# A FRAMEWORK FOR MODELING AND SIMULATION OF CONTROL, NAVIGATION, AND SURVEILLANCE FOR UNMANNED AIRCRAFT SEPARATION ASSURANCE

A Thesis
Presented to
The Academic Faculty

by

Youngjun Choi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
August 2016

# A FRAMEWORK FOR MODELING AND SIMULATION OF CONTROL, NAVIGATION, AND SURVEILLANCE FOR UNMANNED AIRCRAFT SEPARATION ASSURANCE

Approved by:

Professor Dimitri N. Mavris, Advisor
School of Aerospace Engineering
*Georgia Institute of Technology*

Professor Daniel Schrage
School of Aerospace Engineering
*Georgia Institute of Technology*

Professor Eric Feron
School of Aerospace Engineering
*Georgia Institute of Technology*

Dr. Hernando Jimenez
School of Aerospace Engineering
*Georgia Institute of Technology*

Professor John Valasek
Department of Aerospace Engineering
*Texas A & M University*

Date Approved: 8 July 2016

# ACKNOWLEDGEMENTS

First, I would like to express my deepest appreciation to my committee members. I would first like to recognize Prof. Dimitri Mavris, the director of the Aerospace Systems Design Laboratory, for his valuable advice and guidance of my research and career path. During my long journey of Ph.D. thesis writing, he has kept motivating me and continuously forced me to reach my final research goal. He also helped me discover my special hidden talent. I have been enjoying showing my hidden talent at every Christmas event. I appreciate Prof. Daniel Schrages review and critique. I am extremely grateful to Prof. Eric Feron for sharing current research trends and discussing the extension of my research and technical critics. I also thank Prof. John Valasek from Texas A& M for the technical discussion about different aspects of my research work and sharing his empirical research experiences. I would like to express my sincere gratitude to Dr. Hernando Jimenez for the immeasurable amount of support and guidance. He has spent a considerable amount of time to discuss my immature ideas to make them more constructive and concrete, and supported all of my research works. His deep insights and keen arguments helped me through all the stages of my thesis.

A special thank goes to Dr. JongKi Moon from Gulfstream who spent his time to discuss my technical problems and his work experience in person and on the phone. I very much appreciate Prof. Brian German who always encouraged me and allowed me to work in the office of the German Research Group with my colleagues who helped accelerate my research process. I would like to recognize my colleges at the Aerospace System Design Laboratory and the department of the aerospace engineering who have always discussed my research with me in the hallway, Starbucks and

my thesis presentations. I would like to name a few members. Nicky Acrockiam has always supported and motivated me, and proofread my documents, David Pate who is my thesis partner has taught Pate-factor that enhanced the quality of my dissertation, Micheal Patterson (NASA, Langley) discussed my research works in CRC, Marc Canellas has always listened to me about my new algorithms and gave great feedback and comments, and Xiaofan Fei has asked me many creative questions and critics. Andrea Garbo and Giada Abate have always supported and discussed Linux cluster issues and taught me naughty Italian expressions and words. Emre Yilmaz gave some great feedback in my pre-defense. John Dykes has told me diverse new control theories and gave me many new ideas. Erika Brimhall has proofread my thesis document when she was pregnant. Eugina Mendez Ramos has been supportive and fed me to give me energy to work. Ryan coder has always encouraged and supported me all throughout my Ph.D. process. KyungHak Choo has provided me good advice related to the qualifying test and spent fun evenings with me. All of you were always there for me and I will always appreciate your encouragement and the good we shared together.

I want to thank three professors: Professor Patrick Keogh (University of Bath in U.K.) has helped and supported my academic career. Professor Hyun-Ung Oh (Chosun University) has also helped and taught me how to survive in a professional world. Professor Minsig Kang (Gachon University) has provided me numerous advice and taught me fundamental control theories. I especially appreciate my parents, Chulho Choi and Hyunju Kim for their unconditional love and encouragement though my entire life. They have always supported my decisions and provided valuable advices. My father always emphasizes Be healthy and be happy, which made my Ph.D. life full of happy moments. I deeply thank to my grandfather, Jumsuk Choi (1927 2013) who had trusted and waited for this moment.

Lastly, I would like to extend my deepest gratitude to my lovely wife, Eunyoung

An, for her patience, love, constant encouragement and immense supports. She is my best friend, my private English teacher, and the best cook. I am really excited for our next journey with you.

Youngjun Choi

Atlanta, Georgia

July, 2016

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

Unmanned Aircraft Systems (UAS) have gained popularity and attention due to highly flexible mission capabilities and low operating costs compared to manned missions among many other reasons. These advantages have led to various mission concepts such as border control, atmospheric observation, agricultural surveys, communications relay, and surveillance missions. According to the Radio Technical Commission for Aeronautics (RTCA), the future UAS market is forecasted to grow rapidly in the near future. However, in order to accommodate future diverse UAS missions and numerous operations in the national airspace system, several key challenges must be addressed. The major technical challenges are separation assurance, communications, human systems integration, airspace operations, and regulation/certification. Among these challenges, separation assurance has received special attention and is considered to be a critical challenge since it is directly associated with human risk, highly coupled with other disciplinary domains and high degree of difficulty.

Among the Unmanned Aircraft Systems Integration problems in the National Airspace System (UASNAS), the separation assurance challenge is highly complex because of many interactions of the elements in different levels of abstraction and coupling effects between different disciplinary domains. In order to explore this complex separation assurance problem, an analytic model should capture diverse operational scenarios, vehicle dynamics, subsystem functions such as sensor/surveillance, control, navigation and communications, and interactions between various levels of abstraction and different disciplinary domains. This has major implications on the analytic model requirements, particularly with regard to modeling scope, resolution

(or fidelity), and computational expense.

In response to the complex separation assurance problem, this thesis aims to develop Unmanned Aerial Vehicle (UAV) modeling and simulation capabilities for a non-cooperative collision avoidance problem with a ground obstacle, utilizing flight attitude control and guidance, navigation, and control. Second objective is to quantitatively characterize the performance of the non-cooperative collision avoidance as a critical element of separation assurance with regards to system behavior across levels of abstraction and multiple disciplines

To address the first objective, firstly, for the flight attitude control, a new gain-scheduling approach is proposed to mitigate the computational drawback of using conventional methods for real-time flight simulations. The main idea of this proposed method is to create response surface models for trim-inputs and the control gain-set during pre-processing instead of implementing local interpolation, which has been done in conventional gain-scheduling techniques. This a-priori model creates simple functional forms of the trim control inputs and the control gain-set so that it enables a simpler and more computationally efficient gain-scheduling scheme over conventional gain-scheduling methods.

Second, Model Predictive Control (MPC) structure is implemented for a path planning against an obstacle. The MPC structure provides an optimal trajectory based on approximated dynamics, constraints and obstacle information from a sensor. Formulating a simple optimal trajectory problem in MPC structure is an enabler for a fast or real-time trajectory optimization to mitigate computational complexity. However, there is a tendency for the formulated trajectory problem to be oversimplified, which is highly likely to increase optimal trajectory cost. To overcome this issue, this thesis proposes a hybrid optimal collision avoidance methodology using a machine learning technique. The main idea of this proposed methodology is to identify the best avoidance strategy among pre-identified multiple simplified collision

avoidance algorithms through a machine learning technique. During a real-time process with this methodology, the best collision avoidance strategy is selected based on the initial/terminal conditions and sensor information. This methodology will allow the optimal avoidance trajectory to find lower trajectory cost as well as to improve the required computational time.

Third, a two-layer obstacle avoidance algorithm is proposed for a multi-obstacle problem. This two-layer structure allows an unmanned aircraft system to avoid upcoming multiple obstacles with minimal effort. The algorithm includes a global-path optimization that identifies possible approximated avoidance paths from a clustering technique based on obstacle information detected from an airborne sensor, and then selects a path. A local-path trajectory optimization has the same structure of a model predictive control structure with a multi-phase optimal trajectory resulting from approximated dynamics, vehicle constraints, and sensor information. Unlike the conventional on-layer optimal obstacle avoidance algorithm, this proposed two-layer optimal obstacle avoidance algorithm can generate more energy-efficient avoidance trajectory against multiple downstream obstacles.

Forth, a rapid, data-driven and grid-based urban operating urban modeling methodology is proposed for an urban model construction. For the exploration of the UAS urban operation problem, a rapid and realistic urban modeling is a key technique. Although many methodologies in a computer science domain have been introduced, they are highly sophisticated and infeasible for the UAS problem because of computational burdens. To resolve this issue, a novel urban modeling methodology using airborne LiDAR (Light Detection and Range) data is proposed which includes multiple steps: resampling and refining LiDAR raw data, identifying building components, solving a principal component analysis, defining a grid resolution, and generating an entire urban model. The proposed urban modeling technique as a rapid and automatic process enables the exploration of the diverse UAS urban operation scenarios.

Lastly, an integrated experiment is introduced which includes aircraft dynamics, an aircraft controller, an obstacle avoidance algorithm and an urban environment. Using the developed simulation environment, we explore a canonical UAS problem (i.e., collision avoidance problem in San Diego downtown), analyzing sensitivity and interactions between a sensor system and a GNC system in an urban operation. The experiment results are discussed with respect to three perspectives: risk taker, risk averse, and risk nominal. Diverse sensitivity and interaction analyses are performed by various statistical techniques. The results of the representative scenario constitutes a contribution to a strategic decision making process with regards to different risk standards through the sensitivities and interactions.

The research efforts of this thesis enable a full exploration of the UASNAS problem, specifically the obstacle avoidance problem in an urban environment. This new modeling and simulation environment provides insights into coupling and cross-coupling effects between systems, subsystems, or different levels of abstractions that cannot be characterized by a conventional modeling and simulation environment. This simulation environment and introduced analysis methods facilitate the exploration of diverse scenarios and various UAS platforms that allow a strategic decision maker to fully understand the relationships of each system/subsystem component. The information of this research may contribute to a full integration of UAS into NAS in the near future.

# CHAPTER I

# INTRODUCTION

## *1.1 Growing market and applications of UAS*

In recent years, Unmanned Aerial Vehicles (UAV)/drones have gained attentions from commercial investors/companies because of their highly flexible mission scenarios, a low operating cost and a low human risk level [79][78]. These attractive benefits led Google to buy Titan Aerospace, a UAV technology company, to proliferate Internet access around the world as well as to assist global issues, such as monitoring environmental variation and disasters [144]. Other than Google, any leading delivery companies, such as DHL, Amazon and United Parcel Service (UPS), also have invested a large amount of money to develop new types of delivery drones to reduce delivery time and to become a pioneer of a new delivery market [44][8]. The use of UAV operations have been expanded by being utilized not only by commercial companies but also by civil governments [128]. For instance, the Raven UAV supports the observations of suspect cars in the Los Angeles region. The ScanEagle platform monitors illegal fishing activities in the Dry Tortugas National Park. The Global Hawk tracks the movements of tropical storms or hurricanes along the African coast by conducting research on strength variation of the storm and by forecasting their future trajectories. The examples of current and growing UAV investments and usages indicate that in the future, a variety of UAV missions and platforms will be utilized for military, civil government and commercial applications. Thus, there will be a need for the advent of new technologies and technology improvements as well as the creation of new regulations/certification rules for UAV operations.

The growth of UAV platforms and Unmanned Aircraft System (UAS) missions

requires a new rationalized classification or categorization with regard to the UAV performance, operational scenarios, and physical characteristics. The existing classification has limitations to gain a systemic understanding about UAS Integration in the National Airspace System (UASNAS) [57] issues because the existing classification/categorization is based on physical characteristics of manned aircraft. Therefore, the existing methods do not adequately account for unique UAV platforms, such as airships and High Altitude Long Endurance (HALE) platforms.

With the remarkable feature of remote/unmanned control, UAS can conduct a broad range of missions which manned aircraft cannot fulfill due to pilot risk. Notable examples include monitoring missions for a nuclear power plant and surveillance missions by a high altitude airship. Despite the diverse potential uses of UAS, it cannot be fully utilized because of the existing classification, which cannot encompass the entire range of UAS mission scenarios.

A new classification/categorization for UAS is therefore a necessary enabler for evaluating interoperability issues and gaining a systemic understanding of the potential impacts of UAS on the National Airspace System (NAS). However, a standard classification/categorization for UAS does not exist due to a lack of consensus among stakeholders [101]. Most classifications are based on UAV physical characteristics, user classes, mission purposes and operation concepts. Thus, this thesis introduces some key classification methods for a better understanding of UAV operation concepts and diverse missions.

The first classification method is breaking down UAS missions according to user class; these classes include military users, public users, commercial users and private users [9]. The military class includes UAS utilized for military purposes such as reconnaissance and surveillance missions by Predator B [128] and communications relay [37]. The public class is supported by government entities, which include the Federal Aviation Administration (FAA), the Department of Homeland Security (DHS), the

**Table 1:** Current and Potential UAS Applications [9][4]

| Military class | Public class | Commercial class |
|---|---|---|
| Reconnaissance/Surveillance | Atmospheric research | Fish spotting |
| Tactical strike | Border patrol | Remote imaging and mapping |
| Communications relay | Disaster response | Utility inspections |
| Signals intelligence | Hurricane tracking | Mining exploration |
| Maritime patrol | Forest fire monitoring and support | Agricultural applications |
| Penetrating strike | Search and rescue | Communications relay |
| Integrated strike/SEAD | Maritime surveillance | Petroleum spill monitoring |
| Aerial refueling | Law enforcement | Site security |
| Counter air | Humanitarian aid | News media support |
| Airlift | Aerial imaging and mapping | Filming |
| | Drug surveillance and interdiction | Real estate photos |
| | Monitor critical infrastructure | Aerial advertising |
| | Natural hazard monitoring | Cargo |
| | Airborne pollution observation | Crop monitoring |
| | Chemical petroleum spill monitoring | Broadcast services |
| | Communications relay | |
| | Traffic monitoring | |
| | Port security | |

Department of Commerce (DOC) and others. Notable operational scenarios in the public class include scientific missions (e.g. hurricane observation by Global Hawk) and monitoring missions (e.g. coal emission monitor by Aerosonde) [128]. UAS used for business purposes are put into the commercial class. The Amazon and DHL delivery systems by UAV are included in this commercial class. The private class includes all UAS used for private operations such as recreation or UAV competitions. Table 1 summarizes the current and potential applications based on user class.

Another method of UAS vehicle classification is based on physical characteristics and mission features, as shown in Table 2. The types of UAVs are divided into seven groups: nano, micro, small UAS, ultralight aircraft, light sport aircraft, small aircraft and medium aircraft. The FAA classification has four groups and does not include nano, micro or small UAS. Depending on the stakeholders, the classification can be slightly different, but most communities have a similar classification structure.

These classifications are the representative formats of UAV classifications. However, all the applications, missions and types of platform shown in the introduced classifications do not currently exist because of immature technologies and regulation/certification issues. Next, we will discuss UAS market growth in order to observe

**Table 2:** UAS Vehicle Classification [9]

| Platform Type | Weight ($lbs$) | Overall Size ($ft$) | Mission Altitude ($ft$) | Mission Speed (mph) | Range ($Miles$) | Endurance (hrs) |
|---|---|---|---|---|---|---|
| Nano | $< 1$ | $< 1$ | $< 400$ | $< 25$ | $< 1$ | $< 1$ |
| Micro | $1 \sim 4.5$ | $< 3$ | $< 3000$ | $10 \sim 25$ | $1 \sim 5$ | 1 |
| Small UAS | $4.5 \sim 55$ | $< 10$ | $< 10000$ | $50 \sim 75$ | $5 \sim 25$ | $1 \sim 4$ |
| Ultralight Aircraft* | $55 \sim 255$ | $< 30$ | $< 15000$ | $75 \sim 150$ | $25 \sim 75$ | $4 \sim 6$ |
| Light Sport Aircraft* | $255 \sim 1320$ | $< 45$ | $< 18000$ | $75 \sim 150$ | $50 \sim 100$ | $6 \sim 12$ |
| Small Aircraft* | $1320 \sim 12500$ | $< 60$ | $< 25000$ | $100 \sim 200$ | $100 \sim 200$ | $24 \sim 36$ |
| Medium Aircraft* | $12500 \sim 41000$ | TBD | $< 100000$ | TBD | TBD | TBD |

* FAA-Defined Manned Aircraft Weight Categories

the UAS market trends, current UAS integration issues and technological challenges.

UAS Research and development began by the U.S. military in the early 1900s [50]. However, it was not until the mid-1900s when the first UAS (Firebee) was actually flown in combat in Vietnam War. Around the early 2000s, UAS was deployed in several wars, such as Kosovo, Iraq and Afghanistan for tactical purposes. These tactical operations for military purposes have proved the combat effectiveness of UAS. The first advantage of UAS operation is that without a pilot on board, the UAV is able to achieve a longer endurance, a wider flight envelope, a lighter weight, a smaller structure and relaxed maneuvering constraints. Second, the absence of a pilot on board in the UAV protects against pilot risks in dangerous missions. Third, the UAS procurement cost is lower than the acquisition cost of manned aircraft. For these reasons, the investment on UAS development in the United States has dramatically increased starting in the year of 2000, as presented in Figure 1. Despite this trend and these characteristics of UAS, the current UAS investment is mainly limited to military purposes because of regulation/certification issues and technology limitations [128].

According to the forecasting research by the U.S. Department of Transportation [9], both commercial and public (including military) UAS usages will continuously increase. This report also states that the size of the UAS fleets of the federal public agencies will reach approximately 10,000 vehicles by 2035, compared to only a few hundred in 2015. Commercial UAV use will also have a radical growth after 2025, and

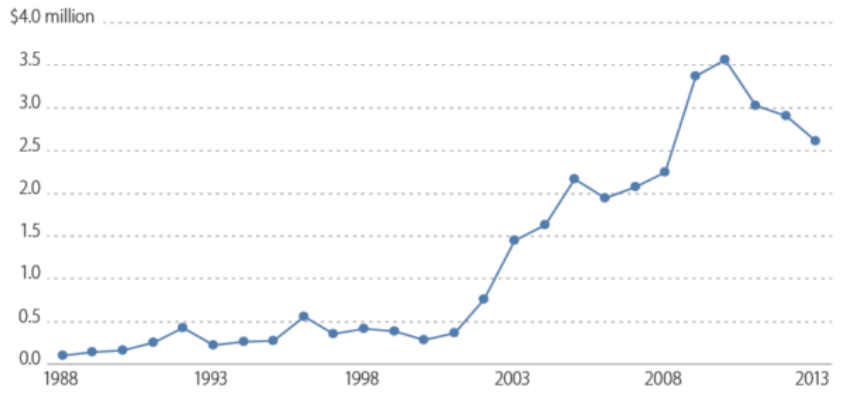



**Figure 1:** UAS Budget 1998 - 2013 [50]

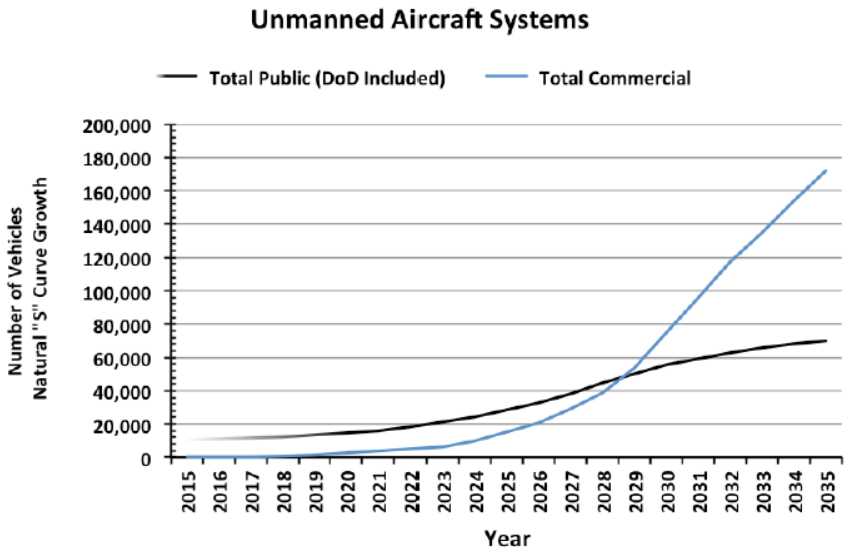



**Figure 2:** Forecast Result of Total UAS [9]

the number of commercial UAS will surpass the total number of public UAS between 2028 and 2030.

Nevertheless, this growth pattern of UAS will not be possible without overcoming several key issues. The first issue is that the current accessing scheme to the National Airspace System for UAS is too limited to utilize a wide range of missions and numerous UAS platforms. The current scheme is based on two certification processes: Certificates of Waiver or Authorization (COA) and Special Airworthiness Certificates, Experimental Category [5][48]. The COA process utilizes for public operations

by public agencies and institutions. The special airworthiness certificate, experimental category, is applied for UAS operations within an assigned test area. Although this process is the current enabler for UAS to access the NAS, this process is not sufficient to accommodate the future growth of UAS with a wide range of missions and numerous UAS platforms [40]. In other words, this COA process is not a long-term solution but rather a short-term solution. Second, there are technology barriers such as communication technologies for ground-to-ground, Air Traffic Controller (ATC) to a pilot/remote operator, Separation Assurance, high precision sensor technologies and collision avoidance algorithms. Third, regulation/certification is needed to facilitate the integration of UAS into the NAS. In the next section, we will discuss the role of each stakeholder and specify the major challenges.

## *1.2 Integration of UAS into the NAS as an emergent imperative*

Integration of UAS into the NAS is imperative for several reasons. First, according to the flight hours of unmanned aircraft in a DoD report [40], in 2011, there were more than one million UAS flight hours, and the number of UAS sorties surpassed the number of manned aircraft sorties. This report also states that for military missions, UAS usage is estimated to replace most manned aircraft usages in the near future. Second, in spite of the current limited UAS operation in civil government missions such as border patrol and disaster response, the forecasting research indicates that most current operations of manned aircraft will shift to UAS operations because of the potential benefits in terms of human safety, operation/acquisition cost and the possible variety of missions [9]. Third, the commercial UAS demand will likely increase due to various potential markets for UAS [128]. However, these optimistic forecasts about the increase in UAS operations will be fulfilled only when all key challenges and barriers are solved. This section will provide an overview of the stakeholders and their roles to provide an understanding of which organizations are involved and

what research activities are being performed in each organization. In addition, key challenges/gaps found through a survey of relevant literatures will be discussed.

