568	
P_s	100%	85.53%	15-25m/s
P_{sb}	52.44%	23.11%	
P_{sa}	78.00%	72.86%	
$Cost_{min}$	0.1500	0.3463	
$Cost_{minb}$	0.1500	0.3463	
$Cost_{mina}$	0.2578	0.3769	
P_s	100%	59.85%	35-45m/s
P_{sa}	51.76%	22.05%	
P_{sb}	78.52%	45.32%	
$Cost_{min}$	0.1500	0.3257	
$Cost_{minb}$	0.1500	0.3257	
$Cost_{mina}$	0.2634	0.3962	

a scenario that the UAV is under maneuver limitation 1 (no thrust due to engine failure and the loss in speed variation capability) in 2D and 3D. The UAV flies toward northeast and detects two dynamic obstacles at about t = 140 s. After calculation, it decides to start the avoidance operation at 60 s to reach maximum efficiency. The speeds of the intruders are between 60 m/s and 75 m/s. The UAV sequentially turns on the anomaly node, the moving obstacle avoidance node and then the normal speed node. After calculating, the UAV detours to the right for 0.7 rad and keeps the same altitude. Finally, the UAV successfully avoids the two intruders. The closest points of approach are around 600 m to 1500 m, larger than the required minimum separation distance of $d_m = 500m$.

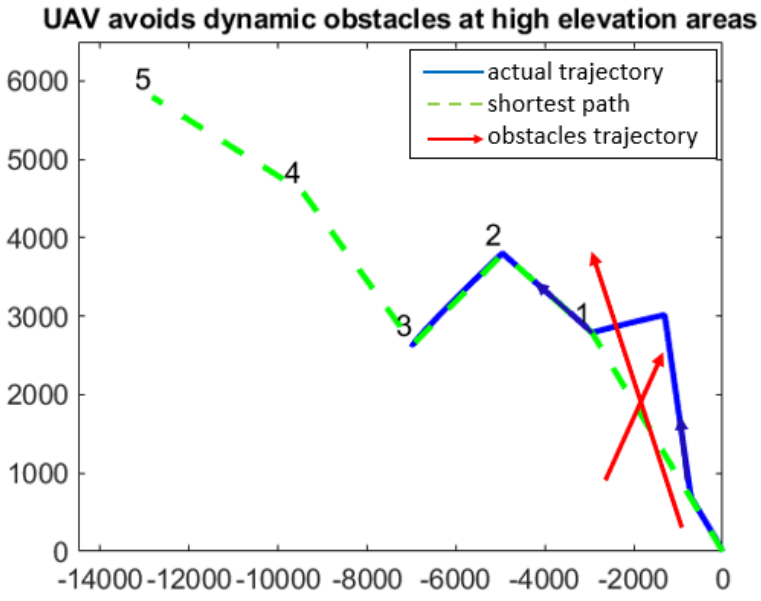


Figure 7.14: Collision avoidance for dynamic obstacles when the UAV is under maneuver limitation 1 (2D plot)

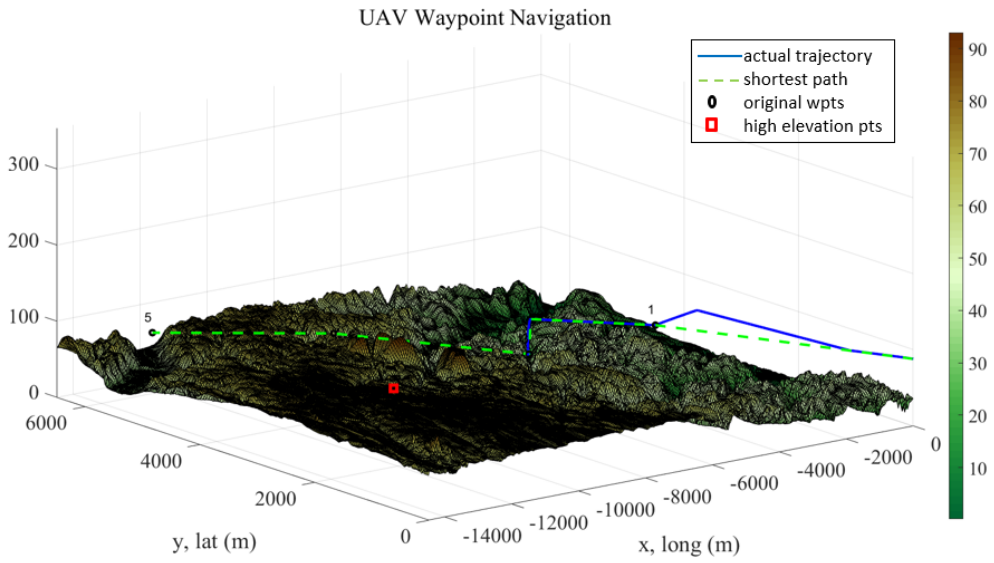


Figure 7.15: Collision avoidance for dynamic obstacles when the UAV is under maneuver limitation 1 (3D plot)

7.7 Conclusion

Based on the comparison, a decision making process is taken for an unhealthy UAV to prevent its collision avoidance success rate from substantially declining and thus improves the safety of an UAV under maneuver limitation.

From the simulation study using the Matlab and Simulink software, we conclude that the decision making algorithm for collision avoidance has several advantages for an UAV under maneuver limitations to manage different types of obstacles, especially for multiple moving intruders. It improves the probability of collision avoidance under maneuver limitation; the safety level of the UAV and the assessment accuracy of the consequences or severity of the maneuver limitation. On average, the collision avoidance success rate has been increased around $30 - 70\%$.

Chapter 8: Obstacle Avoidance of UAV for Time-Varying Obstacles based on Geometry Methods

During the UAS collision avoidance operation, the direction and speed of the obstacles may change, resulting in the unexpected conflicts with the commanded UAS. Thus, for these time-varying obstacles, the conflicts probability map and the cost map are developed for UAS to choose the best collision avoidance solution.

8.1 Model of the Time Varying Obstacles

Assume the safe separation distance between the UAV and the obstacle is d_m and the CPA(close point of approach) between the UAV and the obstacle is $|\overrightarrow{r_{min}}|$.

Without considering the real time position shifting caused by the speed and direction changing of the obstacle, the area surrounding the UAV where $|\overrightarrow{r_{min}}| < d_m$ is the collision zone, which has the possibility of collision P=1. And the area outside is considered to be a safe area with collision possibility P=0.

However, the safe and the dangerous area are not fixed if position shifting is taken into account. Assume the relative position shifting is $\overrightarrow{d_r}$. Since the checking time of the UAV is only several seconds [5], $|\overrightarrow{d_r}|$ is more likely to be small rather than larger. Thus, Gaussian distribution function is applied here to model the $|\overrightarrow{d_r}|$. The

possibility of a safe area in theory changes to dangerous is $P(|\overrightarrow{r_{min}} + \overrightarrow{d_r}| < d_m) > 0$.

Thus, adding buffer additional separation time or distance to avoid collisions in case of the position errors or arrival time error is necessary. In addition, when using the cost function to choose the next head/pitch angle of the UAV, the conflict probability map can be a reference since the lower cost area has the higher probability of collisions.

First, the model of the time-variant obstacles is needed.

The normalized Gaussian distribution is,

$$f_x = \frac{1}{\sigma\sqrt{2\pi}} e^{-0.5(\frac{x-\mu}{\sigma})^2}, \tag{8.1}$$

where μ and σ are the average and the standard variance of the parameter x.

Assume the yaw angle of the obstacle is in detection time is ψ_{obs0} and the its real time yaw angle is ψ_{obs}. Assume the yaw angle our commanded UAV is ψ_{uav0}. Thus, the distribution of the obstacle yaw angle variation is

$$f_{\psi_{obs}} = \frac{1}{\sigma\sqrt{2\pi}} e^{-0.5(\frac{\psi_{obs}-\mu}{\sigma})^2}, \tag{8.2}$$

During the flat(steady) flight process, use value of detection time ψ_{obs0} as μ, the 0.5π as the 3σ,that is $\sigma = 0.167\pi$ hence, the distribution of the obstacle yaw angle is

$$f_{\psi_{obs}} = \frac{1}{0.167\pi\sqrt{2\pi}} e^{-0.5(\frac{\psi_{obs}-\psi_{obs0}}{0.167\pi})^2}, \tag{8.3}$$

8.2 The Safety Map

This section is to build the safety map of the obstacles avoidance. To be easier to clarify, use 2D application as the example. The conflicts possibility for the UAV and time-variant obstacle could be calculated using following steps:

1. Find the collision avoidance solution for UAV with time-invariant obstacles. Using equations in chapter 3.3, get geometry-based obstacle avoidance solutions Ψ_{uav}, where any yaw angles $\psi_{uav} \subseteq \Psi_{uav}$ could let UAV escape from the conflicts. Suppose the initial direction of the obstacle is ψ_{obs0}.

2. For each yaw angle $\psi_{uav} \subseteq \Psi_{uav}$ of the UAV, calculate the obstacle angle ψ_{obs}, for which the UAV cannot use ψ_{uav} as the collision avoidance solution for this obstacle.

3. From step 1 and 2, it is clear that if the yaw angle of the obstacle varies beyond $[\psi_{obs0}, \psi_{obs}]$, the conflict may happen. Suppose the possibility of obstacle's yaw angle varies beyond $[\psi_{obs0}, \psi_{obs}]$ is P_{obs}, then the conflicts possibility for UAV using command angle ψ_{uav} is

$$P_{c\psi} = P_{obs} \tag{8.4}$$

Thus, the safety possibility for UAV using command angle ψ_{uav} is,

$$P_{s\psi} = 1 - P_{obs} \tag{8.5}$$

4. Repeat steps 1-3 and get the safe possibility for all commanded angle ψ_{uav} and build the safety map.

From section 8.1, the new yaw angle of the UAV which could make it avoid the central white collision zone is considered as a collision avoidance solution, e.g. the colorful area in Figure 8.1 is the non-conflict zone. In theory, any area inside the colorful zone is safe for UAV. But if the obstacle changes direction within the checking time of the UAV, the colorful area has the possibility to become the conflict zone in this situation. In this case, the Figure 8.1 presents the different safety levels

of the area with the avoidance solution.

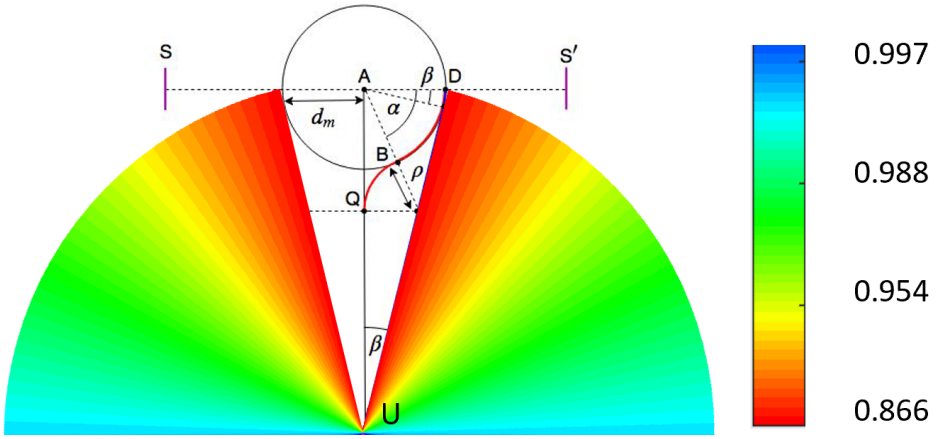


Figure 8.1: Safety map: possibility of avoiding the conflict successfully

It is clear from Figure 8.1, the more far away from the white conflict area, the higher safety level that the UAV could obtain. However, the corresponding cost will also increase. Thus, it is needed to find a balance point between them.

