ABSTRACT

Title of Thesis: Mission and Scenario Planning
for Unmanned Aerial Vehicles
(Path planning and Collision Avoidance systems)

Niloofar Shadab, Master of Science, 2016

Thesis directed by: Professor Huan Xu
Institute for Systems Research

As unmanned autonomous vehicles (UAVs) are being widely utilized in military and civil applications, concerns are growing about mission safety and how to integrate different phases of mission design. One important barrier to a cost-effective and timely safety certification process for UAVs is the lack of a systematic approach for bridging the gap between understanding high-level commander/pilot intent and implementation of intent through low-level UAV behaviors. In this thesis we demonstrate an entire systems design process for a representative UAV mission, beginning from an operational concept and requirements and ending with a simulation framework for segments of the mission design, such as path planning and decision making in collision avoidance.

In this thesis, we divided this complex system into sub-systems; path planning, collision detection and collision avoidance. We then developed software modules for each sub-system.

Mission and Scenario Planning for UAV's
(Path planning and Collision Avoidance systems)

by

Niloofar Shadab

Thesis 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
Master of Science
2016

Dedication

This thesis is proudly dedicated to my wonderful family, my mother Fereshteh, my father Mahmoud, my sister and my brother, who are always there to support me in my journey through life. I want to thank them for their endless love and support.

Acknowledgments

I express my gratitude to all the people who have helped me to make this research possible and because of whom I will have the honor of graduating from an accredit university in the US.

First and foremost I'd like to thank my advisor, Professor Mumu Xu for giving me an incredible opportunity to work on one of the hottest topics in our era. Working with her gave me a chance to learn how to conduct academic research. She was always there to help and provide suggestions for me during these two years that I worked with her as a graduate research assistant. She as a young talented woman and dedicated researcher is definitely a role model for me.

I would also like to thank Millennium Engineering and Integration Company. Without their supports and helps, our research and this thesis would have been a distant dream. I want to specifically thank Mr. Kerry Wisnosky, Mrs. Kajal Pancholi and prof. Wereley for their continuous support and productive suggestions during these two years.

I appreciate all the suggestions and helpful feedbacks that my committee member, Dr. David Lovell and Dr. Jeffrey W. Herrmann, provided. I am gratefully indebted to them for their very valuable comments on this thesis.

I would also like to acknowledge help and support from some of the ISR members. Dr. MacCarthy's help and advices are highly appreciated, also I would thank Mr. Jeff Corial and Ms. Regina King for all the arrangements and supports they did.

I would like to acknowledge some of my teammates, , Scott C. Livingston, William Harris and Josh Gaus who helped me specifically at the early stages of my research. They were such resourceful students that I had the pleasure to work with. Many thanks to Dr. Khazaeli for his valuable advices and helps during the time I was in the USA. I am also very grateful to have fantastic friends who helped me a lot during my study. My best friends, Ladan, Parastoo, Mahshid and Melika who always provided me with best advices and were always there for me when I needed their support. My dear friends, Connor, Arjuna and Sarah who specially helped me be prepared for my thesis defense. In addition, I had many good friends, Niloofar, Roshanak, Mohammad, Ali, Pouya, Navya, Charles, Alborz, Alireza and Melika, my cousins, Emil and Maryam, who made my time here a lot more fun and enjoyable.

Last but not least, I owe my deepest thanks to my family - my mother, Fereshteh, my father, Mahmoud, my sister and my brother who have always stood by me and guided me through my career, and gave me their inordinately and unconditional love and support. Words cannot express the measure of respect that I owe them. I would also like to thank my second family, my uncle Mr. Cyrus Taadolmanesh and his wonderful wife Mrs. Armineh Taadolmanesh for their love and support that made me forget about being away from my parents.

Lastly, thank you all and thank God!

Table of Contents

List of Figures

List of Abbreviations

α	alpha
β	beta
ρ	Turning Angle
ψ	Heading Angle
π^c	Bank Angle
γ^c	Flight Path Angle
τ	Time to Closest Point of Approach
ATC	Air Traffic Control
CAD	Closest Approach Distance
d_c	Closest Distance
d_m	Minimum Separation Distance
d_p	Relative Distance
L	Left
MOE	Measure of Effectiveness
NAS	National Air Space
Q	Steering angle of Dubins Car
q_I	Start Point
q_G	Final Point
R	Right
S	Straight
SARGM	Situational Awareness and Response Guidance Module
TCPA	Time to Closest Point of Approach
t_F	Time to reach to departure
UAV	Unmanned Air Vehicle
U_s	Forward Speed of Dubins Car
V	Air Speed

Chapter 1: Introduction

1.1 Introduction

Unmanned autonomous vehicles (UAVs) are increasingly utilized in military and civilian application due to their potential to provide improved capabilities while increasing manpower efficiency [2] [4]. Current and future domestic applications for UAVs include search and rescue, weather forecasting, law enforcement, border patrol, firefighting, disaster response, precision farming, commercial fisheries, scientific research, aerial photography, mail delivery, infrastructure monitoring and emergency management [4]. As a result of the prevalence of UAVs, particularly in civilian applications, there are growing concerns with regard to the safe integration of UAVs into the national airspace (NAS). The safety and reliability of UAVs are highly reliant on their capability to avoid emergency situations in order to have a safe flight. However, a lack of appropriate systematic method prevents high-level autonomous systems from being widely fielded. Since just by using a systematic approach, designers are able to capture all possible scenarios and trace back low-level behaviors and commands into the high-level mission requirements. Therefore, new techniques for standardized and formalized requirements specification and mission planning of UAVs are needed that take into account discrete decision-making and can be integrated with flight simulation software in order to verify the overall system. Using systems engineering methodologies for solving high-level problems and tracing them into the lower level problems can reduce the risk of failure and catastrophe as they provide a platform for identifying the relationships between the system's elements and predicting the potential malfunctions and failures.

The goal of this thesis is to develop a preliminary flight management system model beginning from real-world problem definition and ending with software implementations and simulation analysis. For this purpose, we demonstrate the entire system design process for a representative UAV flight management system using systems engineering methodologies. Along with developing our system of interest, we demonstrate how model-based systems engineering tools can be used to capture high-level design coupled with low-level constraints.

1.1.1 Model-Based Systems Engineering

Systems engineering is an interdisciplinary approach used in various projects to enable the realization of a successful system and reduce the risk of encountering problems during system operation. A systems engineering approach to a project includes analyzing and deriving stakeholders' needs, documenting requirements and continuing with system design while considering the complete problem and validating the system to ensure it can satisfy stakeholders' needs in an efficient, cost-effective and high quality manner. A model-based system engineering methodology uses formalized applications of modeling to achieve all steps of Systems Engineering methodologies [6]. In this methodology, system requirements, structure, and behavior can be visualized in the conceptual phase of system development as well as later in the life cycle. This method can help systems engineers provide different representations of a system from the standpoint of corresponding concerns and issues of a system [7]. To clarify the importance of systems engineering applications in our problem, we demonstrate the complexity of the mission design and challenges that designers may encounter in mission planning and how they can leverage systems engineering approaches for planning a safe mission.

In mission and scenario planning, various users such as Air Traffic Control and ground controllers, may interact with the flight mission in different phases of the mission, based on the mission requirements. In addition, conceptual operations in different flight states and UAV behaviors and structure can play important roles in organizing the mission. In developing a mission plan for a UAV, one should address different design challenges suchs as flight route, sensor modeling, communication, navigation, threat analysis and 3D visualization. Integrating all these requirements, verifying the entire complex system, as well as reducing failure risk and improving mission safety requires a systematic approach to mission planning. This approach allows us to capture this complex system and detect possible faults and malfunctions in each phase of the system. All of this leads us to conclude model-based systems engineering is a good solution for modeling all required states of our UAV flight.

In this thesis, we utilize a model-based systems engineering approach to capture the requirements, provide a high-level solution to our mission planning problem, and map the generated models into the mathematical models. As a result, as a first step, we formulate our problem and demonstrate the mission requirements and operational concepts. Then, we discuss system architecture as well as functional and behavioral analysis to better understand the system and required functionality our proposed system should have.

1.2 Concept Description

In order for UAVs to accomplish their assigned mission, they need to have a defined representative mission plan and flight plan scenarios. For this reason, identifying and implementing an appropriate path planning system to develop unique flight plans across a wide degree of scenarios is useful. A good flight route has some important attributes: It should provide the optimal path from start node to the final node with respect to the all constraints in the environment. Finally, path planning system is expected to be coded in software and implemented for use onboard UAVs [3].

Furthermore, safe operation of UAVs operated by commercial and military entities in the National Air Space (NAS) is envisioned to require autonomous situational awareness and safe response to situations and anomalies [4] that may constitute hazards to human life and property. The hazardous situations and anomalies may result from loss-of-command-link, violation of flight rules, departure from flight plan, UAV component failures, failure to respond to ATC directives, and the need to sense and avoid nearby air traffic [26]. Software algorithms onboard the UAV must detect and identify these anomalies. In addition, other onboard software algorithms must decide the “safe response” to each identified situation or anomaly, wherein determining such responses requires knowledge of map position, obstacle and terrain features. The onboard “safe response” software must incorporate decision support to either terminate the UAV flight or alter the UAV’s onboard flight plan in accordance with the selected response.

1.2.1 Project Objective

The purpose of the proposed research is to create a Preliminary Design of an Autonomous Intelligent Flight Management System for UAVs that incorporates autonomous situational awareness and safe response to situations and anomalies such as mid-air collision that may constitute hazards to human life and property. For this reason, we generate UAV’s trajectories based on mission waypoints, then we develop a situational awareness software module using collision detection and avoidance algorithms. Afterward, we identify some of the successful and unsuccessful scenarios based on the developed situational awareness module and how environmental and physical constraints of UAV can affect the efficiency of path planning and collision avoidance algorithms. The tasks we covered here are as follows:

1. Execute a prelaunch-uploaded mission plan
2. Identify anomalies (including UAV flight-rule violations, midair collisions, component failures and loss-of-command-link events)
3. Response to such anomalies by generating revisions to the baseline mission plan in a manner that minimizes hazards to human life and property

To accomplish these tasks, we went through the following procedure outlined in Figure 1.1. First, we define a real-world problem, then we define the scope of the problem we want to cover by developing concepts of operations and system requirements. Then, we create a semi-formal model of that system using some UML and SysML diagrams in order to simplify the real world system. Finally, we create our simulation system model to capture some parts of our system model and provide formal analysis based on the simulation results.

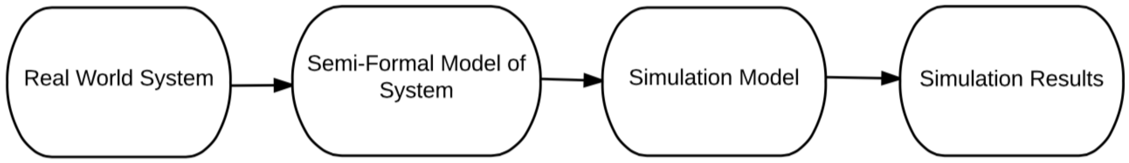


Figure 1.1: Research Procedures: Steps Required for developing a simulation system from an abstract real-world problem. The simulation model is developed based on the model of the real-world system.

1.3 Background

In this section, we review some of the works done in mission planning architecture, path planning and situational awareness systems. First, we have an overview of the UAV operational categories and the new technologies developed to leverage having a safe mission for each of these categories during flight. Then, we discuss about some different algorithms and approaches that address the path planning and sense and avoid system and their related cons and pros.

There are some previous works on system modeling and architecture of mission planner for UAVs. Cristian Atencia [11] provided a reference architecture for mission planner system for multiple UAVs operating in a group to achieve their goal. In this study, UAVs' specifications, zones information and sensors are considered as inputs and the sequence of desired tasks to be done by UAVs are the desired outputs. In another research, Stenger [12] integrated the decision making process using the cognitive agent-based architecture Soar. His system architecture addresses the process in which UAV can autonomously make decision and interact with environment. They used this architecture for capturing UAV behaviors. In another research, George Vachtsevanos [13] provided a platform for high-level architecture of mission planning, trajectory generation and vehicle navigation routins. In this research, the mid-level represents the envelope protection and mode transitioning. He demonstrate the flows of actions and configurations of mission planning system and route planner in high level and mid level architecture respectively.

Defining different flight plans and alternative routes for UAV is vital for the purpose of mission planning and flight management so that the UAV can do the best reaction in a given situation when it is needed. Therefore, path planning and waypoint generation should be also considered as a part of flight management system design to be integrated with SAA. For the purpose of flight plan and alternative routes, there are some algorithms that are proposed. Some path planning algorithms such as RRT [15] which is based on space-state and produces a time-parametrized set of control inputs to move from initial state to the end state, A* [16] which is based on depth first search concept and Dubins Airplane [35] which we discuss it in this research.

UAVs can operate under both cooperative and non-cooperative categories. Those UAVs in the cooperative category are equipped with some technologies such as Traffic Alert and Collision Avoidance System (TCAS) and Automatic Dependent Surveillance- Broadcast (ADS-B). These technologies help UAV to have a safe mission by sharing its position information and getting the same information from other equipped aircraft in the cooperative scenarios. ADS-B for instance, uses GPS or other navigation sources to broadcast its own aircraft position, velocity and other data without being interrogated. With these technologies, UAVs can operate under the control of ATC and detect other equipped vehicles that are able to cooperate with ATC. However, they cannot be utilized in non-cooperative scenarios that the other aircraft are not equipped with the same technologies. This fact provides some troubles for having a safe mission in many other possible scenarios. As a result, when flying in the low altitude where other VFR [14] aircraft are flying, UAVs need to integrate with non-segregated airspace. Thus having a Sense and Avoid (SAA) system seems to be critical for them so they can detect traffic and obstacles and determine the right maneuvers for avoiding them.

For SAA purposes, some algorithms and approaches are proposed which all of them have some advantages and disadvantages. Here we mention and provide an overview of some of them. SAA has five parts; Sensing, Trajectory Prediction, Conflict Detection, Conflict Resolution and Evasion Maneuvers. For the conflict detection and avoidance, Schild [17] provides a set of rules for autonomous separation for UAVs. These rules are base on some optimization tasks and it involves two aircraft sharing the same rules. This approach gives an optimized solution for the scenario in which all aircraft share the same rules and it cannot integrate a specific aircraft into the conflict. The other approach that is used is Game Theory Methods. This approach considers worst case scenario and UAV should avoid all possible maneuvers and disturbances produced by the intruders [18]. Geometric Methods are another approach for SAA problem. Several geometric models are proposed. We go through some of the works done by different authors. Ota et al [19] provided a method for collision avoidance in both the horizontal and vertical planes based on relative geometry between intruder and UAV. In this effort the moving obstacles are described as static obstacles using the concept of "threat map". Bilimoria [20] proposed an optimized geometry solution by minimizing the required velocity vector for avoiding threat. This algorithm works with changing heading angle and velocity vector. However, this gives an optimal solution just for one threat and cannot work for multiple threats. So in the case of multiple collisions, this method works sequentially. There are also some studies on using traffic collision alerting system (TCAS) like collision detection and avoidance systems for UAVs [21]. This approach relies on the vertical commands and needs so many experiences, test flights and very large amounts of data therefore, there is no analytical verification [22] and consideration of physical and operational constraints of UAVs such as turn rate and climb/descent rate. However, statistically speaking, this method can guarantee the high reliability of TCAS.

1.4 Outline of Thesis

This thesis is focusing on the systematic procedure for developing a flight management system. The following chapters are developed based on this procedure.

In Chapter 1, we focused on problem definition. In Chapter 2, we started with, concept of operations and high level mission requirements and then developed semi-formal model of our UAV's mission planner system using SysML and UML diagrams and mapped behaviors of the system to its structure such that they can be traced back to the main requirements of the system. Chapter 3 focuses on developing simulation model based on the semi-formal system architecture and deriving appropriate results that can be used for proper analysis and conclusions. Chapter 4 discusses about the conclusions and the future steps and process should be done. Appendix A is concentrated on path planning algorithm concept we used and the related constraints, assumptions and formulas. Appendix B is dedicated to collision detection and avoidance algorithms used in this research and elaborates the equations, concepts and constraints in more details.

Chapter 2: Semi-formal Flight Management System Design

2.1 Overview

In order to capture functional aspects of the system such as the tasks it should accomplish, we demonstrate and emphasize the system architecture, including system behavior and structure in more details. In this section we discuss the necessary specifications the system should have in order to meet the users' requirements. To begin, we explain the desired inputs and outputs. Then we will walk through the system's functional (operational) requirements to achieve those outputs.

2.2 Mission Overview

In order to determine the operational and functional requirements, it is essential to define representative mission and flight plan scenarios in order to identify, clarify and analyze users' requirements. These are focused on a variety of tasking scenarios characterized by FAA class airspaces A-G [9]. For this thesis, we chose a loitering scenario and captured both high-level and low-level mission requirements related to this phase of flight. Table A.1 shows the main goals of the mission, which are as follows:

1. Autonomous Flight: Achieve controlled take off, flight, loitering and landing.

2. Cover Entire Search Area: Determine target location within defined distance (50ft), fly the search area.

3. Obstacle Detection and Avoidance: Carry out Air Traffic Control (ATC) requirement to remain well clear of other traffics.

Table 2.1: Mission Overview

Mission Overview	High-Level Requirements	Low-Level Requirements
1	UAV shall approach the pre-planned maneuver point	UAV shall capture required information from loitering location
2	UAV shall fly at 1000ft altitude	UAV shall pass specific waypoints during loitering
3	UAV shall loiter for 1 hour	UAV shall Determine target location within defined distance
4	UAV shall resume the flight path along the border	UAV shall detect all intruders within 100ft in loitering phase
5	UAV shall climb back after 1 hour loiter	UAV shall avoid up to 3 intruders at the same time

While this surveillance application is highly in demand due to its potential to be used in civilian applications such as disaster response, firefighting, search and rescue [23], the approach used in this thesis can also extend to the rest of the flight phase such as landing, take-off and cruise as well as to other scenarios.

2.2.1 Operational Concepts

After creating the main goals and mission scenarios, the next step is to provide use case diagrams for our system of interest, the UAV's mission planner, to capture the system and sub-system's behavior. Use-case diagrams are developed using the main goals of a system and show what the users want the system to do. Systems engineers can derive system requirements from use cases and their flows of actions [24]. Thus, we developed two use-case diagrams for the purpose of this example. One is a high-level system's use case diagram in which we show the overall tasks of the mission planner in the sequence of actions that the user might interact with. The second is an obstacle avoidance use-case which is a lower level use-case diagram for our mission. We go through the details of each diagram in the following paragraphs.

Figure 2.1 depicts all the states of the mission planner. These are (1) path planning, (2) trajectory following, (3) sensing, and (4) decision making for emergency situations. In Figure 2.1, the users of the mission are demonstrated. One type of user is the UAV communication systems, which allows the system to send and receive data from sensors and ground controls. Another type is the ground controller, who monitors, manages and tracks the mission and is ready to act in emergency situations. Ground controllers also have permission to cancel or change the mission on board if it is necessary.

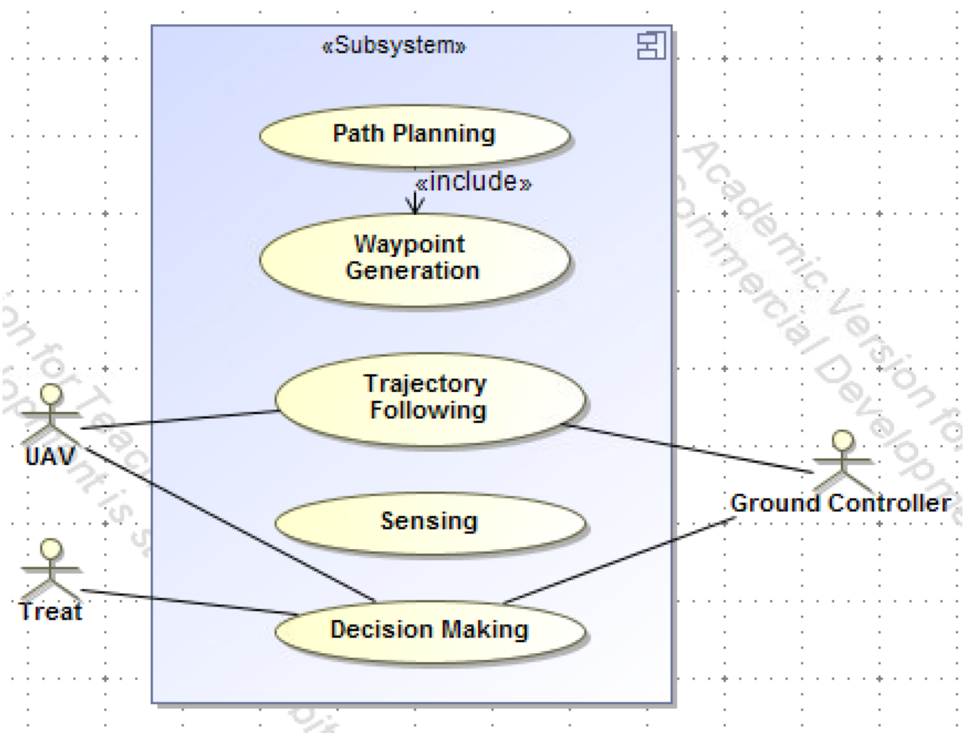


Figure 2.1: Use-case diagram for the entire mission: High-Level Use Case Diagram for Overall Loitering Scenario and different tasks that the UAV must do

To accomplish the mission, the UAV must be safe from any plausible obstacle in its path. Therefore, the UAV should have a reliable collision avoidance system for all mission states, including the loitering scenario we focus on. This leads us to create a sub use case diagram that effectively shows the actions of the collision avoidance system and how it interacts with actors. Figure 2.2 is our sub-level use case diagram for the case in which the UAV should avoid obstacles.