The core stakeholders are the National Aeronautics and Space Administration (NASA), the Federal Aviation Administration (FAA), the Joint Planning and Development Office (JPDO), the Department of Defense (DoD) and the Department of Homeland Security (DHS) [76]. NASA is committed to solving civil UAS integration and addressing key technical challenge areas, assessing technologies, identifying standards and gaps, and identifying required research fields [112]. JPDO is working on defining the Next Generation (NextGen) air transportation system and coordinating all research activities, plans and goals with other stakeholders for NextGen integration [7]. The FAA is building or revising standards/regulations, as well as certification and operational procedures so that UAS can achieve an acceptable level of a safety [48]. The DoD's responsibility for UASNAS integration is supervising the direction of airworthiness and UAS pilot/operator training [39].

Many reports have been published by stakeholders to address the challenges of UAS integration in the NAS [6][39][112][76][7][48]. Based on the literatures, the major challenges can be divided into two categories, non-technical challenges and technical challenges.

In the non-technical area, the NextGen UAS Research, Development and Demonstration Roadmap highlight the issue of sharing research information and coordinating research activities among stakeholders. The research work done by each organization is performed concurrently for their specific communities. These concurrent activities make the UASNA problem difficult to examine the coupling effects between the researches performed by different organizations. For instance, the FAA needs to collaborate with other entities to decide on regulations/certification rules for safe UAS operations in NAS because the regulation/certification process should consider

the characteristics of all types of vehicles, concepts of operations, and the technologies' capabilities. However, sharing this information with other entities is practically impossible due to the concurrent development structure. This process causes overlapping research in research entities, an increase in the cost and a duplication of effort, which leads to the degradation of R&D efficiency.

In the technical area, the major issues are Communications, Separation Assurance (SA)/Self Separation (SS)/Collision Avoidance (CA), Human Systems Integration (HSI), airspace operation and regulation/certification.

Communications challenges include several critical issues. The first issue is that no methodology exists to characterize the impact of the UAS communications system on the current Air Traffic Control (ATC) and the communications system of the UAS platform. The second challenge is the communications frequency allocation issue to protect the safety of the frequency spectrum. Another challenge is defining communications requirements and building a validation process to investigate the integrity of the whole communications system.

The Separation Assurance (SA) area also includes some technical challenges. First, the sensor technologies are not mature. The existing sensor technologies, such as Lidar, radar, and vision sensors have advantages and disadvantages in terms of cost, operational environment (such as weather conditions), and sensor range. Nowadays, to overcome the sensors' shortcomings, sensor fusion technologies have been widely researched. However, there is no standard rule about sensor fusion technologies for separation assurance. Second, the collision avoidance algorithm has a large technical gap. UAS platforms are very diverse; examples include airships, quad-copters and UAVs with distributed propulsion. Therefore, the collision avoidance algorithm should consider the diverse UAV platforms and a variety of sensor systems.

Human Systems Integration (HSI) is also one of the major challenges in UAS integration. Depending on the level of autonomous capabilities (fully autonomous,

semi-autonomous, passive autonomous), the human impact varies on UAS operation with respect to safety level. However, no quantification methodology exists for the exploration of the impact of human interaction. Second, the visualization tools for the weather and air route information, as well as the separation assurance interface to the pilot or remote pilot, are issues, since the efficiency of the visualization system can influence the level of safety in the UASNAS problem.

The Integration of Civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS) Roadmap and the UAS Integration in the NAS Project both address standards/regulations/certification and airspace operation issues. Due to the characteristics of unmanned aircraft, UAS cannot follow the sense and avoid" rule (FAR 91.159) of the national airspace system. This feature complicates the structure of UAS operations. UAS operations require more system elements, such as a communications relay, a control station and a remote pilot to provide information about upcoming obstacles to the UAV platform so that the UAS can perform an avoidance mission in order to satisfy the same level of safety required by the current sense and to avoid the rule for manned aircraft. However, the greater number of system components and interactions between systems in a UAS make the avoidance mission a more difficult problem to design standards/regulations/certification processes. Moreover, UAS operation data do not currently exist. The lack of UAS flight data result in a more challenging problem of developing regulations without understanding the fundamental features of UAS and the interactions between systems.

The airspace operation issue includes integration issues such as automation roles and responsibilities between manned aircraft and UAVs, or between UAVs. Another challenge is that there is no existing analysis approach to evaluate the level of safety in airspace operation.

All in all, UAV integration is absolutely essential due to a large and increasing demand/market for UAS, according to the forecasting research. In order to meet

the demand, there are five key challenges (Communications, Separation Assurance, Human Systems Integration, airspace operation and regulation/certification) from the literature provided by the stakeholders. These major challenges have many sub-challenges which must be solved for successful UAS integration into the NAS.

## *1.3 Characterization of the problem space*

This section introduces key characteristics of the UASNAS problem space to have a better understanding about the UASNAS problem and achieve a systemic approach to solve the UASNAS problem. Before describing the features of the UASNAS problem space, some fundamental definitions associated with the UASNAS problem will be introduced.

According to the Defense Acquisition Guidebook, a system of systems is defined as a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities" [38]. The UASNAS problem is a system of systems problem since it has a set or arrangement of integrated systems.

Next, a system is defined as a functionally, physically, and/or behaviorally related group of regularly interacting or interdependent elements" [35]. The UASNAS problem has several systems, such as UAVs, other airplanes, air traffic controllers, ground stations and communications relays.

The system is composed of subsystems. According to the NASA Systems Engineering Handbook [3], a subsystem is a system in its own right, except it normally will not provide a useful function on its own, it must be integrated with other subsystems to make a system". An example of the subsystem in the UASNAS problem is that a UAV has several subsystem components such as a propulsion system, a guidance, navigation and control system, a sensor system, and a communications antenna.

Based on the basic definitions, three UASNAS features will be discussed. The

first characteristic of the UASNAS problem is coupling effects among many levels of abstraction. For instance, in the separation assurance problem, the UAV system performance parameters, such as maneuverability, can affect the level of safety of the UAS integration. Furthermore, the subsystems (e.g., a satellite communications antenna, a propulsion system, sensor technologies, and a guidance and navigation control system) have a significant impact on the separation assurance capability with respect to safety. The second feature of the UASNAS problem is that many disciplines are highly coupled. As an example of the airspace operation discipline, the performance of the communications can impact the safety of the airspace operation, and the performance of the separation assurance can influence the safety level of the airspace operation as well. Similarly, the degree of human systems integration can contribute to the safety level of the airspace operation. The last characteristic of the UASNAS problem is that the disciplines and levels of abstraction are cross-coupled. That is, the separation assurance capability is influenced by different systems, such as the communications relay, the air traffic controller, the ground station and others. The subsystems of various systems can have a primary effect on determining the safety level in separation assurance as well. The performance and characteristics of these systems and subsystems affect the capabilities in other disciplinary domains.

Note that the UASNAS problem is highly complicated due to the extensive coupling and cross-coupling impacts among different levels of abstraction and the various disciplinary domains. Consequently, a new environment for trade-off studies is necessary which will enable the designer to explore the impact of the interactions between the measures of performance, sometimes in different levels of abstraction and different disciplinary domains.

## *1.4 How is it done today?*

In the previous section, the characteristics of the UASNAS problem space were observed. To explore the UASNAS problem precisely, a new trade-off study environment is required. In this section, current existing trade-off study environments will be examined to identify the gaps.

There are two approaches in order to observe the UAS impact on the NAS: using a real and physical experiment, or using an analytic method (modeling and simulation environment). A key benefit of using real experiments is that one can collect actual flight data and investigate more accurately the actual interaction among the systems or subsystems. Recently, the FAA announced six test sites, which are located in Alaska, Nevada, New York's Griffiss International Airport, North Dakota, Corpus Christi and Virginia [57]. These test facilities were selected to cover diverse locations in different climatic zones and achieve geographical diversity. Each test site has a specific goal to solve in the UASNAS integration problem. The Alaska region has the role of investigating standards for UAS categories and state monitoring and navigation. The Nevada test site will study UAS standards/certification and operator standards. The integration impact on NextGen will be explored in Nevada. The test site located in New York will provide verification and validation research on how to integrate UAS into the congested NAS. The North Dakota area has a plan for the evaluation of technologies' reliability and human factors. The site in the Texas region will review operational procedures and protocols from an airworthiness perspective. Lastly, the Virginia test site will conduct UAS failure experiments to specify the risk areas from technological and operational perspectives. These six test sites will address key challenges such as technologies, operations, safety and human systems integration.

However, the real experiments performed in those sites have limitations in examining the UASNAS problem for several reasons. The first reason is that the experiments

to explore the coupling effects between different disciplinary domains are difficult. Each test site has individual research goals for UASNAS integration. Although each experiment site may produce limited exploration outcomes for the studies of the coupling effects, it is impossible to examine all the coupling and cross-coupling impact in the entire UAS domain. Second, some critical scenarios which can impact human risk directly cannot be implemented through a real experiment. For instance, failure scenarios near a terminal or urban area are impossible to be tested in a real experiment due to the high risk of human safety. Another reason is that it is impossible to perform experimental demonstrations of all types of vehicles or all possible flight scenarios, including emergency scenarios. Conducting experiments on all the diverse mission scenarios and numerous types of vehicles would result in a high cost and a long schedule. Hence, to investigate the various mission scenarios and the impact of the diverse UAV characteristics, an analytic method is a suitable approach.

The representative analytic models include both a high-fidelity model and a low-fidelity model. The high-fidelity model commonly includes detailed flight dynamics like six Degree of Freedom (DOF) and specific sensor models such as a vision sensor, LiDAR sensor, radar sensor and/or sonar sensor. The typical purpose of the high-fidelity model is to validate the proposed aircraft controllers and guidance, and navigation algorithms. Watanabe et al. proposed a collision avoidance algorithm using a vision sensor model [150]. To validate the proposed method, a flight simulation with six DOF vehicle dynamics as well as a detailed vision sensor model, were implemented. Park has also proved the efficiency of obstacle avoidance approach by applying a collision cone through six DOF quad-rotor dynamics and a specific stereo vision model [117]. Another example of a high-fidelity model is that of Salmah, who applied six DOF dynamics to a Model Predictive Control (MPC) approach to validate a collision avoidance method [131]. However, this high-fidelity simulation model is not the most favorable approach for a broad scope analysis for

scenarios such as formation flights, multi-aircraft surveillance and air traffic control since running the high-fidelity model requires a large computational expense. Due to the expense, many modeling and simulation environments with a broad scope choose a low-fidelity model. Agent-based models are well-suited to simulate large portions of the airspace with a multitude of interacting entities while typically ignoring aircraft dynamics altogether. A general implementation assumes a point mass and specifies value ranges for parameters such as velocity, altitude and turning radius, but it can nonetheless become an effective method in examining airspace operations as shown in Ref. [87] for UAS in-transit operations. Encounter models have even lower fidelity because they do not explicitly model the agent interactions, although they have been used to study self-separation function thresholds [151]. A live virtual constructive distributed test environment developed by NASA is a trajectory-based simulation environment with point mass dynamics in order to examine safety and operational challenges [103][109][120]. However, these low-fidelity environments do not provide enough information to address different levels of systemic abstraction or the underlying coupling between the measures of the performance of subsystems, systems, and systems-of-systems.

The UASNAS problem requires an exploration of coupling and cross-coupling effects with respect to different levels of abstraction and different disciplinary domains. The analytic model has to enable a trade-off environment where it is easy for multiple scenarios to be explored with various platforms. Consequently, a favorable balance of breadth, depth and cost is necessary for the modeling and simulation tools.

## *1.5 Identification of a critical area*

The key challenges for UAS integration in the NAS are specified in the previous section. The major technical challenge areas are separation assurance, communications, airspace operation, certification/regulation and human systems integration.

Among the key challenge areas, separation assurance is the most critical area. Many researchers indicate that separation assurance, including self-separation/collision avoidance, is the most critical challenge since it is directly associated with safety issues such as risk of human life and property [99][32][17][51][89].

An FAA report [10] also indicates that See and Avoid" is a significant barrier because of the potential problems resulted from the absence of pilots on board in the UAV, and the immature technologies that are not capable of detecting and avoiding fixed obstacles or moving aircraft/obstacles. Gayer et al. [51] mentions the issues of detection sensor technologies, such as acoustic sensors, radar sensors, LiDAR sensors and vision sensors. These sensors have some limitations. For instance, the vision sensor cannot detect obstacle information under severe weather conditions. The acoustic sensor can acquire obstacle information under the bad weather conditions but is not able to identify low-speed obstacles. Whereas the recently-developed LiDAR sensor has a remarkable accuracy and has outstanding performance under the bad weather conditions, it is not a viable technology for small UAV applications due to its high cost. Due to these drawbacks and limitations of the existing sensor technologies, much research about sensor fusion technology has recently been conducted to overcome sensor drawbacks/limitations.

The current paradigm of separation assurance for unmanned aircraft builds upon that for manned traffic and is structured as a multi-layered framework with overlapping capabilities. It is generally conceived with five main layers: regulatory structure, strategic separation, tactical separation, self-separation, and collision avoidance [89][32]. The regulatory structure includes operating procedures, and airspace interaction rules. Strategic separation addresses separation assurance and conflict resolution with time horizons roughly between 3 minutes and 10+ minutes, and is exercised by air traffic management services. Tactical separation is concerned with separation and conflict resolution with timescales between 2 minutes and 5 minutes, and is managed

by air traffic control services. Self-separation is co-managed by air traffic control and on-board systems for separation assurance with closure margins between 15 seconds and 2 minutes. The last separation assurance layer is collision avoidance that is operated by on-board systems for conflict resolution within 15 seconds approximately. The detect and avoid capability of an unmanned aircraft is a key enabler for self-separation and collision avoidance. Separation assurance across its various layers is a top-level operational capability in the integration of unmanned aircraft systems into the national airspace since it is coupled with critical operational elements and challenges such as sensing/surveillance, communications, functional allocation, and human factors.

## *1.6 Objective of thesis*

The primary goal of this research is to develop an analytic modeling environment for the UASNAS integration problem, specifically for a collision avoidance case under non-cooperative (lost-link) scenarios with a ground obstacle. To meet the overall research objectives, this thesis will accomplish the following tasks:

- Objective 1: to study and develop improvements in modeling and simulation of fully integrated UAS to address the current gaps and to enable systems analysis across levels of abstraction and multiple disciplines

- Objective 2: to quantitatively characterize collision avoidance as a critical element of separation assurance in terms of system behaviors across the levels of abstraction and multiple disciplines

# CHAPTER II

# STATISTICAL GAIN-SCHEDULING METHODS FOR AIRCRAFT FLIGHT SIMULATION

Controllers in the aircraft dynamics model are typically approached with a gain-scheduling method or with a non-linear control method using dynamic inversion by adaptive neural network structure [84].

Gain-scheduling methods are more commonly used mainly due to the reduced computational burden associated with the local linearization of model dynamics inherent compared to other nonlinear approaches [129]. In a general aviation class, because of the small variation in the cruise flight condition, the controller commonly utilizes a fixed-gain approach, but the controller in a small UAS class requires the gain-scheduling structure because of high dynamics variation resulting from agile maneuvers. Gain-scheduling is typically included as part the stability augmentation system (SAS) that define control settings to follow the path trajectory defined by the navigation or autopilot system. The gain-scheduling approach transforms the non-linear aircraft dynamics into a linear time invariant (LTI) equation for given trim conditions and then optimally solves for the feedback gain set. The process is repeated for points of interest within the flight envelope so as to produce a finite one-to-one mapping between operating points and corresponding trim control input and gain values. A scheduling scheme utilizes the a-priori data to generate trim control input and gain values for any operating point within the flight envelope during the simulation. Different gain scheduling schemes with varying degrees of complexity exist. The nearest neighbor approach assigns the trim control input and gain values of the closest a-priori point. Interpolation and blending techniques are also commonly

employed [104]. Gain scheduling presents some inherent shortcomings that have been noted in the literature. The controller can exhibit poor robustness and stability due to deficient gain approximations [73]. Higher order effects in dynamic behavior can be significant but are inherently absent in a linearized model. As a result all higher-order effects, such as the non-linear actuator response or phantom yaw caused by asymmetric vortices, introduce uncertainty and error to the model [155]. To address the issue of higher-order contributions an augmented control structure with adaptive controller has been proposed that can produce more precise predictions of the aircraft dynamics [69]. As can be expected these improvements are attained with increased computation cost and greater complexity of the aircraft controller [130].

We propose a gain-scheduling approach to improve performance and address salient computational drawbacks of conventional gain-scheduling methods. A polynomial regression model is generated a-priori and used in place of nearest neighbor or bivariate interpolation schemes. The polynomial regression model provides accurate trim input solutions and control gain set estimates with a computationally efficient functional form that improves the cost of the overall simulation. Accordingly, we hypothesize that the proposed method offers improvements in computational cost without degradation of controller stability, relative to nearest neighbor and bivariate interpolation gain scheduling methods. In order to test our hypothesis and demonstrate the proposed gain-scheduling approach numerical analysis using flight simulation is performed and the results compared against the two aforementioned conventional gain-scheduling algorithms. This section is written based on the published journal paper [30].

## *2.1 Flight modeling and simulation environment*

In this section, a general flight simulation environment will explore to identify challenges for describing coupling effects and cross-coupling effects between different levels

of abstraction and diverse discipline domains. A general flight control simulator shown in Figure 3. The flight simulation environment is composed of several components. The first component is a guidance and navigation loop, which provides a trajectory based on waypoints determined by a mission scenario. The next component is a flight attitude loop to generate control inputs to follow the trajectory computed in the previous step(guidance and navigation). Another simulation part is a dynamics model including all kinds of system dynamics such as actuator dynamics, vehicle dynamics and others. The simulation environment has a database which have all vehicle information about aerodynamic characteristics, propulsion system specifications and a vehicle weight. The other model of a flight simulator to handle a collision avoidance problem is a sensor model like Radar, Lidar and Vision sensors. The sensors gives an obstacle information to the guidance and navigation loop to update a collision free trajectory. The last component is a world model. The world model has all information about atmosphere and obstacle information such as a size of obstacle and location information.

This thesis focuses on improving the guidance and navigation loop and the aircraft controller since the collision avoidance performance level is highly dependent on the these two loops. These guidance and navigation, control loops highly impact on the level of the flight simulation time, which is a critical component in a fast or real-time simulation environment. If guidance and navigation, control logic has a high complexity model, it would require higher computational expense. On the other hand, these two loops has a simple model it would have some degradation in terms of a control performance, but it leads to run a mission faster. The purpose of the thesis is maintaining the middle level of fidelity and reducing a computational expense of an existing method so that it enables to explore system of systems problem in multiple discipline domains. This thesis will explore the existing UAV modeling and simulation environment to specify the features as a fast time simulation environment and based

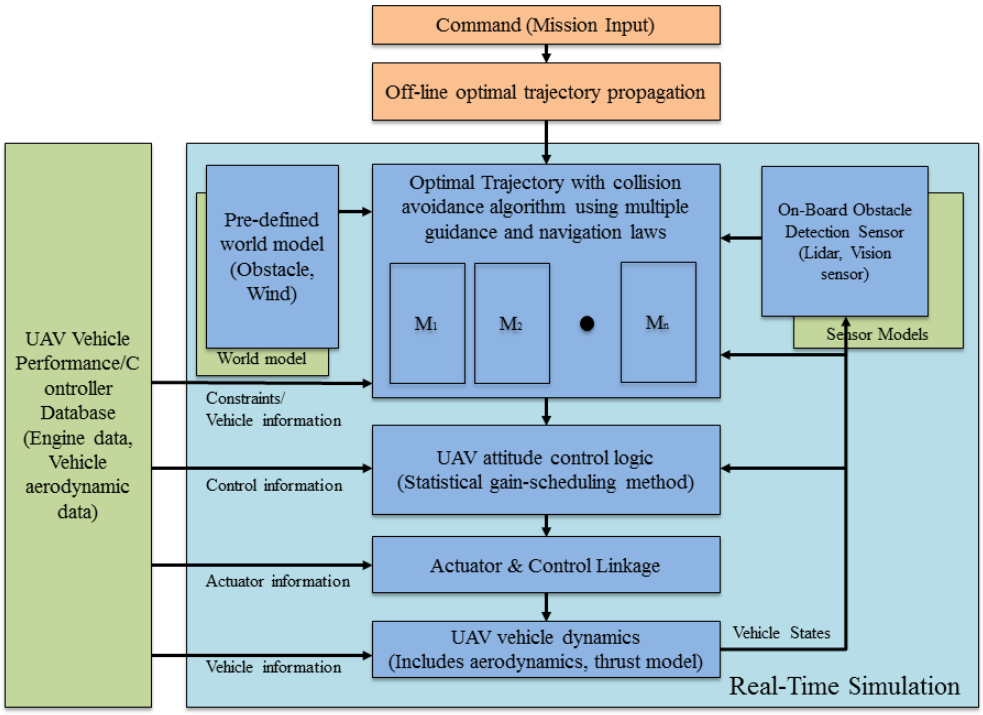



**Figure 3:** UAV flight simulation environment

on that observation new approaches are proposed.

This section focuses on the aircraft controller to improve the computational run-time. The remainder of this chapter is follows: Section 2.1.1 provides about an equation of the UAV motion. Section 2.2 overviews a conventional gain-scheduling approach and elaborates drawbacks/shortcomings with respect to a real-time simulation environment. Based on the observation, a new gain-scheduling is introduced to overcome shortcomings. To demonstrate the proposed gain-scheduling method, numerical simulation and experiment results are discussed.