8.3 The Cost Map

This section describes the cost map for UAV doing collision avoidance. From the cost Equation 3.23 in chapter 3.4 for 3D space,

$$cost = \eta_\psi(\psi_{uav} - \psi_{uav0}) + \eta_\theta(\theta_{uav} - \theta_{uav0}) \tag{8.6}$$

Therefore, a simplified cost function in 2D is,

$$cost = \eta_\psi(\psi_{uav} - \psi_{uav0}) \tag{8.7}$$

Where ψ_{uav0} is the UAV initial direction before starting the avoidance operation. ψ_{uav} is the yaw angle in which UAV could follow to avoid this obstacle.

Based on the Table 3.3 of 3D cost in chapter 3.4 , the corresponding cost table in 2D is developed as Table 8.1, where ψ_{1uav}, ψ_{2uav}, ...ψ_{muav} are all candidate yaw angles of UAV to choose to avoid the collision.

Table 8.1: Cost function to determine command direction in 2D space

$Cost(\psi_{1uav})$	$Cost(\psi_{2uav})$	...	$Cost(\psi_{muav})$

The diagram of the cost map is presented as Figure 8.2, where $\overrightarrow{UA}$ is the UAV original direction, which drives UAV into the white conflict zone. Therefore, a new direction and path is needed. From Figure 8.2, it is clear that the area(path) far away from the original trajectory or the white conflict zone has a higher cost than the path near the original trajectory.

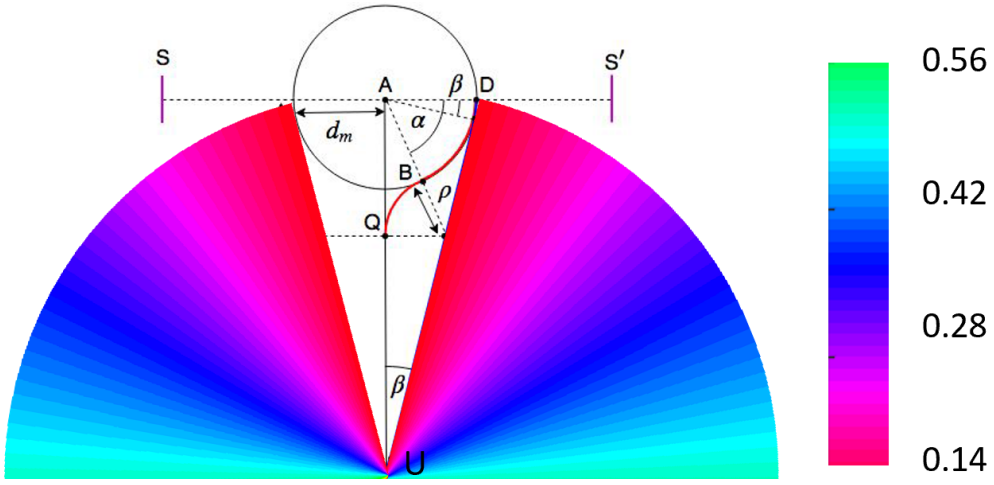


Figure 8.2: Cost distribution of Collision avoidance solutions

8.4 Flexible Selection of the Obstacle Avoidance Solutions

The aim of this section is to decide the least cost collision avoidance solution for UAV with required safety.

Use time-varying obstacles avoidance as an example. An UAV encounters one dynamic obstacle. The velocity of UAV is 20m/s and it flies north, that is 1.57 rad in $0-2\pi$ range. The speed of the obstacle is 15 m/s and it initially heads for southwest at 4.5 rad in $0-2\pi$ range. The obstacle is possible to change its direction but the UAV doesn't know beforehand.

The yaw angles which UAV could operate is shown in Table 8.2. "Y" under the angle means yes, that is the UAV could avoid the obstacles if it selects this angle as the commanded yaw angle. The blank means no.

Table 8.2: Geometric based collision avoidance solutions

0.22	0.37	0.52	0.67	0.82	0.77	1.12	1.27	1.42	1.57
Y	*Y*	*Y*	*Y*	-	-	-	-	-	-
1.72	1.87	2.02	2.17	2.32	2.47	2.62	2.77	2.92	3.07
-	-	-	-	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*

The possibility to avoid this obstacle is shown below in Table 8.3 if the obstacle changes its yaw angle and the variation follows the Gaussian distribution. The cost table corresponding to those collision avoidance solutions are shown in Table 8.4.

In this scenario the given safety requirement is 97% and the yaw angles 0.22, 0.36, 2.62, 2.77, 2.92, 3.07 rad satisfy the requirement. Among them, the least cost

Table 8.3: Possibility of successful collision avoidance

0.22	0.37	0.52	0.67	0.82	0.77	1.12	1.27	1.42	1.57
99.88%	98.74%	92.00%	68.26%	-	-	-	-	-	-
1.72	1.87	2.02	2.17	2.32	2.47	2.62	2.77	2.92	3.07
-	-	-	-	68.26%	92.00%	98.74%	99.88%	100%	100%

Table 8.4: Cost of different collision avoidance solutions

0.22	0.37	0.52	0.67	0.82	0.77	1.12	1.27	1.42	1.57
0.27	0.24	0.21	0.18	-	-	-	-	-	
1.72	1.87	2.02	2.17	2.32	2.47	2.62	2.77	2.92	3.07
-	-	-	-	0.15	0.18	0.21	0.24	0.27	0.3

is the 0.21 with yaw angle to be 2.62 rad. Thus, $\psi = 2.62$ is selected as the final commanded angle for UAV to do the collision avoidance action, as Figure 8.3 shows. The dark blue path with red small triangle in Figure 8.3 is the UAV trajectory while the red path with small cross is the obstacle's trajectory. The UAV avoids the obstacle and then comes back to its resumed path.

8.5 Conclusion

The most important contribution of this algorithm is that it doesn't need to predict the actual changing of the obstacle and then command and control our UAV to avoid that, which is difficult and doesn't fit for online application. Instead, based on the historic statistic data and the character of the obstacle, the possible

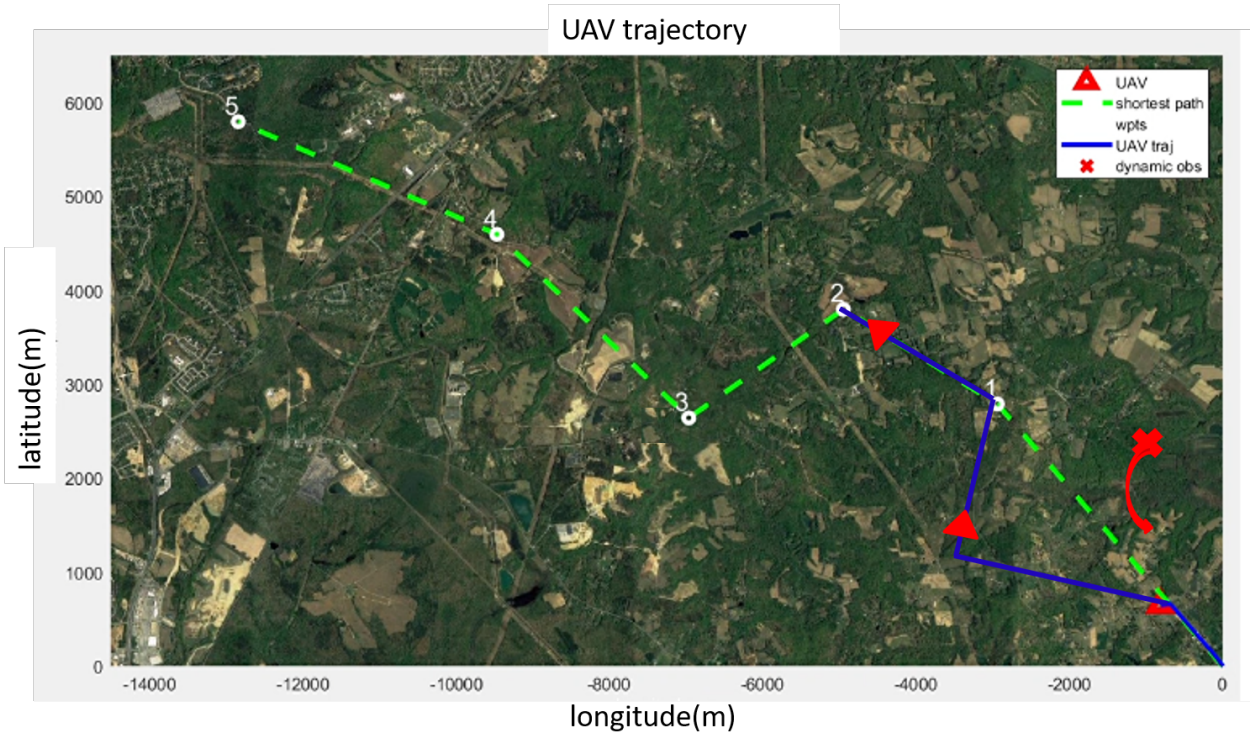


Figure 8.3: UAV avoids a time-varying obstacle

distribution of the obstacle's variation is presented with safety and cost map, and the optimal path can be chosen.

Chapter 9: Success Rates of Geometry Collision Avoidance Algorithm and the Failure Factor Analysis based on Machine Learning

Analysis of the geometry-based collision avoidance success or failure cases can help to improve the obstacle avoidance performance of the UAV. Hence, this chapter will contribute in two aspects. (1) provides the trend of of success rate vs different variables such as number of obstacle, speed range, detection distance and analyze their trend. (2) Uses the machine learning method, especially supervised logistic regression method to analyze and find the weight of these factors impacting on the final result of the UAV collision avoidance. The data used in this chapter is from Monte-carlo simulation in Matlab and Simulink.

9.1 Success Rates and the Experiment Results

This section will provide the collision avoidance success rate variation with the different variables and find the trend of UAV collision avoidance for different obstacles avoidance situation.

Figure 9.1 summarizes the Monte Carlo simulation results of collision scenarios

where the UAV encounters different static obstacles. The success rate decreases as (1) the initial detection distance (the distance between the obstacle and the UAV when the obstacle is first detected) decreases; (2) the number of static obstacles increases.

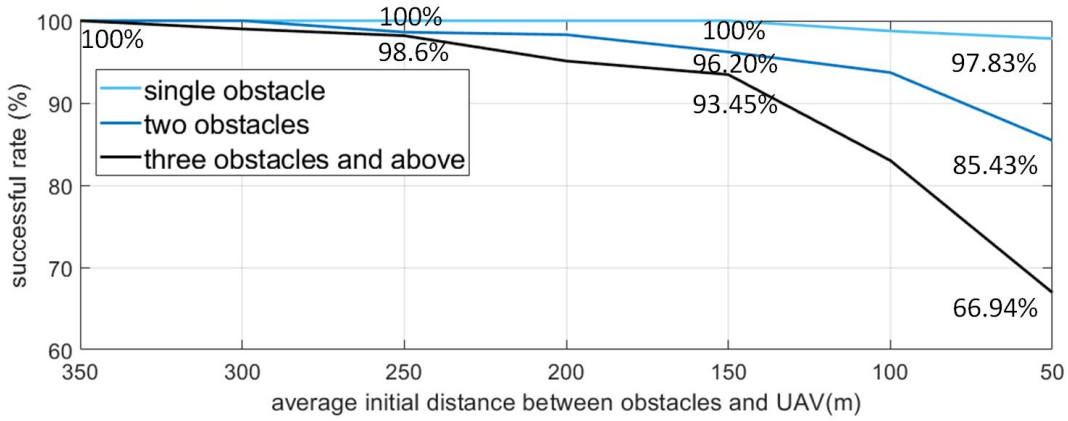


Figure 9.1: FGA static obstacle avoidance success rate

Monte Carlo simulation results for the case of two dynamic obstacles. The success rate decreases as (1) the initial distance (the distance between the UAV and the obstacle when the obstacle is first detected) decreases; (2) the number of obstacles increases, which increases the obstacle density; (3) the average velocity of the dynamic obstacles increases, which reduces the available response time and increases difficulty for the UAV to avoid the obstacles.