As shown in Figure 2.2, while the UAV is loitering and following its trajectory, it should also sense and detect threats and use an appropriate maneuver to avoid collisions. However, based on the type of threats, the detection and avoidance system need to meet different sets of requirements.

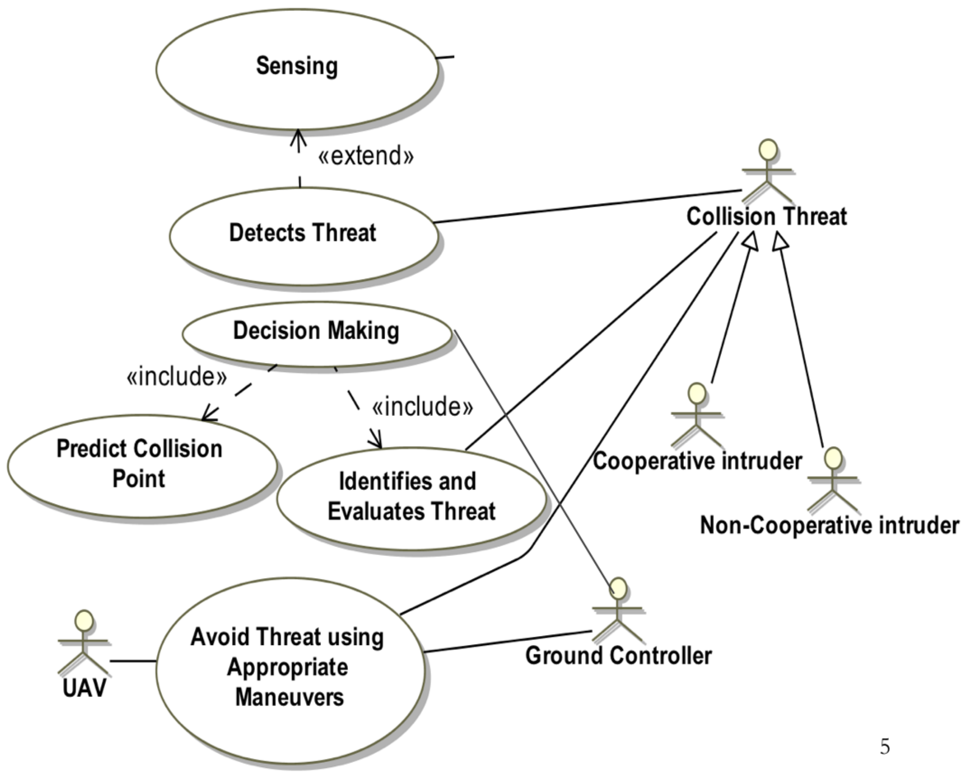


Figure 2.2: Low-level Use-Case: Use Case Diagram for Collision Detection and Avoidance Scenario

It should be noted that there are two types of intruders, known as cooperative intruders and non-cooperative intruders. The difference between these two types is that in the cooperative scenario, the UAV can cooperate with the intruder (another aircraft or UAV) and they can work together to avoid collision. On the other hand, non-cooperative intruders encompass birds, balloons, and other intruders that cannot communicate with our UAV. As a result, the UAV must avoid them entirely by itself. This research focuses on the non-cooperative scenario to simulate the results and develop the algorithm.

In our use case diagrams, there are two important components for mission planning: path planning and trajectory following, and the capability to avoid potential collisions. In order to combine these two components and demonstrate how they can be related to each other, we present the following functional diagram in Figure 2.3.

Figure 2.3 is a context diagram that shows how the system interacts with the environment, their interfaces, and the flow of information. A context diagram is a diagram that captures how the system interacts with outside systems and its environment [27]. We considered documented requirements and a UAV model as inputs for the mission planner system [N.B. In this thesis, we do not consider the problem of modeling the UAV dynamics, instead we consider it as an input and a constraint for UAV mission planning].

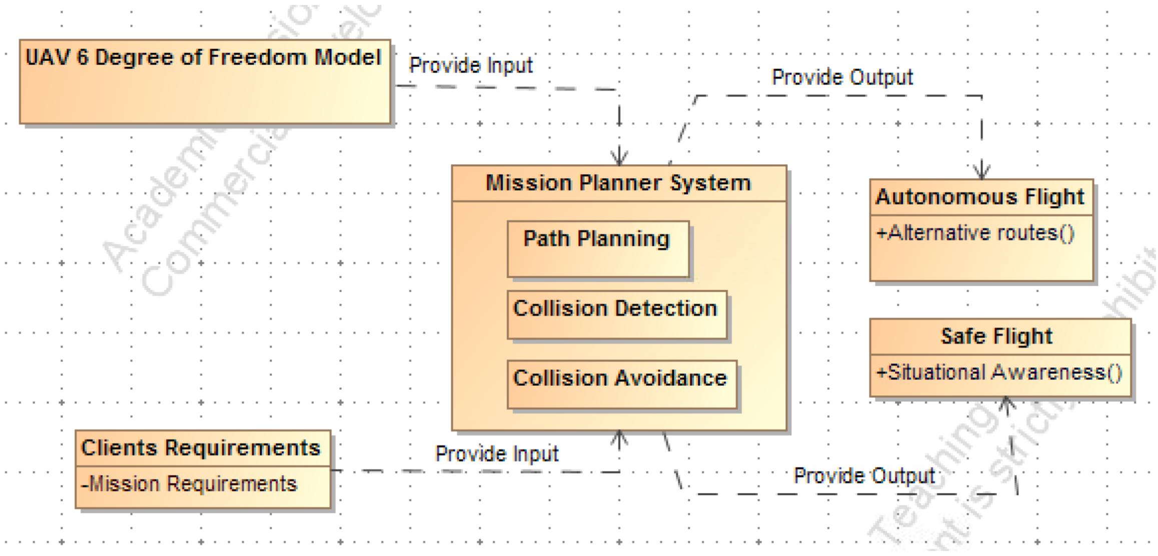


Figure 2.3: Context Diagram: System Context Diagram shows interactions between system and environment and captures desired inputs and outputs of the system.

In Figure 2.3, the required inputs and outputs for the system of interest and how they connect with each other is derived. In this figure, the interoperability of the path planning, sensing and collision detection and avoidance subsystems can be seen. All the information about the UAV model is used as input for the path planning and waypoint generation phase. For collision avoidance, the UAV needs to acquire information about unpredictable obstacles and turbulences from sensors, and use that to predict if there is a risk of an upcoming collision. Then, based on the prediction, it needs to avoid the threat and re-plan the trajectory within a specified time interval. Sometimes sensor data sent to the control station necessitates changes in the scenario planning. In this case, reliability and safety of the sensor are the key parameters for a successful mission.

2.3 System Requirements

In systems engineering methodologies, use case analysis is used for documenting of the functional requirements by providing a set of scenarios which captures how the system can interact with users and other systems. This structure helps us to identify and list main requirements based on proposed scenarios and determine MOE (measure of effectiveness) of the system. In this thesis, we used defined operational concepts and use cases to establish a set of system's measure of effectiveness. In the following section the list of MOEs is provided.

2.3.1 System's Measure of Effectiveness

Setting MOEs for the system, one can answer the question of "Will this system meet the stakeholder's need?" [28]. As a result, MOEs can be defined as "standards against which the capability of a solution to meet the needs of a problem may be judged. The standards are specific properties that any potential solution must exhibit to some extent. MOEs are independent of any solution and do not specify performance or criteria" [28]. Having defined the MOEs and the purpose of using them, we tried to identify these measures for our Mission Planner System. So the Mission Planner System MOEs are as followed:

- The system is able to generate a path that passes all the required waypoints in the loitering scenario of the mission.
- The system is able to provide at least two alternative collision free trajectories.
- The system is able to detect at least dynamic non-maneuvering threats whose trajectories and speeds are predictable by the sensor system.
- The system is able to avoid at least dynamic non-maneuvering threats whose trajectories and speeds are predictable by the sensor system.
- The system is able to avoid at least three simultaneous collisions that are predicted.

Therefore, the final system can be evaluated if it is successful to accomplish of its mission objectives and achieve desired functions or not. These measures are useful to quantify the successfulness of the final system by giving a reference for analyzing if it can accurately correspond to the mission requirements.

2.4 System Behavior

In the previous sections, we considered the users' requirements and how they interface with the system. Next, we develop the system's behaviors and functions. For this purpose, activity diagrams are helpful to visualize the steps, actions, and the parts of the systems that carry out the actions. In the collision detection and avoidance segments, the sequence and flow of actions are important for managing requirements and consistency between design and requirements. We provide the reader with some sections of the activity and sequence diagrams related to the non-cooperative collision detection and avoidance.

2.4.1 Activity Diagram

Activity diagrams are used in the system behavior modeling in order to capture actions states, decisions and merges, object flow and concurrent transitions [29]. In order to identify performance of actions and triggers, we developed activity diagrams for the detection and avoidance sub-system. Figure 3.10 describes the collision detection system, which highly relies on the history of the tracked obstacle to estimate the future trajectory. It shows that the detection section shall predict the point of collision and the time to reach that point in order to provide sufficient information for the collision avoidance section to avoid threats. This prediction is based on time-based information about the history of the obstacle's trajectory and velocity. Sensor systems will ensure the whole system about the possibility of getting this kind of information. If there is sufficient time-based information, the system will estimate the future trajectory of the obstacle. Then it can predict the closest approaching point for UAV and obstacle. In Figure 2.5, the collision avoidance activity, which occurs

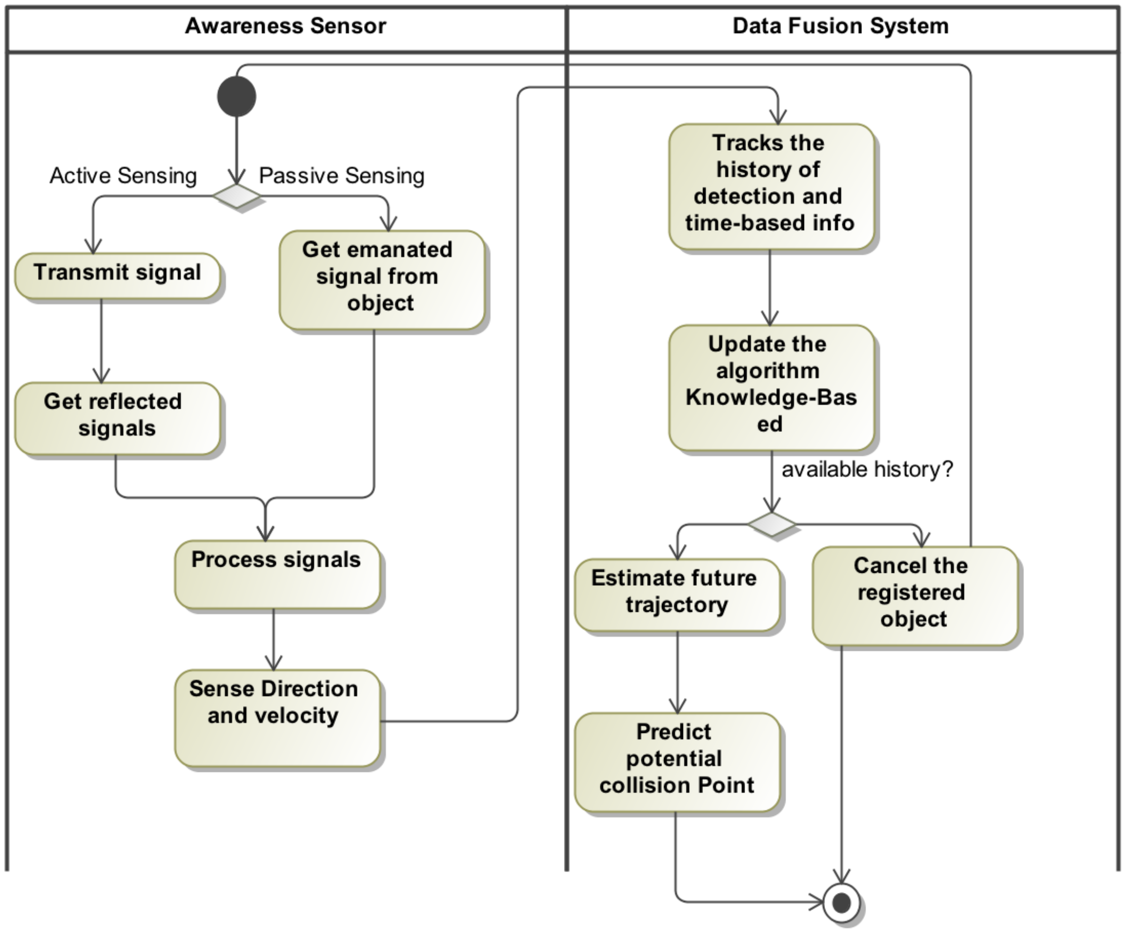

Awareness Sensor
Data Fusion System
Active Sensing
Passive Sensing
Transmit signal
Get emanated signal from object
Get reflected signals
Process signals
Sense Direction and velocity
Tracks the history of detection and time-based info
Update the algorithm Knowledge-Based
available history?
Estimate future trajectory
Cancel the registered object
Predict potential collision Point

Figure 2.4: Detection Activity: Activity Diagram for collision detection showing flows of activities from sensing to data fusion

after detection, is shown. In particular, it shows the steps of determining the best avoidance maneuver and how it applies the maneuver to avoid the obstacle, as well

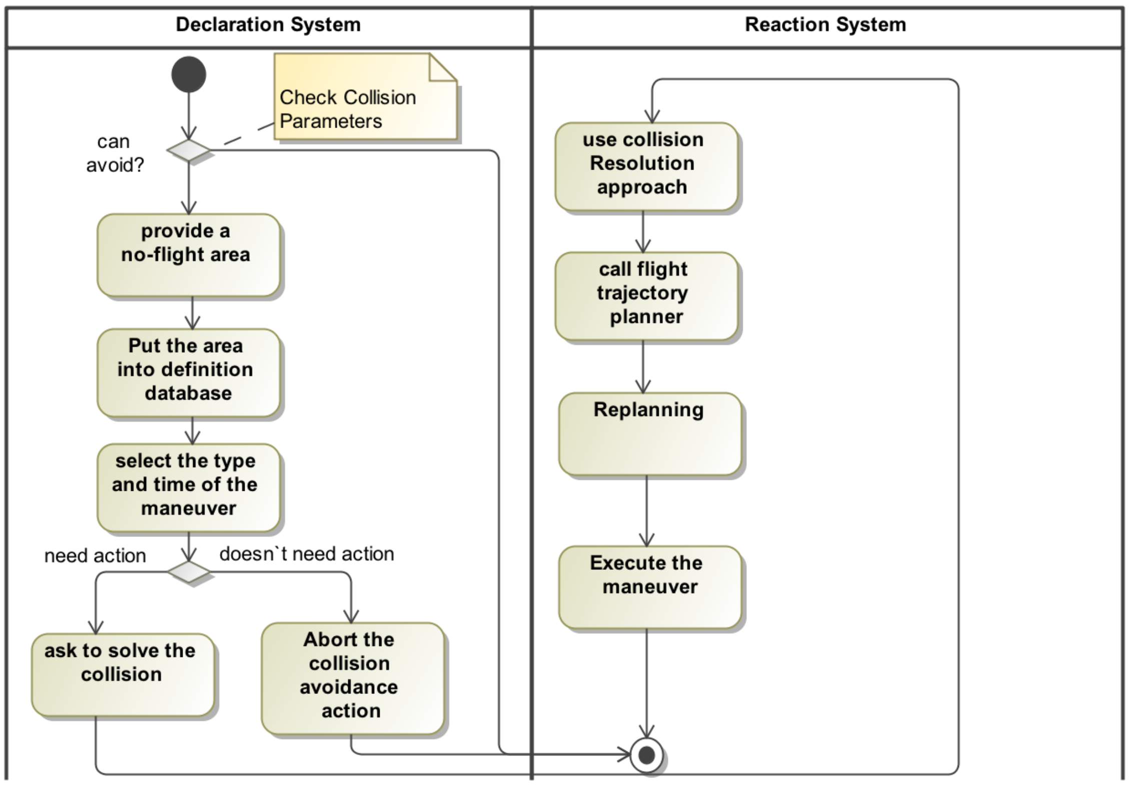

Declaration System
Reaction System
Check Collision Parameters
can avoid?
provide a no-flight area
Put the area into definition database
select the type and time of the maneuver
need action
doesn`t need action
ask to solve the collision
Abort the collision avoidance action
use collision Resolution approach
call flight trajectory planner
Replanning
Execute the maneuver

Figure 2.5: Avoidance Activity: Activity Diagram for collision avoidance showing flows of activities from decision making to reaction maneuvering

as necessity of re-planning and waypoint generation for some small time intervals to avoid the obstacles. The declaration system uses detection information about distance and time to closest approaching point to evaluate if the UAV can avoid the obstacles or not. After this evaluation, the system can provide a no-flight area for the UAV by considering other obstacles and select the most appropriate maneuvers for avoiding. In the next chapter, we demonstrate how collision avoidance algorithm that we use can determine these maneuvers.

2.4.2 Sequence Diagram

A sequence diagram has two dimensions, the vertical lines indicate time for actions and horizontal arrows represent different instances and the flow of information and data [29]. They are used to describe the sequence of interactions between objects in the system. In Figure 2.6, the sequence of actions for non-cooperative collision avoidance is shown. This figure helps us to determine the procedure of the collision avoidance and the priority of different steps. Sensors receive information about the location of obstacles and send that information to the data fusion block for processing. Then, if the information if sufficient, the system tracks the data and estimates the future trajectory. Afterwards, the predicted trajectory is used for determining collision time and collision point. The next step is to provide a no-flight area for UAV and modify its trajectory for some amounts of time. Then, the new waypoints are generated by autopilot and UAV can execute appropriate avoidance maneuver.

2.5 System Structure

Structure in this context is described using components, all the attributes, parts and functions of the components as well as the connections among the components [30]. The parts of the component are described by their properties. The functions of the component contain a reference to the behavioral model of the system. In order to map our system behavior into the system structure, we developed a block definition diagram for the system. It is represented in the following sub-

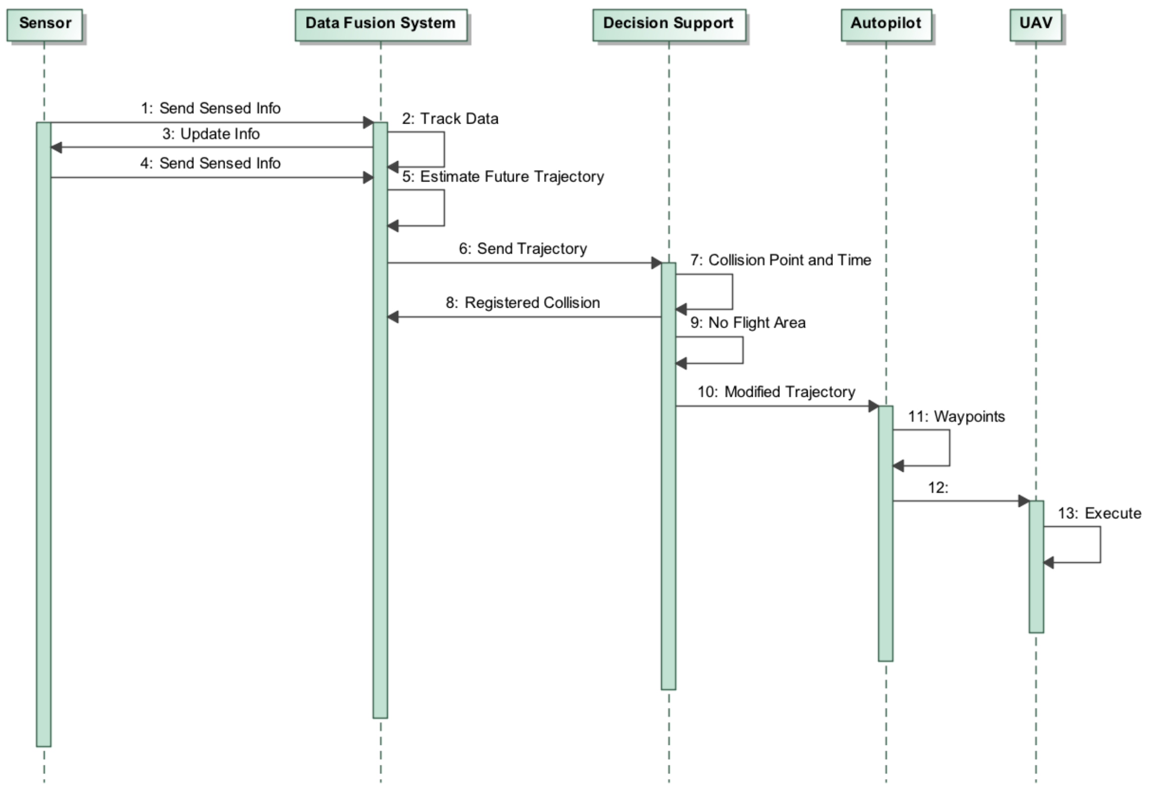


Figure 2.6: System Sequence Diagram: Captured sequence of actions between subsystems and the flow of data among them

section.