### 2.1.1 Equation of UAV motion

There is rich diversity in the platform architectures of unmanned aircraft; numerous planform-propulsion variants exist for fixed wing aircraft, rotorcraft, and airships. In addition, unconventional concepts such as hybrid wing-body or multi-rotor aircraft

are much more pervasive for unmanned applications. This variety presents inherent burdens and difficulties in the development and treatment of flight dynamics and control for unmanned aircraft.[94][36] A conventional propeller-driven fixed wing architecture is selected for simplicity and in consideration of the large portion of unmanned vehicles that it applies to. [94] [102] With a point-mass assumption, the free-body diagram for the aircraft is as shown in Figure 4 and the equations of motion can be stated as follows:

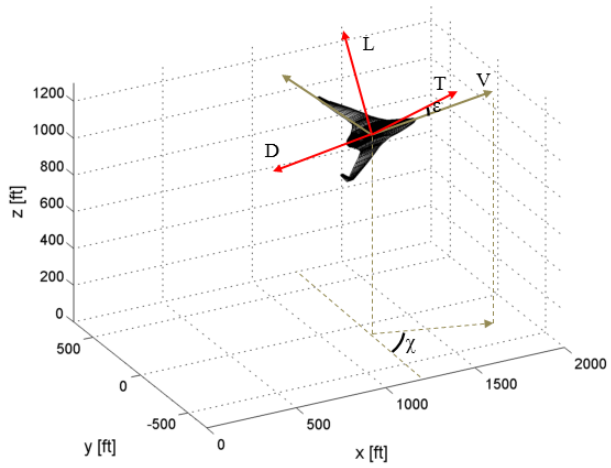



**Figure 4:** UAV free body diagram

$$\dot{x} = v cos\gamma cos\chi \tag{1}$$

$$\dot{y} = v cos\gamma sin\chi \tag{2}$$

$$\dot{z} = v sin\chi \tag{3}$$

$$\dot{v} = \frac{g}{w}\left(T cos\epsilon - D - w sin\gamma\right) \tag{4}$$

$$\dot{\chi} = \frac{g}{wvcos\gamma}\left(Tsin\epsilon + L\right)sin\phi \tag{5}$$

$$\dot{\gamma} = \frac{g}{wv}\left((Tsin\epsilon + L)cos\phi - wcos\gamma\right) \tag{6}$$

where, $x$, $y$ and $z$ are a vehicle position in a global coordinate system. $v$, $\chi$ and $\gamma$ are a velocity, a heading angle, a flight path angle. $D$, $T$, $w$ and $\phi$ are drag, thrust, weight, and bank angle. $\epsilon$ indicates the angle difference between the actual thrust vector and the free stream velocity vector. The lift equation can be defined by

$$L = \frac{1}{2}\rho v^2 S(C_{L0} + C_{L\alpha}\alpha) \tag{7}$$

where, $\alpha$ is an angle of attack, $C_{L0}$ is a coefficient when angle of attack is zero and a lift curve slope $C_{L\alpha}$ is

$$C_{L\alpha} = \frac{\pi AR}{1 + \sqrt{1 + (AR/2)^2}} \tag{8}$$

Here, $AR$ is the wing aspect ratio and $e$ is the span efficiency factor. The following two equations represent a drag model and thrust model.

$$D = \frac{1}{2}\rho V^2 S(C_{D0} + \frac{(C_{L0} + C_{L\alpha}\alpha)^2}{\pi eAR}) \tag{9}$$

$$T = \frac{1}{2}\rho S_p C_p\left[(k_m\delta_t)^2 - v^2\right] \tag{10}$$

where, $C_{D0}$ is a zero-lift drag coefficient and $S$ is a planform area. $S_p$ is the area swept out by the propeller, $C_p$ is an aerodynamic coefficient of the propeller, $k_m$ is a constant indicating the efficiency of the motor and $\delta_t$ is the motor command.

## *2.2 Gain-Scheduling Method*

Conventional gain-scheduling techniques follow the same general process and build upon a common theoretical formulation for the implementation of a linear feedback control to a non-linear system. First, scheduling variables $S^1...S^n$ are chosen to define the flight envelope shown in Figure 5 and all the operating points. A grid of points is defined to span the flight envelope. Normally the grid resolution is chosen based on the sensitivity of vehicle dynamics to scheduling variables. This sensitivity must

be known a-priori, so one may opt for a conservatively dense grid or for an iterative approach that evaluates sensitivity and increases grid resolution in more sensitive regions if needed. An adaptive sampling pursuant to sensitivity information may be chosen, albeit at greater computational expense and complexity. For simplicity we assume two scheduling variables $S^1$ and $S^2$ consistent with common practice, and index values for each of them as $i = 1...n$ and $j = 1...m$. We further assume uniform sampling of the flight envelope for simplicity. The aircraft flight dynamics expressed in equations 1 through 10 can be expressed a function of vehicle states $\mathbf{x}$, control inputs $\mathbf{u}$ and time $t$.

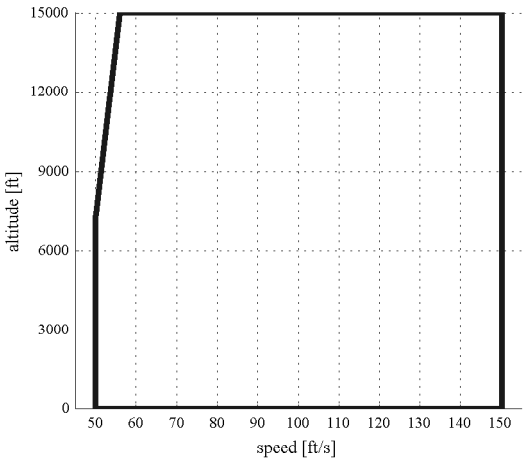



**Figure 5:** Flight envelope

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) \tag{11}$$

For this study vehicle state variables $\mathbf{x}$ are velocity $v$, heading angle $\chi$ and flight path angle $\gamma$. The control inputs $\mathbf{u}$ are motor command $\delta_t$, angle of attack $\alpha$ and back angle $\phi$. The model is locally linearized most commonly with a multivariate Taylor series expansion so that the aircraft state is expressed as the sum of an equilibrium component and a deviation component, as follows:

$$\mathbf{x} = \mathbf{x_{0}}_{ij} + \delta\mathbf{x} \tag{12}$$

$$\mathbf{u} = \mathbf{u_{0}}_{ij} + \delta\mathbf{u} \tag{13}$$

where $\mathbf{x_{0}}_{ij}$ and $\mathbf{u_{0}}_{ij}$ are equilibrium states and control inputs respectively for point $i, j$. Similarly, $\delta\mathbf{x}$ and $\delta\mathbf{u}$ are the perturbation components for the states and the control inputs.

There is an algebraic method and a numerical method for the creation of the linearized model. The algebraic method has some inherent limitations [138] so the numerical alternative is typically preferred. The final mathematical form of linearized dynamics are thus written as:

$$\delta\dot{\mathbf{x}} = \mathbf{A}_{ij}\delta\mathbf{x} + \mathbf{B}_{ij}\delta\mathbf{u} \tag{14}$$

where $\mathbf{A}_{ij}$ is a system matrix of aircraft dynamics at given index i, j, $\mathbf{B}_{ij}$ is a control matrix at given index i, j and $\mathbf{x}$ is the states array. Trim solutions may be found with a number of well established methods and consistent with the assumed order of the flight dynamics model. The next step is designing a linearized controller at each of the prescribed equilibrium points in the flight envelope. There are several design methods available for the linearized controller including the lead-lag method, pole-placement, and optimal controller [82][121]. Robust methods such as H-infinity control [114] can also be applied. The control gain matrix $\mathbf{K}_{ij}$ is thus estimated with the method of choice for each equilibrium point $(S_i^1, S_j^2)$, or $i, j$ for simplicity.

During online aircraft dynamics and controller simulation the gain matrix $\mathbf{K}$ is estimated by a gain-scheduling mechanism as a function of the current scheduling variables. The simplest is the nearest neighbor where the closest equilibrium point $(S_{i*}^1, S_{j*}^2)$ to the current point in the flight envelope $(S^1, S^2)$ is identified, and the corresponding gain matrix $\mathbf{K}_{i*j*}$ is used. The nearest neighbor algorithm is very

simple to implement and is computationally efficient, making it popular in many real-time applications. However, system response features discontinuities for the locus of points equidistant to two or more equilibrium points. The effect is reduced with denser flight envelope sampling requiring greater up-front computation investment and some on-line efficiency degradation for sufficiently large sets. Another popular approach is the use of interpolation techniques such as bilinear interpolation (for the case of two scheduling variables). This approach produces linear interpolants along lines parallel to the scheduling variable axes, and quadratic in all other directions. Extensions to higher order interpolants are straightforward but come at increasing computational burden. Obvious examples include bicubic and spline interpolation [121]. In general, interpolation is recognized to be more computationally expensive than nearest neighbor, even for the lowest allowable order (i.e. linear) and across any number of dimensions. On the other hand, interpolation yields estimates that are generally more accurate, lack discontinuities, and collectively describe a smoother response.

### 2.2.1 Proposed approach - Gain-scheduling with polynomial regression

Conventional methods discussed above outline the fundamental tradeoff between accuracy and computational efficiency. However, both approaches rely on a data set of points in the scheduling variable domain $(S_i^1, S_j^2)$ with corresponding values in the gain set range $\mathbf{K}$, and both approaches feature the same fundamental steps:

- Identify the point(s) in the a-priori set that serve as the basis for generating the estimate of $\mathbf{K}$. For nearest neighbor it is the equilibrium point $(S_{i*}^1, S_{j*}^2)$ closest to the current point. For bilinear interpolation is the four closest equilibrium points, which happen to be corner points surrounding the point of interest.

- Solve for an interpolant of some prescribed order using the selected scheduling variable points as a basis for the solution and its bounding conditions. For

nearest neighbor this step doesn't exist in practice because no interpolant is solved for.

- Evaluate the interpolant at the current point to yield an estimate for gain values $\mathbf{K}$. For nearest neighbor this step is reduced to calling the value of $\mathbf{K}$ that maps to the closest equilibrium point.

The identification of interpolation basis points within a structured set (first step above) as well as the evaluation of a polynomial interpolant of modest order (third step) are both reasonably expected to be fairly inexpensive, whereas solving for the interpolant with scheduling variable points is expected to be the most expensive step. It follows that efficiency improvements of the entire gain scheduling block may be realized by expediting (or eliminating) the generation of the interpolant function. Based on this premise we propose a new gain scheduling method where a single global polynomial interpolant for the entire flight envelope is generated a-priori, effectively reducing the gain scheduler to the evaluation of the polynomial interpolant at the current state. On-line simulation costs are thus reduced. The generation of a global interpolant effectively preserves this as a fundamental step but re-allocates it as an off-line step with some up-front computation cost. The benefits of this approach are reasonably extended to the estimation of trim control inputs so that the latter can be estimated as a polynomial function of scheduling variables generated off-line.

A polynomial interpolant constructed from a structured sample of points in the scheduling variable space is fundamentally equivalent to the creation of a response surface equation from a design of experiments. A response surface takes the form

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i S^i + \sum_{i=1}^{k} \beta_{i,i} S^{i2} + \sum_{i=1}^{k} \sum_{j=1 \neq i}^{k} \beta_{i,j} S^i S^j + \epsilon \tag{15}$$

where, $\beta$ are regression coefficients, $S$ are independent scheduling variables, $k$ is the number of scheduling variables ($k = 2$), $y$ is the dependent response of interest, namely the trim control inputs $\mathbf{u}_{0ij}$ and the optimal control gain $\mathbf{K}_{ij}$. The term $\epsilon$

is the error of the polynomial approximation relative to the true response typically associated with higher order terms omitted. An evenly gridded sample of points in the operating envelope is consistent with a full-factorial design of experiments. The full factorial design is easily justifiable for the commonplace choice of a two scheduling variable setup because it offers the most beneficial sampling of the space for response surface regression while the number of data points $m \times n$ is still reasonable. With the inclusion of more scheduling variables the cardinality of the full factorial set grows quickly and becomes prohibitive. Other designs of experiments may need to be considered in such a case; examples include central composite design (CCD), Box-Behnken, and different variants of space filling designs [110].

Response surface coefficients are most commonly solved for with the least squares method primarily due to its simplicity and guaranteed optimality for the given data set. Alternative methods include step-wise approaches that exploit analysis of variance estimates to determine the inclusion or exclusion of regression terms. The quality of the response surface is commonly evaluated with error statistics including the coefficient of determination $R^2$, residuals, and standard normal distributions for Model Fit Error (MFE) and Model Representation Error (MRE) [110].

We hypothesize that despite the computational cost of generating response surfaces as a priori global approximations, *measurable improvements in computational efficiency can be attained relative to nearest neighbor and bilinear methods* described above. Moreover, we also hypothesize that these *runtime benefits can be attained without degradation of controller stability.*

At the same time, we recognize that global polynomial fits for gain scheduling also present some limitations inherent in response surface methodology. The most salient issue is the non-linearity of the underlying multivariate response vis-a-vis the extent to which the polynomial can approximate it given its order. The polynomial presented in equation 15 is second order, which is sufficient for a surprisingly large

number of relationships in the aircraft performance literature. For phenomena of greater non-linearity the error of a second order polynomial approximation can become unacceptable. A polynomial of higher order may be considered, but the option should be approached with caution. Overfitting may occur, where model fit error improves moderately and model representation error increases significantly, suggesting that the nonlinearlity of the underlying response is not suitably appoximiated by a polynomial. This effect is typically observed near the extremes of the response interval and is referred to as Runge's phenomenon. In the subsection 2.2.3 we address the issue of nonlinearity of polynomial approximations via order increase and logarithmic transformations.

### 2.2.2 Implementation of aircraft dynamics and controller design

We select altitude and speed as scheduling variables consistent with common practice. For simplicity we bound the flight envelope as a rectangle in the scheduling variable space with pairwise combinations of minimum and maximum values of speed and altitude. Other operating points in the envelope are readily defined with a grid resulting from a uniform segmentation of the domain of each variable from the flight envelope presented in Figure 5, so that the total number of the equilibrium points is the product of $m$ uniformly spaced values of speed and $n$ uniformly spaced values of altitude.

In the numerical simulation the aircraft dynamics model is assumed at three degrees of freedom. The states $\mathbf{x}$ include velocity, directional angle and flight path angle $[v\ \psi\ \gamma]^T$ and the control inputs $[T\ \alpha\ \phi]^T$ have thrust and two pseudo control inputs which are Angle of Attack (AoA) and bank angle. We apply the numerical linearization method to the resulting non-linear system.

Optimal controller design is implemented with the Linear Quadratic Regulator

(LQR) technique which is fairly straightforward to implement for a Multi-Input Multi-Output (MIMO) system. From the state space equation of the linearized model presented in equation 14 the control input $\delta\mathbf{u}$ can be defined by LQR theory as follows:

$$\delta\mathbf{u} = \delta\mathbf{u}_T = -\mathbf{K}_{ij}\delta\mathbf{x} \tag{16}$$

where $\mathbf{K}_{ij}$ is the matrix containing a family of control gains and $\mathbf{u}_{Tij}$ is a set of control inputs at given equilibrium conditions. The optimal control inputs minimize the performance index

$$\delta J_{ij} = \int_0^\infty \left(\delta\mathbf{x}^T\mathbf{W}_{1ij}\delta\mathbf{x} + \delta\mathbf{u}^T\mathbf{W}_{2ij}\delta\mathbf{u}\right) \tag{17}$$

where $\mathbf{W}_{1ij}$ and $\mathbf{W}_{2ij}$ are weighting matrices at given flight condition points $i, j$. The control gain matrix $\mathbf{K}_{ij}$ is defined by

$$\mathbf{K}_{ij} = \mathbf{W}_{2ij}^{-1}\mathbf{B}_{ij}\bar{\mathbf{P}}_{ij} \tag{18}$$

$\bar{\mathbf{P}}_{ij}$ is the positive definite solution of the Riccati equation

$$\bar{\mathbf{P}}_{ij}\mathbf{B}_{ij}\mathbf{W}_{2ij}^{-1}\mathbf{B}_{ij}^T\bar{\mathbf{P}}_{ij} - \bar{\mathbf{P}}_{ij}\mathbf{A}_{ij} - \mathbf{A}_{ij}^T\bar{\mathbf{P}}_{ij} - \mathbf{W}_{1ij} = 0 \tag{19}$$

### 2.2.3 Implementation of global polynomial interpolant

In order to examine the proposed global polynomial interpolant against competing approaches it is useful to formulate its generation from an algorithmic perspective readily conducive to implementation in a repeatable and recursive manner. As noted in previous sections the selection, domain bounding, and sampling of scheduling variables are prerequisite. In general the sampling follows a design of experiments chosen based on number of arguments, effects resolution, and corresponding number of function calls. Here we simplify to two scheduling variables $(S^1, S^2) = (v, h)$ so that $(v_i, h_j)$ are sampled in a uniform $n \times m$ full factorial set where $i \in \mathbb{N}|1 \leq i \leq m$ and $j \in \mathbb{N}|1 \leq i \leq n$. The selection of $m$ and $n$ effectively captures the granularity of the

training sample in each ordinate. For all $(v_i, h_j)$ we solve for $\delta\mathbf{u}_{ij}$ and $\mathbf{K}_{ij}$ as shown in the previous section, so that the argument array $(\mathbf{v}, \mathbf{h})$ has corresponding response arrays $\delta\mathbf{u}$ and $\mathbf{K}$. A response surface can be generated for each response variable using the least squares method.

To do so the order $O$ of the response surface must be selected. The response surface shown in Eq (15) is of second order, which is typical for many practical applications. A higher order response surface typically yields approximations with smaller error $\epsilon$. However an indiscriminately high polynomial order is not feasible for various reasons. Higher order terms requires more sample data points so that higher order regression coefficients $\beta$ can be solved for in the least squares regression. Very large data sets may not be realistically attainable for resource intensive data acquisition applications such as computationally expensive simulations and some experimental setups. Even without resource limitations the nonlinear characteristics of the underlying behavior may be such that they cannot be well approximated by a polynomial of high order. Attempting to overcome this inherent barrier with higher polynomial order typically results in 'overfitting', where the error of the polynomial against the regression data set reduces moderately while the validation error against additional data points grows considerably thus diminishing the overall approximation quality of the polynomial. The recommended approach in this and most other applications is to begin with the lowest order expected and to increase the order until some criteria for the quality of the approximation is met.

The least squared regression assumes that error follows a standard normal distribution. Comparison of the model fit error (MFE) $\epsilon_{MFE}$ against this distribution thus offers a suitable basis to evaluate the quality and statistical validity of the polynomial approximation. Model fit error for the $p^{th}$ observation in the data sample is estimated as

$$\epsilon_{MFE,p} = \frac{\Delta y_O}{y_O} = \frac{y_p - y_p^*}{y_O}, \tag{20}$$

where the absolute error $\Delta y$ is the difference between the true value $y$ from the data sample and that approximated by the polynomial, $y^*$. Although several tests for distribution normality exist, it is typical to directly estimate the mean $\mu_{MFE}$ and standard deviation $\sigma_{MFE}$, and evaluate against standard values. We do so with the following inequality condition, allowing for some small deviation of the mean.

$$-0.05 \leq \mu_{MFE} \leq 0.05 \quad \cap \quad \sigma_{MFE} \leq 1 \tag{21}$$

Model representation quality of the polynomial beyond the sample used for its regression via least squares method is evaluated with model representation error (MRE) $\epsilon_{MRE}$. It is calculated in the same way as model fit error $\epsilon_{MFE}$ in Eq (20) but uses a validation points sample, typically drawn at random from the bounded domain of the regression variables. There is no constraint on the size of the MRE data sample, although some rules of thumb exist balancing the practicality and cost of obtaining said sample against the sample size to obtain a statistically significant distribution. In most cases the distribution of model representation error features greater variance relative to model fit error. We account for this trend in the inequality conditions to test for representation error

$$-0.05 \leq \mu_{MRE} \leq 0.05 \quad \cap \quad \sigma_{MRE} \leq 1.5 \tag{22}$$

The coefficient of determination $R^2$ is another convenient and suitable test for the quality of the response surface. It provides a statistical measure of how well the regression approximates the real data. The coefficient can be expressed in many different ways, but is commonly defined on the basis of the sum of squares as follows:

$$R^2 \equiv 1 - \frac{SS_{residual}}{SS_{Total}} = 1 - \frac{\sum_p (y_p - y_p^*)^2}{\sum_p (y_p - \bar{y})^2} \tag{23}$$

Here $y_p$ is the true value of the $p^{th}$ data point, $y_p^*$ is the corresponding value approximated by the polynomial, and $\bar{y}$ is the mean of true value observations

$$\bar{y} = \frac{1}{n} \sum_{p=1}^{n} y_p \tag{24}$$

The coefficient of determination assumes values between 0 and 1, with the latter representing a perfect fit of the polynomial approximation to the regression data set. While values of 0.95 or 0.975 are common standards for a high quality polynomial fit, we opt for the more stringent condition

$$R^2 \geq 0.99 \tag{25}$$

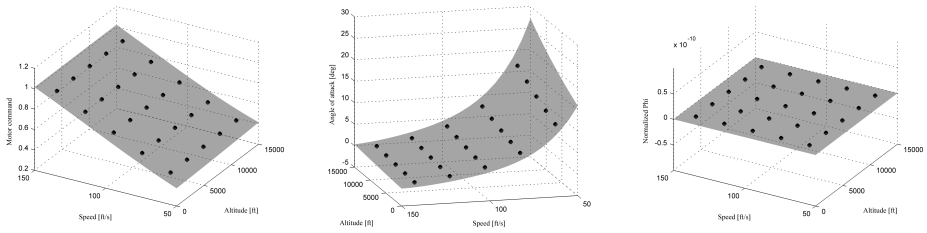
We adopt the conditions expressed in Eq (21), Eq (22), and Eq (25) as the criteria for polynomial approximation quality. For a given scheduling data sample and some prescribed polynomial order $O$ a least squares regression can be conducted and checked against said conditions. We assume here that the data sample is sufficiently large and so designed that polynomial coefficients can be solved. If the quality conditions are not met the order of the polynomial may be increased by one and the process repeated. As discussed above this process should be bound by some maximum polynomial order allowed. In the case that conditions cannot be met even for the highest order allowed then the non-linearity of the true response may be beyond that characterized by the $O_{max}$ order polynomial. A logarithmic transformation of the response is a convenient mechanism to mitigate the non-linearity of the response and fit a polynomial approximation that can then be exponentiated, typically of better quality. For logarithmic transformations the estimation of MFE and MRE is conducted as shown in Eq (20) except that $y_p^*$ is replaced by $exp(y_p^*)$, noting that the least squares regression is performed on $log(\mathbf{Y})$.