Among the above factors, Figure 9.2 summarizes the variation of collision avoidance success rate with initial detection distance and the average velocity of the UAV. If the obstacles' velocity is not larger than the velocity of the UAV (15m/s), 95% successful rate of FGA can be guaranteed for a 300-350m detection distance.

Figure 9.3 shows Monte Carlo simulation results for average obstacle velocity

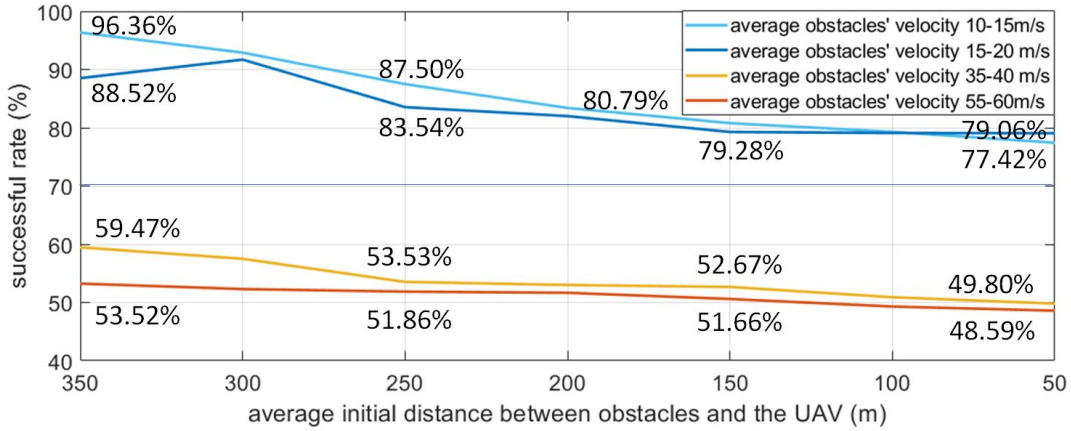


Figure 9.2: FGA dynamic obstacle avoidance success rate with initial distance and obstacles' velocity.

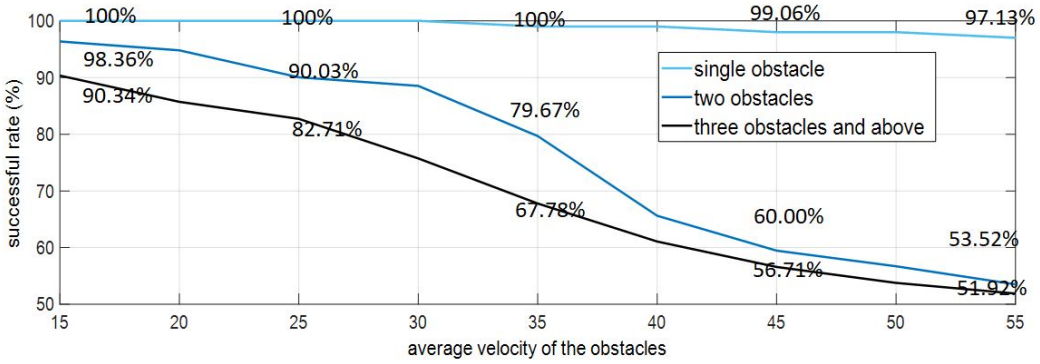


Figure 9.3: FGA dynamic obstacle avoidance success rate with obstacle average velocity and number of obstacles.

and number of obstacles, for a 300m initial distance. In this simulation the velocity of UAV is set to be 15m/s. It is clear that when the obstacles' average velocity increases, the successful avoidance rate goes down quickly. When the average velocity of obstacles are smaller than 30m/s (twice the UAV speed), the successful avoidance rate for two obstacles remains near a 90%. If the obstacle velocity goes up to 45-60m/s (3-4 times of the UAV's speed), the successful avoidance rate decreases to about 60%. It is compatible with the results from bound 2. In addition, for a single obstacle the UAV could avoid it easily with near full percentage while as the number of obstacles going up, the success rate decrease significantly.

9.2 Failure Case Analysis using Machine Learning

This section provides the analysis about the weight of the features impact on the collision avoidance result and outputs the prediction model. That is to say, when given the collision avoidance scenario with number of obstacles, type of obstacles, the detection distance, the speed of UAV and the obstacles and other relative parameters, the success or failure of this conflicts avoidance could be predicted through machine learning models.

Machine learning is used widely in the autonomous vehicles' area, including UAV and the self-driving car. Paper [54] uses the times of human being interfering with the self-driving mode to evaluate the autonomous level of this self-driving robot. Matlab has a good machine learning tool called classification learner, which helps in feature analysis and model production. The instruction for using machine learning

tool including uploading data, training the data and computing the weight will be available at the Appendix A. In this section, only the simulation results and corresponding analysis are presented.

- Prepare for the raw data.

For analyzing the weight of the factors which have impact on UAV collision avoidance result, we prepared 5000 cases of the geometric based obstacle avoidance, including successful and failure cases. Those cases are all simulated from the Matlab/Simulink software instead of hardware tests since thousands of hardware tests are not available at this moment. First, create a data format .mat named "dataset.dat" in the Matlab. The "dataset.dat" includes the features chosen to be analyzed: number of obstacles, ratio of the speed by obstacles and the UAV, detection distance, relative angle, relative altitudes and the results of successful collision avoidance or not. Figure 9.4 shows the 25 scenarios of the 5000 scenarios.

Note that 'obsNum' means the number of the obstacles. 'speedRatio' means the ratio of obstacles' average speed divided by UAV's speed. 'detectDis' means the average distance between the obstacles and the UAV when they are detected. 'relAngle' represents the direction difference. 'altDif' means whether the obstacles and the UAV are almost in the same altitude plane or they have difference in altitude, for which 1 represents they have difference while 0 means no. 'result' states whether the collision avoidance operation is successful with 1 success and 0 for failure.

- Train the data and select the model.

1 obsNum	2 speedRatio	3 detectDis	4 relAngle	5 altDif	6 result
1	1.7000	210.7473	0.0293	1	1
1	1.8586	296.1688	0.0528	1	1
1	1.0150	295.8419	0.0359	1	1
2	1.8674	261.3280	0.0433	0	0
1	1.8160	225.9596	0.1239	1	1
1	1.7368	251.1080	0.0192	1	1
1	1.2077	241.4454	0.0199	1	1
1	1.7878	229.8686	0.0410	1	1
1	1.2317	242.4003	0.1153	1	1
1	1.8566	238.1005	0.0211	1	1
3	1.5683	265.4581	0.0407	0	0
2	1.5125	267.2102	0.0677	0	1
2	1.3790	234.8851	0.0283	1	1
1	1.6163	267.2886	0.0277	1	1
1	1.2695	214.8322	0.0042	1	1
2	1.6644	252.7406	0.0633	1	1
1	1.5248	202.0950	0.0050	1	1
1	1.9149	250.7805	0.1013	1	1
1	1.9800	232.4888	0.0398	1	1
1	1.4761	201.4393	0.0840	1	1
1	1.7445	201.0899	0.1458	1	1
2	1.5527	240.0518	0.0534	0	1
2	1.6836	264.2869	0.0677	1	1
1	1.1958	256.8039	0.1106	1	1
2	1.3054	247.0310	0.0631	1	1

Figure 9.4: Features have effect on success rate of UAV collision avoidance

Classification learner will train the data in dozens to one hundred models and adjusts the parameters. In this case, the Coarse Tree and Coarse KNN has the highest overall predicted correctness rate of 94.9% with the proper true positive and true Negative rate. Figure 9.5 shows the Coarse Tree model with true negative(96.6%), true positive(94.8%) and low enough false positive (3.4%), false negative(5.2%), which is pretty good for the collision avoidance area.



Figure 9.5: Train the prediction model

True positive means the UAV avoids the obstacles in real and the prediction is right. True negative means in real world the UAV failures to avoid the obstacles and the prediction gives the right result. The false positive means that in real world the UAV avoids the obstacle but the model gives the wrong prediction. The false negative means that in real world the UAV failures the obstacle avoidance but the

model gives the wrong prediction. The higher value of the diagonal(blue block), which represents the true positive and true negative and the lower value of the auti-diagonal(pink block), which represents the false positive and false negative, the better the model is.

Then, output the model to workspace. Therefore, when given a collision avoidance scenario with above features, a prediction of whether the UAV could avoid the potential collision successfully will be produced.

- Find the weights of the factors impacting the UAV obstacle avoidance.

The last is to find the importance level of the features impact on the collision avoidance success rate. Pick one feature out and remain another four features each time. Then run the simulation and compare with the results produced by all features to find the most weighted feature.

From the simulation results, it is concluded that in geometry based obstacle avoidance process, the number of the obstacles, speed ratio of the obstacle, detection distance and the relative altitude are all very important features and contribute 22%, 23%, 26% 25% on the failure of collision avoidance. E.g. In Figure 9.6, lacking the consideration of the obstacles number will significantly decrease the prediction correctness. Among the five features, the only less important feature is the direction difference between the obstacles and the UAV, contributes for 4% on the failure cases. Thus, the UAV with more flexibility to accelerate or decelerate is more capable to avoid obstacles. Increasing the sensor detection distance to find the obstacles earlier also help with the UAV collision avoidance.