2.5.1 Block Definition Diagram

Having explained functionality and behavior, we now demonstrate the system's structure by using a block definition diagram. A block definition diagram is useful for showing the system's module and can be used in software development and simulation of the system. Block definition diagrams can accept values, parts, operations and attributes, which allow it to be easily converted to code. In this thesis we focus on the procedure and provide the reader with an example of simulation results that are extracted from the block definition diagram.To simplify this, our simulation only looked at the set of requirements for a non-cooperative intruder, which is explained in the following paragraph. The simulation will be discussed in more detail in section 3. This is shown in Figure 2.7, which depicts the functionality of the entire system.

As discussed earlier, one of the first steps in mission planning is generating trajectories. Other parts of the system are sensing, collision detection and collision avoidance sub-systems needed for mission safety. All parts have their own operations and values. For example, the path planning part includes trajectory following operation with the constraints on transferring latitude, longitude waypoints into Cartesian model and vice versa as well as using path planning algorithm (Dubins Airplane Model in this example). In collision detection, the system shall predict the obstacle's trajectory in the near future and detect if there will be a collision based on the speed and trajectory of the UAV, these operations are dependent on having some values such as obstacle and UAV's speed and positions, closest approaching distance and time to closest approaching distance which means that collision detection module needs to calculate these values in order to have a correct detection. The collision avoidance part shall make the decision on how to avoid the potential collision by changing speed, turning radius and altitude. Figure 2.7 shows each of these parts in mission planning. In each block, constraints demonstrate the method and formula for developing codes for each block. This block diagram helps us design and develop the simulation system based on the operations, constraints and values defined in the block definition diagram. It also divides the system into different modules and sub-systems that simplify implementation and integration phases of simulation system design.

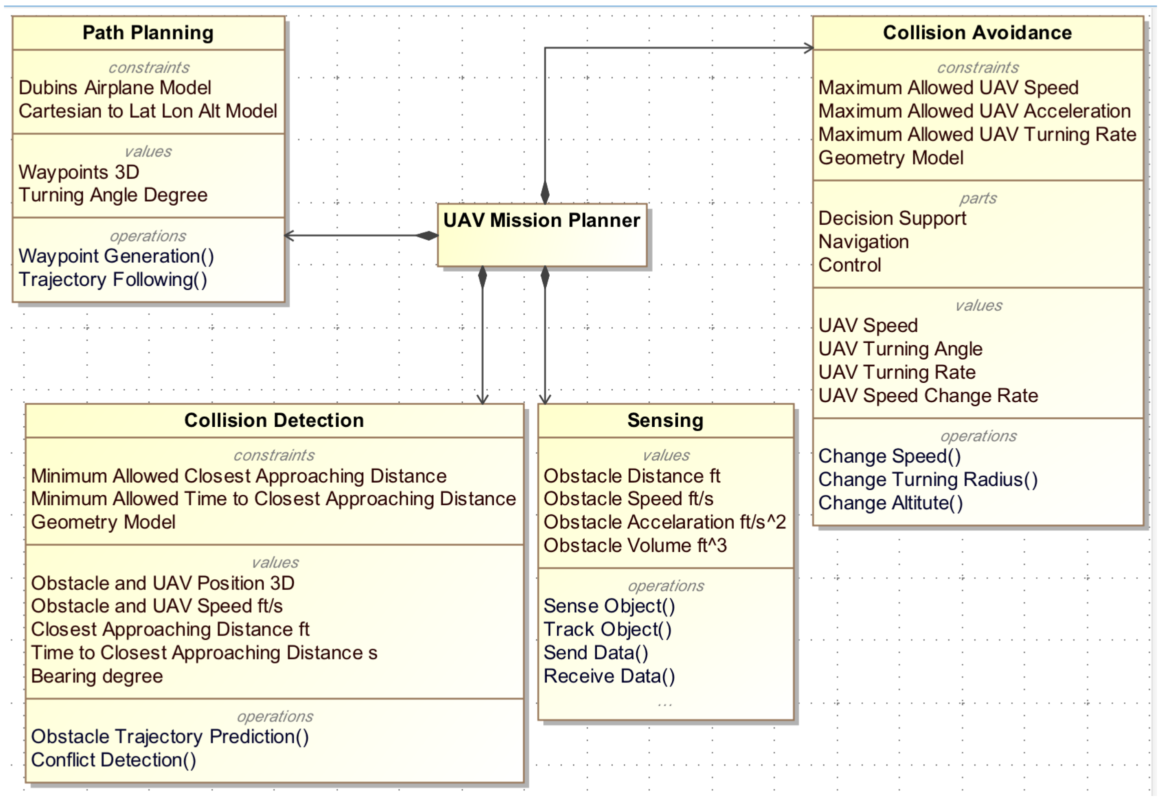


Figure 2.7: Class Diagram: System Block Definition Diagram that shows four modules of the system. These modules are: (1) path planning, (2) collision detection, (3) sensing, and (4) collision avoidance.

Chapter 3: Software Modules (Formal System Design)

3.1 Overview

To integrate a complex system consisting of different sub-systems, simulations are used to analyze and predict the system's behavior in some situation. Developing simulations can help us to examine the algorithms that we finally want to implement in our real system and identify any problems or specific results and patterns that might be seen in the actual system. Having these analysis, we are able to design systems with more information so they are better consistent with the system models and requirements. The simulation system is derived from the system's model and tries to capture all aspects of the system's model. However, in the real world, the entire system cannot be simulated as there are always many constraints for simulating all parts of the system. Therefore, it is important to clarify what the goals of the simulation system are and what parts of the system are going to be simulated. As it is shown in Figure 1.1, simulation software is developed from the mission planner model to simulate different segments of the mission planner model. In the previous sections, we provided a semi-formal model for a mission planner system and in this section, we provide simulation systems for path planning and collision avoidance parts of our developed model.

In order to visualize how the simulation system traces back to the semi-formal model of the system, we provided a context diagram for the simulation system itself and defined system model as an input for the simulation system. We also defined user-interface block to represent the user requirements which have interfaces with the simulation systems through the simulation's user-interface. Additionally, the simulation software can simulate the functionality of the system. Then the hardware codes can be developed in order to provide 3D visualization of the mission and integrate them into UAV system. The inputs and outputs and their interactions are shown in Figure 3.1.

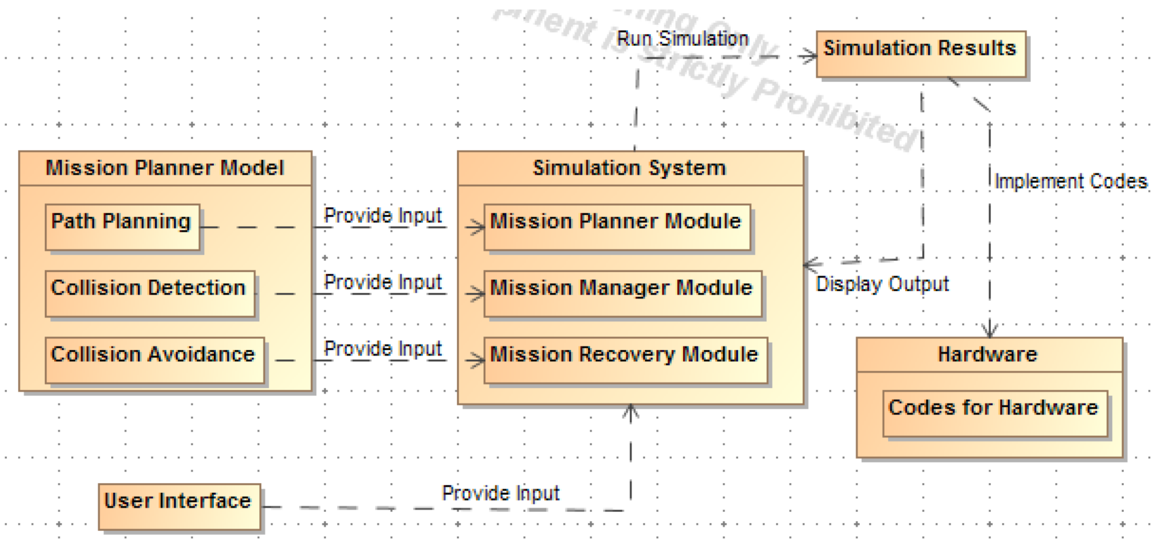


Figure 3.1: Simulation Context Diagram: Simulation Context Diagram shows interactions between system and environment and captures desired inputs and outputs of the system

In order to show the flow of the data and the structure of the simulation, we develop the simulation internal block diagram which shows the interaction between parts of the simulation software and how data is transferred between them [31]. Figure 3.2 shows the internal structure of our simulation system. The simulation parts are as follows: (1) Mission Planner User Interface, (2) Mission Planner, (3) Mission Manager, (4) Mission Recovery, (5) Mission Planner Display Engine. The mission Planner User Interface and Mission Planner Display Engine are related to the users' inputs, outputs, and the simulation results. The Mission Planner determines the UAV's trajectories, waypoints and search area. The Mission Manager is responsible for determining possible collisions and calculating collision parameters such as point of closest approach and time to point of closest approach. The Mission Recovery calculates required speed change, turn rate and altitude change for avoiding obstacles. In Figure 3.2, the flow of information between these parts is also depicted.

In this chapter, we demonstrate a simulation that represents how our UAV mission planning system meets the customer's requirements and achieves the required functionality. Our requirement analysis and the artifacts that we created allow us to identify relevant data for the simulation and organize our numerical simulation. For this purpose, first we show the trajectory generation for the overall mission, then we go through the obstacle avoidance and generalize it by simulating collision avoidance for 3 obstacles at the same time (which was one of the mission requirements).

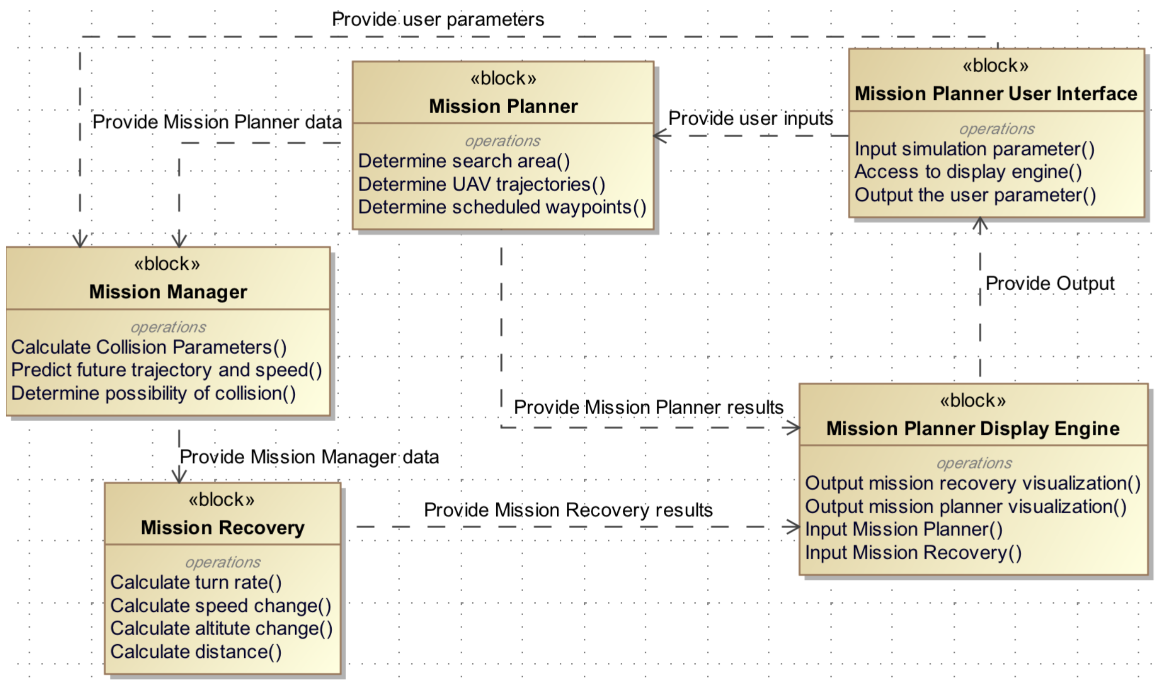

Provide user parameters
«block»
Mission Planner
operations
Determine search area()
Determine UAV trajectories()
Determine scheduled waypoints()
Provide Mission Planner data
Provide user inputs
«block»
Mission Planner User Interface
operations
Input simulation parameter()
Access to display engine()
Output the user parameter()
«block»
Mission Manager
operations
Calculate Collision Parameters()
Predict future trajectory and speed()
Determine possibility of collision()
Provide Output
Provide Mission Planner results
«block»
Mission Planner Display Engine
operations
Output mission recovery visualization()
Output mission planner visualization()
Input Mission Planner()
Input Mission Recovery()
Provide Mission Manager data
«block»
Mission Recovery
operations
Calculate turn rate()
Calculate speed change()
Calculate altitute change()
Calculate distance()
Provide Mission Recovery results

Figure 3.2: Simulation Structure: Simulation Internal Structure and the Flow of Information between different parts of the simulation software is shown

3.1.1 Simulation System's Measure of Effectiveness

The simulation system that we developed is also treated as a system, so it also should be assigned some measure of effectiveness to evaluate the ultimate simulation system can satisfy the functional requirements. For this reason, we still need to have a list of MOEs in order to examine and analyze the simulation system. Below are Mission Planner Simulation System MOEs:

- The simulation system is able to generate collision free paths for loitering scenario with the given inputs of the UAV's physical and behavioral constraints and information about the environment and mission waypoints.

- The simulation system is able to calculate collision parameters for at least 3 simultaneous collisions by having the predicted linear trajectory, position and velocity of the threats and the information of the UAV's motion.
- The simulation system is able to perform collision analysis based on the collision parameters for at least three simultaneous collisions.
- The simulation system is able to propose solutions for avoiding detected collisions by controlling turn rate and velocity change for the scenarios in which threats' behavior are predictable and linear.

3.2 Path Planing Module

After modeling our simulation system, the first step in developing our simulation is to create the UAV trajectory based on the mission scenario. For the purpose of path planning, we chose Dubins Airplane model [32]. Some of the reasons why we picked this model include: 1. it considers a moving object at a constant forward speed. 2. It has some constraints on maximum bank angle, heading angle and UAV airspeed V. 3. It does not consider wind speed in the path planing equations and is treated as some disturbances during flight and the flight controller tries to reject them [33] . The assumptions regarding Dubins Airplanes algorithm can be match with the assumptions we provided for our UAV mission requirements. It also can include UAV physical constraints and capabilities. Although further information about Dubins algorithm can be found in more details in Appendix A, we have an over view of this algorithm later in this section.

3.2.1 Simulation Module Requirements

After selecting an appropriate algorithm for path planning, we need to derive functional and performance requirements to develop the algorithm and create simulation software for our intended purpose. As a result, we established a list of simulation software requirements in order to get the desired results. In the following lists, you can see the path planning simulation functional and user-interface requirements for a non-cooperative scenarios. These requirements are all derived from MOEs of the simulation system:

1. Generate waypoints between main waypoints using Dubins Path algorithms.
2. Develop candidate paths between main waypoints.
3. Choose the optimal Dubins path between each of two waypoints among all possible generated Dubins paths.
4. Develop 3D trajectory for UAV using Dubins algorithms for 3D environment.
5. Transfer waypoints XYZ coordinate to altitude-latitude-magnitude coordinate using the transforming equations.

Path planning simulation user-interface requirements for a non-cooperative scenario are as followed:

1. The system's path planning part shall allow user to change initial simulation parameters for UAV.
   (a) The system shall allow user to input the main waypoints.

(b) The system shall allow user to specify UAV maneuvering constrains.

i. The user shall specify the minimum allowed spiral radius for UAV.

ii. The user shall specify the maximum allowed flight angle for UAV.

2. The system shall be capable of outputting the path planning outputs.

(a) The system shall output the trajectories' waypoints.

(b) The system shall output the optimal Dubins path

3.2.2 Module Architecture

In Appendix A we provided all concepts, assumptions and equations related to Dubins car model and how it can be extended to Dubins airplane. However, in order to start, we developed a data structure model of the software module so we could identify sequence of steps and flow of data required in the module and architect the simulation software. This data structure diagram is developed based on the required inputs and operations for the Dubins airplane model. So the input data for calculating Dubins airplane include desired start and end waypoints, flying altitude, main mission waypoints and some UAV physical constraints, such as Maximum and minimum allowed UAV speed, maximum and minimum UAV turn rate [33] . Those inputs are related to the UAV mission, other inputs related to simulation specifications are time step (time to update calculations) and simulation total run-time. These inputs will be used by Dubins airplane algorithm which is implemented in the simulation software to generate paths by calculating the appropriate turn rate, banking angle and heading angle of UAV. Here you can see the data structure for the path planning module. This diagram demonstrates different types of data, input data, output data as well as internal data.

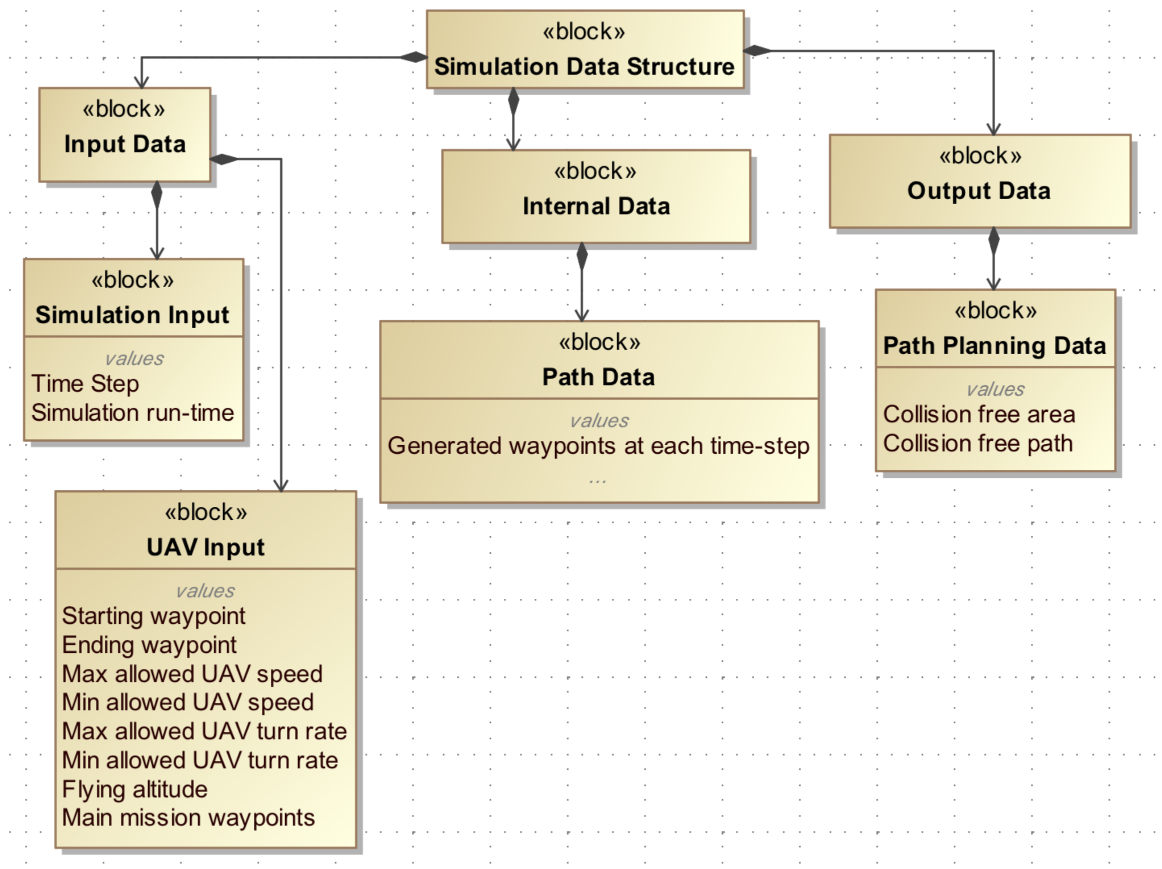


Figure 3.3: Path Planning Data Model: Data Structure diagram which demonstrates the data types for the input, output and internal data of path planning module

3.2.3 Software Implementation

As it is elaborated more in Appendix A, Dubins Cars algorithm is based on three motion primitives, left, right and straight. This algorithm proves that we just need some combination of these three motion primitives in order to generate a path between two points, so the possible combination of these motion primitives could be left-straight-right, right-straight-left, right-left-straight and so on [35]. Dubins Airplane extends this concept in a three dimensional environment with a difference that all curves and straight lines turn into some helical and straight lines [33]. So as to use Dubins Airplane model, we first created simple and primitive Dubins paths between each two nodes that are shown in Figure A.2. After that we can generate paths for multiple nodes just by using generated simple Dubins Airplanes paths. The following figures are some examples of generated paths based on mission waypoints. Figure 3.4 demonstrates a path generated by using RSR (Right, Straight, Right) motion primitive combinations.