Algorithm 1 states the aforementioned recursive regression, as follows:

**Algorithm 1** Surrogate modeling procedure for gain-scheduling method

Inputs: $\mathbf{S}$, a $m \times n$ array of $(v_i, h_j)$ pairs, $1 \leq i \leq m$ and $1 \leq i \leq n$
System matrix $\mathbf{A}_{ij}$, control matrix $\mathbf{B}_{ij}$, weighting matrices $\mathbf{W}_{1ij}$ and $\mathbf{W}_{2ij}$
Compute trim control inputs $\mathbf{u}_{0ij}$ and optimal control gains $\mathbf{K}_{i,j}$ at equilibrium points
**for** i=1:m **do**
  **for** j=1:n **do**
    Solve:$\mathbf{u}_{0ij}$
    Solve:$\bar{\mathbf{P}}$ ▷ Eq (19)
    Solve:$\mathbf{K}_{i,j}$ ▷ Eq (18)
  **end for**
**end for**
**for** $\mathbf{Y} = \mathbf{u}_0, \mathbf{K}$ **do**
  $O = 1$
  $C$ =FALSE
  **while** $C$ = FALSE **do**
    $O = O + 1$;
    $\mathbf{Y_{pred}}$=LeastSquares($\mathbf{Y}$,$\mathbf{S}$,$n$)
    $(\mu_{MFE}, \sigma_{MFE}, \mu_{MRE}, \sigma_{MRE}, R^2)$=FitStats($\mathbf{Y}$,$\mathbf{Y_{pred}}$) ▷ Eq (21), Eq (22), and Eq (23), or log transform equivalent
    $C$ = Eval: Eq (21) AND Eq (22) AND Eq (25)
    **if** $O = O_{Max}$ **then**
      $O = 1$
      $Y = log(Y)$, Replace:FitStats, LogFitStats
    **end if**
  **end while**
**end for**

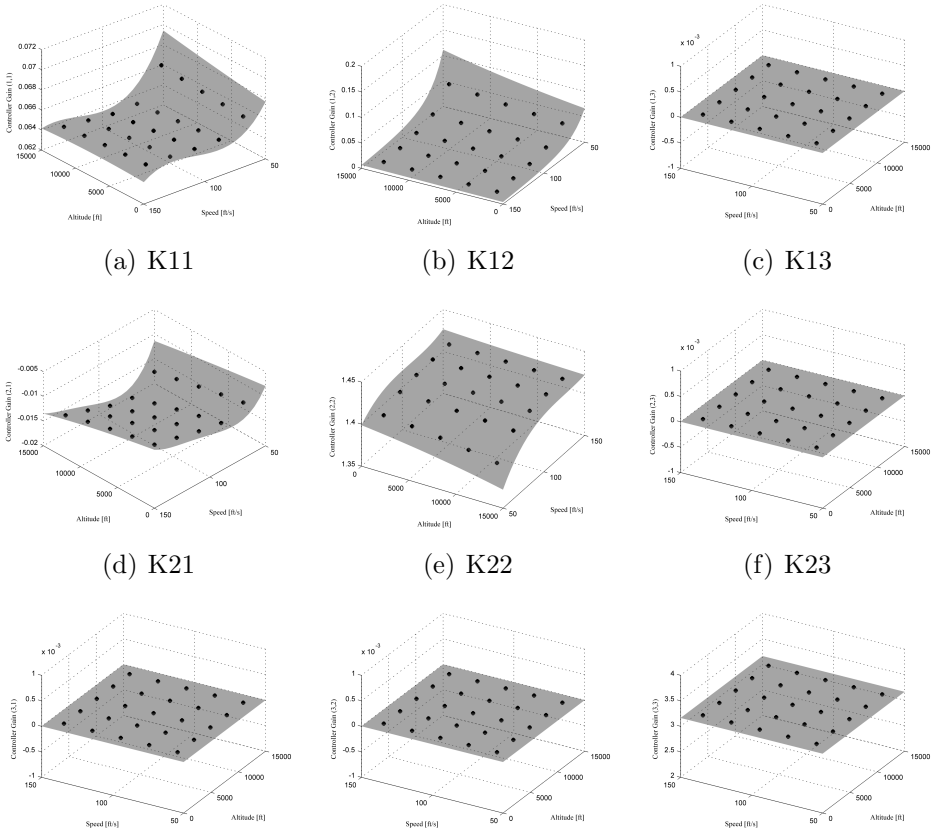
We illustrate the process and its final outcome by bounding the scheduling variable domain with $v\ [ft/s] : 50 \leq v \leq 150$ and $h\ [ft] : 0 \leq h \leq 15,000$, and sampling uniformly along each dimension with $n = m = 5$ and we eliminates the sampling points outside of flight envelope region presented in Figure 5. Figure 6 and Figure 7 illustrate the results of the response surface regressions for the trim control inputs and the control gain values respectively. The 25 training points are shown as black dots. For the trim control inputs shown in Figure 6 the bank angle is zero in the entire flight envelope since the equilibrium conditions is only defined under a level flight condition. Visual inspection of the response surfaces for the motor command and angle of attack suggest greater sensitivity to speed variation than to altitude. This trend is observed as well in the regressed polynomials for the control gain matrix **K** values as shown in Figure 7. In these polynomial approximation of the control gain matrix **K**, K13, K23, K31 and K32 are approximately zero since our simplified equations of motion do not account for coupling effects between bank angle and other variables (thrust and angle of attack).




(a) Motor command (b) Angle of attack (c) Bank angle

**Figure 6:** Surrogate model for the trim inputs

As an initial comparative assessment of control inputs (motor and angle of attack) and gains (K11, K12, K21 and K22) obtained via the three scheduling methods we evaluate a random sample of 100 points in the scheduling variable space. We quantify the average accuracy of the estimates against optimal solutions with $R^2$. Table 3 summarizes $R^2$ values, and Figure 8 illustrates the data set as actual vs.




(a) K11 (b) K12 (c) K13

(d) K21 (e) K22 (f) K23

(g) K31 (h) K32 (i) K33

**Figure 7:** Surrogate models for the control gain matrix **K**

predicted plots for the 100 data points, color-coded according to gain-scheduling methods. These results indicate that the global polynomial approach has comparable (modestly better) accuracy relative to bilinear interpolation, and both are notably more accurate than nearest neighbor. The control gain K11 is most sensitive to the choice of method, and is explained by its association to the low speed region.

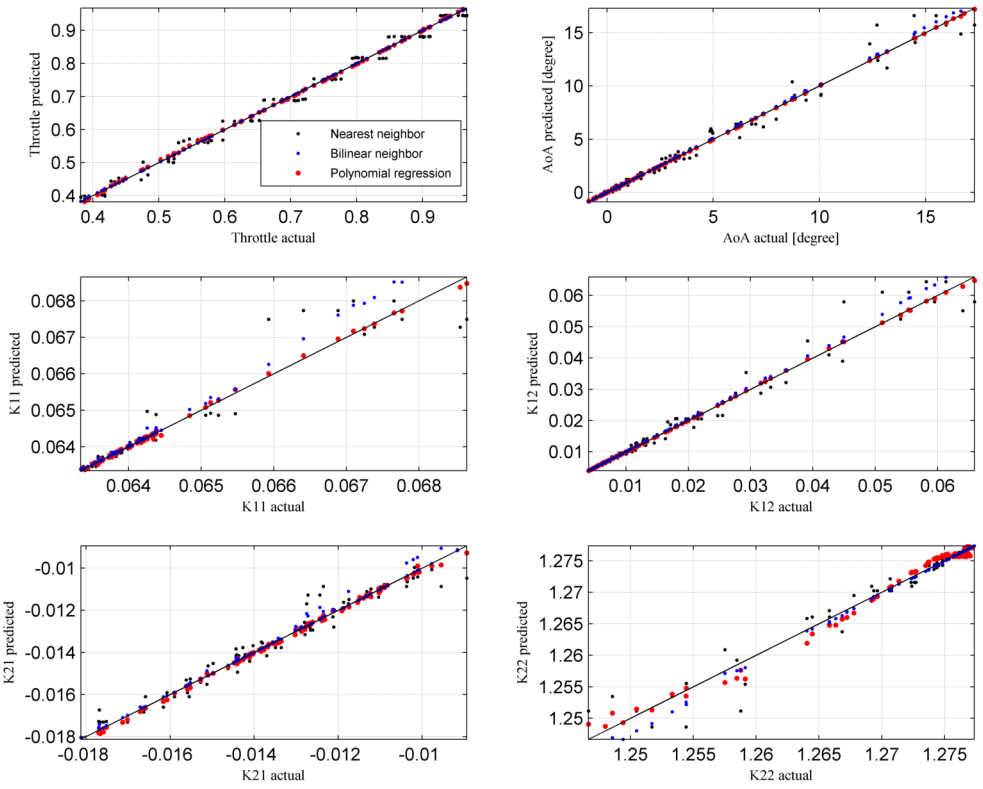



**Figure 8:** Predicted vs. Actual plot

**Table 3:** $R^2$ results

| | Nearest Neighbor | Bilinear Interpolation | Global Polynomial |
|---|---|---|---|
| Motor command | 0.99074 | 0.99999 | 0.99982 |
| Angle of attack | 0.9817 | 0.9988 | 0.99997 |
| K11 | 0.91475 | 0.96196 | 0.99819 |
| K12 | 0.97139 | 0.99487 | 0.99981 |
| K21 | 0.9609 | 0.99288 | 0.99754 |
| K22 | 0.96359 | 0.99109 | 0.98461 |

## *2.3 Results and discussion*

### 2.3.1 Computational cost

Our hypothesis states that the global polynomial incurs in online and off-line computational cost that is the same or better than nearest neighbor and bilinear methods. To test the hypothesis we measure the actual off-line and online computational processing time for the nearest neighbor, bilinear interpolation, and global polynomial approaches. Off-line processing pertains to the solution of the trim control inputs $\mathbf{u}_{0ij}$ and the optimal control gain matrix $\mathbf{K}_{ij}$ for the $n \times m$ set of conditions in the scheduling variable space. Off-line computational time thus increases with the cardinality of the set. In the case of the global polynomial method off-line processing also includes the generation of the response surface approximations via least square regression as illustrated in Algorithm 1. The regression process is also expected to increase with the size of the data sample, albeit less so than the solution of $\mathbf{u}_{0ij}$ and $\mathbf{K}_{ij}$. Online processing pertains to the actual simulation of the UAS flight dynamics, including the variable scheduling function applied to the trim control inputs $\mathbf{u}_{0ij}$ and the optimal control gain matrix $\mathbf{K}_{ij}$, utilizing each of the three methods under consideration. As discussed in subsection 2.2.1 value scheduling involves three basic steps: the identification of points that are the basis for the interpolation, solving for the interpolant, and evaluating the interpolant. The first step is believed to be of very low computational cost, marginally sensitive to the granularity of the data set, and not applicable for the global polynomial. The second step is believed to be the most expensive and only applicable to the bilinear interpolation. The evaluation of the interpolant in the third step is also believed to be inexpensive and modestly dependent on its order, and is not applicable to nearest neighbor.

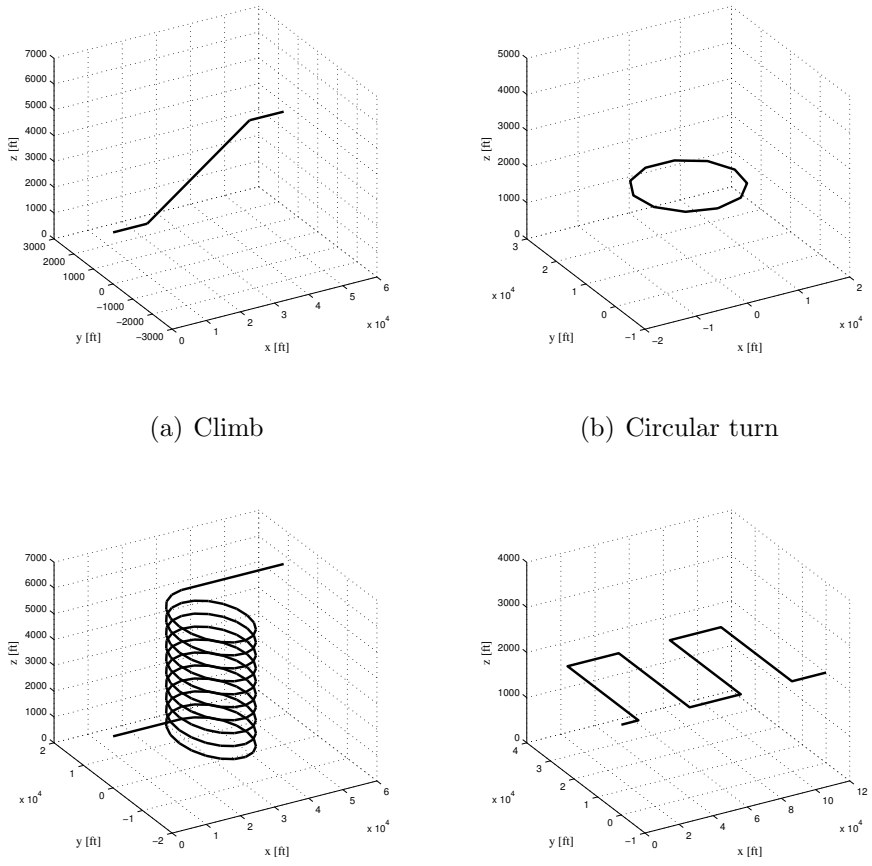
The hypothesis test is designed to explicitly quantify the effect of the selected gain scheduling approach and of the data set size on computational time, and to provide relevance and representativeness to UAS applications treated in this paper.

To address the former we conduct tests with data sets containing 16, 25, 64, and 100 trim points arranged in a square design within prescribed velocity and altitude bounds. To address the latter, we simulate four different maneuvers selected from the literature based on their importance and pervasiveness across different UAS mission profiles. More specifically, we utilize the RTCA Operational Services and Environmental Definition (OSED) for UAS [128] which outlines seven basic UAS mission profiles and categorizes them as point-to-point (P2P), planned aerial work (PAW), and unplanned aerial work (UAW). We identify four elemental flight maneuvers from explicit commentary in Ref. [128] or implicitly through our own interpretation and understanding of the mission profiles: Climb, Circular Turn, Circular Climb (spiral climb), and Horizontal Step. Table 4 summarizes the four elemental maneuvers and their presence across the seven basic UAS mission profiles. A circle indicates that the maneuver is observed in a given mission, a cross indicates the maneuver does not occur, and a triangle indicates that maneuver is understood or expected to occur albeit lack of explicit textual reference. Figure 9 illustrates the elemental maneuvers.

**Table 4:** UAS Elemental maneuvers and basic mission profiles

| Mission | Law enforcement | Marine monitoring | Environmental sensing | Cargo Delivery | Border surveillance | Hurricane research | Environmental monitoring |
|---|---|---|---|---|---|---|---|
| Mission | P2P | P2P | P2P | P2P | P2P | P2P | P2P |
| Type | PAW | PAW | PAW | | PAW | PAW | PAW |
| | UAW | UAW | PAW | | UAW | UAW | |
| Climb | O | O | O | O | O | O | O |
| Cir. turn | X | X | O | X | O | O | X |
| Cir. climb | X | X | △ | X | △ | △ | X |
| Hor. step | X | O | O | X | X | X | O |
| O - Observed | | | | | | | |
| X - Not observed | | | | | | | |
| △ - No information | | | | | | | |

Simulations were conducted on a commercially available desktop computer with a 3.40GHz Intel(R) Core(TM) i7-2600 processor and a 8GB RAM. Results for off-line computational cost, presented in Table 5, confirm expected trends discussed above. For all scheduling methods off-line cost increases with number of points in the scheduling variable space. For nearest neighbor and bilinear interpolation the cost is the same since the off-line process is exactly the same, namely solving for trim




(a) Climb

(b) Circular turn

(c) Circular climb

(d) Horizontal step

**Figure 9:** Simulation scenarios

control inputs and the control gain set. Results indicate that off-line cost is linear with number of points, with a fixed minimum cost of 0.59 seconds and an incremental cost of 0.0355 seconds per trim point. For the proposed method using a global polynomial interpolant results also show off-line cost increasing linearly with number of points. Incremental cost is also observed at 0.0355 seconds per trim point, whereas fixed minimum cost is 0.12 seconds greater by virtue of the global polynomial regression. The computational time for the regression increases with the number of trim points, but negligibly within the 0.01 second accuracy used in reporting these results. This approximately constant increment in off-line cost represents a 8.6% increase for 16 points decreasing linearly to 2.8% for 100 points. Altogether, the we consider this penalty in off-line processing to be very modest.

**Table 5:** Off-line computational time in seconds [s] for nearest neighbor, bilinear interpolation, and global polynomial, with reference data sets of varying size

| No. of Points | Nearest Neighbor | Bilinear Interpolation | Global Polynomial |
|---|---|---|---|
| 16 | 1.17 | 1.17 | 1.28 |
| 25 | 1.47 | 1.47 | 1.59 |
| 64 | 2.86 | 2.86 | 2.98 |
| 100 | 4.14 | 4.14 | 4.26 |

For online computational cost we record the computer processing time associated with the evaluation of the scheduling function. Because this function is evaluated a multitude of times in each simulation, we report results as mean and standard deviation values in Table 6. We also record the entire simulation time, the total time spent on scheduling function calls, and the number of scheduling function calls within the simulation, all summarized in Table 7

From results in Table 6 we observe that the global polynomial features the best function evaluation time of all three methods, followed by nearest neighbor which is 2.7 times greater on average, and then by bilinear interpolation taking 5.5 times as long on average. As expected, within each method the function call time is not dependent on the definition of the maneuver or the resolution of the reference data

**Table 6:** Scheduling function (online) evaluation time, mean and standard deviation in miliseconds [ms] for nearest neighbor, bilinear interpolation, and global polynomial, with reference data sets of varying size

| | No. of | Mean[ms] | | | | Standard Deviation [ms] | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Points | Climb | Cir. Turn | Cir. Climb | Hor. Step | Climb | Cir. Turn | Cir. Climb | Hor. Step |
| Nearest Neighbor | 16 | 0.492 | 0.489 | 0.490 | 0.490 | 0.067 | 0.018 | 0.046 | 0.050 |
| | 25 | 0.490 | 0.488 | 0.490 | 0.491 | 0.043 | 0.023 | 0.044 | 0.049 |
| | 64 | 0.492 | 0.491 | 0.490 | 0.490 | 0.049 | 0.034 | 0.031 | 0.045 |
| | 100 | 0.490 | 0.489 | 0.491 | 0.490 | 0.041 | 0.042 | 0.045 | 0.038 |
| Bilinear Interpolation | 16 | 1.000 | 0.996 | 1.005 | 0.998 | 0.100 | 0.037 | 0.073 | 0.062 |
| | 25 | 0.999 | 0.999 | 0.998 | 0.998 | 0.059 | 0.055 | 0.044 | 0.043 |
| | 64 | 0.998 | 1.000 | 0.997 | 0.999 | 0.030 | 0.046 | 0.056 | 0.070 |
| | 100 | 1.011 | 1.005 | 1.001 | 1.000 | 0.078 | 0.049 | 0.045 | 0.031 |
| Global Polynomial | 16 | 0.184 | 0.181 | 0.180 | 0.181 | 0.224 | 0.013 | 0.024 | 0.043 |
| | 25 | 0.181 | 0.184 | 0.182 | 0.182 | 0.014 | 0.038 | 0.022 | 0.026 |
| | 64 | 0.182 | 0.182 | 0.182 | 0.182 | 0.013 | 0.038 | 0.027 | 0.033 |
| | 100 | 0.182 | 0.182 | 0.182 | 0.182 | 0.041 | 0.012 | 0.027 | 0.036 |

**Table 7:** Total simulation time [s], number of scheduling function calls, and total scheduling function time [s] for nearest neighbor, bilinear interpolation, and global polynomial, with 100 point reference data set

| | | Nearest Neighbor | Bilinear Interpolation | Global Polynomial |
|---|---|---|---|---|
| Total Simulation Time [s] | Climb | 8.64 | 11.59 | 7.31 |
| | Cir. Turn | 10.29 | 14.09 | 8.73 |
| | Cir. Climb | 111.02 | 153.44 | 93.79 |
| | Hor. Step | 40.04 | 55.02 | 33.82 |
| Number of Function Calls [-] | Climb | 4,775 | 5,016 | 5,018 |
| | Cir. Turn | 5,879 | 6,172 | 6,176 |
| | Cir. Climb | 64,522 | 67,648 | 67,677 |
| | Hor. Step | 23,141 | 24,280 | 24,271 |
| Total Scheduling Function Time [s] | Climb | 2.33 | 5.07 | 0.91 |
| | Cir. Turn | 2.87 | 6.2 | 1.12 |
| | Cir. Climb | 31.68 | 67.68 | 12.33 |
| | Hor. Step | 11.33 | 24.29 | 4.42 |

set used to generate the approximation. Standard deviation for the evaluation time sample is shown to be one order of magnitude smaller than the mean thus indicating an acceptably modest degree of variation.

Results collected for the entire experimental set indicate that for each variable scheduling method the total simulation time for each maneuver is insensitive to the number of reference points. Accordingly, we only report results in Table 7 for the case of 100 points noting that they are almost identical to those for simulations using reference data sets with 16, 25, and 64 points.