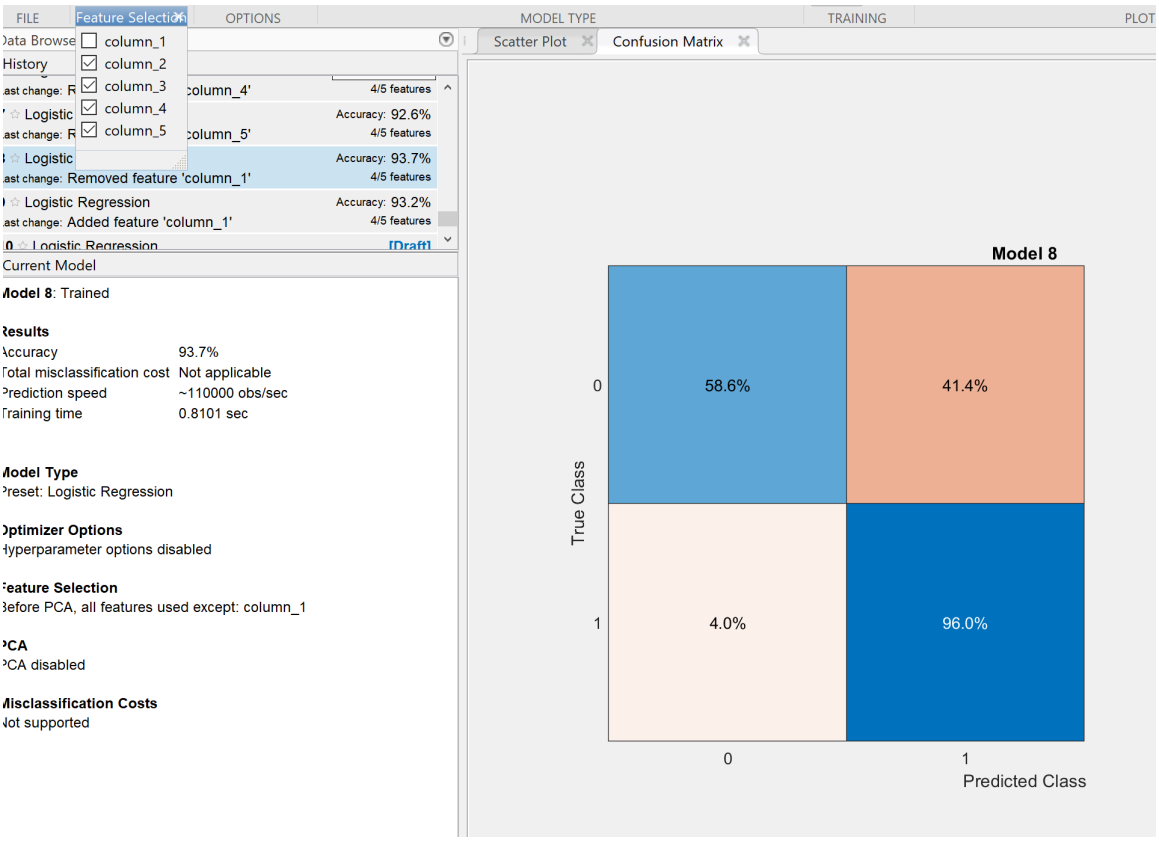


Figure 9.6: Decide the weight of each feature

Chapter 10: SITL Simulator and Hardware Experiment of Fixed-Wing Aircraft

The work of this thesis is implemented in the Simulink with UAV dynamic model, Software-In-The-Loop (SITL) vehicle simulator and hardware on board fixed-wing UAV aircraft with control center to be validated. There are five sections in this chapter with the description of each setup and result. Section 1 describes the Simulink setup and the simulation results. Section 2 explains the software in the loop experiments. Section 3 presents the work of hardware test on aircraft RC while section 4 is the further hardware implement with the control center. Section 5 discusses about the future work. All of the results verify the efficiency of the research.

10.1 Simulink Model and Test Environment Setup

A dynamic behavior model of UAV is built in the simulink for path planning, control and execution in an environment with randomly generated obstacles. There are three levels of the simulink model from high architecture to the basic level. (1) Overall model with five modules, Figure 10.1, (2) module of path planning including several collision avoidance functions, Figure 10.2, and (3) dynamic obstacles avoidance

function, Figure 10.3. They will be explained in sequence.

Figure 10.1 presents the overall flight model of UAV in a test environment. It includes five major modules. From left to right, they are: Detection, Path planning, Control, Actuator and Execution.

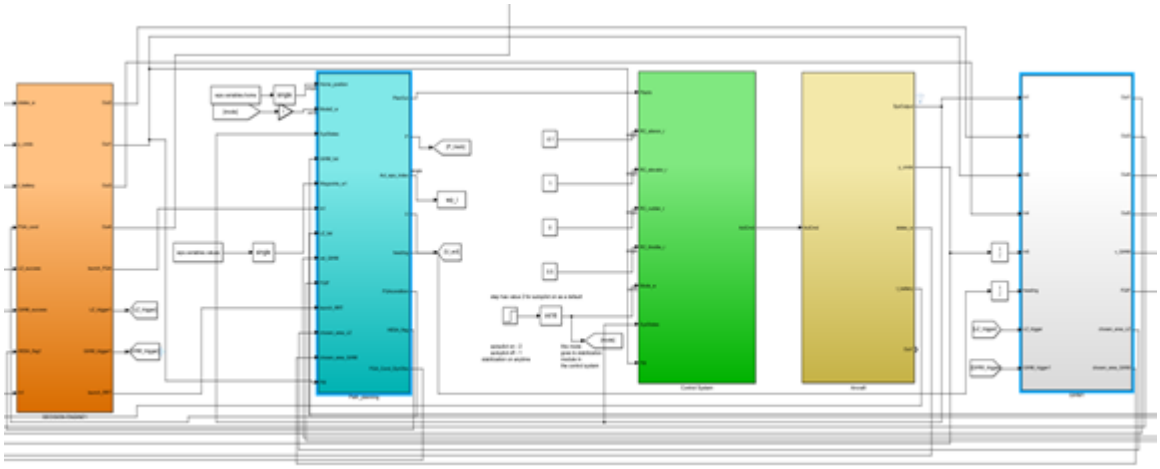


Figure 10.1: High architecture of the robot's dynamic system

- Detection. This part receives the information from virtual obstacles simulated and analyze their speed, direction, safe separation distance to determine the threaten level and check whether a collision avoidance operation is needed.

- Path planning. This part follows the detection part. If the original trajectory of UAV is threaten by the obstacles and a new path is needed, the path planning part will be active and generates the conflicts-free trajectory for the UAV.

- Control. After the path planning part issue a new yaw angle, pitch angle and velocity for UAV to avoid the conflicts, control module, which includes several

level of different controllers, e.g. PID controller would give the reference to UAV, compare the difference between UAV current states and commanded states and control the UAV to reach the set value.

- Actuator. This part is the model of rudder, aileron, elevator and throttle for the UAV, which are connected with the roll, yaw, pitch angle and speed of the UAV.

- Execution. This is the final module of the UAV dynamic model. It is designed to connect with the software engineering part to the hardware implement, transfer all information of its upstream to its downstream. Then measures the data of the aircraft and gives the feedback to the control module.

Figure 10.2 presents the path planning module of UAV, which is the middle level of the whole model. It includes three functions and they are,

- Obstacle generator. This function generates 20-30 obstacles with randomly direction, speed, distance. The information will broadcast to the detection module to determine. In each case, around 1-5 obstacles will become really threaten for the UAV.

- Static obstacle avoidance. This function helps UAV deal with the static obstacles such as electric power lines, buildings and blocked areas. The avoidance applies FGA algorithm and aims at operating collision avoidance at a least cost and least effect way.

- Dynamic obstacle avoidance. This function helps UAV deal with the dynamic obstacles such as other aircraft, birds. The avoidance applies FGA algorithm and aims at operating collision avoidance at a least cost and least effect way.

In this module, first, collision avoidance command is calculated and determined. Then send to control module and the follow modules to operate. After obstacle avoidance is done, the path planning will let the UAV search for the waypoints located in the original trajectories and track them.

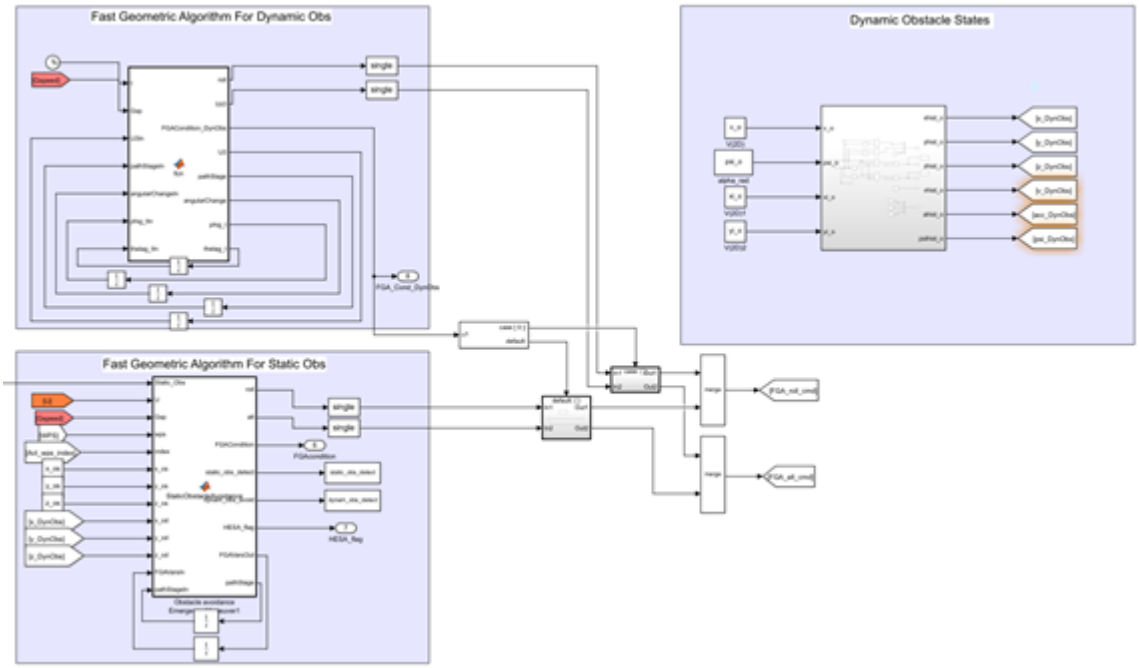


Figure 10.2: Path planning module including collision avoidance in the simulink

Figure 10.3 presents the dynamic obstacle avoidance function of UAV, which is the bottom level of the model. In this function, the UAV is assigned different stages after conflicts alarm given. Stage 1, continue flight in initial path until critical

avoidance time based on FGA. Stage 2, leave original path for collision avoidance. Stage 3, Check if the risk is relived.

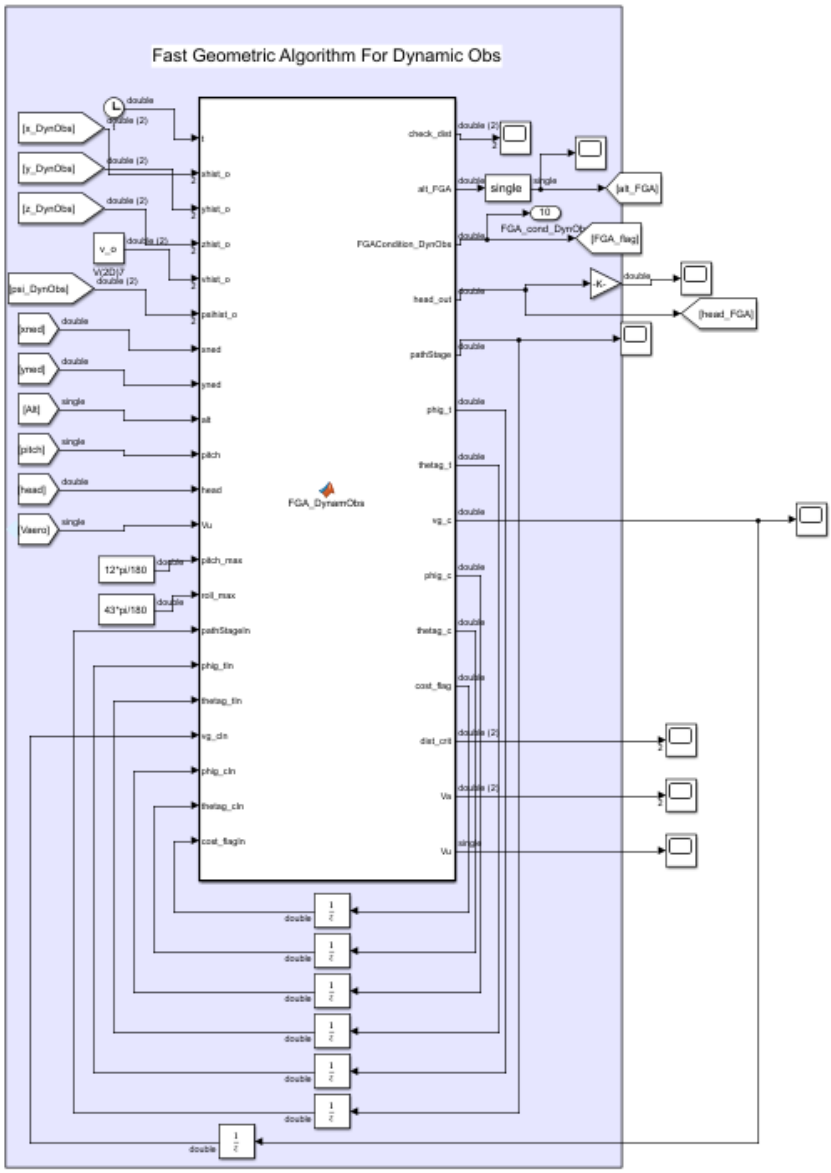


Figure 10.3: Dynamic obstacles avoidance function in the simulink

Here is a video shows the working mechanism and the simulation processes.