In Figures 3.4 and 3.5, the UAV loiters in some specific locations and passes specific waypoints. In order to generate this path, we used Dubins Airplane method which allows us to generate an optimal solution to the path planning problem. Both paths include 4 main waypoints which are defined from UAV's flight mission. In all these paths we considered that our UAV is flying between two waypoints with a low altitude difference so we did not consider high altitude flight for our UAV. Having this assumption, we can assure that UAV does not need additional maneuver to reach the desired altitude (if the altitude difference between start and end nodes is big, Dubins Airplane algorithm, due to some constraints regarding bank angle and heading angle cannot generate path using just 3 motion primitives. In this case, UAV shall have some extra maneuvers to reduce altitude difference [33]. However, the generality of path planning problem will not be damaged by using this assumption.

Below, we provided some examples of Dubins Airplane generated by different motion primitives such that from one node to another node, there is a path generated

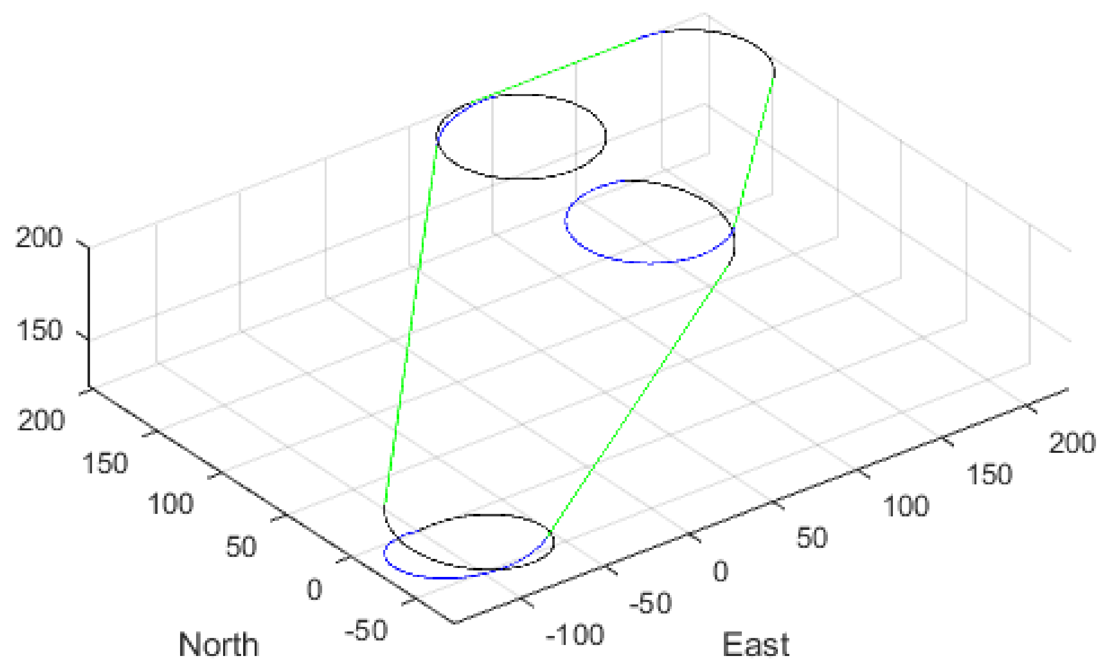


Figure 3.4: Loitering Scenario 1: Path Planning and Waypoint Generation for the overall loitering scenario in which UAV is loitering above an area and it turns around for sometimes so the start and end nodes are considered the same. Start node: [0, -200, -125], end node: [0, -200, -125]. The coordinates indicate North, East and Altitude respectively

based on one type of motion primitive combinations.

3.3 Situational Awareness Module

Situational awareness module includes sensing, collision detection and collision avoidance modules. In this thesis, we aimed to develop software modules for collision detection and avoidance for some of the possible scenarios that might happen for UAVs. However, we considered that sensor system can work properly and sense

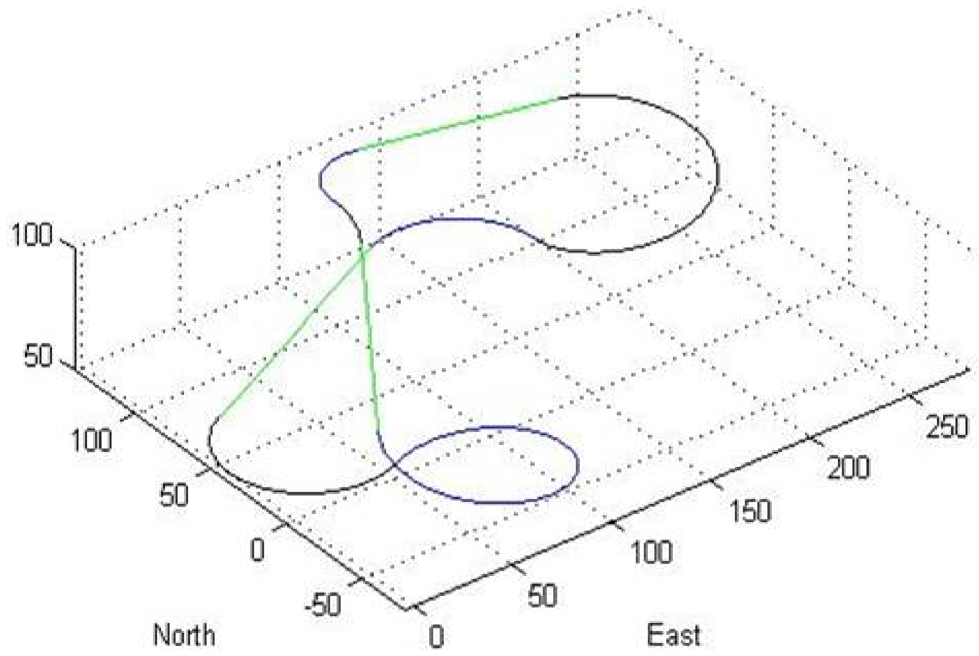


Figure 3.5: Loitering Scenario 2: Path Planning and Waypoint Generation for the overall loitering scenario in which UAV is loitering above an area and it turns around for sometimes so the start and end nodes are considered the same. Start node: [50, 100, 100], end node: [50, 100, 100]. The coordinates indicate North, East and Altitude respectively

the possible threads in the UAV's trajectories. Having said these assumptions, we created a table in which all possible scenarios for UAV and threads are shown. In this thesis, we tried to cover some of these scenarios and how our simulation results will change. As it is shown in the table 3.1, various scenarios can be defined by considering multiple conditions for initial conditions of UAV and updated equations of motion of obstacles during avoidance maneuver.

In this research, we considered all these scenarios and compared how effective

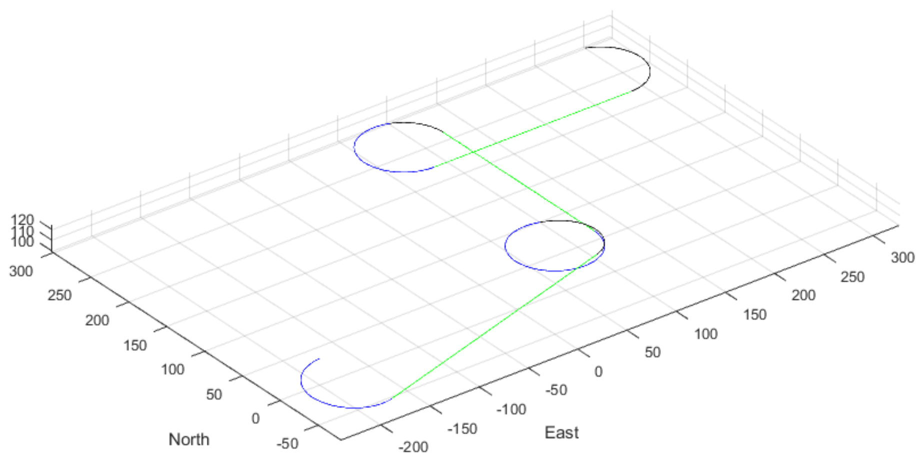


Figure 3.6: LSL Dubins Airplane: Path Planning and Waypoint Generation using LSL (Left, Straight, Left) motion primitives. This path is produced using 4 waypoints, these waypoints are as followed: Start node: [-200, 25, 0], end node: [300, 300, 50], middle nodes:[100, 100, 50], [100, 300, 50]. The coordinates indicate North, East and Altitude respectively

our collision avoidance algorithm is in different cases and situations. It should be mentioned that all the simulations for multiple collisions, considered up to three simultaneous collisions. Although, the platform of the simulation is general enough to be able to be extended to more than three collisions at the same time.

Avoiding all these collision situations depends on some physical and mechanical constraints of UAVs. For having a successful avoidance maneuvers, it is very important to examine extreme values of UAV specifications. These boundary values include but not limited to maximum and minimum UAV's achievable speed, maximum allowed UAV's turn rate, maximum heading angle, maximum and minimum UAV tangent acceleration. These parameters may limit the maneuverability and

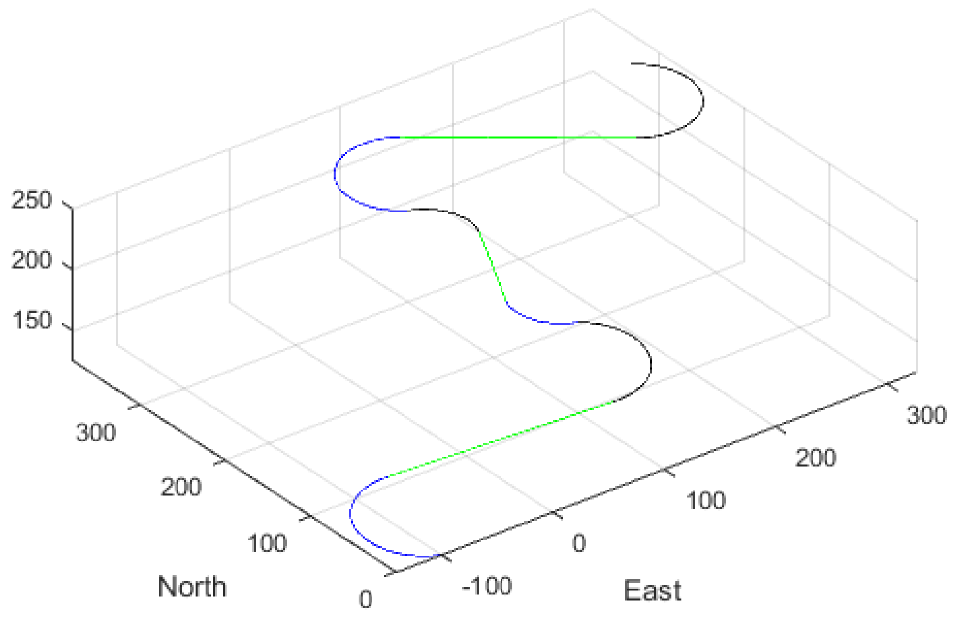


Figure 3.7: RSL Dubins Airplane: Path Planning and Waypoint Generation using RSL (Right, Straight, Left) motion primitives. This path is produced using 4 waypoints, these waypoints are as followed: Start node: [-100, 0, 100], end node: [300, 300, 250], middle nodes:[150, 200, 150], [150, 300, 200].The coordinates indicate North, East and Altitude respectively

agility of UAV. Since these factors are some design factors, they should either be designed based on the importance of UAV's desired collision avoidance capabilities or they provide restrictions for UAVs in order to avoid collisions. Although, the collision avoidance algorithms used in this paper, are practical in avoiding obstacles, as they are developed based on UAV's physical constraints, they cannot provide UAVs with efficient avoidance maneuvers in all possible scenarios. Therefore, it is very critical to identify the situations, in which UAV is not agile enough or maneuverable enough to do required maneuvers. The other important factor that should be

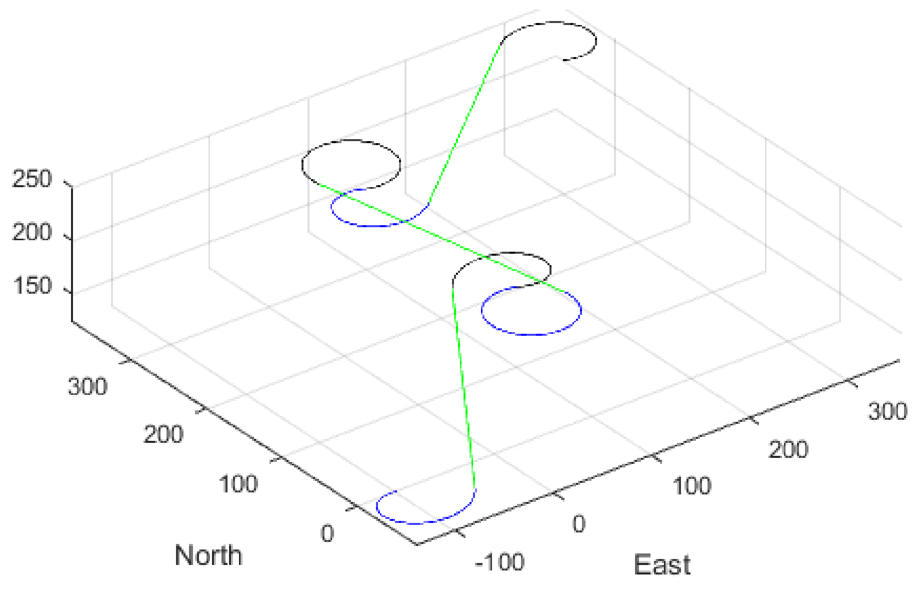


Figure 3.8: LSR Dubins Airplane: Path Planning and Waypoint Generation using LSR (Left, Straight, Right) motion primitives. This path is produced using 4 waypoints, these waypoints are as followed: Start node: [-100, 0, 100], end node: [300, 300, 250], middle nodes:[150, 200, 150], [150, 300, 200].The coordinates indicate North, East and Altitude respectively

evaluated when getting simulation results, is the time required for UAVs to process the commands and apply maneuvers. So the gap time between understanding commands and executing them, can cause some inaccuracies in calculation of required time to avoid collision. In order to mitigate those inaccuracies, the amounts of response lag in executing required command, should be determined. This response lag will vary for different UAVs. However, in this work we assumed ideal situation which means there is no lag between system's command and UAV execution.

Table 3.1: Possible Collision Scenarios

Number of Threads	V_u	V_a	a_u	a_a
One	Constant	Constant	0	0
One	Constant	Changing	0	Constant
One	Constant	Changing	0	Changing
One	Changing	Changing	Constant	Constant
One	Changing	Changing	Changing	Changing
Multiple	Constant	Constant	0	0
Multiple	Constant	Changing	0	Constant
Multiple	Constant	Changing	0	Changing
Multiple	Changing	Changing	Constant	Constant
Multiple	Changing	Changing	Changing	Changing

3.3.1 Collision Detection Module

After completing the path planning and waypoint generation, we need to calculate how the UAV can successfully detect and avoid the obstacles that may appear in its path. We used the Geometry model to calculate how the UAV detects the collision and how it decides to avoid the obstacle. This algorithm is purely relies on the relative geometry between UAV and obstacles by defining important relative velocities, relative angles and vectors along with or perpendicular to the UAV's velocity, obstacles velocity as well as relative velocity [39]. In Appendix B, all assumptions, constraints and rules for implementing this algorithm are discussed in more details. In order to realize how exactly this algorithm works and how its equations are derived, we refer the reader to Appendix B. In the following sections, we concentrate more on the procedures of designing and implementing collision detection software module starting with simulation requirements and ending with software implementation.

3.3.1.1 Collision Detection Simulation Requirements

As previously mentioned in the path planning module, the first step toward designing and implementing a simulation module, is identifying simulation functional and user-interface requirements. For this purpose, we provided a list of operational and user-interface requirements in this section. By recognizing these requirements, one knows how to implement the algorithm in order to meet these requirements. In the following list, the requirements for a non-cooperative scenario for the collision detection simulation module are listed:

1. The system shall determine the probability of collision.
    (a) The system shall determine closest distance between intruder and UAV.
    (b) The system shall determine time to closest distance between intruder and UAV.
    (c) The system shall compare the closest distance with safety circle (minimum allowed distance between intruder and UAV)

(d) The system shall Compare time to closest distance with look-ahead time

(e) The system shall determine if collision will occur

(f) The system shall do all the procedure for each of the detected targets.

Collision detection simulation user-interface requirements for a non-cooperative scenario are as follows:

1. The system shall be capable of outputting the detection outputs.

    (a) The system shall output the closest distance between all targets and UAV.

    (b) The system shall output the time to closest distance between all targets and UAV.

    (c) The system shall output the collision/no collision decision for each target.

2. The system's collision detection module shall allow user to change the initial simulation parameters for UAV safety criteria

    (a) The user shall specify the look ahead time.

    (b) The user shall specify the minimum allowed distance between UAV and other objects.

3.3.1.2 Detection Simulation Data Structure

After identifying requirements, we know what the desired outputs and functionality of the module are. The next step would be finding out what operations, data and procedures are needed to achieve all those requirements. For instance, in order to find the closest distance between intruder and UAV, simulation needs some data of both UAV and intruders velocity and positions. Thus, for capturing all these required data, one possibility is to develop a data-structure model of the simulation, since it is really important to identify all types of data that are used or generated in the software so the data flow of software can define all required steps and sequence of actions to use input data and generate output data. As a result, Figure 3.9, depicts all input, output and internal data for the detection module. Since the detection module should receive some data both from UAV and intruders, we separate these data into two types of data: Threat Inputs and UAV Inputs. As the same as path planing data structure, there are some simulation inputs such as Time-step and Simulation run-time. The output data is defined considering the functional requirements for collision detection module. This includes closest distance, time to closest distance and binary output of collision/no-collision alert. The internal data includes all data generated while simulation is running such as updated data for distance between UAV and threat at each iteration.

3.3.1.3 Software Implementation

The detection algorithm that we used, works based on the UAV's minimum and maximum allowed speed, acceleration and turn rate as well as look ahead time and minimum allowed distance between UAV and obstacles [39]. In Figure 3.10, the UAV calculates the closest point of approach and the time to closest point of approach for each of the approaching obstacles. In this simulation, the assumption

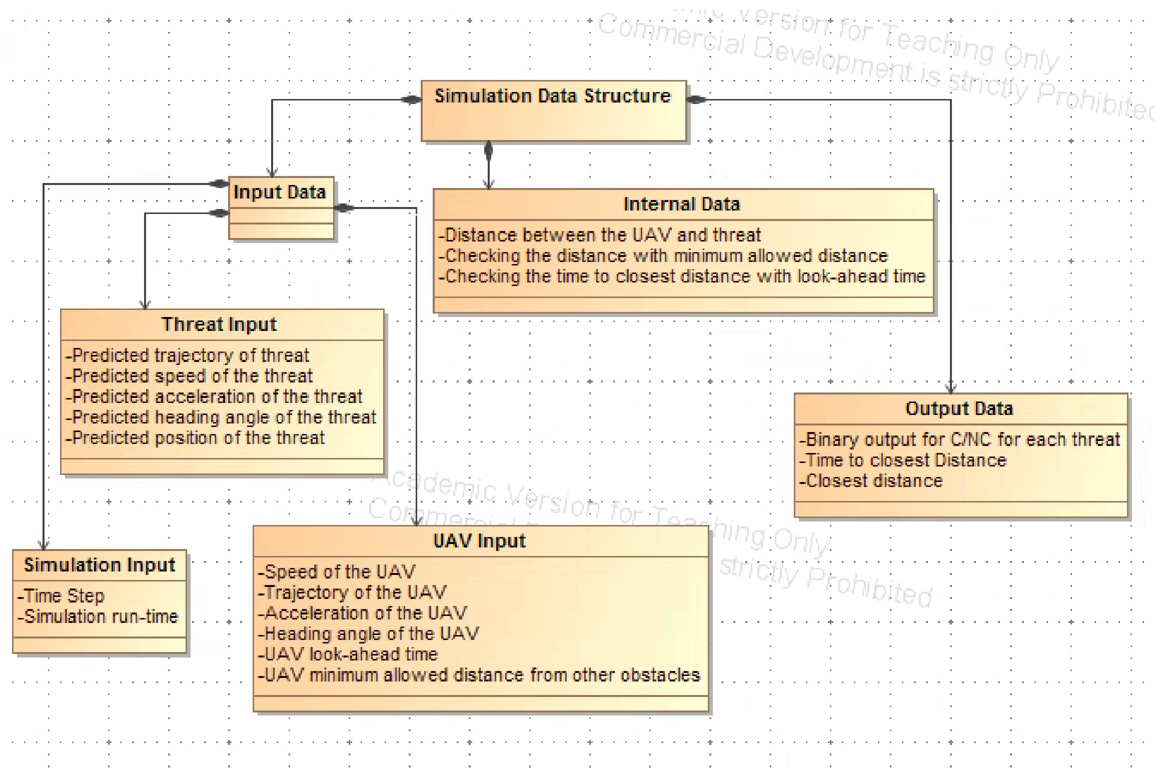

Simulation Data Structure
Input Data
Internal Data
-Distance between the UAV and threat
-Checking the distance with minimum allowed distance
-Checking the time to closest distance with look-ahead time
Threat Input
-Predicted trajectory of threat
-Predicted speed of the threat
-Predicted acceleration of the threat
-Predicted heading angle of the threat
-Predicted position of the threat
Output Data
-Binary output for C/NC for each threat
-Time to closest Distance
-Closest distance
Simulation Input
-Time Step
-Simulation run-time
UAV Input
-Speed of the UAV
-Trajectory of the UAV
-Acceleration of the UAV
-Heading angle of the UAV
-UAV look-ahead time
-UAV minimum allowed distance from other obstacles

Figure 3.9: Detection Data Model: Data Structure diagram which demonstrates the data types for the input, output and internal data of collision detection module

is that the obstacles' velocity and heading angles are constant during the detection period.