Results depict a consistent trend in total simulation time across the four maneuvers for all gain scheduling methods. Climb takes the least amount of time followed by circular turn, horizontal step, and circular climb. The same trend can be observed

via number of function calls. This difference is attributed to the duration of the simulated maneuvers themselves as defined in this study. The climb maneuver is the most brief, whereas the circular climb is the most lengthy. Within the total simulation time we identify the time allocated for scheduling function evaluation. Online simulation time for functions other than variable scheduling includes vehicle dynamics functions, atmosphere conditions function, and geometry conversion function, among others. These functions are called the same number of times and with no variation in evaluation time within each iteration. They are identical across method, maneuver, and reference data size. Only small variations are observed as minimal perturbations due to fluctuations in computer processing, and are on average 1.2 [ms] per scheduling function call iteration.

It is worth noting that the number of function calls for bilinear interpolation and global polynomial are almost same and the biggest difference between the two methods is 29 function call difference relative to a nominal 67,667 value is noted for the circular climb maneuver. However, for the nearest neighbor more significant differences in the number of function calls are noted. To explain this difference we first note that the number of function calls is related to the simulated maneuver flight time and the simulation engine sampling time as follows:

$$N = \frac{t_{flight}}{t_{sampling}} + 1 \tag{26}$$

where $N$ is the number of function calls, $t_{flight}$ is the simulated flight time, and $t_{sampling}$ is the sampling time. For instance, for the nearest neighbor method the *Climb* maneuver required 4,775 function calls (See Table 7), which corresponds to the 477.4 second duration of the simulated maneuver and the 0.1 second sampling time of the simulation engine. The additional function in Eq (26) is the first evaluation at flight time 0.0 that initializes the flight dynamics simulation. All methods and all maneuvers features the same 0.1 second simulation sampling time.

The difference in number of function calls of the nearest neighbor method relative to the bilinear interpolation and the global polynomial is traced to differences in the control gain set and trim solution estimation which have a direct impact on control inputs, overall vehicle response, and ultimately on the duration of the flight maneuver. Estimates produced with bilinear interpolation and global polynomial are similar, at least when compared to those produced by nearest neighbor, and result in maneuvers with practically the same duration. The nearest neighbor algorithm yields noticeably different inputs and gain sets, and therefore has noticeably different flight times. The flight time with nearest neighbor is found to be consistently lower than that with the other two scheduling methods, an unexpected result that could be thought to be favorable. However we note that even with less function calls the total variable scheduling time and total simulation time is greater than that for the global polynomial. The quality of the nearest neighbor approximation as discussed in the next subsection also places this observation in context and suggests that this reduced flight time may in fact be misleading as the controller stability with this method deviates from the optimum.

Overall we find that compared to bilinear interpolation and nearest neighbor the global polynomial method presents very small penalties in off-line processing costs, while providing significant improvements in online time. The latter is improved by a factor of 2.7 relative to bilinear interpolation and much more significantly relative to nearest neighbor with a factor of 5.5. Collectively the evidence here reported supports the stated hypothesis on runtime.

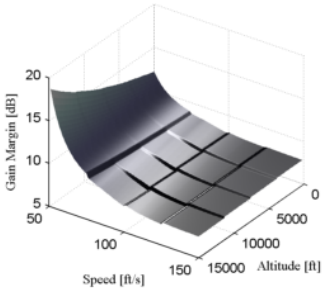
### 2.3.2 Controller stability and performance

The hypothesis for the proposed gain scheduling method with a global polynomial states that the resulting controller stability and performance is the same or better than nearest neighbor and bilinear methods.

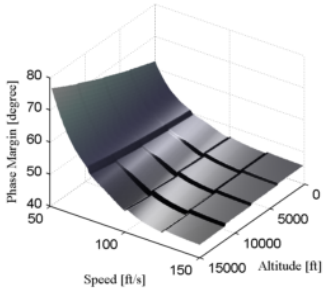
To evaluate controller stability and performance we conduct multivariate gain and phase margin analysis on the gain value approximations $\mathbf{K}$ produced by each of the gain-scheduling approaches. Gain and phase margin are well known measures of stability in closed-loop, dynamic-control systems. We use multivariate gain and phase margin analysis that simultaneously considers the response of each controller output ($x_i$: velocity $v$, heading angle $\chi$, and flight path angle $\gamma$), relative to each controller input signal ($u_i$: motor command $\delta_t$, angle of attack $\alpha$, bank angle $\phi$). Like its univariate analogue, multivariate margin analysis provides a measure of the tolerance of the close-loop system before becoming unstable. This technique has been implemented in many stability analysis, for instance, for of the F/A-18 aircraft in the falling leaf mode [28].

Figure 10 presents multivariate margin analysis results as surface plots of gain margin and phase margin on the scheduling variable space $(v, h)$ for each of the three value scheduling methods. Visual inspection of the results reveals the step-like response of gain and phase margin in figures 10(a) and 10(b) for the nearest neighbor method. These undesirable discontinuities in the response are expected and occur for the locus of points that are equidistant to two (or more) adjacent solution points. In contrast the gain and phase margin for bilinear interpolation and global polynomial appear to be smooth and within comparable value bounds. For all methods gain and phase margin are governed by velocity and notably insensitive to altitude.

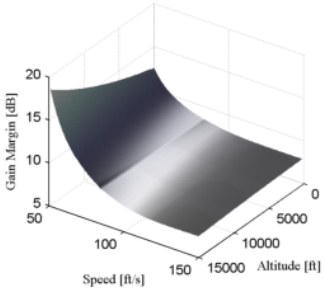
To evaluate controller stability for each of the gain scheduling methods we characterize the deviation of gain and phase margin relative to the optimal trim and gain solution. To do so we define 10,000 ($i = 1, \ldots, 100$, $j = 1, \ldots, 100$) equilibrium points in a square grid design of the scheduling variable space and solve for the optimal control gain-values $\mathbf{K}_{ij}^{opt}$. High-altitude low-speed points of the square grid that lay outside the flight envelope are removed, resulting in 9,837 points. We estimate corresponding multivariate gain and phase margin values $\mathbf{GM}_{ij}^{opt}, \mathbf{PM}_{ij}^{opt}$ for each point.



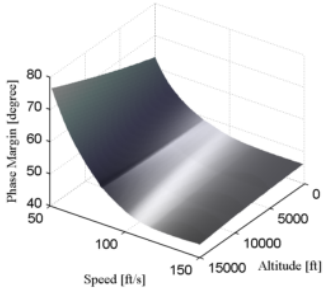

(a) Gain margin - nearest neighbor



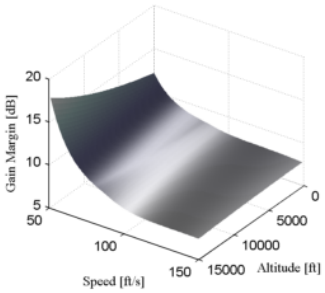

(b) Phase margin - nearest neighbor



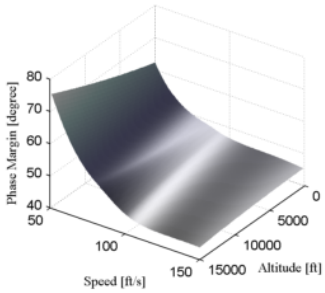

(c) Gain margin - bilinear interpolation




(d) Phase margin - bilinear interpolation




(e) Gain margin - global polynomial




(f) Phase margin - global polynomial

**Figure 10:** Multivariate gain and phase margin for three candidate scheduling methods

The process is repeated with each of the gain scheduling methods, and the margin deviation estimated as the difference relative to $\mathbf{GM}_{ij}^{opt}$ and $\mathbf{PM}_{ij}^{opt}$ respectively for each point $(v_i, h_j)$:

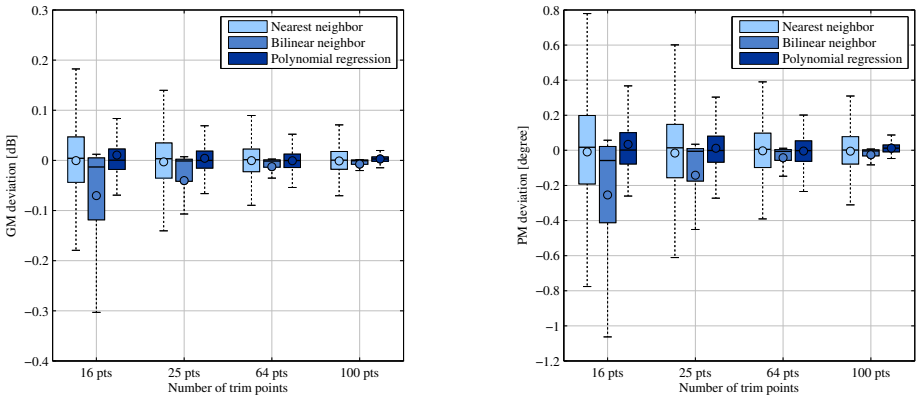
$$\begin{aligned} \Delta GM &= GM_{ij}^{opt} - GM_{ij} \\ \Delta PM &= PM_{ij}^{opt} - PM_{ij} \end{aligned} \tag{27}$$

Statistical results of the analysis are graphically summarized as box plots in Figure 11. The deviation relative to optimal solution margin values is presented for each of the gain scheduling methods and for each of the four grid resolutions tested (16, 25, 64, and 100 points in square design). In each box plot the box encompasses the second and third quartile ($25^{th}$ to $75^{th}$ percentile), the bars near the extremes encompass $5^{th}$ to $95^{th}$ percentiles, the bar near the middle indicates the median, and the circle indicates the mean.

Results for each method indicate that a higher number of reference solution points results in smaller stability deviations relative to the optimal controller design. This effect can be reasonably expected because approximations of the gain values $\mathbf{K}$ for points other than those in the reference set are more accurate. The trend is readily observed in the box plots as a decreasing deviation from the optimal solution with mean and median values closer to zero as well as decreasing data sample dispersion. In consideration of the magnitude of the deviations observed, in the order of $+/-1dB$ and $+/-0.4deg$, the effect of number of trim points on deviation relative to optimal solution margins is measurable but not of any major practical significance.

Results also show that, for any given number of reference solution points, the proposed global polynomial approximation features small stability deviation from the optimal controller. These deviations are of the same order of magnitude as those observed for nearest neighbor and bilinear interpolation. Although results indicate slightly smaller margin deviations for the proposed polynomial regression the numerical nature of this assessment and the measured magnitude of the effect do not suggest

a stability margin benefit. The findings none the less support the hypothesis that relative to nearest neighbor and bilinear interpolation the polynomial gain scheduling does not observe degradation of controller stability.




(a) Gain margin

(b) Phase margin

**Figure 11:** Statistic analysis for Gain margin and Phase margin

As an additional means to characterize modeling accuracy we perform numerical simulations showing the time-response characteristics of each gain-scheduling method. The assessment is done in the style of work by Morelli [108][107] who used time-response to doublet control inputs to quantify the modeling accuracy of a multivariate power series expansion for aerodynamic coefficient predictions. For the modeling accuracy comparison with the three scheduling techniques we define a reference trajectory as shown in Figure 12 and investigate the time response arising from each gain-scheduling method. Results indicate that gain-scheduling with the bilinear interpolation and the proposed polynomial approach have very similar time-response. The nearest neighbor method can give rise to some loss of smoothness in the speed response due to the switching control structure. We conclude that the proposed polynomial gain-scheduling has time-response quality comparable to bilinear interpolation.

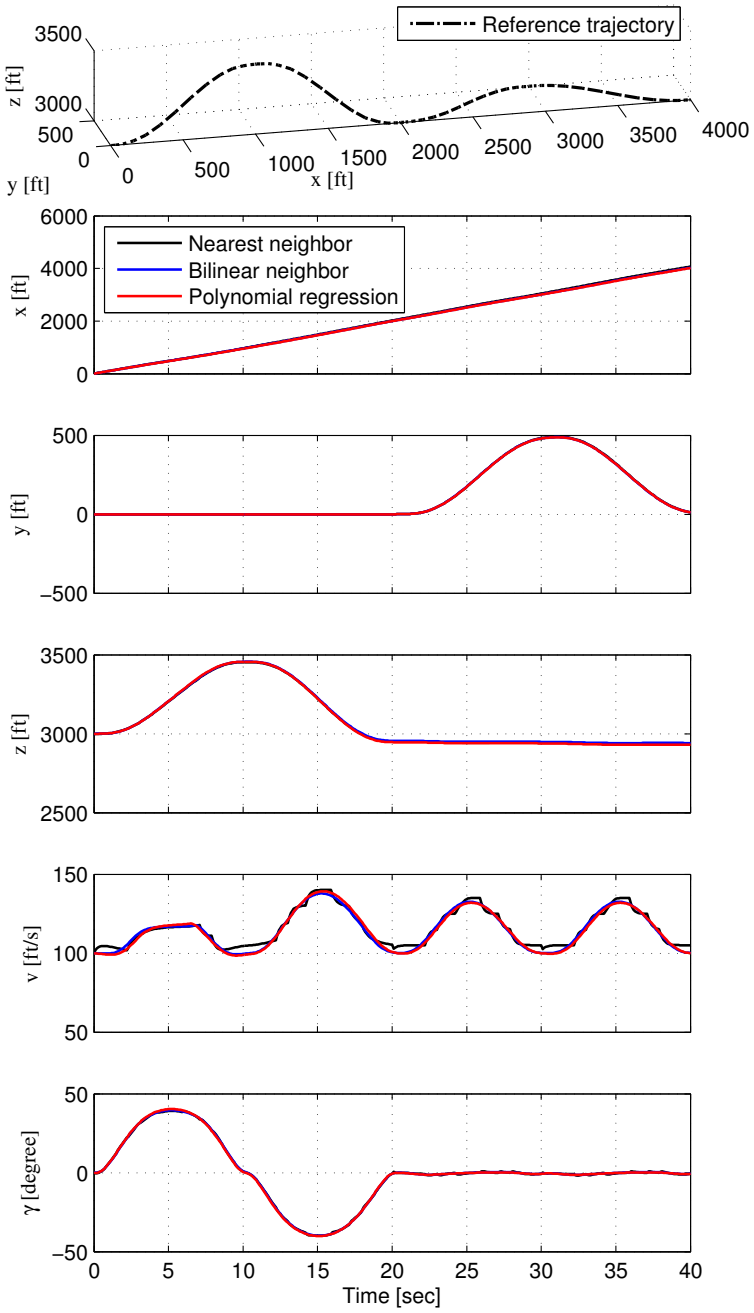



**Figure 12:** Time response

## *2.4 Conclusion*

Analysis of increasingly complex operational airspace concepts, such as separation assurance functions for unmanned aircraft, require modeling and simulation capabilities with breadth of scope to capture multi-agent interactions and depth of scope to accurately model aircraft flight dynamics, navigation, and control governing said interactions. Runtime improvements without loss of accuracy or fidelity are paramount enablers. The global polynomial approximation proposed for variable scheduling of controller gain and trim inputs replaces conventional local approximation via nearest neighbor or bilinear interpolation algorithms. The hypothesis presented contends that, relative to the two conventional methods considered, the proposed method concurrently provides improvements in computational cost and controller stability. To test this hypothesis we conduct numerical flight dynamics simulations with the three variable scheduling methods for four aircraft maneuvers using scheduling variable solution sets with 16, 25, 64, and 100 points. Computational runtime is recorded for each simulation case and used to conduct a comparative assessment across methods. Results support our hypothesis and show that the proposed global polynomial method reduces runtime by a factor of 2.6 over the nearest neighbor approximation, and by a factor of 5.4 over bilinear interpolation. In the interest of transparency and fairness we account for the computational cost of the polynomial regression in off-line runtime. We find that the penalty is negligible, in the order of 0.1 [s]. Assessment of controller stability attained with each method is measured as the deviation of multivariate gain and phase margin from optimal controller values. Margin deviation is statistically assessed using a uniform $100 \times 100$ sampling in the scheduling variable space, and repeated for the four scheduling variable solution sets. Qualitative inspection of box plots readily reveals that the margin deviation is smallest for the proposed method for any given number of reference solution points. Moreover, the global polynomial produces practically no deviation from optimal solution with a 25

point reference data sample whereas the bilinear interpolation method only begins to reach this level of stability with the 100 point reference sample. Overall the evidence produced through numerical flight dynamics simulations support the hypothesis and characterize the improvements of the proposed method over conventional approaches. This statistical gain-scheduling technique is especially more efficient for the multi-UAV problem because each vehicle employs the proposed gain-scheduling structure. For example, if n UAVs are simulated, the computational improvement is n times than a single UAV simulation. However, in the actual hardware implementation, this proposed technique can have a limited improvement depending on the on-board hardware processor. For instance, if the on-board hardware process is powerful, the impact of the proposed gain-scheduling technique may not be significant to improve the computational process.

# CHAPTER III

# COLLISION AVOIDANCE ALGORITHM USING OPTIMAL CONTROL THEORY

Trajectory optimization is a method to compute a trajectory with the minimum cost function (i.e., Performance index) that satisfies dynamic constraints and initial/terminal/boundary conditions. The optimal trajectory is widely utilized for diverse applications in autonomous vehicles, launch vehicles, and aircraft. This chapter provides an overview of the fundamental optimal trajectory problem and existing numerical methods. Based on the optimal trajectory problem, we introduce a mathematical formulation of the optimal collision avoidance algorithm.

## *3.1 General optimal trajectory problem*

This section provides the overview of the trajectory optimization problem based on Bolza problem with an unspecified final time. The objective of the trajectory optimization is to identify the trajectory with a minimum cost and to satisfy all constraints. To compute the optimal trajectory, we define a performance index. The performance index (cost function) to find the optimal trajectory in $[t_0 \quad t_f]$ can be written as

$$Min\ J(\mathbf{u}) = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t)dt + \Psi(\mathbf{x}(t_f), t_f), \tag{28}$$

where $L$ is a transient cost function (i.e., Lagrange cost or running cost) and $\Psi$ is a terminal cost function (i.e., Mayor cost). $\mathbf{x}(t)$ is a vehicle state vector, $\mathbf{u}(t)$ is a control input, $t_0$ is an initial time, and $t_f$ is a terminal time. Vehicle state are captured as dynamic constraints. The vehicle dynamic constraints can be defined as a non-linear

differential equation:

$$\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{u}(t), t) \tag{29}$$

The dynamic equation is $F : [t_0 \;\; t_f] \times D \times U \rightarrow R^n$, the states are $D \subseteq R^n$, and the control inputs are $\mathbf{u} \in U_{adm} = \mathbf{u}[t_0 \; t_f] \rightarrow R^m, PWC(Piece\,Wise\,Continuous), \mathbf{u}(t) \in U, \forall t \in [t_0 \;\; t_f]$. $U_{adm}$ is an admissible control input.

The boundary conditions are $\mathbf{x}(t_0) = \mathbf{x}_0 \subseteq D$, and $\mathbf{x}(t_f) = \mathbf{x}_f \subseteq D$. The state constraints can be stated as

$$\begin{aligned} &\mathbf{x}(t_0) \in \mathrm{X}_0 \subseteq D \\ &\mathbf{x}(t_i) \in \mathrm{X}_i \subseteq D, \; i = 1, 2, ..., k \\ &\mathbf{x}(t) \in \mathrm{X}_i \subseteq D, \; i = 1, 2, ..., k, \; \forall t \in [t_0, t_1] \end{aligned} \tag{30}$$

In Equation 30, the first state constraint is the initial time constraint, and the second state constraints indicate a mid-point constraint. The last constraints are the general state constraints for all time. From the given cost function, dynamic constraints, states constraints and boundary conditions, and the necessary conditions for the optimal trajectory can be derived from the calculus of the variations [85] [27] [93]. The necessary conditions can be different mathematical forms depending on the given boundary conditions and constraints. This paper focuses on the trajectory optimization with given initial/terminal conditions and an unspecified terminal time that can be formulated as the optimal trajectory problem with a free final time. The augmented performance index to derive the necessary conditions can be expressed as follows

$$\hat{J}(\mathbf{x}, \mathbf{u}, \lambda, t) = \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t) + \lambda^T (f(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}}) dt + \Psi(\mathbf{x}(t_f), t_f) \tag{31}$$

In the unspecified final time case, the variations by the admissible weak variation $\mathbf{v}$ are $\mathbf{u}+\epsilon\mathbf{v}$ and $t_f+\epsilon\tau$ over $[0 \; t_f+\epsilon\tau]$. Note that the perturbed control is $\mathbf{u}+\epsilon\mathbf{v} \in U_{adm}$.