10.2 Software In The Loop (SITL) Simulator

In addition to Simulink model built by ourselves, which might have the simplification on dynamic model and environment, SITL provides the implement of collision avoidance algorithm into a commercial aircraft simulator software to validate its application efficiency. The SITL part in this section and hardware work in next section is conducted by my teammate Eward, Lina and me using my FGA collision avoidance algorithm and mainly cite from my previous work [1].

10.2.1 Ardupilot and DroneKit

The software Ardupilot [55] is used in this thesis for the software in the loop simulation (SITL) and the software preparation for hardware experiments. Ardupilot is an open source advanced commercial software of autopilot system in the unmanned vehicles, which could run on-board of both air and ground vehicles. The most advantage feature of Ardupilot and the reason to choose Ardupilot to conduct the SITL simulation research is that, the simulated flights with the full model in ArduPilot can be conducted on a real world vehicle without additional testings. Ardupilot is always work with the API DroneKit for implementing customized algorithm.

- Ardupilot

Ardupilot is a C++ based software to maximum the coding efficiency and it is consistent with both Linux and Windows operating systems. To better simulating the real UAS model, Ardupilot develops several relative function blocks such as full

kinematic model, full dynamic model with six degrees of freedom, different types of controllers including but not limited to PID controller and hardware interface. Ardupilot software also provides custom flight dynamic models (FDM) for customers to design kinematics and dynamic models needed for their vehicles [1].

Autopilot software always works the ground control station (GCS) for pilots or players to control or observe the unmanned vehicle from the ground [56]. GCS could afford a heavy computations, however, sending commands to the vehicles after computation on the GCS induce the communication loss, latency and slow down the action for the UAV. Therefore, on-board computation for obstacle avoidance and path planning for UAV is preferred in online application. FGA has light computation load and can be executed easily by the avionics interface on-board the vehicle. For our simulations, Linux system is used for Ardupilot implementation. The FDM model in Ardupilot is used for generating sensor data.

Figure 10.4 shows the architecture of the SITL setup [1] and here are the main variables used in the Figure 10.4.

ArduPlane.elf: The autopilot program in ArduPilot software

Sim Plane: FDM, provided alongside ArduPilot for SITL

FlightGear: 3D visualization program

MAVProxy: ground control station

DroneKit Scripts: Python API to provide the autonomous logic

- DroneKit

FGA collision avoidance algorithm in SITL simulation is implemented through

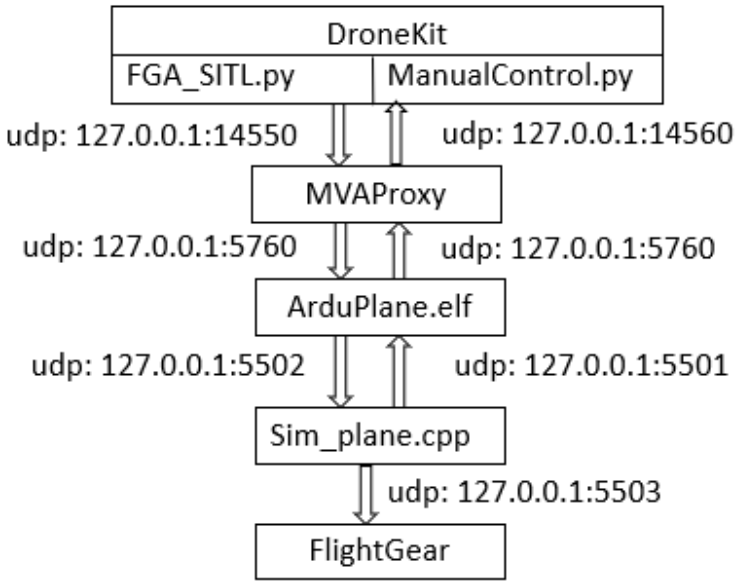


Figure 10.4: SITL Architecture

DroneKit, a Python API. The DroneKit allows the users to implement Python scripts that can communicates with the UAV or other unmanned vehicles by by sending out MAVLink commands. In addition to DroneKit API, another Python script was used together to translate keyboard inputs to PWM (Pulse Width Modulation) values can be used by control surfaces of the aircraft.

The implementation and links of DroneKit is shown in Figure 10.4

10.2.2 SITL Simulation Results

Figure 10.5 presents a screen of the SITL simulator [1]. In the figure, the UAV is avoiding a static obstacle. The simulator video showing the avoidance process is provided in the supplemental materials.

Figure 10.6 shows a visualization software used in the SITL runs. The simulation

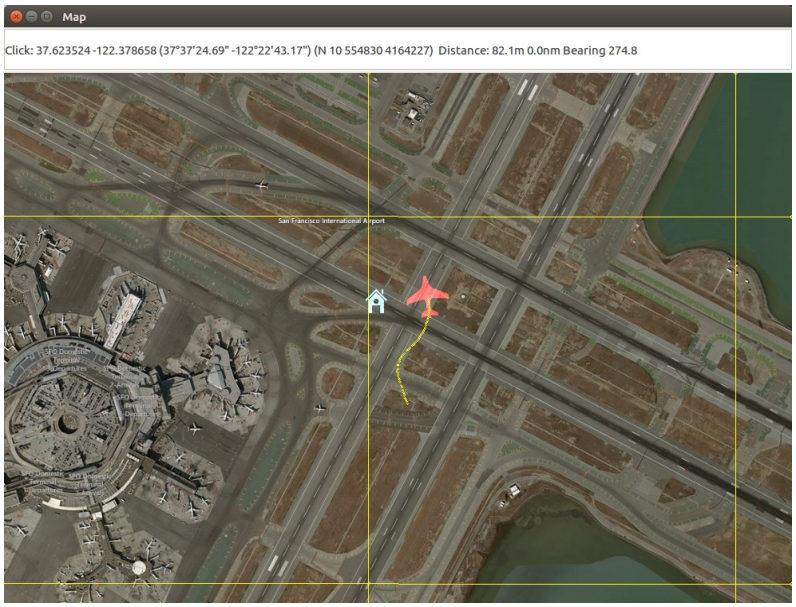


Figure 10.5: SITL simulator: Obstacle avoidance of static obstacle [1]



Figure 10.6: SITL with FlightGear visualization [1]

uses the data from San Francisco International Airport to generate the virtual environment and a static obstacle with safe separation distance $d_m = 150m$, placed on the intended path of the UAV.

Figure 10.7 shows the three flights of the aircraft in emergency avoidance maneuvers. The aircraft avoids conflicts of a virtual static obstacle with required protected radius from multiple angles.

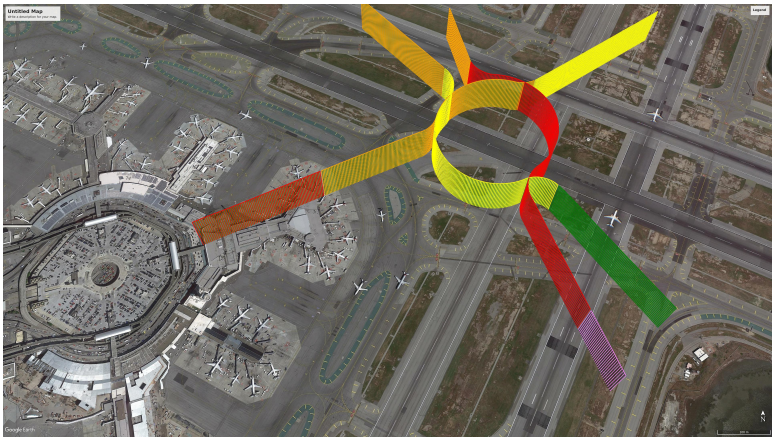


Figure 10.7: Critical avoidance SITL implementation: Flight paths and FGA avoidance, Google Earth view [1]

10.3 Hardware Implementation

For the hardware test, a shelf aircraft RC Apprentice 15e was instrumented with the NAVIO2 board, telemetry and GPS [1]. The FGA was implemented through DroneKit and MavLink in the NAVIO2 board, where the Ardupilot flight software runs. Figure 10.8 explains architecture of the hardware implementation.

For obstacle setup, the true static obstacles such as lines, buildings are chosen. For dynamic obstacles, virtual obstacles were simulated with pre-determined GPS

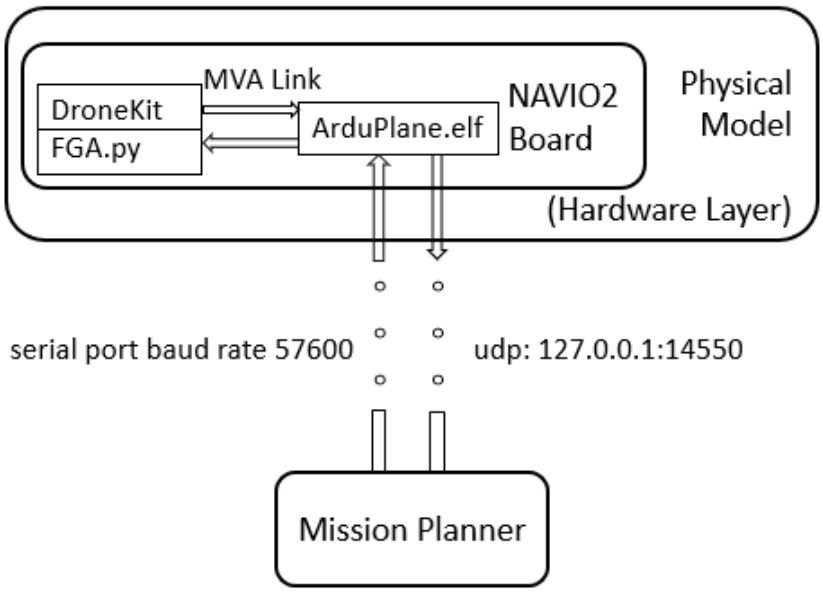


Figure 10.8: Hardware Implementation Architecture

coordinates. They have their specified safe distance d_m and starting location in GPS. The UAV and its ailerons will follow a desired avoidance maneuver including going straightly or turning a circle based on physically permitted maximum bank angle and the associated turning radius.

Algorithm 9 explains how FGA algorithm was implemented in the NAVIO2 board [1] (My teammate Edward summarized this algorithm). In FGA algorithm the UAV will turn at a maximum deviation to avoid the obstacle and its protected area. The algorithm implementation makes use of the dynamic control that ArduPilot provides with its flight modes. The algorithm decides whether there is a collision and subsequently calculates the new heading angle with expected turn radius based on Equation 3.1 to avoid the obstacle.