3.3.2 Collision Avoidance Module

When the collision detection module predicts one or more future collisions, it is time for collision avoidance to take some actions and determine the best maneuvers to avoid possible collisions. As we mentioned before, for the collision avoidance algorithm, we also used Differential Geometry Modeling. Based on this algorithm, UAV can control both its velocity and heading angle to stay well clear of other

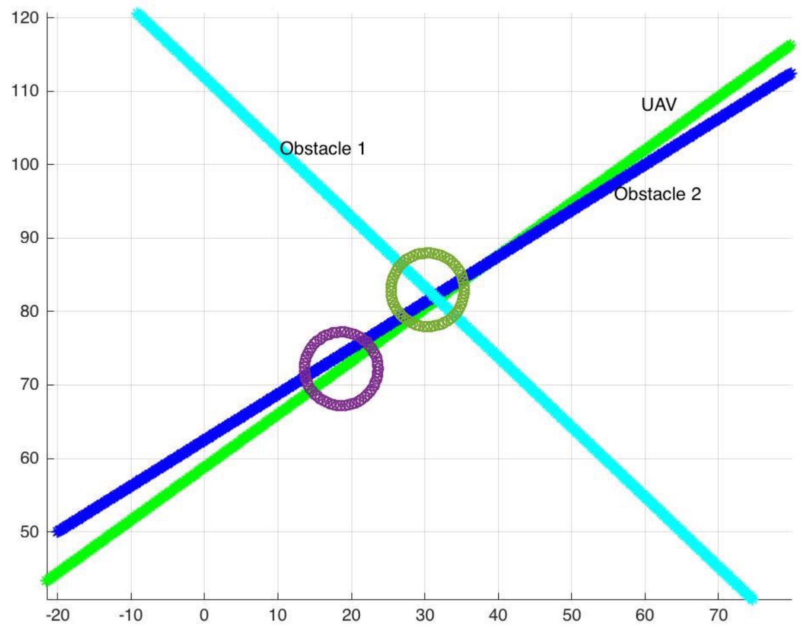


Figure 3.10: Detection simulation result: Collision Detection of two Obstacles at the same time. The circles indicate the first detected collision points where the distance between UAV and the obstacle is less than the minimum allowed distance for UAV and obstacle.

obstacles. In Appendix B, the equations for finding desired speed and heading angle and the approach for solving the collision problem are provided. Finding out the desired speed and heading angle at each iteration, the algorithm calculates required speed rate and heading angle rate to reach the desired amounts of speed and heading angle at each time step [39].

3.3.2.1 Collision Avoidance Simulation Requirements

The same as the previous module design, we start with collision avoidance simulation requirements to identify the required functions and capabilities of the module. As a result, the list of collision avoidance simulation functional requirements for a non-cooperative scenario are as follows:

1. The system shall determine the collision avoidance solution.
   (a) The system shall determine the speed rate.
   (b) The system shall determine the heading angle rate.
   (c) The system shall determine the velocity at each time step
   (d) The system shall determine the angle at each time step.
   (e) The system shall check if at each time step the distance between intruder and UAV remains above safety circle.
   (f) For multiple collisions at the same time, the system shall consider the maximum relative heading angle among all heading angles between target and UAV
   (g) For multiple collisions at the same time, the system shall consider the union of all conflict sectors as a conflict resolution.

Collision avoidance simulation user-interface requirements for a non-cooperative scenario are as follows:

1. The system's collision avoidance part shall allow user to change the initial simulation parameters for UAV maneuverability.

   (a) The user shall specify the maximum allowed UAV velocity.

   (b) The user shall specify the minimum allowed UAV velocity.

   (c) The user shall specify the maximum allowed UAV velocity rate.

   (d) The user shall specify the maximum allowed UAV heading angle.

   (e) The user shall specify the maximum allowed UAV turn rate.

2. The system shall be capable of outputting the avoidance outputs.

   (a) The system shall output the velocity of UAV at each time step.

   (b) The system shall output the velocity rate of UAV at each time step.

   (c) The system shall output the heading angle of UAV at each time step.

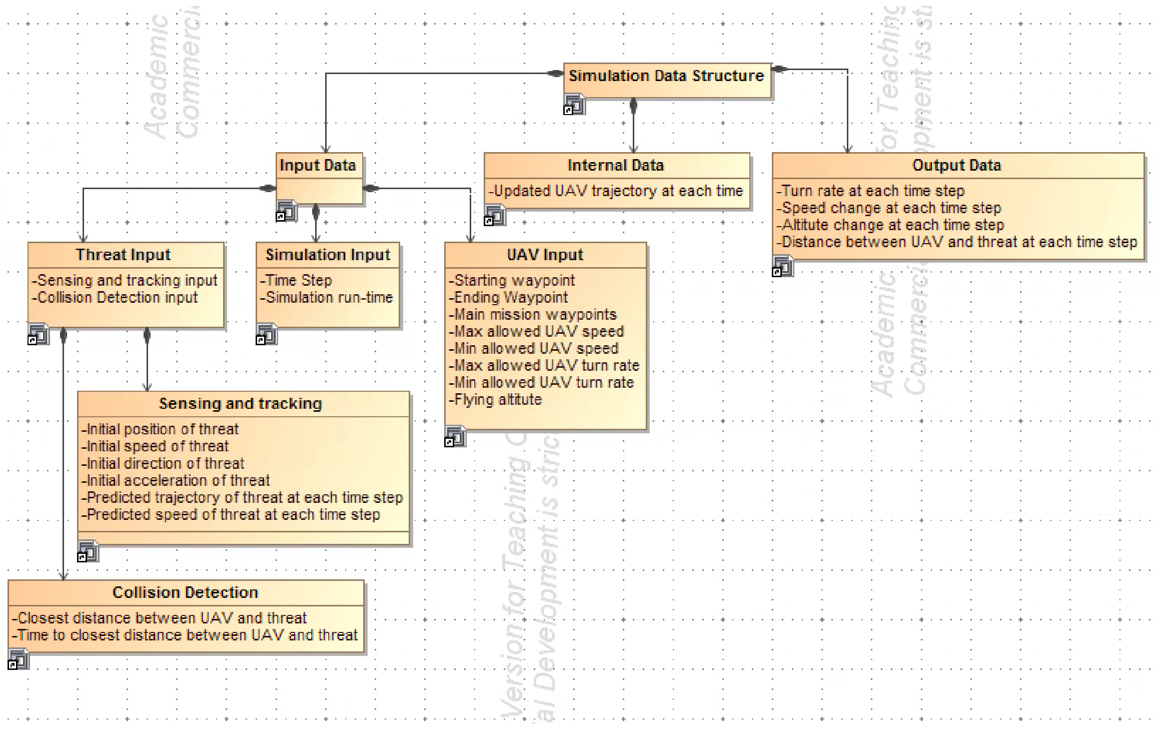
   (d) The system shall output the turn rate of UAV at each time step.

   (e) The system shall output the updated position of UAV at each time step.

   (f) The system shall output the updated distance between UAV and all targets.

3.3.2.2 Avoidance Simulation Data-Structure

The avoidance data structure also has three types of data: input, output and internal data. The input data for this module consists of UAV inputs, Simulation inputs and Threats inputs. Sensor systems and collision detection module provide the collision avoidance part with the required data for threats. As you can see in Figure 3.11, the collision detection part provides data of closest distance between UAV and threat as well as time to closet distance, the data that gained from sensor are the position and speed of the threats and the predicted trajectory of threat at each time step. The required output data would be turn rate and speed rate at each time step as well as updated distance between UAV and threats at each time step.



Simulation Data Structure
Input Data
Internal Data
-Updated UAV trajectory at each time
Output Data
-Turn rate at each time step
-Speed change at each time step
-Altitute change at each time step
-Distance between UAV and threat at each time step
Threat Input
-Sensing and tracking input
-Collision Detection input
Simulation Input
-Time Step
-Simulation run-time
UAV Input
-Starting waypoint
-Ending Waypoint
-Main mission waypoints
-Max allowed UAV speed
-Min allowed UAV speed
-Max allowed UAV turn rate
-Min allowed UAV turn rate
-Flying altitute
Sensing and tracking
-Initial position of threat
-Initial speed of threat
-Initial direction of threat
-Initial acceleration of threat
-Predicted trajectory of threat at each time step
-Predicted speed of threat at each time step
Collision Detection
-Closest distance between UAV and threat
-Time to closest distance between UAV and threat

Figure 3.11: Avoidance Data Model: Data Structure diagram which demonstrates the data types for the input, output and internal data of collision avoidance module

3.3.2.3 Collision Avoidance Algorithm

3.3.2.4 Software Implementation

After implementing the collision avoidance algorithm, the simulation results will be generated to analyze the avoidance maneuvers and the possibility of the success of UAV to avoid the collision. Therefore, we provided some examples of the avoidance simulation to evaluate and examine the implemented algorithm. In Figure 3.12, we see that the UAV changes its heading angle to avoid an obstacle. In this specific scenario, the UAV does not need to change its speed to avoid the obstacle. The second plot in Figure 3.12 shows how the UAV and obstacle are able to remain far enough from each other. The first plot depicts the changes of UAV heading angle in order to avoid the obstacle. As it was mentioned before, this obstacle is non-cooperative, so the UAV shall do all avoiding maneuvers on its own. This is an example of a scenario in which both UAV and obstacles have constant initial speed and heading angle and during the avoidance maneuver, the obstacle don't change its heading angle or/and speed.

The next step in developing our simulation is to generalize the collision avoidance part from avoiding one obstacle into avoiding three obstacles. For this purpose, we simulate the scenario in which the UAV should simultaneously avoid multiple obstacles [42]. Figure 3.13 indicates a situation in which the UAV should change both its turning angle and speed to avoid the collision. This simulation result is an example of a scenario in which all obstacles and UAV have constant initial speed

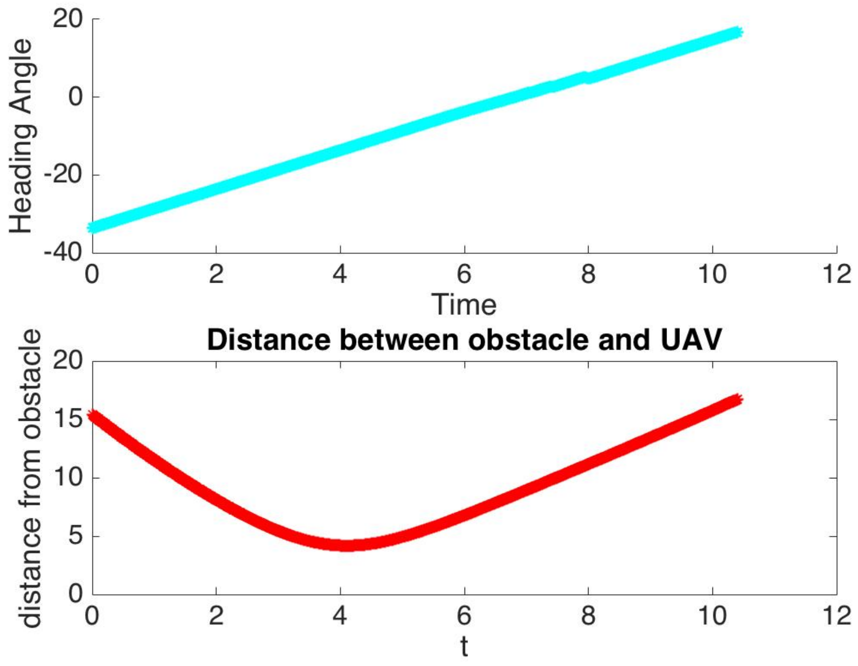


Figure 3.12: Collision Avoidance from one Obstacle: UAV heading angle (top), and distance between obstacles and UAV (bottom)

and heading angle. Moreover, all obstacles have linear trajectories and constant speed during the avoidance maneuvers.

In order to be certain that our avoidance algorithm works properly, we captured the distances between all three obstacles and UAV all the time . Figure 3.14 shows how the obstacles and UAV are far enough from each other based on the minimum allowed distance between UAV and obstacles. This distance is also one of the requirements.

The other possible scenario could be the one in which at least one of the obstacles doesn't have constant speed and heading angle. We extended the collision avoidance algorithms for the situation that obstacles are changing their speeds.

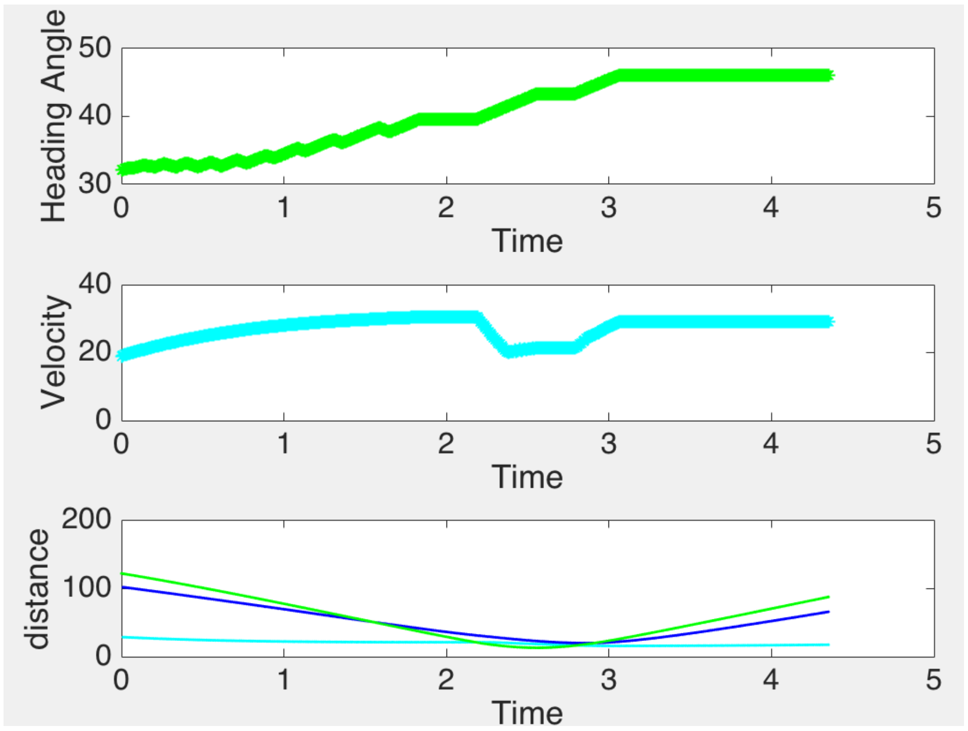


Figure 3.13: Multiple Collision Avoidance: UAV's Velocity and Heading Angle Change During Time so the distance between obstacles and UAV remain above minimum safe distance

Thus, we simulate a scenario that involves 3 different obstacles with changing speeds but with a constant rate (Constant acceleration). This scenario should be evaluated by collision detection algorithm and be updated during the whole simulation. In the case that Figure 3.15 shows, we can see how this fact can affect on the avoidance maneuvers in comparison with the previous scenario in Figure 3.13.

As it is shown in Figure 3.15, the greater the accelerations of the obstacles, the sharper the slopes of heading angle and speed are. The UAV also ended up with greater final heading angle and speed in order to avoid all obstacles in the scenario that the obstacles have a greater tangential acceleration which is shown

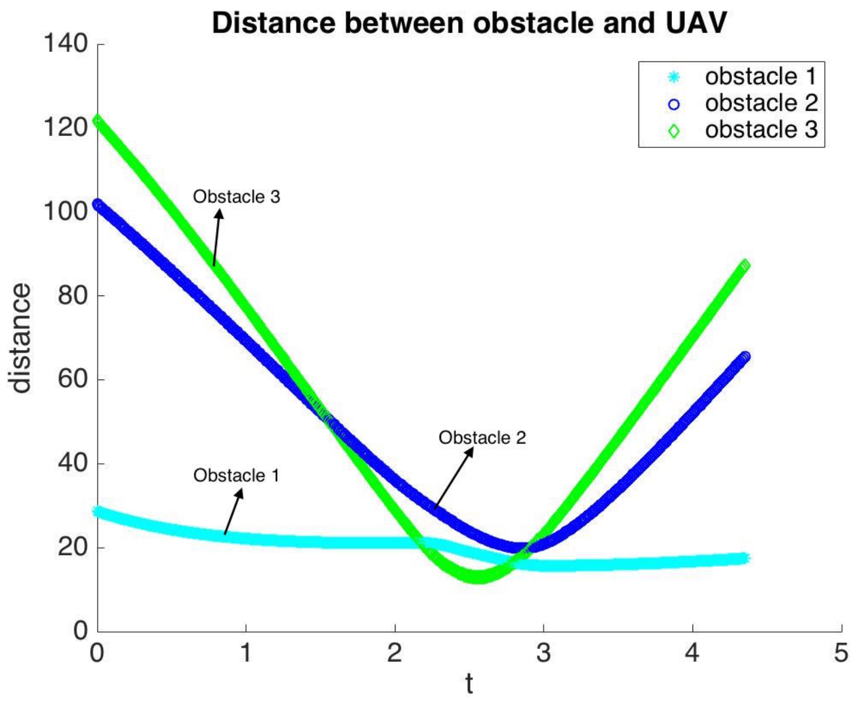


Figure 3.14: Distance Captured: UAV and All Three Obstacles During Avoidance Time. 10m was considered as the minimum allowed distance

in the bottom picture of Figure 3.15. This result indicates that if the obstacle's velocity and acceleration becomes greater, the UAV needs to reach higher heading angle and velocity in order to avoid the obstacle. This means that at the same time, the UAV requires higher speed rate and turn rate to perform avoidance maneuver. Thus, this fact may increase the possibility of the UAV uses the maximum physical capabilities including maximum turn rate and speed rate for conflict resolution. For this reason, if the obstacles' velocity or acceleration is high enough that the UAV requires greater turn rate and speed rate in order to avoid them, then the UAV will face some difficulties for avoiding the possible collision. In this case, it should use its maximum physical capabilities to have the best maneuvers that it can.

We designed two other tests in order to see how the algorithm responds to the obstacles with changing velocity in other scenarios . The first simulation tests the situation in which each obstacle has a different acceleration, the second simulation is designed to test the scenario that the accelerations also change during time. However, these changes are linear. Figure 3.16 represents both these scenarios.

The collision point in the bottom picture of Figure 3.16 occurs earlier in comparison with simulation result in the top picture. It also consists of harsher turn rate and speed rate slopes and the UAV ends up with the greater heading angle and velocity. In the top picture of Figure 3.16, the accelerations of the obstacles are greater than the accelerations in Figure 3.15. As a result, the distances between the UAV and obstacles are smaller than the ones in Figure 3.15. This fact supports the conclusion we had for the simulation results in Figure 3.15 about the possible scenario in which the UAV cannot properly do avoiding maneuver due to its physical constraints. In fact, for the simulation result shown in the bottom picture of Figure 3.16, the distance between the UAV and obstacles falls bellow the minimum safe distance which was set as 10m. However, they do not collide with each other because the distance does not reach zero.

3.4 Analysis and Comparison of Simulation Results

As it was mentioned earlier, the collision avoidance algorithm highly depends on both the initial condition and the physical constraints of the UAV such as maximum turning rate and maximum speed rate. In the previous simulations shown in

Figures 3.15 and 3.16, we examined the effect of changing obstacle's specifications on the avoidance maneuver. Now in order to realize how different specifications of UAV can have effects on collision resolution, we came up with a series of simulation results in which we set various initial conditions based on the UAVs physical constraints. The ultimate goal of these sets of simulation is to identify any patterns or rules for predicting the success of an avoidance maneuver before starting it. However, in this thesis we didn't develop any relationship between different factors and we consider this part as the future work.

The first and second sets of results include some scenarios that the UAV's maximum allowed turn rate is changed in order to examine how the avoidance maneuvers could change for different range of UAV's initial speed.

In Figure 3.17, a set of simulations is shown that indicates how the proposed maneuvers change if the starting speed of UAV changes. As it can be seen in Figure 3.17, in the right diagram that captures a situation in which the initial speed of UAV is greater than the left simulation result in Figure 3.17, UAV has a successful avoidance maneuver that ended up with a higher final speed in comparison to other diagram. In the left diagram in Figure 3.17, UAV has the lower initial speed and ended up its maneuver with the higher turn angle.