The variation from the incremental is

$$\delta\hat{J}(\mathbf{x},\mathbf{u},\lambda,t)=\frac{1}{\epsilon}\lim_{\epsilon\to 0}[\hat{J}(\mathbf{x}+\epsilon\eta,\mathbf{u}+\epsilon\mathbf{v},\lambda,t)-\hat{J}(\mathbf{x},\mathbf{u},\lambda,t)]. \tag{32}$$

The variation equation can be rewritten using the augmented performance index in Equation 31.

$$\delta\hat{J}(\mathbf{x},\mathbf{u},\lambda,t)=\frac{1}{\epsilon}\lim_{\epsilon\to 0}[\int_{t_0}^{t_f+\epsilon\tau}(L(\mathbf{x}+\epsilon\eta,\mathbf{u}+\epsilon\mathbf{v},t))\,dt+\lambda^T(f(\mathbf{x}+\epsilon\eta,\mathbf{u}+\epsilon\mathbf{v},t)-\dot{\mathbf{x}}-\epsilon\dot{\eta})dt$$
$$+\Psi(\mathbf{x}(t_f+\epsilon\tau),\epsilon\eta(t_f+\epsilon\tau),tf+\epsilon\tau)-\int_{t_0}^{t_f}\left(L(\mathbf{x},\mathbf{u},t)dt+\lambda^T(f(\mathbf{x},\mathbf{u},t)-\dot{\mathbf{x}})\right)dt+\Psi(\mathbf{x}(t_f),t_f)] \tag{33}$$

The first integral term can be divided into two integral terms that are from the initial time $t_0$ to the final time $t_f$, and from the final time $t_f$ to the final time plus the perturbed time $t_f+\epsilon\tau$:

$$\delta\hat{J}(\mathbf{x},\mathbf{u},\lambda,t)=$$
$$\frac{1}{\epsilon}\lim_{\epsilon\to 0}[\int_{t_0}^{t_f}\left(L(\mathbf{x},\mathbf{u},t)+\epsilon\frac{\partial L}{\partial x}\eta+\epsilon\frac{\partial L}{\partial u}\mathbf{v}+\lambda^T(f(\mathbf{x},\mathbf{u},t)+\epsilon\frac{\partial f}{\partial x}\eta+\epsilon\frac{\partial f}{\partial u}\mathbf{v})-\dot{\mathbf{x}}-\epsilon\dot{\eta}\right)dt+\mathbf{o}_1(\epsilon)$$
$$+\int_{t_f}^{t_f+\epsilon\tau}(L(\mathbf{x}+\epsilon\eta,\mathbf{u}+\epsilon\mathbf{v},t))\,dt+\lambda^T(f(\mathbf{x}+\epsilon\eta,\mathbf{u}+\epsilon\mathbf{v},t)-\dot{\mathbf{x}}-\epsilon\dot{\eta}))dt+$$
$$\Psi(\mathbf{x}(t_f+\epsilon\tau),\epsilon\eta(t_f+\epsilon\tau),t_f+\epsilon\tau)$$
$$-\int_{t_0}^{t_f}\left(L(\mathbf{x},\mathbf{u},t)dt+\lambda^T(f(\mathbf{x},\mathbf{u},t)-\dot{\mathbf{x}})\right)dt+\Psi(\mathbf{x}(t_f),t_f)] \tag{34}$$

In Equation 34, $\mathbf{o}_1(\epsilon)$ presents high order terms in the first integral term. In Equation 34, the second integral can be rewritten using the Taylor series expansion

$$\delta\hat{J}(\mathbf{x},\mathbf{u},\lambda,t)=$$
$$\frac{1}{\epsilon}\lim_{\epsilon\to 0}[\int_{t_0}^{t_f}\left(L(\mathbf{x},\mathbf{u},t)+\epsilon\frac{\partial L}{\partial x}\eta+\epsilon\frac{\partial L}{\partial u}\mathbf{v}+\lambda^T(f(\mathbf{x},\mathbf{u},t)+\epsilon\frac{\partial f}{\partial x}\eta+\epsilon\frac{\partial f}{\partial u}\mathbf{v})-\dot{\mathbf{x}}-\epsilon\dot{\eta}\right)dt+\mathbf{o}_1(\epsilon)$$
$$+\epsilon\tau(L(\mathbf{x},\mathbf{u},t))+\mathbf{o}_2(\epsilon)+\Psi\left(x(t_f)+\epsilon(\frac{\partial x}{\partial t})|_{t_f}\tau+\eta(t_f))+\mathbf{o}_3(\epsilon),t_f+\epsilon\tau\right)$$
$$-\int_{t_0}^{t_f}(L(\mathbf{x},\mathbf{u},t)dt+\lambda^T(f(\mathbf{x},\mathbf{u},t)-\dot{\mathbf{x}}))dt+\Psi(\mathbf{x}(t_f),t_f)], \tag{35}$$

where $\mathbf{o}_2(\epsilon)$ and $\mathbf{o}_3(\epsilon)$ are high order terms resulting from the Taylor series expansion. By reorganizing the equation, the variation equation is

$$\begin{aligned}
&\delta\hat{J}(\mathbf{x},\mathbf{u},\lambda,t) = \\
&\frac{1}{\epsilon}\lim_{\epsilon\to 0}[\int_{t_0}^{t_f}\left(L(\mathbf{x},\mathbf{u},t)+\epsilon\frac{\partial L}{\partial x}\eta+\epsilon\frac{\partial L}{\partial u}\mathbf{v}+\lambda^T(f(\mathbf{x},\mathbf{u},t)+\epsilon\frac{\partial f}{\partial x}\eta+\epsilon\frac{\partial f}{\partial u}\mathbf{v})-\dot{\mathbf{x}}-\epsilon\dot{\eta}\right)dt+\mathbf{o}_1(\epsilon) \\
&\quad+\epsilon\tau(L(\mathbf{x},\mathbf{u},t))+\mathbf{o}_2(\epsilon)+\Psi(x(t_f),t_f)+\epsilon\frac{\partial\Psi}{\partial x}\left(\frac{\partial x}{\partial t}|_{t_f}\tau+\eta\right)|_{t_f}+\epsilon\frac{\partial\Psi}{\partial t_f}\tau+\mathbf{o}_4(\epsilon) \\
&\quad-\int_{t_0}^{t_f}(L(\mathbf{x},\mathbf{u},t)dt+\lambda^T(f(\mathbf{x},\mathbf{u},t)-\dot{\mathbf{x}}))dt+\Psi(\mathbf{x}(t_f),t_f)]. \quad (36)
\end{aligned}$$

$\mathbf{o}_4(\epsilon)$ entails high order terms with the terminal cost. For the simplification of the variation equation, we assume that the high order terms are small, which means negligible. From this assumption, the final simplified equation can be therefore rewritten as follows

$$\begin{aligned}
&\delta\hat{J}(\mathbf{x},\mathbf{u},\lambda,t) = \\
&\int_{t_0}^{t_f}\left[\left(\frac{\partial L}{\partial x}+\lambda^T\frac{\partial f}{\partial x}+\dot{\lambda}^T\right)\eta+\left(\frac{\partial L}{\partial u}+\lambda^T\frac{\partial f}{\partial u}\right)\mathbf{v}\right]dt-\lambda^T(t_f)\eta(t_f)+\lambda^T(0)\eta(0) \\
&\quad+\tau L|_{t_f}+\frac{\partial\Psi}{\partial x}\left(\frac{\partial x}{\partial t}\tau\right)|_{t_f}+\frac{\partial\Psi^T}{\partial x}\eta(t_f)+\frac{\Psi}{\partial t_f}\tau|_{t_f} \quad (37)
\end{aligned}$$

From the final variation equation, the first order necessary conditions can be stated as

$$\frac{\partial L}{\partial u}+\lambda^T\frac{\partial f}{\partial u}=0 \quad (38)$$

$$\dot{\lambda}^T=-\left(\frac{\partial L}{\partial x}+\lambda^T\frac{\partial f}{\partial x}\right) \quad (39)$$

$$\lambda^T(t_f)=\frac{\partial\Psi^T}{\partial x}|_{t_f} \quad (40)$$

$$\left(L+\frac{\partial\Psi^T}{\partial x}\left(\frac{\partial x}{\partial t}\right)+\frac{\partial\Psi}{\partial t_f}\right)|_{t_f}=0 \quad (41)$$

The term $L+\lambda^T f$ that is named Hamiltonian $H$. The first order necessary conditions can also be expressed with respect to Hamiltonian $H$.

$$\frac{\partial H}{\partial u}=0 \quad (42)$$

$$\dot{\lambda}^T = -\frac{\partial H}{\partial x} \tag{43}$$

$$\lambda^T(t_f) = \frac{\partial \Psi^T}{\partial x}|_{t_f} \tag{44}$$

$$\left(H + \frac{\partial \Psi}{\partial t_f}\right)|_{t_f} = 0 \tag{45}$$

In the first necessary conditions, Equation 39 and Equation 43 are called the costate equation or the adjoint equation. Equation 41 and Equation 45 are the transversality condition.

In general engineering systems such as an autonomous vehicle, and manipulate systems, states and control inputs are bounded by constraints. For example, a thrust of spacecraft cannot exceed over a maximum thrust. The maximum climb angle for a commercial aircraft can be limited due to safety reasons. Therefore, we discuss solving a constrained optimal trajectory problem.

If Hamiltonian can be a linear function of a control input $u$, the optimal controller does not exist in the first order necessary condition because the optimal controller $\frac{\partial H}{\partial u}$ is not a function of $u$. In this case, the Pontryagin's minimum principle (i.e., Pontryagin's maximum principle) is a way to generalize the optimal control problem with control and state constraints. If the control is bounded to ($u_{min} \leq u \leq u_{max}$), Pontryagin's minimum principle:

$$H^* = H^*(x^*(t), u^*(t), \lambda^*(t), t) \leq H^*(x^*(t), u(t), \lambda^*(t), t) \tag{46}$$

$$u^* = argmin \quad H^*(x^*(t), u(t), \lambda^*(t), t), \quad u \in u_{adm}, \quad \in [t_0, t_1], \tag{47}$$

where $x^*(t)$ is an optimal state, $u^*(t)$ is an optimal control and $\lambda^*(t)$ is an optimal Lagrange multiplier. The Pontryagin theory implies that if a control is bounded, the control input to minimize Hamiltonian is optimum. This theory can also be applicable to the problem with unbounded controls [85]. The Pontryagin's minimum principle approach has an issue when Hamiltonian is linearly dependent on the controller $u$. The optimal controller cannot be specified because the necessary condition of the optimal

controller is not a function of the control input $u$. This case is called *singular control* or *singular arc*. To determine the optimal control $u^*$, one approach is repeated to differentiate $\partial H/\partial u$ until the control input $u$ appears. To be the optimal singular controller, the control input $u$ should meet the following necessary conditions:

$$(-1)^k \frac{\partial}{\partial u}\left[\left(\frac{d}{dt}\right)^{2k}\left(\frac{\partial H}{\partial u}\right)\right] \geq 0 \quad k = 0, 1..., \tag{48}$$

where $m = 2k$ is the order of derivative of $\partial H/\partial u$ and k is the order of singular arc. This condition is named $Kelley$'s condition

The function of Legendre necessary condition is to find a weak local minimizer of the cost function $J(t, x, \dot{x})$ [152]. This Legendre necessary condition is weaker than Weierstrass necessary condition. The Legendre condition is a more practical approach because of its simpler mathematical form. The Legendre necessary condition is that if $x^*$ is a weak local minimizer of the cost function $J(t, x, \dot{x})$, then

$$L_{\dot{x}\dot{x}}(t, x^*, \dot{x}^*) \geq 0 \quad \forall t \in [t_0, t_f] \tag{49}$$

From the calculus of variations and the necessary condition of the cost function, $J(t, x, \dot{x})$ can be written as

$$J(\eta) = \int_{t_0}^{t_f} L_{xx}\eta^2 + 2L_{x\dot{x}}\eta\dot{\eta} + L_{\dot{x}\dot{x}}\dot{\eta}^2 dt \geq 0 \tag{50}$$

Legendre condition defines the candidate solutions. To determine the minimizer, we can consider the following Jacobi condition defined as

$$\frac{d}{dt}\left(L_{x\dot{x}}\eta + L_{\dot{x}\dot{x}}\dot{\eta}\right) = L_{x\dot{x}}\dot{\eta} + L_{xx}\eta \tag{51}$$

If this Jacobi condition is satisfied, conjugate points will not be appeared as/on $[t_0\ t_f]$, which is the weak local minimizer. The sufficient condition of the cost function $J(t, x, \dot{x})$ is that a weak local minimizer $x^*$ should be a smooth function. This smooth function should also satisfy the following three conditions: Euler-Lagrange equation, Legendre condition, and Jacobi condition.

In the constrained optimization problem, the cost function of the continuous optimal trajectory in $[t_0 \;\; t_f]$ without any terminal cost function can be written as

$$Min\ J(\mathbf{u}) = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t)dt, \tag{52}$$

where $L$ is a transient cost function and the final time is free. The vehicle dynamics can be defined with non-linear differential equation:

$$\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{u}(t), t) \tag{53}$$

In this equation, the function is $F : [t_0 \;\; t_f] \times D \times U \rightarrow R^n$, the states are $D \subseteq R^n$, the control inputs are $\mathbf{u} \in U_{adm} = \mathbf{u}[t_0 \;\; t_f] \rightarrow R^m, \;\; PWC, \;\; \mathbf{u}(t) \in U, \forall t \in [t_0 \;\; t_f]$. If the control input $u$ is a local minimizer of the cost function $J(u)$, the co-state equation will satisfy the following function:

$$\dot{\lambda} = -\frac{\partial H^T}{\partial x}, \;\; t \in [t_0 \;\; t_f] \tag{54}$$

The co-state at the final time is $\lambda(t_f) = 0$ since the states $\mathbf{x}$ at the terminal time are free. If the optimal control input equation is zero, the following condition can be defined

$$\frac{\partial H}{\partial u} = 0. \tag{55}$$

The second order condition should also be considered to specify the optimal controller. The condition to be a local minimizer of $J(u)$ is

$$\delta^2 J(u) = \frac{1}{2} \int_{t_0}^{t_f} \left[ \delta \mathbf{x}^T H_{xx}(t) \delta x + 2\delta \mathbf{x}^T H_{xu}(t) v + v^T H_{uu}(t) v \right] dt \geq 0 \tag{56}$$

A necessary condition of the optimal controller is

$$v^T H_{uu}(x(t), u(t), \lambda(t)) v \geq 0 \tag{57}$$

This necessary condition in the equation 57 is called the $Legendre-Clebsch$ condition. To solve the second order variation of the cost function, the additional necessary conditions should be considered. From the equation 56 when the control input $u$ is a

local minimizer, the admissible control variation $v$ should not be negative. This fact makes the *accessory minimization problem*.

$$J_{acc}(v) = \frac{1}{2}\int_{t_0}^{t_f} \left[\delta \mathbf{x}^T H_{xx}(t)\delta x + 2\delta \mathbf{x}^T H_{xu}(t)v + v^T H_{uu}(t)v\right] dt \tag{58}$$

The dynamic equation is

$$\delta \dot{\mathbf{x}} = A(t)\delta \mathbf{x}(\mathbf{t}) + B(t)u(t). \tag{59}$$

The boundary condition is $\delta x(0) = 0$. The Hamiltonian of the accessory minimization problem is

$$H = \frac{1}{2}\delta \mathbf{x}^T H_{xx}(t)\delta x + \delta x^T H_{xu}(t)v + \frac{1}{2}v^T H_{uu}(t)v + \lambda^T(A(t)\delta \mathbf{x}(\mathbf{t}) + B(t)u(t)). \tag{60}$$

The optimal control is given by

$$\frac{\partial H}{\partial u} = H_{xu}^T(t)\delta x + B^T(t)\lambda + H_{uu}(t)u. \tag{61}$$

The adjoint equation is

$$\dot{\lambda}^T = -\frac{\partial H}{\partial \delta x} = -H_{xx}(t)\delta x - H_{xu}(t)u - A(t)^T\lambda \tag{62}$$

It is assumed that $H_{uu}$ satisfies $Legendre - Clebsh$ condition in Equation 57. From Equations 59, 61 and 62, the state space equation can be represented as

$$\begin{Bmatrix} \delta \dot{\mathbf{x}} \\ \delta \dot{\lambda} \end{Bmatrix} = \begin{bmatrix} A - BH_{uu}^{-1}H_{xu}^T & -BH_{uu}B^T \\ -(H_{xx} - H_{xu}H_{uu}^{-1}H_{xu}^T) & -(A^T - H_{xu}^{-1}B^T) \end{bmatrix} \begin{Bmatrix} \delta \mathbf{x} \\ \delta \lambda \end{Bmatrix} \tag{63}$$

The boundary conditions are $\delta \mathbf{x}(0) = 0$ and $\delta\lambda(t_f) = 0$. The matrix form can be rewritten with respect to a state transition matrix through solving the homogeneous solution of the different equation

$$\dot{\mathbf{\Phi}}(t) = \mathbf{M}(t)\mathbf{\Phi}(0), \tag{64}$$

where the state transition matrix is $\mathbf{\Phi}(t) = [\delta x(t) \quad \delta\lambda(t)]^T$. In this matrix form, if $detM_{12}(t_c)$ is zero, $t_c \in (0 \ \ t_f)$, the $t_c$ is a conjugate point. In other words, the control

input $u$ is not a local minimizer in the optimal trajectory problem. For the optimal trajectory problem with unspecified final time, the final time needs to be specified. This final time can be computed from the first order necessary conditions mentioned from Equations 42 to Equation 45.

### 3.1.1 Numerical method

This section discusses optimal solution methodologies to find an optimal trajectory problem described in previous section. Figure 13 illustrates the representative methods for solving the optimal trajectory problem.

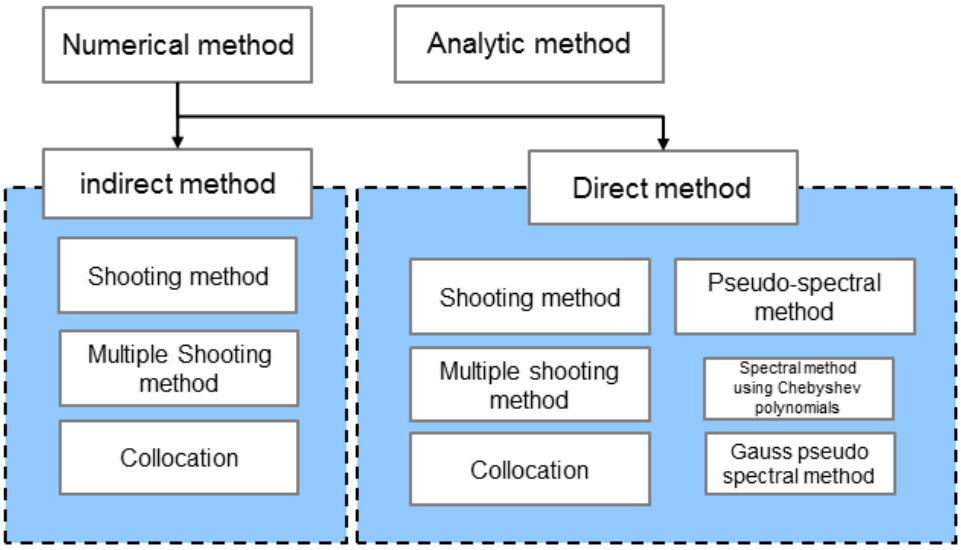



**Figure 13:** Methodologies for solving an optimal control problem

The first order necessary conditions of the optimal trajectory problem can be defined from the calculus of variations and Pontryagin's maximum principle, which is based on a cost function, initial conditions and constraints [85][27][19]. This first order necessary conditions can convert the optimal control problem into two points boundary value problem (TBVP). This two points boundary value problem can be solved by an analytic approach or numerical techniques. Most analytic techniques for the optimal trajectory is not solvable because of the mathematical complexity derived

from the first order necessary conditions and the sufficient conditions [24][27]. Practical engineering problems like an optimal trajectory for a launch vehicle to reach Mars or other planets cannot be solved analytically due to their complex dynamics and constraints. Consequently, the iterative numerical approach is a suitable approach to overcome the complexity issue of the analytic solution. The numerical methods have two types of techniques which are an indirect method and a direct method [19][149][124]. The indirect method requires the analytic expression about the first order necessary conditions of the optimal trajectory problem from the calculus of variations. From the first order necessary conditions, the indirect approach solves the candidate optimal trajectory solutions, which are called extremals, based on the given boundary conditions and constraints. The final optimal trajectory is one of the candidate solutions to satisfy the necessary conditions with the lowest cost. The advantages of the indirect method are that the final optimal solution is highly accurate, and the solution does not violate the necessary conditions since the candidate solutions satisfy the first order necessary conditions. However, this indirect technique has several drawbacks. The first drawback is that solving the first order necessary conditions is not trivial because the first order necessary conditions must be solved analytically, not numerically. Second drawback is a small convergence bound. In other words, initial guesses can hugely affect the convergence of the final optimal trajectory solution since gradient methods (ex. steepest descent algorithm) or Newton method is commonly employed for the trajectory optimization [85]. The guesses for the initial co-states' values are also very difficult because the co-states do not have physical meanings. Moreover, the designer of the optimal trajectory has to derive a switching structure from the given constraints. Identifying the switching structure may not be trivial depending on the optimal trajectory problem complexity. Notable examples of the indirect method are the shooting method, multiple shooting method and indirect collocation method [83][122][123][124].

The direct method converts the continuous optimal trajectory problem into a finite discrete optimal trajectory problem with algebraic constraints. The direct method is also named Nonlinear Program (NLP). The benefits of the direct method are that it does not require the analytic solution for the first order necessary condition. In addition, unlike the convergence issue of the indirect method, the convergence area is larger. Namely, the level of the convergence is less related to the initial guess for the initial states or initial co-states. There are two types of direct methods. One is the method to parameterize control input. The direct shooting method and the direct multiple shooting method are included in this category [124]. The other method is a collocation method using a spectral method. Examples of the spectral method are the spectral method using Chebyshvy polynomial, Legendre pseudo spectral method, Legendre polynomials, Legendre pseudo spectral method [148][147][124]. Stryk proposed a hybrid approach to have both benefits of the direct and indirect method [149]. The hybrid approach combines the large convergence feature and the multiple shooting strategy with the high accuracy. The hybrid approach is verified from the examples of the Brachistochrone problem and the Apollo reentry problem [149]. In the optimal trajectory problem, the co-state estimation is critical because the estimated co-state is applied to verify the optimality conditions, mesh refinement and sensitivity analysis. The co-state can be estimated from solving approximation to the co-state dynamics from a post-processing or the method based on the relationship of KKT (Karush-Kuhn-Tucker) multipliers in NLP and the continuous co-states from the sensitivity analysis. The recent technique to estimate co-state is a pseudo spectral method, which the co-state maps a principle from the relationship between KKT multipliers of NLP and co-state estimations. The drawback of this technique is that the boundary points cannot be held from the KKT multipliers and the co-state estimations. Therefore, it may not satisfy the co-state dynamics or boundary

conditions. To solve this problem, Benson suggested Gauss pseudo-spectral transcription optimal control [19]. In details, this method uses a direct transcription method through a parameterization technique for the states and the controls by a global polynomial collocated at Gauss points. This technique produces more precise co-state estimation. To be more specific, the KKT conditions of NLP are exactly equivalent to the first order necessary condition from the discretization. Consequently, this Gauss pseudo-spectral technique has the benefits of both of the direct method and indirect method. This method is also very stable and robust, and in spite of the unidentified switching structure, this method provides the result of the trajectory optimization. Lastly, this convergence speed is exponentially fast. Because of these benefits, the Gauss Pseudospectral Method (GPM) provided by the open-source software GPOPS is implemented as a numerical solver for the optimal collision avoidance problem. The Gauss pseudo-spectral method is suggested by Benson, and it is advanced and validated from several practical case studies performed by Huntington, et al [19][20][62][61][63]. This technique uses an orthogonal collocation method based on the Legendre-Gauss points. GPOPS software provides MATLAB interface with the non-linear programming problem solver SNOPT [54][55].