As described in the chapter 3, the FGA algorithm (implemented in the script)

Algorithm 9 Implementation of FGA algorithm [1]

1: **function** FGA(`uav,obs`) ▷ The ownship and obstacle

2: **while** `uav.mode` $\neq$ `CRUISE` **do** `<wait>`

3: **end while**

4: **if** objectDetected **then**

5: $P_{xy} \leftarrow$ PositionalAngle(`uav.pos, obs.pos`)

6: $\theta_{t0} \leftarrow P_{xy} -$ `uav.heading` ▷ current vehicle heading

7: $r \leftarrow$ norm(`uav.pos, obs.pos`) ▷ 2D distance to the obstacle

8:

9: **if** $|r \cdot sin\theta_{t0}| <$ `obs.dm` **then** ▷ Collision projected

10: $\theta_i \leftarrow$ `uav.heading`

11: ▷ Initial heading before deviation

12: $\rho_{calc} \leftarrow$ TurnRadius(`uav.vel, uav.bankAngle`)

13: $d_c \leftarrow \sqrt{(\texttt{obs.dm} + \rho)^2 + (r \cdot sin\theta_{t0} + \rho)^2)}$ ▷ Critical distance

14: $t_c \leftarrow \frac{d_c}{\texttt{uav.vel}}$ ▷ Critical Time

15: $\alpha \leftarrow cos^{-1}(\frac{\rho + r \cdot sin|\theta_{t0}|}{\rho + \texttt{obs.dm}})$ ▷ Max Deviation Angle

16: $t \leftarrow$ startTimer()

17: **while** norm(`uav.pos,obs.pos`) $\leq d_c$

18: **and** $t \geq t_c$ **do** `<wait>`

19: ▷ Waiting on either t_c or d_c to be met

20: **end while**

21: `uav.mode` ← `FBWB`. ▷ Setting vehicle to FBWB mode

22: `dir` ← SIGN(`uav.heading` − POSANGLE(`uav.pos, obs.pos`))

23: TURN(`uav, obs`, $\theta_i, \alpha, 1500 + dir \cdot 500$)

24: ▷ Turning away from obstacle until tangent

25: `pwm` ← PWMREQ(`uav.vel, obs.dm, dir`)

26: TURN(`uav, obs`, θ_i, α, `pwm`)

27: ▷ Turning along obstacle perimeter

28: TURN(`uav,obs`, $\theta_i, 0, 1500 +$ `dir` $\cdot 500$)

29: ▷ Passed obstacle, turn to initial heading

30: `uav.returnRoll()` ▷ Sets the aircraft roll back to 0

31: `vehicle.mode` ← `CRUISE`

32: **while** `vehicle.mode` ≠ `CRUISE` **do**

33: `<wait>`

34: ▷ Continues in CRUISE mode. Breaks if otherwise specified

35: **end while**

36: **end if**

37: **end if**

38: **end function**

simultaneously checks for the critical distance, critical time and safety. It alarms the UAV if it meets those critical thresholds or it has already reached the critical point when obstacle is detected. The decision control for triggering FGA is presented in Figure 10.9.

For our test, the timer is set on-board for minimizing the errors. In theory, the data available is always accurate and the all variables of UAV perform ideally. However, there is always the possibility in real hardware implementation for a physical model to have small uncertainty or estimation errors. Thus, the on-board timer could mitigates the errors caused by low sampling time from the GPS and decrease the risk of the late collision avoidance maneuver.

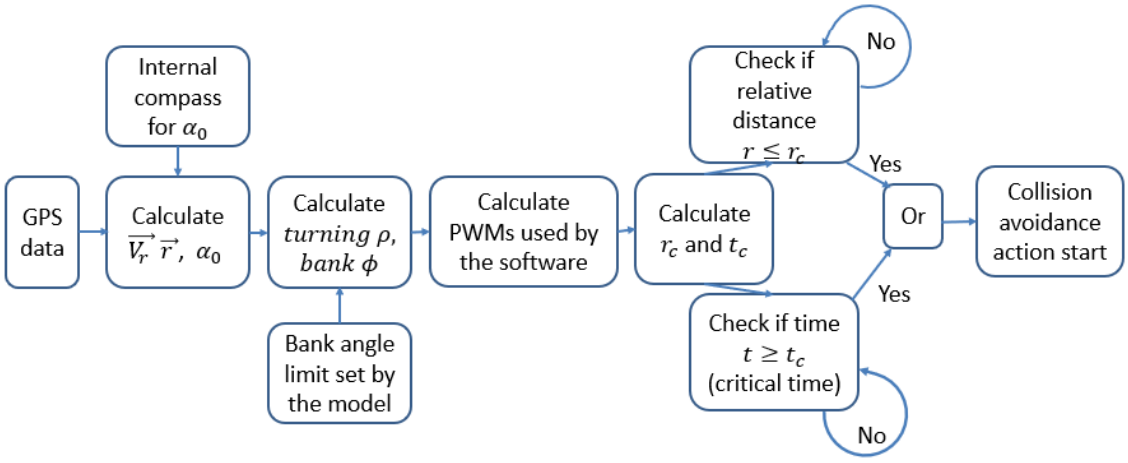


Figure 10.9: Distance and Time trigger for physical implementation

In the test case on the attached video, the vehicle is put in to fly-by-wire B (FBWB) mode, which is a mode of constant altitude and allows the roll angle to be controlled by pilot via RC control and to have the associated variation.

When there is a difference between the UAV's current yaw angle ψ and the commanded yaw angle ψ_c, the UAV will use the turning angle rate ω, the turning radius ρ and the corresponding bank angle ϕ to reach the desired value.

$$\omega = \dot{\psi} = K_p(\psi_c - \psi) + K_i \int (\psi_c - \psi) + K_d \frac{d(\psi_c - \psi)}{dt}, \tag{10.1}$$

$$\rho = \frac{V}{\omega}; \phi = tan^{-1}(\frac{V^2}{11.26\rho}). \tag{10.2}$$

Where $\phi_{min} \leq \phi \leq \phi_{max}$. If required bank angle ϕ is beyond the range, then Equation 10.3 is applied here to calculate the minimum turning radius ρ:

$$\rho_{min} = \frac{V^2}{11.26 tan \phi_{max}} \tag{10.3}$$

The hardware flight test is implemented in the customized Apprentice RC aircraft, which is shown in Figure 10.10 and the two videos of hardware experiment result is provided in detail on my research [1].



Figure 10.10: Hardware Implementation: Apprentice Aircraft

10.4 Hardware Setup with Control Center

The FGA collision avoidance algorithm was implemented into a hardware in the loop simulated environment with control center, which was developed by ajoint collaboration between University of Maryland and Lina from Millennium Engineering. In this setup, the collision avoidance module ran in the onboard AIFMS software, which include: global overall path planning from start point to destination and local path planning for collision avoidance, decision making for hazard mitigation, ground impact mitigation and emergency path landing including landing or path planning with maneuver limitation. Figure 10.11 presents the hardware setup including the control surface and the communication.

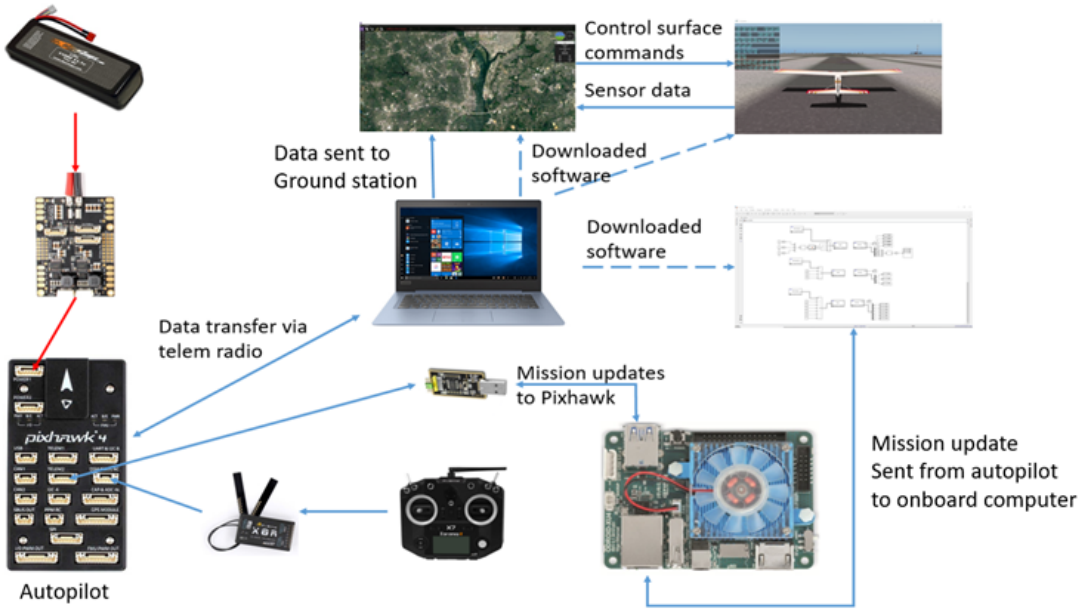


Figure 10.11: Hardware Set up with the control center

The HITL setup of this section is the further step extended from section 10.3. It consists of a ground station computer called Ground Control with visualization software Xplane, a UAV with an autopilot board, RC radio transmitter & receiver, and on-board computer. Other ground stations like Mission Planner and other visualization software like Gazebo can also be used. In the hardware in the loop setup, the airplane was simulated through the Xplane environment, where all the variables involved in realistic flight were provided by the simulated environment.

This HITL test is more close to the real flight task since it allows for the programming of winds and turbulence. Simulated obstacles were successfully avoided by the UAV which had the FGA collision avoidance algorithm. The avoidance was achieved in three dimension space, with combinations of vertical and horizontal angular deviations. In addition, the collision avoidance module was interfaced with other hazard mitigation modules such as ground impact mitigation module and emergency landing module for a holistic approach for autonomous vehicles' navigation and path planning.

10.5 Future Work

There are two interesting aspects worth to be discussed in the future.

- The obstacles perform dangerous actions

All of the discussions in this dissertation are based on a precondition that the intruder doesn't have the hostility to the UAV and thus will not perform at a dangerous way to collide with our commanded UAV. However, if in a combat

situation, the FGA algorithm needs the improvement to (1) avoid the intruders which might or intend to catch our UAV, and (2) revise the cost function to let the UAV choose an optimal obstacles-free path with least cost.

- All UAV are equipped with FGA algorithm

This part discusses a situation if all UAV and moving intruders are implemented with the FGA collision avoidance algorithm. Assume there are two UAV U_a and U_b meet each other on the air. If one of them performs FGA avoidance action while another keeps resumed path, it is safe for both of them. However, if both of them perform avoidance algorithm individually, then it is possible that the two UAVs choose the same direction to avoid the conflicts, which makes the collision happen quicker. Thus, the communication or the checking time and the states updating time should be considered in the further steps.

Appendix A: Using Matlab Machine Learning Tool to Get the Prediction Model and the Weight of the Factors

Matlab has a good machine learning tool called classification learner, which helps in feature analysis and model production. We prepared 5000 cases of the geometric based obstacle avoidance, including successful and failure cases and use machine learning method to analyze the causes. The steps are presented below.

The first step is prepare for the raw data. Create a data format .mat named dataset.dat in the Matlab. It includes the features chosen to be analyzed: number of obstacles, ratio of the speed by obstacles and the UAV, detection distance, relative angle, relative altitudes and the results of successful collision avoidance or not. Figure A.1 shows the 25 scenarios of the 5000 scenarios.