In the Figure 3.17, avoidance maneuvers for various initial speed can be totally different from each other both in terms of heading angle pattern and speed pattern. The slope of speed will be smoother if the initial speed of the UAV becomes greater. However, despite the amount of initial heading angle of UAV, the slopes and patterns of heading angle change are very similar before the closest approaching point for different initial velocities.

In the next set of simulation results, we changed the maximum allowed turn rate for UAV between 5 deg/s and 20 deg/s in order to examine the possible changes in turn rate, speed rate and avoidance maneuvers of the UAV. For this purpose. As a result, in this set we provided four simulation results with different UAV maximum turn rate. Figure 3.18 shows the details of heading angle, speed and distance between UAV and obstacles for each of these scenarios.

As it can be seen in Figure 3.18, the overall trend for speed rate and turn rate is the same among all four simulation results. The only difference is that the maximum heading angle that the UAV reaches will increase by increasing the maximum allowed turn rate. However, in this set of simulation, there is a little difference in speed rate of UAV. This means that the distance between UAV and obstacles can be slightly bigger in the bottom right simulation than the top left, due to the higher maneuverability of the UAV in terms of maximum allowed turn rate.

From Figure 3.18, one can conclude that changing in the input values and constraints causes some changes in the UAV's final heading angle and speed. The other important conclusion is that for a specific scenario, the proposed avoidance maneuver is the same for different range of inputs. The only thing that might differ for doing the proposed maneuver, is the sharpness and smoothness of heading angle and speed change of the UAV which is highly dependent on the UAV's constraints and initial conditions.

As we discussed earlier in this chapter that due to the physical constraints of the UAV, the collision avoidance algorithm might not be successful in all scenarios, we examined several situations to see if our collision avoidance algorithm fails in at least one of them or not. At the end, we encountered a scenario in which the UAV is not agile enough to avoid the obstacles. In Figure 3.19, it is shown that the UAV cannot properly avoid one of the obstacles. Figure 3.19 demonstrates that in the time interval between t=1.9s and t=2.1s (less than 0.2 s to the collision point (t=2.3 s)), the heading angle rate is 5 deg/s. Although, the UAV applies its maximum turn rate capability, the distance between UAV and obstacle 3 falls bellow the minimum allowed distance (10m). Thus, despite of the fact that the distance is a bit above zero, it is still bellow the minimum allowed distance and is considered as a collision.

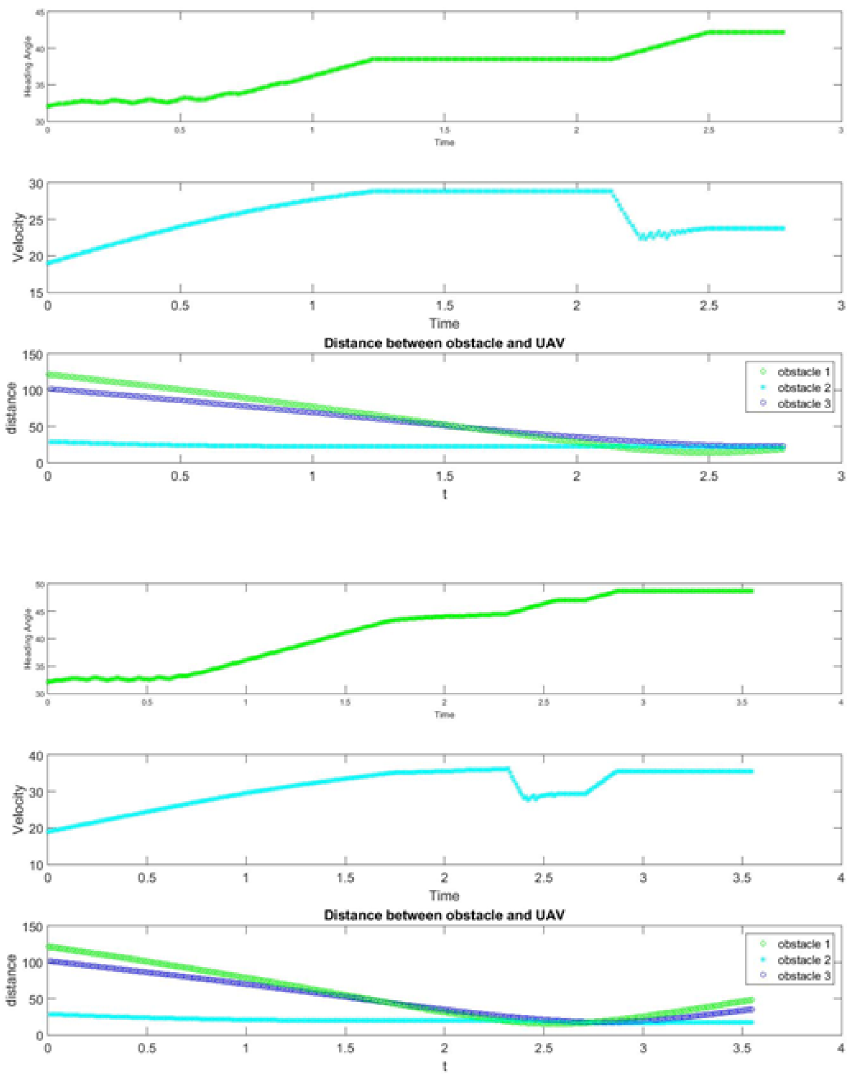


Figure 3.15: Avoidance From Threats with Constant Tangent Acceleration: Obstacles' Velocity Changes During Time with a constant rate. The top image indicates a scenario in which the 3 obstacles have a constant acceleration both in [X, Y] coordinates. The amounts of accelerations are as followed ([-1, 1], [-1, 1], [-1, 1]) m/s^2. The bottom pictures simulates a situation in which all three obstacles have a constant acceleration both in [X, Y] coordinates. The amounts of accelerations are as followed ([2, 1], [2, 1], [2, 1]) m/s^2.

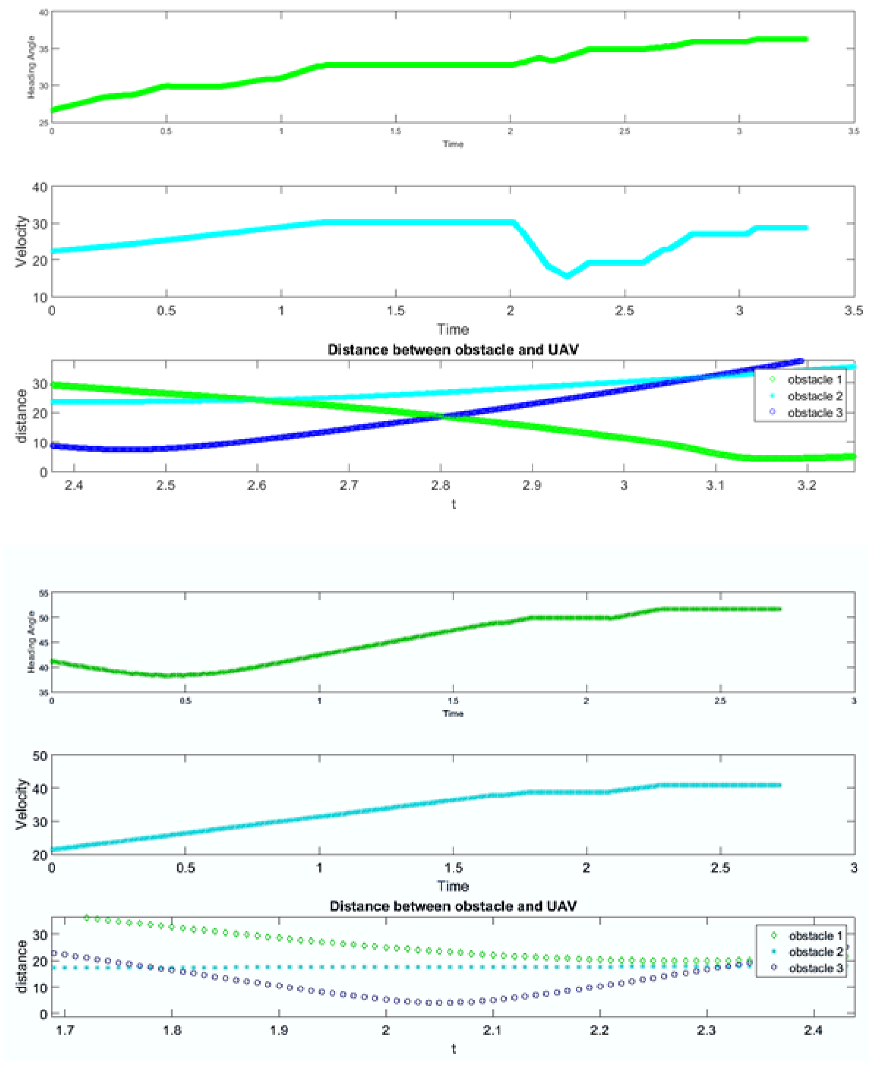


Figure 3.16: Avoidance From Threats with Changing Tangent Acceleration- Obstacles' Velocity Change During Time. The top image indicates a scenario in which the 3 obstacles have a constant acceleration in XY coordinates. The amounts of accelerations are as follows ([-3, 10], [10, -4], [5, 4]) m/s^2. The bottom pictures simulates a situation in which all three obstacles have a changing acceleration with linear equations in XY coordinates. The amounts of accelerations are as follows ([-3t, 10t], [10t, -4t], [5t, 4t]) m/s^2 where t is time. In both pictures, the turn rate is 15 deg/s and the speed rate is 20 m/s^2

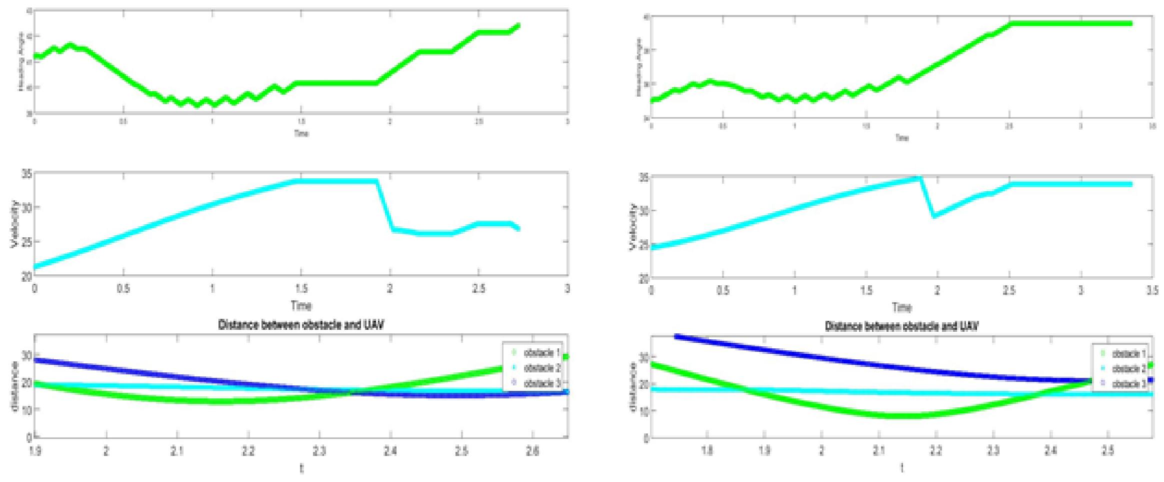


Figure 3.17: Avoidance Results for different initial UAV's speed: The UAV's speed in the simulations from left to right are as follows: [16, 14], [20,14] m/s in two dimensional X-Y coordinate relative to obstacles. The maximum turn rate for UAV is 5 deg/s and maximum speed rate is 10 m/s^2. Each simulation consists of 3 diagrams, UAV heading angle, UAV speed, distance between UAV and each obstacle

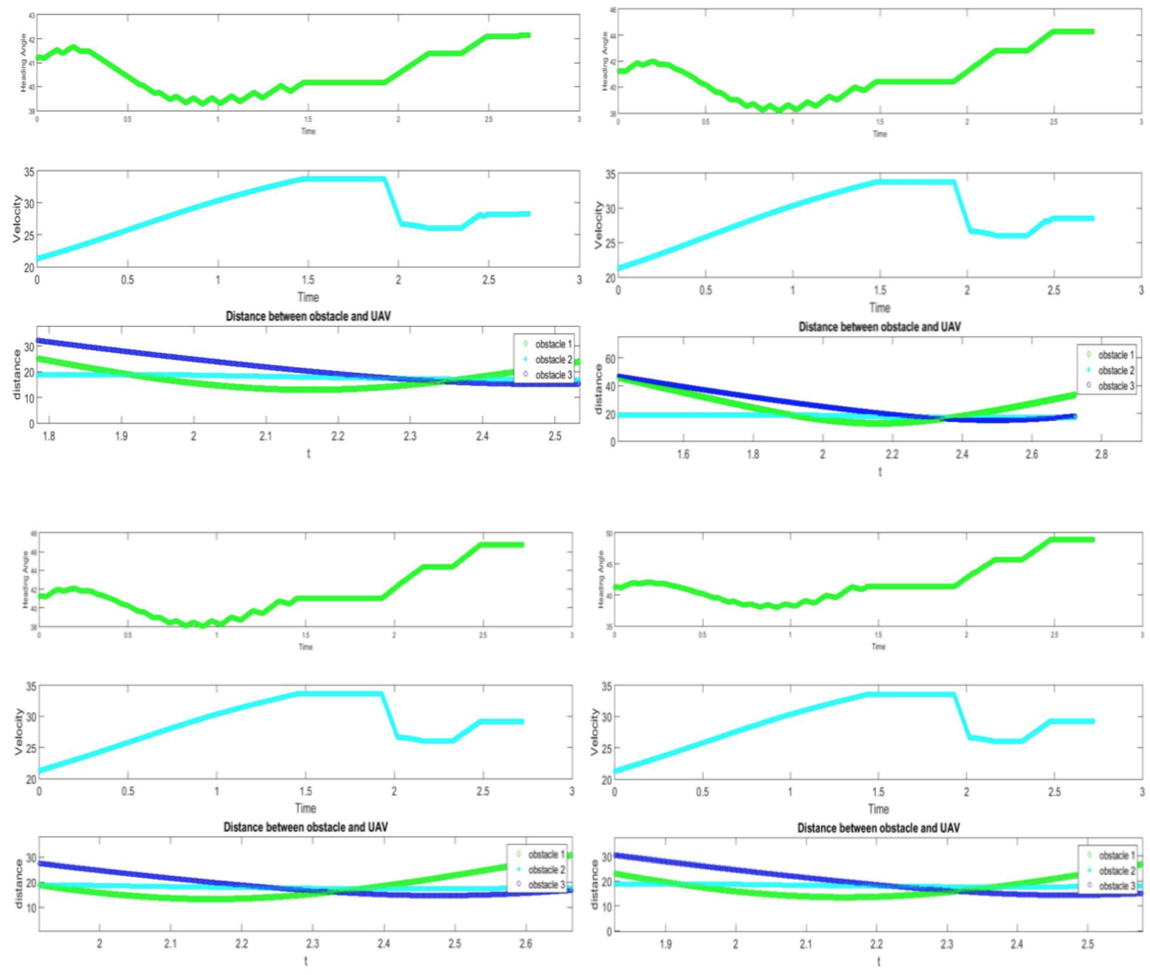


Figure 3.18: Avoidance Results for different UAV's maximum turn rate and a fixed initial UAV speed: The UAV's maximum turn rate for the top left simulation result 5 deg/s. This amount for the top right diagrams is 10 deg/s and for the bottom left and bottom right pictures are 15 deg/s and 20 deg/s respectively

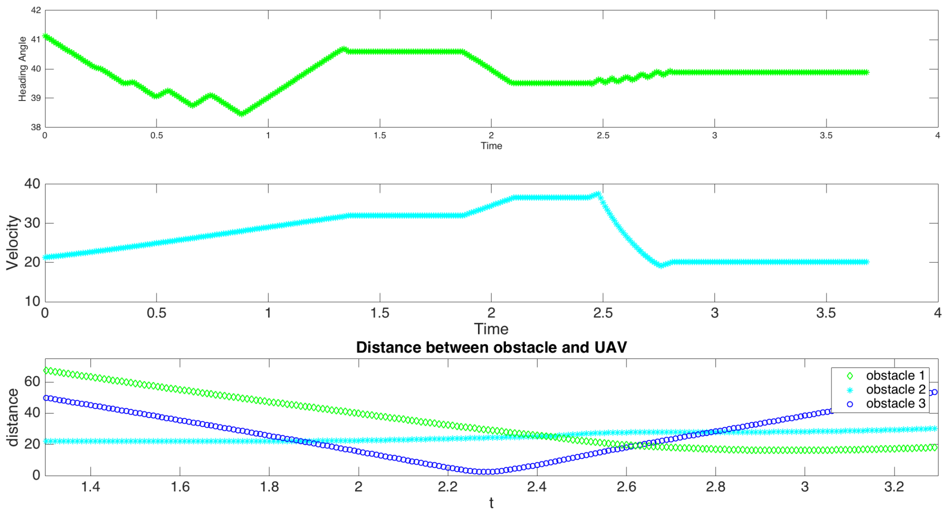


Figure 3.19: Collision Avoidance Simulation Results for the scenario in which the UAV cannot avoid all the obstacles. The amount of turn rate is 5 deg/s.

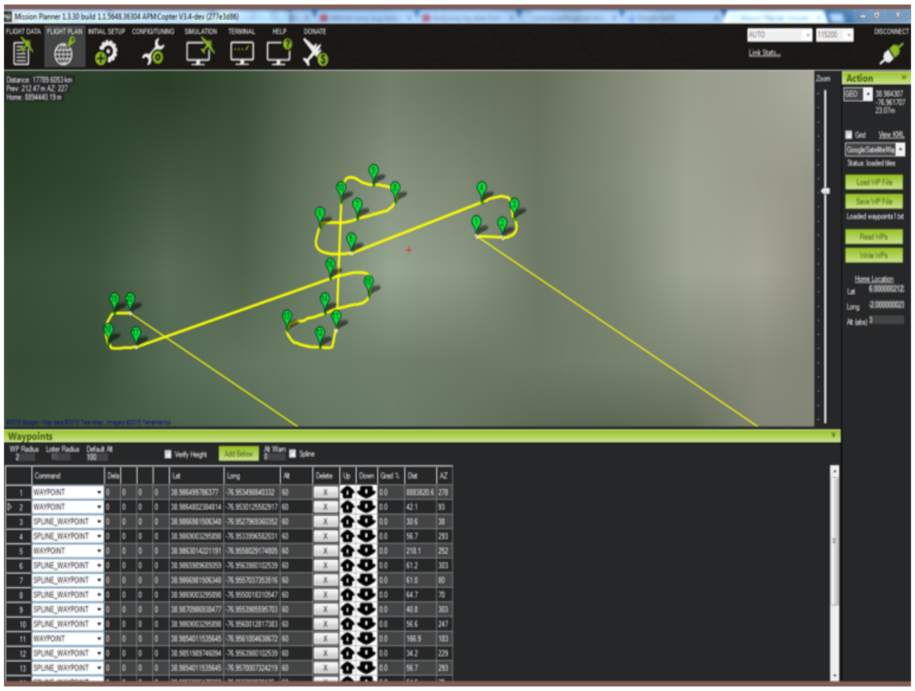
Chapter 4: Conclusion

4.1 DISCUSSION AND FUTURE WORK

A model based systems engineering approach applied to this project, helps us in formalizing requirements analysis and requirement identification. System artifacts ensure the consistency between design and requirements so the simulation results and final mission planning design will satisfy the stakeholders' needs. The simulation results helped us to identify and verify the robustness of the algorithms for different scenarios.

The next step after developing path planning and collision avoidance modules is to integrate them into the mission planner and use onboard UAVs. We already implemented the trajectories generated by our Path Planning module into the Mission Planner which is an open source simulation software used for mission design for UAVs and aircraft. For this purpose, we got the generated waypoints in our Path Planning module, then we converted the XYZ coordinates of the waypoints into the latitude- longitude- altitude coordinates. After that we insert the XML file consisting of the desired waypoints into the mission planner simulation software. We got the following results which are similar to the trajectory the Path Planning module generated. Figure 4.1 shows the trajectory of the UAV in Mission Planner using the implemented waypoints. The trajectory in Mission Planner can be compared to Dubins Airplane trajectory in Figure 4.2. However, integrating path planning with collision avoidance and implementing the collision avoidance results into the Mission Planner would be our next step.

In the future, we will explore the details of the detection and sensing parts, which would include some challenges about sensor systems, tracking objects and trajectory predictions. Thus, sensor modeling would also be considered for our future works to determine if the system can satisfy detection and sensing requirements. We will try to focus more on the most challenging part for the non-cooperative collision, which is the ability of UAV to deal with threats' maneuvers that are made at the very last seconds before or right after decision making process and avoidance maneuvering calculation. So we need to cover more possible scenarios for collision avoidance than what we covered in this thesis. Also communication and navigation accuracy play important roles in accomplishing a safe mission. Considering these parts would be challenging and increase the uncertainty of the results. Furthermore, a possible future work could be finding a relationship between the UAV's physical constraints and the feasibility of the avoidance maneuver. This could be a condition checking instead of running the whole simulation system.