#### *3.1.1.1 Gauss Pseudospectral method*

In this section, the *Gauss pseudospectral method* will be introduced. The Gauss pseudo-spectral method is proposed by Benson and extended by Huntington [19][20][62]. To formulate an optimal trajectory problem, the Bolza optimal control problem will be stated. In the Bolza optimal control problem, the performance index can be represented as follows:

$$Min\ J(\mathbf{u}) = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t)dt + \Psi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f), \tag{65}$$

where $L$ and $\Psi$ are respectively a transient cost function and a terminal cost function. The vehicle dynamics can be defined with non-linear differential

$$\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{u}(t), t) \tag{66}$$

The dynamic constraint $F \rightarrow R^n$ in $t \in [t_0 \ t_f]$, the states are $D \subseteq R^n$, and the control inputs are $\mathbf{u} \in U_{adm}$ and $\mathbf{u}[t_0 \ \ t_f] \rightarrow R^m$. The boundary conditions at the initial time and terminal time can be expressed as

$$\Phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) = 0 \tag{67}$$

The constraints can be written as

$$C(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0, \quad t \in [t_0 \ \ t_f] \tag{68}$$

The Gauss pseudo-spectral method for solving the continuous Bolza optimal trajectory problem described from Equation 65 to Equation 68 requires fixed time interval for a Nonlinear Program problem. Namely, Bolza optimal problem can be transformed into the optimal control problem with the fixed time according to the following transformation algebraic equation.

$$\tau = \frac{2t}{t_f - t_0} - \frac{t_f + t_0}{t_f - t_0} \tag{69}$$

This algebraic equation transforms the continuous optimal control problem in $t \in [t_0 \quad t_f]$ into the optimal control problem with the fixed time $\tau \in [-1 \quad 1]$. The transformed discrete cost function of Bolza optimal problem is given as

$$Min \ J(\mathbf{u}) = \frac{t_f - t_0}{2} \int_{\tau_0}^{\tau_f} L(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) dt + \Psi(\mathbf{x}(\tau_0), t_0, \mathbf{x}(\tau_f), t_f) \tag{70}$$

The dynamic constraints can be transformed to

$$\frac{2}{t_f - t_0} \frac{d\mathbf{x}}{d\tau} = F(\mathbf{x}(\tau), \mathbf{u}(\tau), t_0, t_f) \tag{71}$$

The transformed boundary condition and constraints can be written as

$$\Phi(\mathbf{x}(\tau_0), t_0, \mathbf{x}(\tau_f), t_f) = 0 \tag{72}$$

$$C(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) \leq 0 \tag{73}$$

The above equations are called *transformed continuous Bolza problem*. The next step of the Gauss pseudo spectral method is making a discretized mathematical formula, known as Nonlinear Program (NLP), from the transformed continuous Bolza problem described from Equations 70 to 73. The discretized equations for the state, the control and the co-state functions can be defined by the approximation technique of Lagrange interpolation polynomial. The Lagrange interpolation equations $L_i(\tau)$ are given as

$$L_i(\tau) = \prod_{j=0, j\neq i}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j}, \quad (i = 1, ..., N) \tag{74}$$

The differential equation of the Lagrange interpolating polynomial can be written as

$$\dot{L}_i(\tau) = \sum_{i=0}^{N} \frac{\prod_{j=0, j\neq i}^{N} (\tau_k - \tau_j)}{\prod_{j=0, j\neq i}^{N} (\tau_i - \tau_j)}, \quad (i = 1, ..., N) \tag{75}$$

The state and control equations can be derived from the previous Lagrange interpolation equation and the differential equation of Lagrange polynomial equation. The approximated state equation is

$$\mathbf{x}^L(\tau) = \sum_{i=0}^{N} \mathbf{x}(\tau_i) L_i^{\mathbf{x}}, \quad (i = 1, ..., N), \tag{76}$$

where $\mathbf{x}^L$ is the approximated state, and $L_i^{\mathbf{x}}$ is the Lagrange interpolation function for the state. The derivative of the state equation is

$$\dot{\mathbf{x}}^L(\tau) = \sum_{i=0}^{N} \mathbf{x}(\tau_i) \dot{L}_i^{\mathbf{x}}, \quad (i = 1, ..., N) \tag{77}$$

where $\dot{L}_i^{\mathbf{x}}$ is the derivative of the Lagrange polynomial, which is be written in Equation 75. The control equation can be approximated in the same manner of the state approximation using Lagrange interpolating polynomial.

$$\mathbf{u}^L(\tau) = \sum_{i=0}^{N} \mathbf{u}(\tau_i) L_i^{\mathbf{u}}, \quad (i = 1, ..., N) \tag{78}$$

The two Lagrange polynomial equations approximating the state $L_i^{\mathbf{x}}$ and the control $L_i^{\mathbf{u}}$ satisfy the following properties.

$$L_i^{\mathbf{x}}(\tau_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{79}$$

$$L_i^{\mathbf{u}}(\tau_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{80}$$

The previous discussion of the approximation technique can be provided to define the formal Gauss pseudo-spectral discretization. The cost function can be written as Equation 70

$$Min\ J(\mathbf{u}) = \frac{t_f - t_0}{2} \sum_{k=1}^{K} w_k L(\mathbf{x}_k, \mathbf{u}_k, \tau_k; t_0, t_f) dt + \Psi(\mathbf{x}(\tau_0), t_0, \mathbf{x}(\tau_f), t_f) \tag{81}$$

The integral part of the cost function is transformed by the quadrature approximation (Gaussian quadrature) which makes a weighted sum of the function at a specified point within the integration range [139]. Dynamic constraints can be restated from Equation 71 and Equation 77 as a residual equation.

$$\mathbf{R}_k = \sum_{i=0}^{K} \mathbf{x}(\tau_i) \dot{L}_i^{\mathbf{x}}(\tau) - \frac{t_f - t_0}{2} f(\mathbf{x_k}, \mathbf{u}_k, \tau_k; t_0, t_f) = 0 \tag{82}$$

The final states of the continuous form is

$$\mathbf{x}(t_f) = \mathbf{x}(t_0) + \int_{t_0}^{t_f} f(\mathbf{x}(t), \mathbf{u}_t, t) dt \tag{83}$$

The final states in the fixed time, $t \in [-1 \ \ 1]$, can be transformed as follows

$$\mathbf{x}(\tau_f) = \mathbf{x}(\tau_0) + \frac{t_f - t_0}{2} \int_{-1}^{1} f(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) d\tau \tag{84}$$

Using Gaussian quadrature approximation, the boundary condition of the final state can be expressed as

$$\mathbf{R}_{tf} = \mathbf{x}(\tau_f) - \mathbf{x}(\tau_0) - \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k f(\mathbf{x}_k, \mathbf{u}_k, \tau_k; t_0, t_f) d\tau, \quad (k = 1, ...., N) \tag{85}$$

$$\Phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) = 0 \tag{86}$$

The constraints can be written as

$$C(\mathbf{x}_k, \mathbf{u}_k, \tau_k; t_0, t_f) \leq 0, \quad (k = 1, ..., N) \tag{87}$$

The discretization points are a set of Legendre-Gauss points, which is the roots of the $K$th degree Legendre polynomial. Consequently, NLP of the optimal Bolza problem is that the cost function is Equation 81, and the algebraic constraints are Equations 82, 85, 86 and 87. The NLP optimal cost function of the Bolza problem can be rewritten to represent the augmented performance index.

$$Min\ \hat{J}(\mathbf{u}) = \Psi(\mathbf{x}(\tau_0), t_0, \mathbf{x}(\tau_f), t_f) + \frac{t_f - t_0}{2} \sum_{k=1}^{K} w_k L(\mathbf{x}_k, \mathbf{u}_k, \tau_k; t_0, t_f) dt \tag{88}$$

$$+ \bar{\Pi}^T \phi(\mathbf{x}(\tau_0), t_0, \mathbf{x}(\tau_f), t_f) - \sum_{k=1}^{K} \bar{\lambda}_k^T \mathbf{C}(\mathbf{x}_k, \mathbf{u}_k, \tau_k; t_0, t_f) \tag{89}$$

$$- \sum_{k=1}^{K} \bar{\mu}_k^T \mathbf{R}_k - \bar{\mu}^T \mathbf{R}_{tf} \tag{90}$$

From the augmented performance index, we can solve the first order necessary conditions for the optimality, which can be defined through the calculus of the variation. The details to derive the first order necessary conditions is well described in the reference [64]. For the Gauss Pseudospectral Method, we implement the trajectory optimization software GPOPS to solve the optimal collision avoidance problem.

## *3.2 Optimal collision avoidance trajectory strategy*

In the previous section, the general optimal trajectory problem and the numerical techniques to solve an optimal trajectory problem are discussed. The introduced optimal trajectory problem is an open-loop control strategy, but it does not have a feedback structure to compensate the tracking error. The three general optimal control structures for the integration of guidance and navigation and an aircraft attitude controller can be implemented.

The first structure is a tracking control scheme without the feedback structure for the compensation of the tracking error. The optimal trajectory defines an optimal controller $\mathbf{u}(t)$ based on the performance index (cost function), initial/terminal conditions, boundary conditions and dynamic constraints. From the definition of the optimal controller $\mathbf{u}(t)$, the optimal reference states $\mathbf{x}_{ref}$ can be specified. The guidance and navigation loop from the optimal trajectory function gives this optimal reference states $\mathbf{x}_{ref}$ to the flight control loop. The flight controller using the reference states information generates the control inputs $\mathbf{u}(t)$ to follow the optimal trajectory path $\mathbf{x}_{ref}$. The advantage of this tracking structure is a computationally efficient and simple since the integrated flight controller does not have the feedback loop. However, due to the non-existence of the feedback scheme when the tracking controller cannot follow accurately, the tracking error will be accumulated.

Another optimal trajectory structure is the optimal state feedback control scheme. This technique is also called a policy optimization technique. The role of the optimal state feedback control is finding an optimal controller input $\mathbf{u}(x)$ instead of finding an optimal controller $\mathbf{u}(t)$. The result of the optimal state feedback controller input $\mathbf{u}(x)$ is Hamilton-Jacobi-Bellman (HJB) equation, which is nonlinear Partial Differential Equation (PDE). This state feedback control approach has a difficulty to solve numerically in a real-time manner due to the high complexity of the nonlinear PDE equation [27]. Therefore, the optimal state feedback control scheme is not an adequate structure for the real-time trajectory problem.

The last approach is *Model Predictive Control* (MPC). This method is also called *Receding Horizon Control* (RHC), or *Moving Horizon Optimal Control*. The model predictive control predicts a future trajectory, future states' vector based on the current states $\mathbf{x}_c$ over time horizon at each sampling instant. This process is repeated at every sampling step. Unlike the tracking control structure, the MPC approach continuously updates a new trajectory over a finite horizon that provides

some levels of robustness against perturbations and modeling uncertainties [16]. This MPC structure commonly applies a simplified dynamic model to the future reference trajectory with constraints so that it can improve a computational efficiency better compared to the optimal state feedback control scheme. Therefore, MPC structures have been implemented in many flight real-time simulations [141][21][34][106][81][105]. For these reasons, the model predictive control structure is adopted to the integrated flight controller.

The block diagram of the overall flight control system is shown in Figure 14. The proposed flight simulation structure have two loops including the flight control logic and the guidance and navigation loop. The flight controller is designed by the proposed statistic gain-scheduling method addressed in the earlier chapter. The guidance and navigation structure will be discussed to explain how to formulate the optimal collision avoidance problem in the following sections.

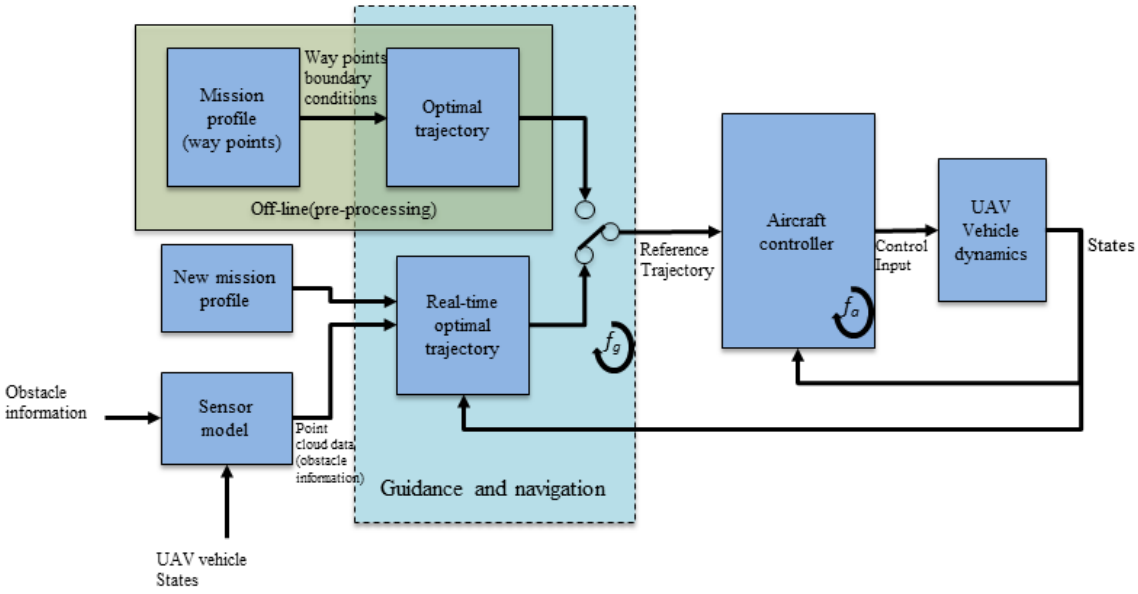



**Figure 14:** Two-layer collision avoidance framework

### 3.2.1 Collision avoidance framework

To formulate a trajectory optimization problem in a collision avoidance situation, we must define the steps of collision avoidance such as detection and avoidance. The sequence of collision avoidance in cooperative and non-cooperative collision problems have been actively researched. The cooperative collision avoidance problem is that unmanned aircraft systems can receive information about fixed or moving obstacles from other systems such as ground stations, chasers, and air traffic controllers. Notable systems of the cooperative avoidance problem that provide obstacle information are the traffic collision avoidance system (TCAS) and the automatic dependent surveillance broadcast (ADS-B) system. Unlike the cooperative collision avoidance problem, the non-cooperative situation is that unmanned aircraft systems receive obstacle information from sub-systems. A common example is on-board sensor systems: LiDAR, radar, and vision types of sensors. Because of complex sequence of these cooperative/non-cooperative collision avoidance problems, we introduce several concepts of the sequence of the collision avoidance from the following literatures.

Andrew[89] addresses a generic process about the steps of conflict detection, awareness of obstacles, and identification of the specific required tasks to achieve separation assurance in cooperative/non-cooperative cases. Yazdi[70] also has constructed an autonomous collision avoidance algorithm based on a comprehensive architecture for collision avoidance situations that include cooperative and non-cooperative avoidance problems suggested by Barfield [17]. The collision avoidance architecture suggested by Yazdi is based on a sphere that characterizes each of the four stages, which are determined by available information from the TCAS and airborne sensors.

Based on the introduced frameworks of a collision avoidance sequence from literature, we establish a comprehensive architecture of an autonomous collision avoidance to formulate an optimal collision avoidance trajectory problem. First, we define two collision avoidance spheres shown in Figure 15. The first, $R_d$, is a detection distance

defined by the maximum distance range of the sensor. The second sphere, $R_s$, is a safe distance that is the required separation distance that the UAV must maintain to protect against the collision. In Figure 15, $AZ$ and $EL$ are the range of the azimuth angle and the elevation angle in the sensor model, respectively.

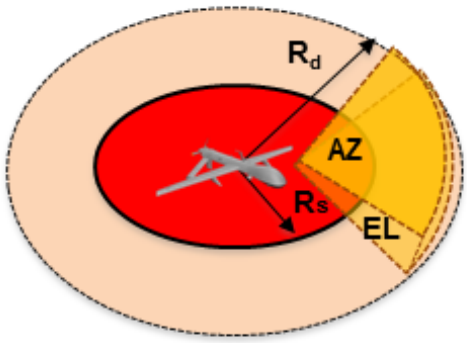



**Figure 15:** Definition of shperes for a collision avoidance framework

Figure 16 depicts the architecture of the collision avoidance framework, which is based on the definitions of two spheres. The new collision avoidance framework is composed of four phases. In the first phase of obstacle detection and awareness, the UAV detects obstacles and collects information about the obstacle locations and sizes from the on-board airborne sensor. Then, based on the sensor data information and UAV states information, the UAV evaluates the possibility of the collision with the obstacle. In the second phase, once the probable collision is identified, the UAV updates a new optimal trajectory based on the UAV dynamic constraints and boundary conditions to avoid the identified obstacle, and then the UAV starts the avoidance mission. In the third phase, after the avoidance mission, the UAV returns to the original path, and in the last phase, returns to its original trajectory.

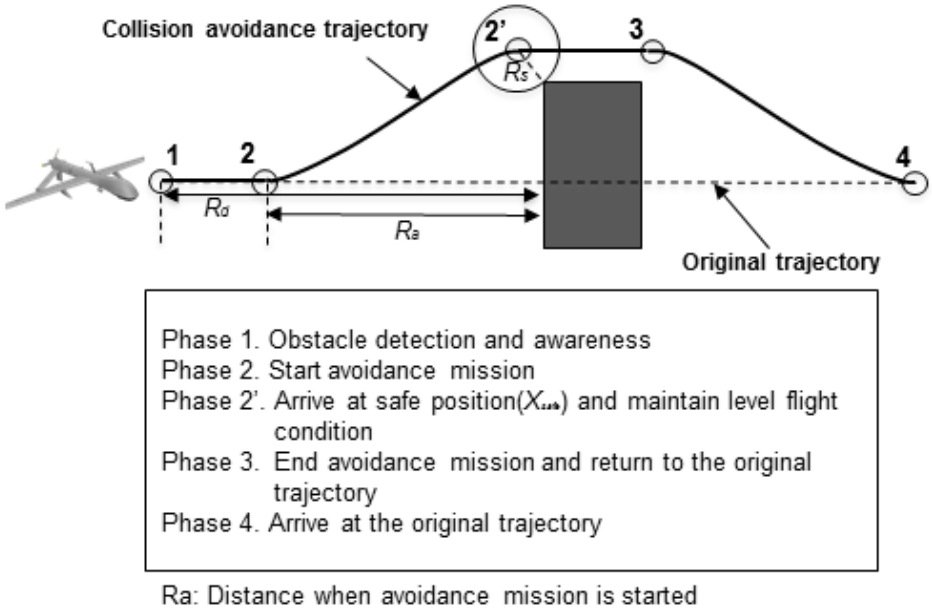



**Figure 16:** Framework of collision avoidance

### 3.2.2 Problem formulation of the optimal collision-free collision avoidance algorithm with minimal effort

This section discusses the formulation of the optimal trajectory problem in a collision avoidance situation. We will introduce benchmark algorithms from literatures, and after observing the advantages and disadvantages of the benchmarking algorithms, we will suggest new collision avoidance algorithms

An objective function of the optimal collision avoidance problem can define a time-optimal trajectory problem, a minimal-effort (minimal-energy) trajectory problem, or the combination of the minimal-effort and time problems with weight factors. The minimal-time approach, which minimizes the duration of time in hostile environments, was employed in a number of studies [140][106][81]. However, the minimal-time approach requires high energy consumption because of aggressive maneuvers in which a UAV quickly escapes from an obstacle. Having consumed most of its energy, a UAV may not be able to complete a given mission. In other words, high energy requirements cause the degradation of a mission feasibility. To avoid this situation, we

adopt the energy-based optimal collision avoidance that minimizes energy consumption. The optimal collision avoidance trajectory problem with a minimal effort can be defined as

$$J = \frac{1}{2}\int_{t}^{t_f} \mathbf{u}^T \mathbf{W} \mathbf{u} \; dt, \tag{91}$$

where $\mathbf{u}$ is the control input vector ($\mathbf{u} \in \mathbb{R}^3$), $\mathbf{W}$ is the $3 \times 3$ weighting matrix, $t$ is the current time, and $t_f$ is the terminal time. The vehicle dynamics is defined through a simplified kinematic model that is the first order approximation model. The first order approximation model for the guidance has been adopted by several researchers [65] [106][81] to improve on-line computational time. The state space model of the approximated vehicle dynamics is

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{a}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{0}}_{3\times3} & \tilde{\mathbf{I}}_{3\times3} & \tilde{\mathbf{0}}_{3\times3} \\ \tilde{\mathbf{0}}_{6\times6} & & \tilde{\mathbf{I}}_{3\times3} \\ & & \tilde{\mathbf{A}} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \\ \mathbf{a} \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{0}}_{6\times3} \\ \tilde{\mathbf{B}} \end{bmatrix} \mathbf{a}_c, \tag{92}$$

where $\mathbf{x}$ is the position vector $[x \quad y \quad z]^T$, $\mathbf{v}$ is the velocity vector $[u \quad v \quad w]^T$, $\mathbf{a}$ is the acceleration vector, and $\mathbf{a}_c$ is command acceleration $[a_x \quad a_y \quad a_z]^T$. These states are expressed in a navigation coordinate system. $\tilde{\mathbf{I}}_{3\times3}$ is a $3 \times 3$ that identities matrix. $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{0}}_{n\times m}$ are defined as

$$\begin{aligned} \tilde{\mathbf{A}} &= \begin{bmatrix} -\frac{1}{\tau_x} & 0 & 0 \\ 0 & -\frac{1}{\tau_y} & 0 \\ 0 & 0 & -\frac{1}{\tau_z} \end{bmatrix}, \\ \tilde{\mathbf{B}} &= \begin{bmatrix} \frac{\kappa_x}{\tau_x} & 0 & 0 \\ 0 & \frac{\kappa_y}{\tau_y} & 0 \\ 0 & 0 & \frac{\kappa_z}{\tau_z} \end{bmatrix}, \\ \tilde{\mathbf{0}}_{n\times n} &= \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}. \end{aligned} \tag{93}$$

$\tau_x$, $\tau_y$, and $\tau_z$ are time constants with respect to three acceleration states $[a_x \ a_y \ a_z]^T$. $\kappa_x$, $\kappa_y$, and $\kappa_z$ are the gains of the first order approximation. According to the developed collision avoidance architecture, once a UAV detects an obstacle, an on-board computer checks the possibility of a collision with an obstacle. If the UAV recognizes the obstacle as a potential threat, a new optimal collision avoidance trajectory will be computed and updated. When the optimal collision avoidance trajectory is initiated, the initial conditions at time $t$ can be expressed as:

$$\begin{aligned} \mathbf{x}(t) &= [x(t) \quad y(t) \quad z(t)]^T \\ \mathbf{v}(t) &= [u(t) \quad v(t) \quad w(t)]^T \\ \mathbf{a}(t) &= [a_x(t) \quad a_y(t) \quad a_z(t)]^T \end{aligned} \tag{94}$$

In a real-time simulation, formulating terminal constraints for a collision avoidance problem is critical. This complex formulation for terminal constraints may increase required computational resources to solve the optimal trajectory problem. In contrast, a simple formulation may result in an increase of the optimal trajectory cost. To avoid this problem, we have to construct an appropriate formulation for the optimal collision avoidance problem.