Note that 'obsNum' means the number of the obstacles. 'speedRatio' means the ratio of obstacles' average speed divided by UAV's speed. 'detectDis' means the average distance between the obstacles and the UAV when they are detected. 'relAngle' represents the direction difference. 'altDif' means whether the obstacles and the UAV are almost in the same altitude plane or they have difference in altitude, for which 1 represents they have difference while 0 means no. 'result' states whether the collision avoidance operation is successful with 1 success and 0 for failure.

1 obsNum	2 speedRatio	3 detectDis	4 relAngle	5 altDif	6 result
1	1.7000	210.7473	0.0293	1	1
1	1.8586	296.1688	0.0528	1	1
1	1.0150	295.8419	0.0359	1	1
2	1.8674	261.3280	0.0433	0	0
1	1.8160	225.9596	0.1239	1	1
1	1.7368	251.1080	0.0192	1	1
1	1.2077	241.4454	0.0199	1	1
1	1.7878	229.8686	0.0410	1	1
1	1.2317	242.4003	0.1153	1	1
1	1.8566	238.1005	0.0211	1	1
3	1.5683	265.4581	0.0407	0	0
2	1.5125	267.2102	0.0677	0	1
2	1.3790	234.8851	0.0283	1	1
1	1.6163	267.2886	0.0277	1	1
1	1.2695	214.8322	0.0042	1	1
2	1.6644	252.7406	0.0633	1	1
1	1.5248	202.0950	0.0050	1	1
1	1.9149	250.7805	0.1013	1	1
1	1.9800	232.4888	0.0398	1	1
1	1.4761	201.4393	0.0840	1	1
1	1.7445	201.0899	0.1458	1	1
2	1.5527	240.0518	0.0534	0	1
2	1.6836	264.2869	0.0677	1	1
1	1.1958	256.8039	0.1106	1	1
2	1.3054	247.0310	0.0631	1	1

Figure A.1: Features have effect on success rate of UAV collision avoidance

Second, upload the data into Matlab. Open the Matlab and run the dataset.dat and it will be stored in the workspace part of the Matlab.

Third, open classification learner and create new session. Choose the dataset.dat from workspace such as Figure A.2 presents. For the validation of the right part, click on the Holdout Validation since it is aimed at dealing large dataset(The cross-validation is used for small dataset). Choose 25% held out, which represents that 75% data is used for machine learning training and the remain 25% for the testing.

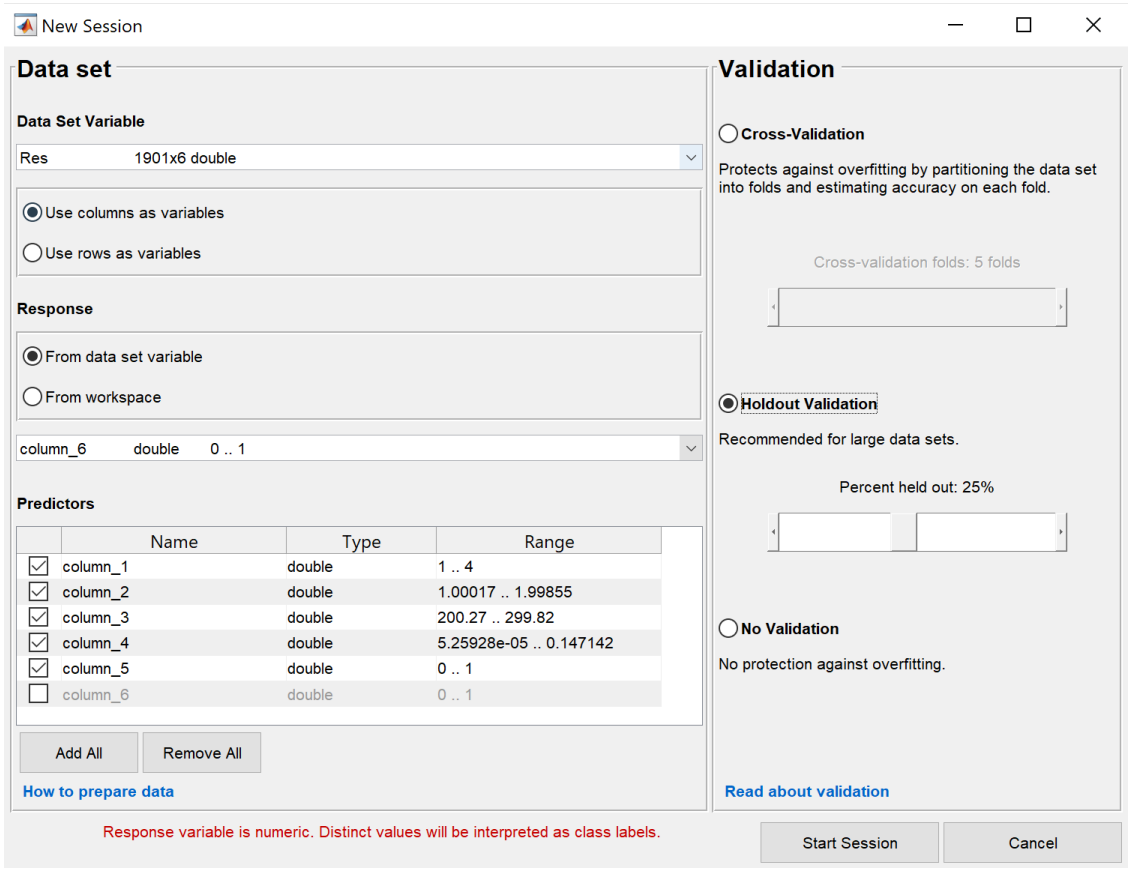


Figure A.2: Choose the data from workspace

Forth, train the data. Choose "all quick to train" bottom and click the green train bottom to run. Then choose the model with highest prediction accuracy Figure

A.3. In this case, the Coarse Tree and Coarse KNN has the highest predicted correctness rate of 94.9% with the proper true positive and true Negative rate.



Figure A.3: Train the prediction model

True positive means the UAV avoids the obstacles in real and the prediction is right. True negative means in real world the UAV failures to avoid the obstacles and the prediction gives the right result. The false positive means that in real world the UAV avoids the obstacle but the model gives the wrong prediction. The false negative means that in real world the UAV failures the obstacle avoidance but the model gives the wrong prediction. The higher value of the diagonal(blue block), which represents the true positive and true negative and the lower value of the auti-diagonal(pink block), which represents the false positive and false negative, the better the model is. Figure A.3 shows an excellent model with very high true negative(96.0%), true

positive(94.7%) and low enough false positive (4.0%), false negative(5.3%)

Fifth, output the model to workspace through export selection as in Figure A.4. Then, when given a collision avoidance scenario with above features, it would predict whether the UAV could successfully avoid the potential collisions.

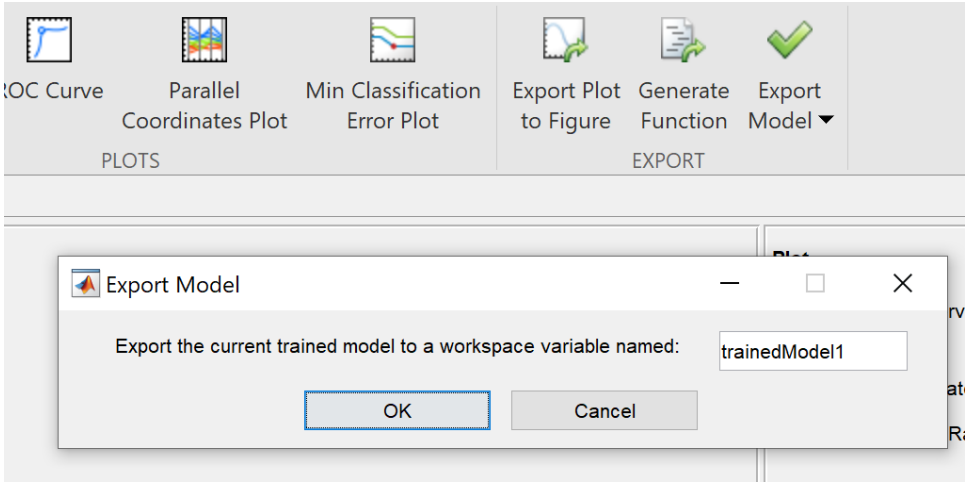


Figure A.4: Export the prediction model

Sixth, find the importance level of the features impact on the collision avoidance success. Open the feature selection as Figure A.5 shows.

Select one feature out and keep another four features each time as Figure A.6 shows. Then run the simulation and compare with the results produced by all features to find the most weighted feature.

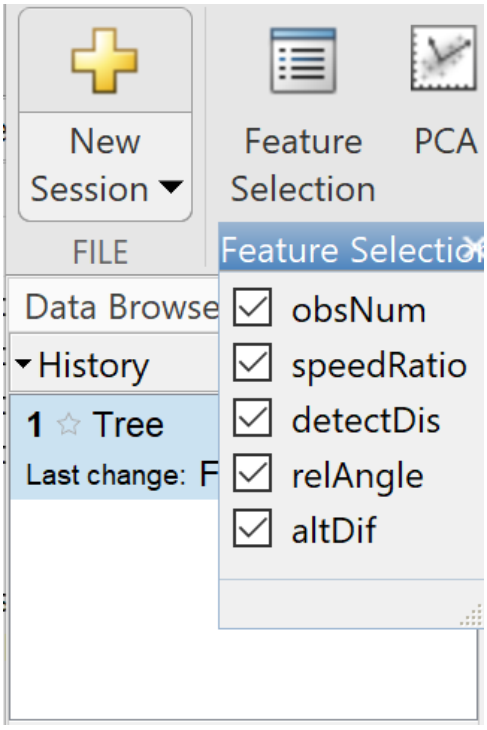


Figure A.5: Decide the weight of each feature

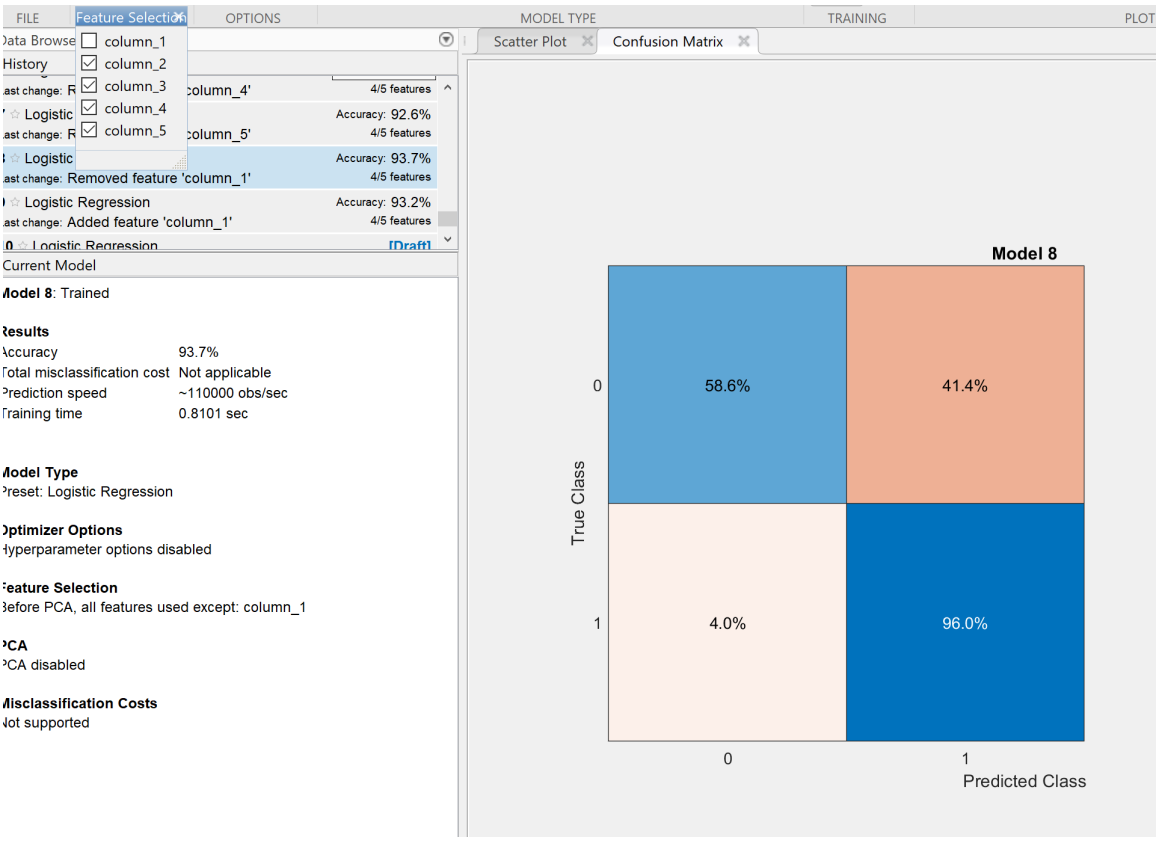


Figure A.6: Decide the weight of each feature

Bibliography

[1] Z. Lin, L. Castano, E. Mortimer, and H. Xu, "Fast 3d collision avoidance algorithm for fixed wing uas," *Journal of Intelligent & Robotic Systems*, pp. 1–28, 2019.