Distance: 17789.6053 km
Prev: 212.47 m AZ: 227
Home: 8894440.19 m

Loaded waypoints1.txt

WP Radius 2 | Loiter Radius | Default Alt 100 | Alt Warn 0

	Command	Dela				Lat	Long	Alt	Delete	Up	Down	Grad %	Dist	AZ
1	WAYPOINT	0	0	0	0	38.986499786377	-76.953498840332	60	X	⇧	⇩	0.0	8883820.6	278
2	WAYPOINT	0	0	0	0	38.9864802384814	-76.9530125582917	60	X	⇧	⇩	0.0	42.1	93
3	SPLINE_WAYPOINT	0	0	0	0	38.9866981506348	-76.9527969360352	60	X	⇧	⇩	0.0	30.6	38
4	SPLINE_WAYPOINT	0	0	0	0	38.9869003295898	-76.9533996582031	60	X	⇧	⇩	0.0	56.7	293
5	WAYPOINT	0	0	0	0	38.9863014221191	-76.9558029174805	60	X	⇧	⇩	0.0	218.1	252
6	SPLINE_WAYPOINT	0	0	0	0	38.9865989685059	-76.9563980102539	60	X	⇧	⇩	0.0	61.2	303
7	SPLINE_WAYPOINT	0	0	0	0	38.9866981506348	-76.9557037353516	60	X	⇧	⇩	0.0	61.0	80
8	SPLINE_WAYPOINT	0	0	0	0	38.9869003295898	-76.9550018310547	60	X	⇧	⇩	0.0	64.7	70
9	SPLINE_WAYPOINT	0	0	0	0	38.9870986938477	-76.9553985595703	60	X	⇧	⇩	0.0	40.8	303
10	SPLINE_WAYPOINT	0	0	0	0	38.9869003295898	-76.9560012817383	60	X	⇧	⇩	0.0	56.6	247
11	WAYPOINT	0	0	0	0	38.9854011535645	-76.9561004638672	60	X	⇧	⇩	0.0	166.9	183
12	SPLINE_WAYPOINT	0	0	0	0	38.9851989746094	-76.9563980102539	60	X	⇧	⇩	0.0	34.2	229
13	SPLINE_WAYPOINT	0	0	0	0	38.9854011535645	-76.9570007324219	60	X	⇧	⇩	0.0	56.7	293

Figure 4.1: Trajectory generated in Mission Planner using Dubins Airplane Waypoints

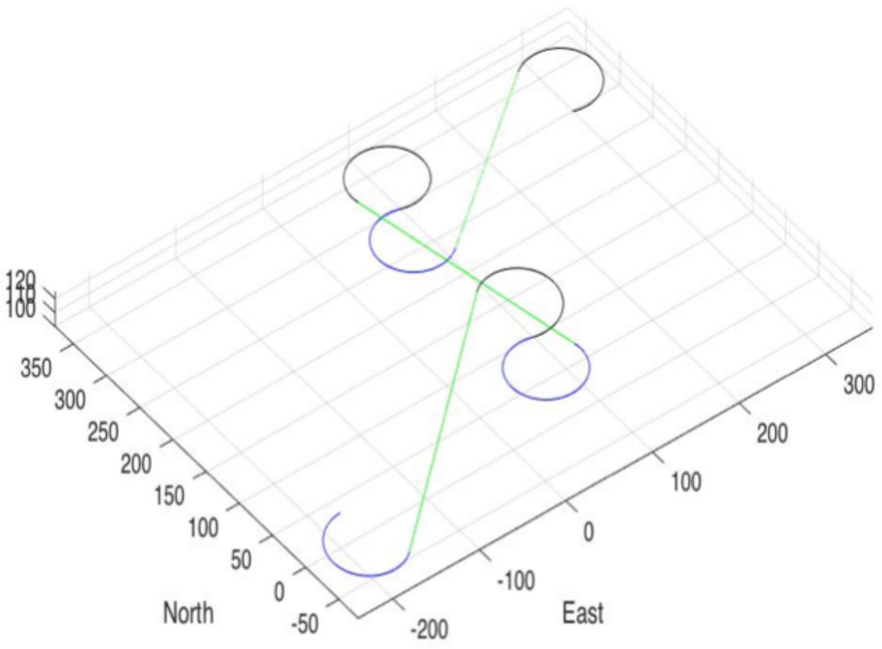


Figure 4.2: Trajectory generated in Path Planning simulation module using the start and end points

Appendix A: Dubins Airplane Model

A.1 Dubins Car Model

Dubins car is a path planning algorithm using the concept of a moving car at constant forward speed, $u_s = 1$, so it cannot move backward. The other critical constraint and assumption related using this algorithm is the maximum steering angle ϕ_{max} , which results in a minimum turning radius ρ_{min} [34] [35]. The Dynamics of the moving car is as follows:

Considering the fact that the car cannot move sideways, the back wheels turn instead of sliding. Therefore, if all four wheels are simultaneously turned toward a direction, then the car follows a desired curve. However, to have this maneuver, the car has some rolling constraints which makes it difficult or impossible to do all the turning maneuvers. We consider the whole car as a rigid body with the origin of the center rear axle (x,y). As a result the configuration of the car is defined by $q = (x, y, \theta)$ where θ is the angle between horizon and the car axis. The x-axis is along the main axis of the car. Other important variables in defining the motion of the car are as follows: 1. s denotes the speed, 2. ϕ denotes the steering angle 3. L is the distance between front and rear wheels and 4. ρ is the radius of the circle the car travels with a constant ϕ.

Having defined all the notation, we can represent the motion of the car as a set of equations.

$$\begin{aligned} \dot{x} &= f_1(x, y, \theta, s, \phi) \\ \dot{y} &= f_2(x, y, \theta, s, \phi) \\ \dot{\theta} &= f_3(x, y, \theta, s, \phi) \end{aligned} \tag{A.1}$$

considering the fact that in a small time interval of , we have $dy/dx = \dot{y}/\dot{x}$ and $\tan\theta = \sin\theta/\cos\theta$ and as a result, we have $dy/dx = \tan\theta$. The equation bellow is derived:

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{A.2}$$

The equation will be satisfied if $\dot{x} = s\cos\theta$ and $\dot{y} = s\sin\theta$. To derive the equation for $\dot{\theta}$ we define w as the distance traveled by the car and we note that $dw = \rho(d\theta)$. From trigonometry, $\rho = L/\tan\phi$. All of this leads to the following equation:

$$d\theta = \tan\phi/Ldw. \tag{A.3}$$

As we know that $\dot{w} = s$ the equation becomes

$$\dot{\theta} = s/L\tan\phi. \tag{A.4}$$

As the car travels between q_I and q_G, we can minimize the length of the curve of car's path. Due to ρ_{min}, this is considered as a shortest path problem [34] [35]. If $\rho = 0$, then there is no curvature bound. As a result the criteria to optimize the path is

$$L(q, u) = \int_0^{t_F} \sqrt{\dot{x^2}(t) + \dot{y^2}(t)}dt \tag{A.5}$$

in which t_F is the time at which the car reaches to the departure. since the speed is constant we can simplify the system to

$$\begin{aligned} \dot{x} &= \cos\theta \\ \dot{y} &= \sin\theta \\ \dot{\theta} &= u \end{aligned} \tag{A.6}$$

If for simplicity, we can assume that $\tan\phi = 1$. The following results also hold for any ϕ_{max} between $(0, \pi/2)$. The Dubins car algorithm shows that the shortest

Table A.1: Dubins Path

Motion	Symbol	Steering: u
Straight	S	0
Left	L	1
Right	R	-1

path between any two waypoints can be derived using combination of those three motion primitives (Straight, Left, Right). The basic idea is that for Straight motion primitive, the car travels straight ahead and for the Left and Right primitives the car turns as sharply as possible to the left or right, respectively. So the possible combinations of these three motion primitives are

$$L_\alpha R_\beta L_\gamma, R_\alpha S_\beta L_\gamma, L_\alpha S_\beta R_\gamma, L_\alpha S_\beta L_\gamma, R_\alpha S_\beta R_\gamma, R_\alpha L_\beta R_\gamma \tag{A.7}$$

The α, β, γ signs indicate the amount of total rotation for each motion. In this case we can derive the time related to each motion primitives. The shortest path between two points can always be derived by one of these words. Here you can see the Dubins curves

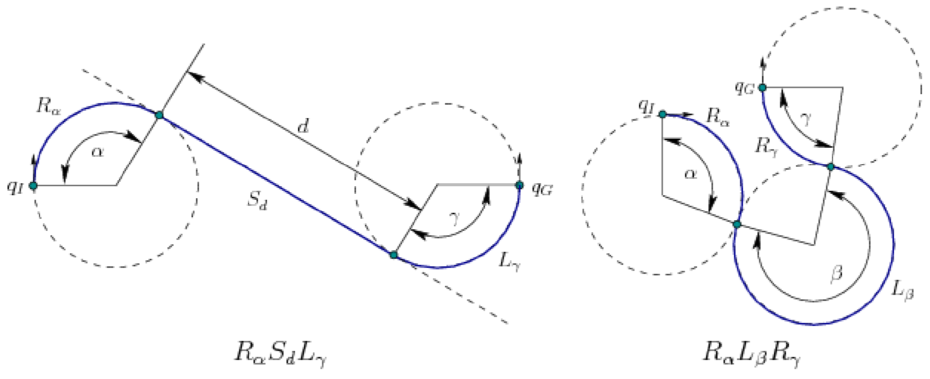


Figure A.1: The trajectories for two possible combinations of Dubins car motion primitives [35]

A.2 Extending Dubins Car to Dubins Airplane

For an airplane flying in the air, wind speed can have some effects on the path that aircraft tries to follow. However, wind effects are not considered in the path planning equations and are treated as some disturbances during flight and the flight controller tries to reject them [33]. Because the effects are not known before the moment they act on the aircraft. Therefore, Dubins Airplane algorithm can be developed without considering the wind effects. One difference between Dubins car and Dubins Airplane is that Dubins Car's path is the combination of some curves and straight lines based on the car's steering angles, while Dubins Airplane's path is the combination of straight lines and helical paths based on flight path angles and banking angles. Furthermore, in addition to turn rate constraint we had in Dubins

Car, the Climb rate constraint is added to Dubins Airplane algorithm. The whole idea of Dubins Airplane is that the Dubins paths are generated by intersecting two planes. So if we are able to formulate these two planes, we can get the path using an arbitrary point and the direction of the generated line.

Ignoring the wind effects, we can consider Dubins Airplanes paths relative to the inertial environment. In this case, that the effects of wind are not accounted when formulating equations of motions, the airspeed V equals to the ground speed, the heading angle ψ equals to the course angle which means we assume there is no side-slip angle and the flight path angle equals to the air-mass-referenced flight path angle [33]. The very basic Dubins Airplane equation of motions are as follows if we consider $(r_n, r_e, r_d)^T$ as the starting position of UAV.

$$\begin{aligned} r_n &= V\cos\psi \\ \dot{r_e} &= V\sin\psi \\ \dot{r_d} &= u_1 \qquad |u_1| \leq 1 \\ \dot{\psi} &= u_2 \qquad |u_2| \leq 1 \end{aligned} \tag{A.8}$$

$$\begin{pmatrix} \dot{r_n} \\ \dot{r_e} \\ \dot{r_d} \end{pmatrix} = \begin{pmatrix} V\cos\psi\cos\gamma \\ V\sin\psi\cos\gamma \\ -V\sin\gamma \end{pmatrix} \tag{A.9}$$

These equations are based on the assumptions that there are some constraints on airspeed V, flight path angle γ^c and the bank angle ϕ^c just the same as the

Dubins Car algorithm. So the Dubins Airplane should satisfies these constraints

$$\begin{aligned} \phi^c &\leq \bar{\phi} \\ \gamma^c &\leq \bar{\gamma} \end{aligned} \tag{A.10}$$

The relationship between the heading angle ψ and the bank angle γ is given by the coordinated turn condition:

$$\dot{\psi} = (g/V)\tan\phi \tag{A.11}$$

where g is the gravity acceleration.

As a result, th equations of motion for the airplane in the desired situation and with the assumption that there is no violation between the autopilot command and the response of UAV to the commands are as follows [33] :

$$\begin{aligned} r_n &= V\cos\psi\cos\gamma^c \\ \dot{r_e} &= V\sin\psi\cos\gamma^c \\ \dot{r_d} &= -V\sin\gamma^c \\ \dot{\psi} &= (g/V)\tan\phi^c \end{aligned} \tag{A.12}$$

These equations of motion for airplane based on Dubins Airplane model, can be used based on constraints of airplane's physical capabilities. So there are limits on the bank angle and flight-path angle. So $\psi^c < \bar{\psi}$ and $\gamma^c < \bar{\gamma}$. As we have the kinematic relationships for UAV motions in 3 dimensional, the next step is to find the desired planes and the intersection of them to generate a desired Dubins Airplane. We can consider the velocity vector of the UAV along the intersection by defining the equation bellow in which we used the second partial derivatives for each of these two planes:

$$V_{(r)} = 1/2{\alpha_1}^2(r) + 1/2{\alpha_2}^2(r) \tag{A.13}$$

In order to assure that the Dubins Airplane follows the intersection, we use $\partial V_{(r)}/\partial r$. When the Dubins Airplane is on the generated line, it must be perpendicular to both $\partial\alpha_1/\partial r and \partial\alpha_2/\partial r$. So we can be sure that the Dubins Airplane is also in the right direction of its trajectory. Having explained these facts, the desired velocity vector would be:

$$u' = -K_1 \partial V_{(r)}/\partial r + k_2(\partial\alpha_1/\partial r \times \partial\alpha_2/\partial r) \tag{A.14}$$

We also need to normalize A.14 to make it equal to V, airspeed of UAV. So we have:

$$u = Vu'/||u'|| \tag{A.15}$$

As a result, $u = (u_1, u_2, u_3)^T$ is equal to the A.9. In this regard, we can solve equations for $\gamma^c, \psi^d and \phi^c$ using equations A.14 and A.9. Then we have:

$$\begin{aligned} \gamma^c &= -sat_{\bar{\gamma}}[\sin^{-1}(u_3/V)] \\ \psi^d &= atan2(u2, u1) \\ \phi^c &= sat_{\bar{\phi}}[K_{\bar{\psi}^c}(\psi^d - \psi)] \end{aligned} \tag{A.16}$$

Where atan2 is the four quadrant inverse tangent, $K_{\bar{\psi}^c}$ is positive and Sat function is defined as bellow

$$Sat_a[x] = \begin{cases} a & \text{if } x \geq a \\ -a & \text{if } x < -a \\ x & \text{Otherwise.} \end{cases} \tag{A.17}$$

After deriving motion kinematics and the Dubins Airplane path formula, we need to generate both Straight lines and Helical paths based on these principals for Dubins Airplane paths. In order to find more details of the equations for Helical and straight line paths we refer you to [33]. How ever, we provide the reader with the final equations for these paths. The Straight-line paths and the desired velocity vector are derived using A.18 and the Helical paths and its desired velocity vector are derived from A.19.

$$\begin{gathered} P_{line}(c_l, \psi_l, \gamma_l) = \{r \in R : R = C_L + \sigma q_l, \sigma \in R\} \\ u'_{line} = K_1(n_{lon}{n_{lon}}^T + n_{lat}{n_{lat}}^T)(r - c_l) + k_2(n_{lon} \times n_{lat}) \end{gathered} \tag{A.18}$$

Where n_{lon} is the unit vector perpendicular to the longitudinal plane defined by q_l and n_{lat} is the unit vector perpendicular to the lateral plane defined by q_l.

$$\begin{gathered} P_{helix}(c_h, \psi_h, \gamma_h, R_h, \lambda_h) = \{r \in R : \alpha_{cyl}(r) = 0 \& \alpha_{pl}(r) = 0\} \\ u'_{helix} = K_1(\alpha_{cyl}\partial\alpha_{cyl}/\partial r + \alpha_{pl}\partial\alpha_{pl}/\partial r) + \lambda K_2(\partial\alpha_{pl} \times \partial\alpha_{cyl}) \end{gathered} \tag{A.19}$$

Where $\alpha_{cyl}(r)$ and $\alpha_{pl}(r)$ equal to:

$$\begin{cases} \alpha_{cyl}(r) = ((r_n - c_n)/R_n)^2 + ((r_e - c_e)/R_h)^2 - 1 \\ \alpha_{pl}(r) = ((r_d - c_d)/R_h) + \tan\gamma_h/\gamma_h(\tan^- 1((r_e - c_e)/(r_n - c_n)) - \psi_h) \end{cases} \tag{A.20}$$

The Dubins airplane path between two nodes can be derived using Dubins motion primitives as we talked earlier in this section. So here you can see the possible path between two specific nodes can be created using the combinations of these motion primitives.

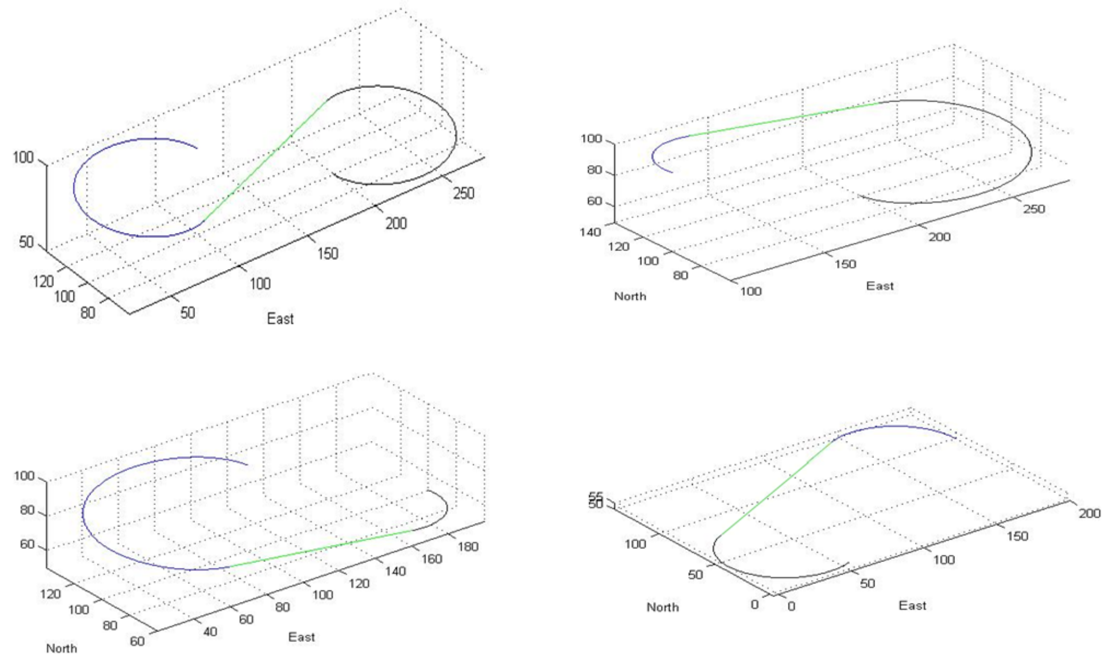


Figure A.2: The trajectories for possible combinations of Dubins airplanes primitives. From top left picture to bottom right picture the motion primitives are LSR, RSL, LSL and RSR

A.2.1 Dubins Airplane Path

There are three different scenarios for developing the Dubins Airplanes depending on the altitude difference between the start point and the end point. These three cases are named, low altitude, medium altitude and high altitude. [33]. The minimum turn radius for Dubins Airplane is

$$R_{min} = (V^2/g)\tan\bar{\phi} \tag{A.21}$$

We consider the altitude between the start and end points to be low if

$$|z_{de} - z_{ds}| \leq L_{car} R_{min} \tan\bar{\gamma} \tag{A.22}$$

The altitude is medium if

$$L_{car}R_{min}\tan\bar{\gamma} < |z_{de} - z_{ds}| \leq [L_{car}R_{min} + 2\pi R_{min}]\tan\bar{\gamma} \tag{A.23}$$

where the term πR_{min} shows the adding one orbit at radius $_{min}$ to the path length. Th altitude is to be considered as high altitude if

$$|z_{de} - z_{ds}| > [L_{car}R_{min} + 2\pi R_{min}]\tan\bar{\gamma} \tag{A.24}$$

The low altitude case occurs when the airplane can reach the end point from the start point using the simple Dubins path while satisfying the $\gamma^c < \bar{\gamma}$ constraint. The high altitude case occurs when the altitude is too high to satisfy the flight path angle constraint. So in order to extend the Dubins path is to generate some certain numbers of spiral turns at the beginning or end of the path. The medium altitude case occurs when the altitude is too low to have complete spiral turns and is too high to have the Dubins path while satisfying the flight path angle constraints. In this case certain maneuvers added to the Dubins path in order to complete the path between the start and end nodes.