[2] H. Xiang and L. Tian, "Development of a low cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav)," *Biosystems engineering*, vol. 108, no. 2, pp. 174–190, 2011.

[3] R. J. Kephart and M. S. Braasch, "See-and-avoid comparison of performance in manned and remotely piloted aircraft," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 5, pp. 36–42, 2010.

[4] I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Physical review letters*, vol. 113, no. 23, p. 238701, 2014.

[5] P. Angelov, *Sense and avoid in UAS: research and applications*. John Wiley & Sons, 2012.

[6] J. Linchant, J. Lisein, J. Semeki, P. Lejeune, and C. Vermeulen, "Are unmanned aircraft systems (uas s) the future of wildlife monitoring? a review of accomplishments and challenges," *Mammal Review*, vol. 45, no. 4, pp. 239–252, 2015.

[7] C. von Essen and D. Giannakopoulou, "Analyzing the next generation airborne collision avoidance system," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2014, pp. 620–635.

[8] M. Huerta, "Integration of civil unmanned aircraft systems (uas) in the national airspace system (nas) roadmap," *Federal Aviation Administration, Retrieved Dec*, vol. 19, p. 2013, 2013.

[9] FAA, "Reported uas sightings (april 2020-june 2020) (ms excel)," *https* : $//www.faa.gov/uas/resources/public_records/uas_sightings_report/$, 2020.

[10] J. Haugen and L. Imsland, "Monitoring moving objects using aerial mobile sensors," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 2, pp. 475–486, 2016.

[11] C. Luo, S. I. McClean, G. Parr, L. Teacy, and R. De Nardi, "Uav position estimation and collision avoidance using the extended kalman filter," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 6, pp. 2749–2762, 2013.

[12] S. Saha, A. Natraj, and S. Waharte, "A real-time monocular vision-based frontal obstacle detection and avoidance for low cost uavs in gps denied environment," in *Aerospace Electronics and Remote Sensing Technology (ICARES), 2014 IEEE International Conference on*. IEEE, 2014, pp. 189–195.

[13] Y. Kwang and Y. Kwang, "Performance simulation of radar sensor based obstacle detection and collision avoidance for smart uav," in *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th*, vol. 2. IEEE, 2005, pp. 10–pp.

[14] K. Al-Mutib, M. AlSulaiman, M. Emaduddin, H. Ramdane, and E. Mattar, "D* lite based real-time multi-agent path planning in dynamic environments," in *2011 Third International Conference on Computational Intelligence, Modelling & Simulation*. IEEE, 2011, pp. 170–174.

[15] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*. IEEE, 2013, pp. 1–8.

[16] K. Sigurd and J. How, "Uav trajectory design using total field collision avoidance," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003, p. 5728.

[17] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[18] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE, 1991, pp. 1398–1404.

[19] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[20] B. I. Kazem, A. H. Hamad, and M. M. Mozael, "Modified vector field histogram with a neural network learning model for mobile robot path planning and obstacle avoidance." *Int. J. Adv. Comp. Techn.*, vol. 2, no. 5, pp. 166–173, 2010.

[21] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[22] Y. Zhang and G. Wang, "An improved rgb-d vfh+ obstacle avoidance algorithm with sensor blindness assumptions," in *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, 2017, pp. 408–414.

[23] Y. Lin and S. Saripalli, "Path planning using 3d dubins curve for unmanned aerial vehicles," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 296–304.

[24] P. Pharpatara, B. Hérissé, and Y. Bestaoui, "3-d trajectory planning of aerial vehicles using rrt," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 1116–1123, 2017.

[25] P. Liu, H. Yu, and S. Cang, "Optimized adaptive tracking control for an underactuated vibro-driven capsule system," *Nonlinear Dynamics*, vol. 94, no. 3, pp. 1803–1817, 2018.

[26] ——, "Trajectory synthesis and optimization of an underactuated microrobotic system with dynamic constraints and couplings," *International Journal of Control, Automation and Systems*, vol. 16, no. 5, pp. 2373–2383, 2018.

[27] T. Yu, J. Tang, L. Bai, and S. Lao, "Collision avoidance for cooperative uavs with rolling optimization algorithm based on predictive state space," *Applied Sciences*, vol. 7, no. 4, p. 329, 2017.

[28] P. Yang, K. Tang, J. A. Lozano, and X. Cao, "Path planning for single unmanned aerial vehicle by separately evolving waypoints," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1130–1146, 2015.

[29] D.-S. Jang, H.-J. Chae, and H.-L. Choi, "Optimal control-based uav path planning with dynamically-constrained tsp with neighborhoods," *arXiv preprint arXiv:1612.06008*, 2016.

[30] D. Bozhinoski, A. Bucchiarone, I. Malavolta, A. Marconi, and P. Pelliccione, "Leveraging collective run-time adaptation for uav-based systems," in *Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on*. IEEE, 2016, pp. 214–221.

[31] H. Jing-Lin, S. Xiu-Xia, L. Ri, D. Xiong-Feng, and L. Mao-Long, "Uav real-time route planning based on multi-optimized rrt algorithm," in *Control And Decision Conference (CCDC), 2017 29th Chinese*. IEEE, 2017, pp. 837–842.

[32] D. Pascarella, S. Venticinque, and R. Aversa, "Autonomic agents for real time uav mission planning," in *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. IEEE, 2013, pp. 410–415.

[33] A. Mujumdar and R. Padhi, "Reactive collision avoidance of using nonlinear geometric and differential geometric guidance," *Journal of guidance, control, and dynamics*, vol. 34, no. 1, pp. 303–311, 2011.

[34] J.-W. Park, H.-D. Oh, and M.-J. Tahk, "Uav conflict detection and resolution based on geometric approach," *International Journal of Aeronautical and Space Sciences*, vol. 10, no. 1, pp. 37–45, 2009.

[35] S. J. Guy, J. Van Den Berg, M. C. Lin, and D. Manocha, "Geometric methods for multi-agent collision avoidance," in *Proceedings of the twenty-sixth annual symposium on Computational geometry*. ACM, 2010, pp. 115–116.

[36] Z. Daniels, L. Wright, J. Holt, and S. Biaz, "Collision avoidance of multiple uas using a collision cone-based cost function," *Computer Science and Software Engineering Department, Auburn University, Tech. Rep. CSSE12-07*, 2012.

[37] J. Seo, Y. Kim, and A. Tsourdos, "Differential geometry based collision avoidance guidance for multiple uavs," *IFAC Proceedings Volumes*, vol. 46, no. 19, pp. 113–118, 2013.

[38] Y. Watanabe, A. Calise, E. Johnson, and J. Evers, "Minimum-effort guidance for vision-based collision avoidance," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2006, p. 6641.

[39] X. Yang, L. M. Alvarez, and T. Bruggemann, "A 3d collision avoidance strategy for uavs in a non-cooperative environment," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 315–327, 2013.

[40] C. Yoo, A. Cho, B. Park, Y. Kang, S. Shim, and I. Lee, "Ads-b hils test for collision avoidance of smart uav," in *2011 Tyrrhenian International Workshop on Digital Communications-Enhanced Surveillance of Aircraft and Vehicles*. IEEE, 2011, pp. 253–257.

[41] H.-S. Shin, A. Tsourdos, and B. White, "Uas conflict detection and resolution using differential geometry concepts," *Sense and avoid in UAS: Research and applications*, vol. 62, 2012.

[42] J. Goss, R. Rajvanshi, and K. Subbarao, "Aircraft conflict detection and resolution using mixed geometric and collision cone approaches," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 4879.

[43] D. Yang, D. Li, and H. Sun, "2d dubins path in environments with obstacle," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[44] Z. Lin, L. Castano, and H. Xu, "A fast obstacle collision avoidance algorithm for fixed wing uas," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 559–568.

[45] Q. Wang and J. Zhang, "Mpc and tgfc for uav real-time route planning," in *Control Conference (CCC), 2017 36th Chinese*. IEEE, 2017, pp. 6847–6850.

[46] Z. Shi and W. K. Ng, "A collision-free path planning algorithm for unmanned aerial vehicle delivery," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 358–362.

[47] Z. Lin, L. Castano, and H. Xu, "Uav collision avoidance with varying trigger time," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1257–1264.

[48] K.-Y. Kim, J.-W. Park, and M.-J. Tahk, "Uav collision avoidance using probabilistic method in 3-d," in *2007 International Conference on Control, Automation and Systems*. IEEE, 2007, pp. 826–829.

[49] J. Nilsson, M. Ali, P. Falcone, and J. Sjoberg, "Predictive manoeuvre generation for automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 418–423.

[50] H. Andersen, W. Schwarting, F. Naser, Y. H. Eng, M. H. Ang, D. Rus, and J. Alonso-Mora, "Trajectory optimization for autonomous overtaking with visibility maximization," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–8.

[51] Z. Lin, L. Castano, and H. Xu, "Uav maneuver limitatio on obstacle avoidance," in *in proceeding of IEEE Access*, 2021.

[52] K. W. Williams, "A summary of unmanned aircraft accident/incident data: Human factors implications," FEDERAL AVIATION ADMINISTRATION OKLAHOMA CITY OK CIVIL AEROMEDICAL INST, Tech. Rep., 2004.

[53] M. Coombes, W.-H. Chen, and P. Render, "Reachability analysis of landing sites for forced landing of a uas," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 635–653, 2014.

[54] J. J. Dudley and P. O. Kristensson, "A review of user interface design for interactive machine learning," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 8, no. 2, pp. 1–37, 2018.

[55] A. D. Team, "Ardupilot autopilot suite," *http : //ardupilot.com/Accessed*, pp. 05–20, 2016.

[56] L. Meier, J. Camacho, B. Godbolt, J. Goppert, L. Heng, M. Lizarraga *et al.*, "Mavlink: Micro air vehicle communication protocol," *Tillgänglig: http : //qgroundcontrol.org/mavlink/start[Hämtad 2014-05-22]*, 2013.