Appendix B: Differential Geometry Concept

B.1 Introduction

Both Collision Detection and Avoidance algorithms are used the differential geometry concepts. one is controlling UAV heading angle and the other is controlling ground speed. This algorithm can be used for one or multiple collisions at the same time. The algorithm is also used the principals of airborne collision avoidance systems confirming to TCAS. This study limits the analysis to non-cooperating UAVs and intruders. Some of the assumptions that are considered for developing this algorithm are

- Vehicle dynamics are presented by point mass in Cartesian coordinates on R^2.
- The threats are non-cooperative and non-maneuvering.
- The threats have been sensed by the UAV's sensors so the deterministic positions and velocity vector of the intruders are determined. So the UAV can predict the future trajectories of threats based on the current position and velocity vectors and their linear projections.

B.1.1 Differential Geometry Concept

In order to develop the collision algorithms, we should describe the concepts of the conflict and detection and resolution of the conflict. We used the same concept for all of these definitions. We used the concepts proposed in [37] and [38]. Considering that the intruder is sensed by the sensors, the UAV establishes a sightline between itself and the intruder. This sightline vector is given by

$$r = r_a - r_u. \tag{B.1}$$

considering the assumptions that the velocity of both intruder and UAV is constant, then the differential of equation B.1 is

$$\dot{r}t_s + r\dot{\theta}_s n_s = v_a t_a - v_u t_u \tag{B.2}$$

where n_s is the basis vector normal to the sightline for UAV and the t_s and t_a are the basis vectors along to the sightline for UAV and intruders, respectively. In the figure bellow you can see the deferential geometry related to the UAV and intruder.

Components of the relative velocity vector along and normal to the sightline are as follows. These equations are derived from the B.2 and dot product of t_s and n_s to the B.2 equation, respectively

$$\begin{aligned} \dot{r} &= v_a t_s.t_a - v_u t_s.t_u \\ r\dot{\theta}_s &= v_a n_s.t_a - v_u n_s.t_u \end{aligned} \tag{B.3}$$

We can also derive the relative acceleration along and normal to the sightline by

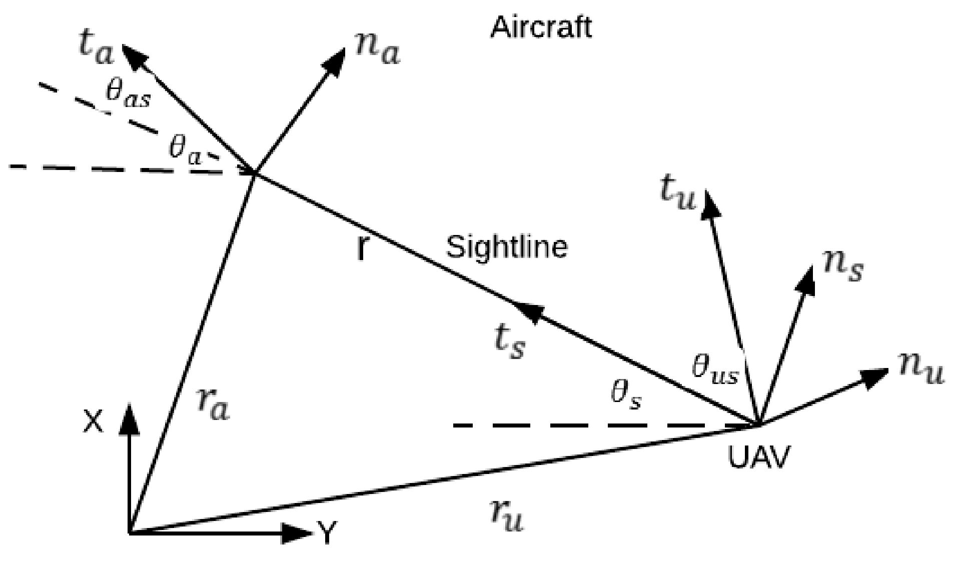


Figure B.1: Geometry of the UAV relative to the threat in the Cartesian coordinate

modifying equation B.2 and using the Serret-Frenet[] equations. So we have

$$\begin{aligned}(\ddot{r} - r\dot{\theta_s}^2) &= v_a^2 k_a t_s.n_a - v_u^2 k_u t_s.n_u \\ (r\ddot{\theta_s} + 2\dot{r}\dot{\theta_s}) &= v_a^2 k_a n_s.n_a - v_u^2 k_u n_s.n_u\end{aligned} \tag{B.4}$$

B.1.2 Detection Algorithm

This algorithm investigates the collision condition and if the UAV may have a collision with the threats or not. Because both the UAV to sightline angle θ_{us} and threat to sightline angle θ_{as} are considered to remain constant, the collision detection triangle doesn't change shape, it just shrinks as the UAV and threat move towards each other. If the distance between the UAV and the threat is or will be smaller than the minimum separation of d_m within a specific time, a collision will be detected because they disobey the minimum allowed separation distance. So the CAD and TCPA concepts are used to detect the collision. The algorithms and equations are used and modified from previous researches presented in [39], [40], [42] and [43]. For a non-maneuvering threat, the CAD d_c can be derived by projecting the relative position vector along the sightline:

$$d_c = r \sin\theta \tag{B.5}$$

Where θ is the angle between the sightline and the relative velocity vector. if the d_p is the relative distance to the CPA, we can define it as

$$d_p = r \cos\theta \tag{B.6}$$

so the TCPA can be derived as follows

$$\tau = d_p / v_r \tag{B.7}$$

So we can detect the collision if the d_c is smaller than the minimum separation of d_m and the TCPA is in the future and is before the Look-ahead time T. for the multiple collision, we can extend the above conditions for the multiple threats which are as follows

$$\begin{pmatrix} d_{c1} < d_m \text{ and } \tau_1 \in [0, T) \\ \\ d_{c2} < d_m \text{ and } \tau_2 \in [0, T) \\ \\ \ldots \\ d_{cn} < d_m \text{ and } \tau_n \in [0, T) \\ \end{pmatrix} \tag{B.8}$$

B.1.3 Collision Avoidance

There are two approaches for resolving the collision, the first approach is by controlling the heading angle of the UAV and the second is by controlling both heading angle and speed of the UAVs. For the first approach, the UAV speed is greater than the threat's speed and for the second approach the speed of the threat is greater than the speed of UAV. In order to get the most optimal solution, we consider the vector that guarantees the condition $d_m = d_c$, so the resolution is provided such that satisfies the minimum separation distance. In order to consider probable maneuvers that the threat can have and we don't account for in our formulation, we can scale up the d_m so the collision resolution will be more reliable. As it is obvious, there are two solutions for clockwise and anti-clockwise maneuvers. In this section we just consider the clockwise solution.

B.1.3.1 Approach I

For this approach and the clockwise solution, we can derive the direction of the relative velocity vector, θ_r is

$$\theta_r = \theta_m = \theta_s + \theta_d \tag{B.9}$$

In the Figure B.2, you can see the conflict resolution for clockwise rotation.

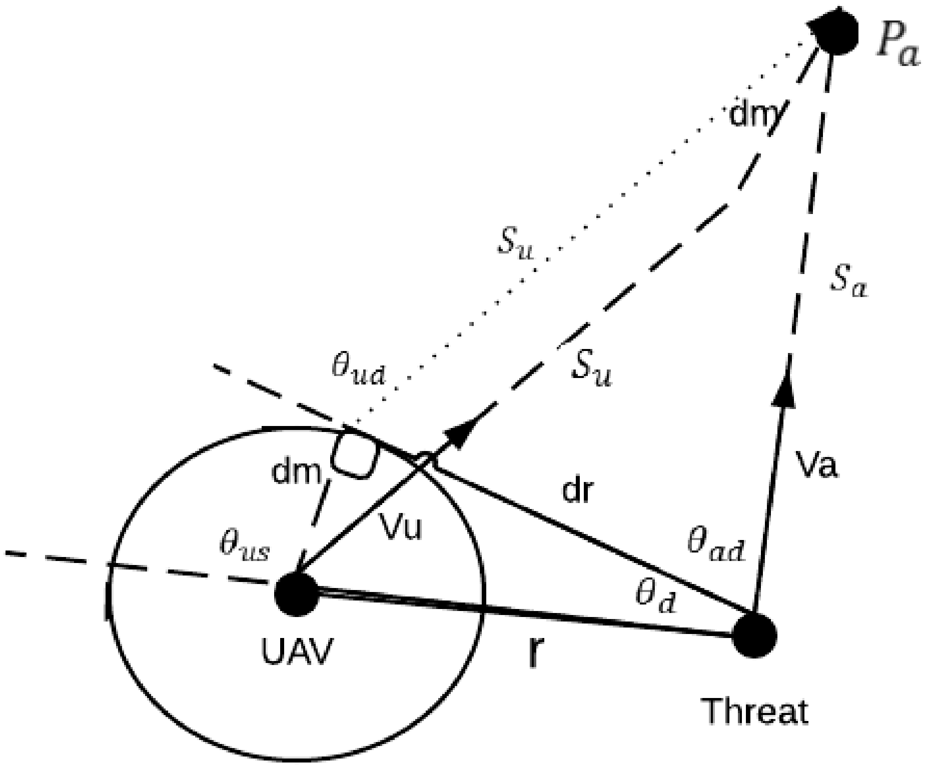


Figure B.2: Conflict resolution geometry for clockwise rotation

The matching condition is derived

$$v_u = v_r + v_a$$
$$t_u = \alpha t_m + (1/\gamma)t_a \tag{B.10}$$

The alpha is the v_r/v_u ratio. Bellow in Figure B.3 you can see the clockwise resolution for the minimum separation.

As the geometry relations between different angles and vectors are shown in figure B.3, we can derive α and d_r/s_a using cosine rules:

$$d_r/s_a = -\cos\theta_a m \pm \sqrt{\gamma^2 - \sin^{\theta am}} \tag{B.11}$$

So given γ and $\dot{r}/s_a$ in equation B.11, the velocity ration equals to

$$\alpha = (-\cos\theta_{am} \pm \sqrt{\gamma^2 - \sin^2\theta_{am}})/\gamma \tag{B.12}$$

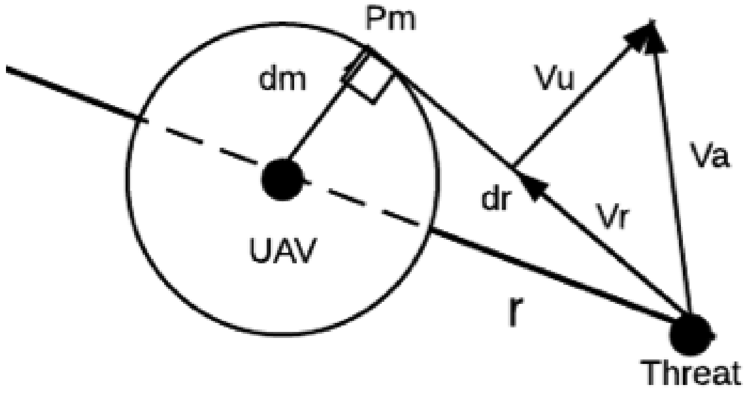


Figure B.3: Geometry for minimum separation for the clockwise solution

where $d_r = \sqrt{r_2 - d_m^2}$ and $\theta_{am} = \theta_{as} - \theta_d$. But the heading angle is not the same as the desired one in order to achieve the matching condition. So the heading angle should reach the desired one by using an algorithm to change the heading angle smoothly within a specific time. So we can define heading angle as $\theta_e = \hat{\theta_{um}} - \theta_{um}$ which means the angle difference between the desired UAV tangent vector $\hat{t_u}$ and the current UAV tangent vector t_u. In order to derive the regulating algorithm for the heading angle, we use Lyapunov function [41] and determine the time derivative of matching condition for minimum separation solution. So we have

$$-1/\gamma|\dot{\theta}_{am} \leq \dot{\hat{\theta}}_{um} \leq 1/\gamma|\dot{\theta}_{am} \tag{B.13}$$

As we assumed that the threat is non-maneuvering, we have $\dot{\theta_a} = 0$. Therefore, $\dot{\theta_{am}} = -\dot{\theta_m}$. Using this assumption and the resolution geometry for $\dot{\theta_m}$, the equation B.13 can be modified to

$$\dot{\theta_m} = (v_r/\sqrt{r^2 - d_m^2}) \sin\theta_d \mp \theta{-1}/\gamma(v_r/\sqrt{r^2 - d_m^2}) \leq \dot{\hat{\theta}}_{um} \leq 1/\gamma(v_r/\sqrt{r^2 - d_m^2}) \tag{B.14}$$

so the heading angle rate in order to reach the desired heading angle for UAV, can be derived as follows

$$\dot{\theta}_u = (1 + 1/\gamma)(v_r/\sqrt{r^2 - d_m^2})sign(\theta_e) + K\theta_e \tag{B.15}$$

Where $K > 0$ and $sign(\theta_e) = |\theta_e|/\theta_e$. The curvature of UAV can be derived from the equation B.15 and $k_u = \dot{\theta}_u/V_u$.

B.1.3.2 Approach II

In this section we consider some scenarios in which the speed of threat is greater than the speed of UAV, so the UAV should adjust its speed and heading angle to avoid the collision. The same as the previous approach there are desired speed and heading angle that UAV should have for conflict resolution. So we should investigate the velocity relation in order to determine the avoidance solution. The velocity relation is given by

$$\hat{v}_u^2 = \hat{v_r}^2 + v_a{}^2 - 2\hat{v_r}v_a \cos \pi - \hat{\theta}_{am} = \hat{v_r}^2 + v_a{}^2 - 2\hat{v_r}v_a \cos \hat{\theta}_{am} \tag{B.16}$$

so we have:

$$\hat{v}_u = \sqrt{\hat{v_r}^2 + v_a{}^2 + 2\hat{v_r}v_a \cos \theta_{am}(\text{B.17})}$$

The same as the previous section, by using a simple Lyapunov function for V and considering the stability situation, we have the speed rate and heading angle rate for UAV in order to reach the desired V and θ as follows:

$$\begin{aligned} \theta_u &= (v_r/\sqrt{r^2 - d_m^2})\theta_e + K_1\theta_e \\ \dot{v}_u &= (v_a v_r/\sqrt{r^2 - d_m^2})v_e + K_2 v_e \end{aligned} \tag{B.18}$$

Bibliography

[1] Shadab, Niloofar, and Huan Xu. "A systematic approach to mission and scenario planning for UAVs." 2016 Annual IEEE Systems Conference (SysCon). IEEE, 2016.

[2] Bachkosky, J. M., et al. Roles of unmanned vehicles. No. NRAC-03-1. NAVAL RESEARCH ADVISORY COMMITTEE ARLINGTON VA, 2003.

[3] Bortoff, Scott A. "Path planning for UAVs." American Control Conference, 2000. Proceedings of the 2000. Vol. 1. No. 6. IEEE, 2000.

[4] Civil UAV capability assessment, NASA, 2006. Interim report, http://www.nasa.gov/centers/dryden/research/civuav/index.html (last accessed 21/3/2011).

[5] Andrew P. Sage, William B. Rouse, Handbook of Systems Engineering and Management, John Wiley & Sons, 2009

[6] INCOSE Systems Engineering Vision 2020, September 2007, Version 2.03

[7] IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. September 2000

[8] Evans, Andrew R. "The hazards of unmanned air vehicle integration into unsegregated airspace." The University of York, York (2006).

[9] Aviation Handbook, FAA Regulation and Policies, 2014, http://www.faa.gov/regulations_policies/handbooks_manuals/aviation/ pilot_handbook/media/PHAK%20- %20Chapter%2014.pdf

[10] Unmanned aircraft systems operations in the US. national airspace system, Federal Aviation Administration, Interim Operational Approval Guidance 08-01, 2008

[11] Ramrez Atencia, Cristian Oliver. "Modelling Unmanned Vehicles Mission Planning problems as Constraint Satisfaction Problems." (2014).

[12] Stenger, A., B. Fernando, and M. Heni. Autonomous Mission Planning for UAVs: A Cognitive Approach. Deutsche Gesellschaft fr Luft-und Raumfahrt-Lilienthal-Oberth eV, 2013.

[13] Vachtsevanos, George, et al. "From mission planning to flight control of unmanned aerial vehicles: strategies and implementation tools." Annual Reviews in Control 29.1 (2005): 101-115.

[14] "Instrument Flying Handbook", Instrument Flight Rules (defined), Oklahoma City, OK: Federal Aviation Administration, 2008, pp. G?9, retrieved 2010-11-27

[15] Yang, Kwangjin, and Salah Sukkarieh. "3D smooth path planning for a UAV in cluttered natural environments." Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008.

[16] Tulum, Kamil, Umut Durak, and S. K. Yder. "Situation aware UAV mission route planning." Aerospace conference, 2009 IEEE. IEEE, 2009.

[17] Schild, R., and J. K. Kuchar. "Operational Efficiency of Maneuver Coordination Rules for Airborne Separation Assurance System." PROGRESS IN ASTRONAUTICS AND AERONAUTICS 193 (2001): 665-676.

[18] Lachner, Rainer. "Collision avoidance as a differential game: Real-time approximation of optimal strategies using higher derivatives of the value function." Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on. Vol. 3. IEEE, 1997.

[19] Ota, Tomoki, M. G. Nagati, and Dong-Chan Lee. "Aircraft collision avoidance trajectory generation." Proceedings of the AIAA Guidance, Navigation, and Control Conference. 1998.

[20] Bilimoria, Karl D. "A geometric optimization approach to aircraft conflict resolution." AIAA guidance, navigation, and control conference and exhibit. Reston, VA: AIAA, 2000.

[21] Kuchar, James, et al. "A Safety Analysis Process for the Traffic Alert and Collision Avoidance System (TCAS) and See-and-Avoid Systems on Remotely

Piloted Vehicles." AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit, September 20-23 2004, Chicago, Illinois. 2004.

[22] Tomlin, Claire, George J. Pappas, and Shankar Sastry. "Conflict resolution for air traffic management: A study in multiagent hybrid systems." Automatic Control, IEEE Transactions on 43.4 (1998): 509-521.

[23] A. Stenger, B.Fernando, M. Heni, Autonomous Mission Planning for UAVs, A Cognitive Approach, German Aerospace Congress, Berlin, Germany, 2012.

[24] Muhairat, Mohammad I., and Rafa E. Al-Qutaish. "An approach to derive the use case diagrams from an event table." 8th WSEAS,?Int. Conference on Software Engineering, Parallel and Distributed Systems. 2009.

[25] Data Integration Glossary, U.S. Department of Transportation, August 2001.

[26] Hayhurst, Kelly J., et al. "Unmanned aircraft hazards and their implications for regulation." 25th Digital Avionics Systems Conference, 2006 IEEE/AIAA. IEEE, 2006.

[27] Choubey, Manoj Kumar. IT Infrastructure and Management (For the GBTU and MMTU). Pearson Education India, 2011.

[28] Smith, Neill, and Thea Clark. "A framework to model and measure system effectiveness." 11th ICCRTS (2006).

[29] van der Aalst, Wil MP. "Inheritance of dynamic behaviour in UML." MOCA 2 (2002): 105-120.

[30] Goel, Ashok K., Spencer Rugaber, and Swaroop Vattam. "Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language." Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23.01 (2009): 23-35.

[31] Friedenthal, Sanford, Alan Moore, and Rick Steiner. "OMG systems modeling language (OMG SysML?) tutorial." INCOSE Intl. Symp. 2006.

[32] Chitsaz, Hamidreza, and Steven M. LaValle. "Time-optimal paths for a Dubins airplane." Decision and Control, 2007 46th IEEE Conference on. IEEE, 2007.

[33] Beard, Randal W., and Timothy W. McLain. "Implementing dubins airplane paths on fixed-wing UAVs." Contributed Chapter to the Springer Handbook for Unmanned Aerial Vehicles (2013).

[34] Dubins, Lester E. "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents." American Journal of mathematics 79.3 (1957): 497-516.

[35] Chitsaz, Hamidreza, and Steven M. LaValle. "Time-optimal paths for a Dubins airplane." Decision and Control, 2007 46th IEEE Conference on. IEEE, 2007.

[36] Kayton, Myron, and Walter R. Fried. Avionics navigation systems. John Wiley Sons, 1997.

[37] Dowek, Gilles, and Csar Munoz. "Conflict detection and resolution for 1, 2,..., N aircraft." Proceedings of the 7th AIAA Aviation, Technology, Integration, and Operations Conference, AIAA-2007-7737, Belfast, Northern Ireland. 2007.

[38] Galdino, Andr L., Csar Munoz, and Mauricio Ayala-Rincn. "Formal verification of an optimal air traffic conflict resolution and recovery algorithm." WoLLIC. Vol. 4576. 2007.

[39] White, B. A., Shin, H. S., and Tsourdos, A., "UAV obstacle avoidance using differential geometry concepts", IFAC World Congress 2011, Milan, Italy, 2011.

[40] Oh, Hyondong, et al. "Rendezvous and standoff target tracking guidance using differential geometry." Journal of Intelligent Robotic Systems 69.1-4 (2013): 389-405.

[41] Lyapunov, A. M. (1992-08-28). General Problem of the Stability Of Motion. CRC Press. ISBN 9780748400621.

[42] Shin, H. S., White, B.A., and Tsourdos, A., "Conflict detection and resolution for static and dynamic obstacles", Proceedings of AIAA GNC 2008, August 2008, Honolulu, HI, AIAA 2008-6521

[43] Angelov, Plamen. Sense and avoid in UAS: research and applications. John Wiley Sons, 2012.