To simplify the formulation for an optimal trajectory problem, Schouwenaars et al.[143] introduces the mixed-integer linear program (MILP) for collision avoidance against a fixed object. Through binary constraints, this technique allows an optimal avoidance trajectory to avoid a rectangular obstacle. This methodology was validated by a simulation in a horizontal plane (i.e., a two-dimensional simulation environment). Yoshiaki et al. have extended the MILP approach to the receding-horizon mixed-integer linear program (RH-MILP) [88]. These two collision avoidance methodologies are computationally tractable, and they allow a vehicle to safely avoid fixed obstacles. However, the MILP requires multiple optimal solutions that can identify the optimal constraint, so it is commonly implemented with the Branch and Bound method to choose the optimal avoidance trajectory solution with a constraint from the set of

binary constraints.

In addition to the MILP, two-dimensional rule-based approaches can be used to simplify the collision avoidance algorithm suggested by Moon and advanced by Kang [106] [81]. The rule-based approach suggested by Moon defines safe positions based on the maximum and minimum horizontal and vertical points from the sensor information about the detected obstacle. These safe positions, which are the considered minimum required separation distance, are implemented by a trajectory optimization as a terminal constraint. The final optimal trajectory is the minimal time with the lowest cost. However, since safe positions are multiple, this collision avoidance method requires a safe position to solve multiple trajectory optimization problems that increase computational expenses. Unlike Moon, Kang proposed a simpler collision avoidance approach based on a rule-based approach that can only solve one optimization problem through selecting a safe positon without solving multiple trajectory optimization problems. Because this rule-based approach is mathematically simple and computationally efficient, we, therefore, select this rule-based collision avoidance algorithm as a benchmark case.

For the benchmark case of the rule-based collision avoidance algorithm, we need to identify final states. The final position of a UAV must be placed outside of an obstacle that satisfies the minimum safe distance. To satisfy this condition, we can specify the terminal position $\mathbf{x}(t_f)$, which is the safe position, and define $t_f$ as free final time because our trajectory optimization problem is the minimum-energy problem. We also assume that terminal velocity and acceleration conditions are in a level-flight condition, which is a zero acceleration condition and constant velocity. The following

conditions represent the terminal constraints,

$$\begin{gathered} t_f = free \\ \mathbf{x}(t_f) = [x(t_f) \quad y(t_f) \quad z(t_f)]^T = \mathbf{x}_{safe} \\ \mathbf{v}(t_f) = [v_x(t_f) \quad 0 \quad 0] \\ \mathbf{a}(t_f) = [0 \quad 0 \quad 0] \end{gathered}, \tag{95}$$

where $\mathbf{x}_{safe}$ is the vector of the safe position, which can be determined from the detected obstacle information and the safe distance. In this paper, we introduce the definition of the safe position from the rule-based collision avoidance approach proposed by Kang [81]:

$$x_{safe} = min\, x_{ob} - r_s \tag{96}$$

$$y_{safe} = \begin{cases} min\, y_{ob} - r_s,\ if\ \ |min\, y_{ob} - y(t)| \leq |max\, y_{ob} - y(t)| \\ y(t),\ if\ z_{safe} \neq z_0 \\ max\, y_{ob} + r_s,\ if\ \ |min\, y_{ob} - y(t)| \geq |max\, y_{ob} - y(t)| \end{cases}. \tag{97}$$

$$z_{safe} = \begin{cases} max\, z_{ob} + r_s,\ if\ y_{safe} = y_0 \\ z_0,\ if\ y_{safe} \neq y(t) \end{cases}. \tag{98}$$

$[x_{ob} \quad y_{ob} \quad z_{ob}]^T$ is the position information of the detected obstacle. The on-board sensor is assumed to be LiDAR sensor, which provides point cloud information about obstacles. In the simulation model, LiDAR model does not include uncertainties such as noise. $r_s$ is the safe distance.

During the avoidance maneuver phase, a UAV velocity and acceleration must be operated within a flight envelope. To include these restrictions in the optimal trajectory formulation, we must define them as the path constraints of a flight envelope so that the UAV maintains its velocity and acceleration within the flight envelope. The

velocity path constraint can be written as follows:

$$\begin{aligned} 0 \leq u \leq u_{max} \\ v_{min} \leq v \leq v_{max} \\ w_{min} \leq w \leq w_{max} \\ V_{min} \leq \sqrt{u^2 + v^2 + w^2} \leq V_{max} \end{aligned} , \tag{99}$$

where $v_{min}$, and $w_{min}$ are low speed limits and $u_{max}$, $v_{max}$, and $w_{max}$ are high speed limits specified by its flight envelope. Another path constraint is the load factor conditions. To simplify the load factor conditions, we define the following load factor constraints instead of ellipsoid load factor constraints,

$$\begin{aligned} |a_x| \leq g_{xmax} \\ |a_y| \leq g_{ymax} \\ |a_z + g| \leq g_{zmax} \end{aligned} , \tag{100}$$

In this table, $g_{max}$ is the maximum allowable acceleration defined by the V-n diagram. That is, the maximum $g_{max}$ in a low-speed region can be identified from the stall effect which is a significant constraint of flight conditions, and the maximum $g_{max}$ in a high-speed region can be load constraints caused by a structural damage which is a critical constraint in a high-speed operation [14][127]. The last path constraint is the constraint of the minimum flight attitude that minimizes a collision risk with ground facilities. This minimum flight altitude constraint can be added as follows:

$$z \geq z_{min} \tag{101}$$

Table 8 summarizes the rule-based collision avoidance algorithm that includes initial, terminal, and path constraints.

In this table, $P = [P_x \quad P_y \quad P_z]$ is the point cloud information about a detected obstacle from an airborne sensor.

The rule-based collision avoidance strategy is computationally efficient because it consists of simple formulations, which are derived by the safe position concept.

**Table 8:** Rule-based optimal collision avoidance algorithm

| | |
|---|---|
| Path constraints | $t_f = free$<br>$\mathbf{x}(t_f) = [x(t_f) \quad y(t_f) \quad z(t_f)]^T = \mathbf{x}_{safe}$<br>$\mathbf{v}(t_f) = [u(t_f) \quad 0 \quad 0]$<br>$\mathbf{a}(t_f) = [0 \quad 0 \quad 0]$<br>$z \geq z_{min}$<br>$x_{safe} = min\ P_x - r_s$<br>$y_{safe} = \begin{cases} min\ P_y - r_s,\ if\ \ \lvert min\ P_y - y(t)\rvert < \lvert max\ P_y - y(t)\rvert \\ y(t),\ if\ \ \lvert min\ P_y - y(t)\rvert = \lvert max\ P_y - y(t)\rvert \\ max\ P_y + r_s,\ if\ \ \lvert min\ P_y - y(t)\rvert > \lvert max\ P_y - y(t)\rvert \end{cases}$.<br>$z_{safe} = \begin{cases} max\ P_z + r_s,\ if\ y_{safe} = y(t) \\ z(t),\ if\ y_{safe} \neq y(t) \end{cases}$. |
| Velocity constraints | $0 \leq u \leq u_{max}$<br>$v_{min} \leq v \leq v_{max}$<br>$w_{min} \leq w \leq w_{max}$<br>$V_{min} \leq \sqrt{u^2 + v^2 + w^2} \leq V_{max}$ |
| Load factor constraints | $\lvert a_x\rvert < g_{xmax}$<br>$\lvert a_y\rvert < g_{ymax}$<br>$\lvert a_z + g\rvert < g_{zmax}$ |

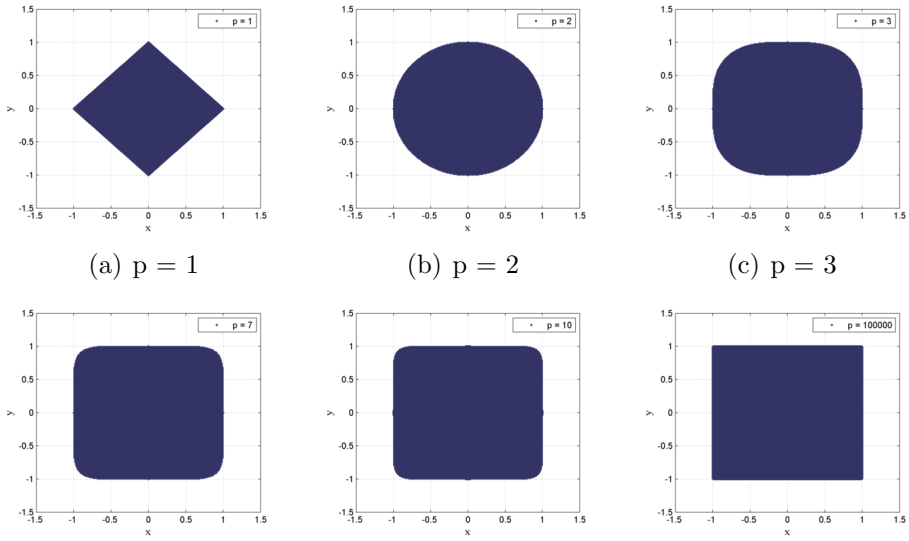
However, it has several drawbacks. The first drawback is that since its formulation does not include a next waypoint, the trajectory cost of a computed optimal avoidance trajectory may be higher than an actual optimal avoidance trajectory. Second potential drawback is that because of the hard waypoint constraint from the safe-position, the optimal trajectory cost may increase in certain state conditions. To be more specific, the final avoidance trajectory always passes one of the safe positions, but an actual optimal trajectory may not pass the selected safe position; thus, this restriction by the safe position concept may raise the optimal trajectory cost. These two drawbacks lead to the following research question: *How can a collision avoidance trajectory problem that reduces computational expense and minimizes maneuver effort be constructed?*

To consider a next waypoint in the formulation of an optimal avoidance trajectory, one can apply a multi-phase optimal trajectory problem that leads to a better optimal avoidance trajectory with respect to the optimal trajectory cost. Next, to solve the

restricted safe position concept, we can formulate a more sophisticated optimal problem that provides more flexibility to the final optimal collision avoidance trajectory. In other words, the terminal constraints of a new optimal collision avoidance can be formulated by the $L_p$ norm. Terminal conditions using the $L_p$ norm have been implemented in the optimal collision avoidance problem [81][156]. These papers applied the $L_p$-norm formulation to build the formulation of an optimal avoidance trajectory. Xu applied this collision avoidance algorithm to off-line trajectory generation, and Kang implemented it in receding-horizon on-line trajectory generation. The $L_p$ norm commonly measures the length of a vector in $p$ dimensional space. The mathematical expression of the $L_p$-norm is

$$\|x_n\|_{L_P} = \left( \sum_{n=1}^{\infty} |x_n|^p \right)^{1/p}. \tag{102}$$

When $p$ is two, it is called the Euclidean norm, which is well-known as a unit circle shape. As $p$ increases to infinity, the $L_p$ norm becomes a square shape that is the maximum norm (i.e., uniform norm). Examples of the $L_p$ norm as $p$ increases are exhibited in Figure 114.




(a) p = 1 (b) p = 2 (c) p = 3

(d) p = 7 (e) p = 10 (f) p = $10^5$

**Figure 17:** $L_P$-norm examples

The $L_p$-norm examples illustrate that the boundary shape varies according to the $p$ value. This feature of the $p$ value can be used to formulate the shape of a detected obstacle from an airborne sensor. In other words, the mathematical expression of the $L_p$ norm can be implemented to formulate a path constraint of an optimal avoidance trajectory. In the $L_p$-norm expression, the $p$ value is assumed to be ten because the shape is approximately a cuboid shape. Based on the $L_p$-norm constraint, we can formulate the multi-phase optimal collision avoidance problem described in Table 9.

Because of the more precise constraints by the inequality constraint and the addition of the next waypoint constraint to the multi-phase optimal problem, the formulation of the multi-phase optimal collision avoidance can yield a lower optimal trajectory cost of a UAV than that of the rule-based avoidance approach. In this

**Table 9:** Optimal collision avoidance algorithm using $L_P$-norm

| | | |
|---|---|---|
| Phase 1 | Path constraints | $\mathbf{x}(t_1) = [x(t_1) \quad free \quad free]^T$<br>$\mathbf{v}(t_1) = [u(t_1) \quad 0 \quad 0]^T$<br>$\mathbf{a}(t_1) = [a_x(t_1) \quad 0 \quad 0]^T$<br>$x_{t_1} = min\, P_x - r_s$<br>$z \geq max[min\, P_z, \quad z_{min}]$<br>$0 \leq \|c(y,z)\|_p = \left[\left\|\frac{y-y_c}{a}\right\|^p + \left\|\frac{z-z_c}{b}\right\|^p\right]^{\frac{1}{p}} - 1 \leq L_{Pmax}$ |
| | Velocity constraints | $0 \leq u \leq u_{max}$<br>$v_{min} \leq v \leq v_{max}$<br>$w_{min} \leq w \leq w_{max}$<br>$V_{min} \leq \sqrt{u^2+v^2+w^2} \leq V_{max}$ |
| | Load factor constraints | $\|a_x\| < g_{xmax}$<br>$\|a_y\| < g_{ymax}$<br>$\|a_z + g\| < g_{zmax}$ |
| Phase 2 | Path constraints | $t_f = free$<br>$\mathbf{x}(t_f) = [x(t_f) \quad y(t_f) \quad z(t_f)]^T = \mathbf{x}_t$<br>$\mathbf{v}(t_f) = [u(t_f) \quad 0 \quad 0]$<br>$\mathbf{a}(t_f) = [0 \quad 0 \quad 0]$ |
| | Velocity constraints | $0 \leq u \leq u_{max}$<br>$v_{min} \leq v \leq v_{max}$<br>$w_{min} \leq w \leq w_{max}$<br>$V_{min} \leq \sqrt{u^2+v^2+w^2} \leq V_{max}$ |
| | Load factor constraints | $\|a_x\| < g_{xmax}$<br>$\|a_y\| < g_{ymax}$<br>$\|a_z + g\| < g_{zmax}$ |
| Corner conditions | | $\lambda^T(t_1^+) = \lambda^T(t_1^-)$<br>$H^T(t_1^+) = H^T(t_1^-)$<br>$H_u^T(t_1^+) = H_u^T(t_1^-)$ |

table, $t_1$ is the terminal time in the first phase, $a$ is the width that includes the the

half width of the detected obstacle and a safe-distance, and $b$ is the width that includes the half height of the detected obstacle and a safe-distance. These $a$ and $b$ are computed from the point cloud information $\mathbf{P}$.

To analyze the optimal trajectory cost and the required computational time of the both optimal avoidance algorithms, which are the rule-based algorithm and the optimal collision avoidance algorithm using the $L_p$ norm, we perform case studies. The experiment cases only vary initial velocity conditions, and fix other vehicle conditions and an obstacle. The conditions for the case studies are summarized in Table 10, which lists the initial/terminal conditions, sensor specifications, and obstacle information. We select seven initial velocity vectors: $V_1 = [100 \;\; 0 \;\; 0]$, $V_2 = [100 \;\; 5 \;\; 10]$, $V_3 = [100 \;\; -5 \;\; 10]$, $V_4 = [100 \;\; -10 \;\; 5]$, $V_5 = [100 \;\; 10 \;\; 5]$, $V_6 = [100 \;\; 10 \;\; 10]$ and $V_7 = [100 \;\; -10 \;\; 10]$.

**Table 10:** Initial conditions and assumptions for case studies

| Variable name | Variable | Value | Etc. |
|---|---|---|---|
| Initial condition | $\mathbf{x}_0$ $[ft]$ | [0 0 800] | |
| | $\mathbf{v}_0$ $[ft/s]$ | $[u_0 \; v_0 \; w_0]$ | |
| | $\mathbf{a}_0$ $[ft/s^2]$ | [0 0 0] | |
| Terminal condition | $\mathbf{x}_{tf}$ $[ft]$ | [4000 0 800] | |
| | $\mathbf{v}_{tf}$ $[ft/s]$ | [100 0 0] | |
| | $\mathbf{a}_{tf}$ $[ft/s^2]$ | [0 0 0] | |
| Field of View | Distance range $S_r[ft]$ | 2500 | |
| | Azimuth range $S_{AZ}[deg]$ | 45 | |
| | Elevation range $S_{EL}[deg]$ | 45 | |
| Obstacle information | Volume center $[ft]$ | [2000 0 500] | |
| | Side length $[ft]$ | [100 500 1000] | [width depth height] |
| | Euler rotation $[deg]$ | [0 0 0] | [roll pitch yaw] |

Figures 18 and 19 are the results of optimal collision avoidance trajectories according to the seven initial velocities. As we can reasonably expect, when the initial velocity vector is $V = [100 \;\; 0 \;\; 0]$, the optimal trajectory of the rule-based approach selects the top safe position. In other cases, the optimal avoidance trajectories are dependent on the initial horizontal velocity (i.e., y-direction velocity in the navigation frame). In Figure 19, results of the optimal collision avoidance trajectory using the $L_p$

norm (i.e., the PNORM approach), show that unlike those of the rule-based approach, the optimal trajectories of the PNORM are not associated with the initial velocity direction because the $L_p$ norm formulation is not a function of the velocity that provides more flexible trajectories. For a qualitative comparison assessment of the two optimal collision avoidance algorithms, the optimal trajectory cost and the computational time are measured because the algorithms have to minimize energy consumption as well as remain computationally efficient for fast-time simulation. The results of the comparison of the two algorithms with respect to two aspects are depicted in Figure 19. The results illustrate that the rule-based optimal collision avoidance algorithm is computationally favorable because of the simplified mathematical formula, but it requires higher optimal trajectory cost because of the ignored target position in the optimal trajectory formulation and the intermediate trajectories limited by the safe position concept. In contrast to the rule-based optimal collision avoidance algorithm, the PNORM approach is optimally favorable, but it increases computational expense due to the complex constraints. Overall, the experimental results imply that more complex inequality constraints for the optimal collision avoidance algorithm incur higher computational expense.

From the previous case studies, we observe that the computational time is coupled with the complexity of the terminal constraints, and the optimal trajectory cost is highly correlated with the terminal constraints. Therefore, if we build a less complex formulation of the optimal collision avoidance problem than the PNORM approach, it may slightly degrade the optimal trajectory cost but improve computational runtime. Instead of using the two-dimensional inequality constraints of the PNORM approach, we suggest a new collision avoidance algorithm that has simpler one-dimensional inequality constraints and improve computational efficiency. The new optimal collision avoidance trajectory is summarized in Table 11. The new optimal collision avoidance

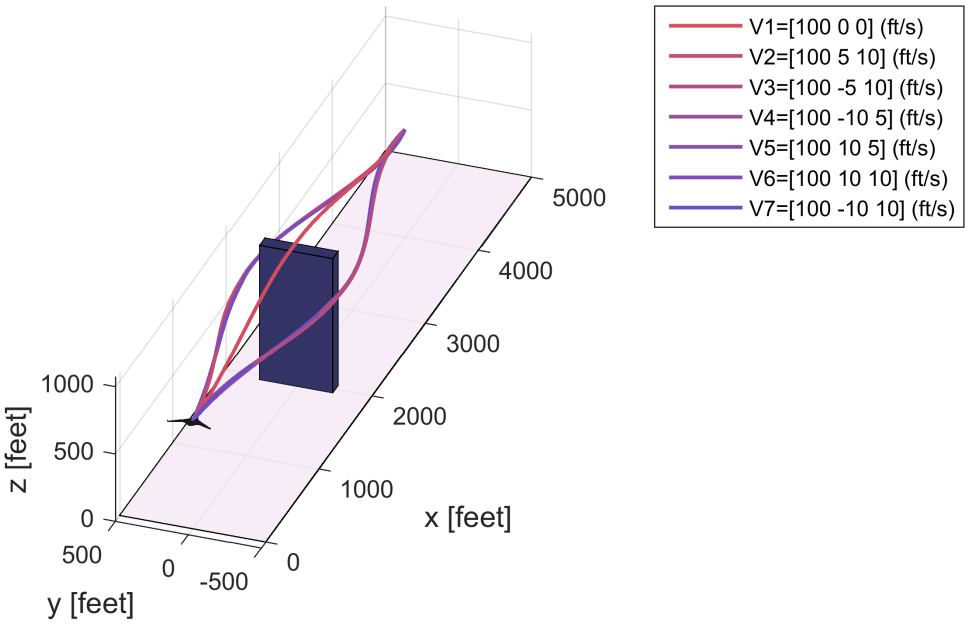



**Figure 18:** Rule-based optimal collision avoidance trajectory suggested

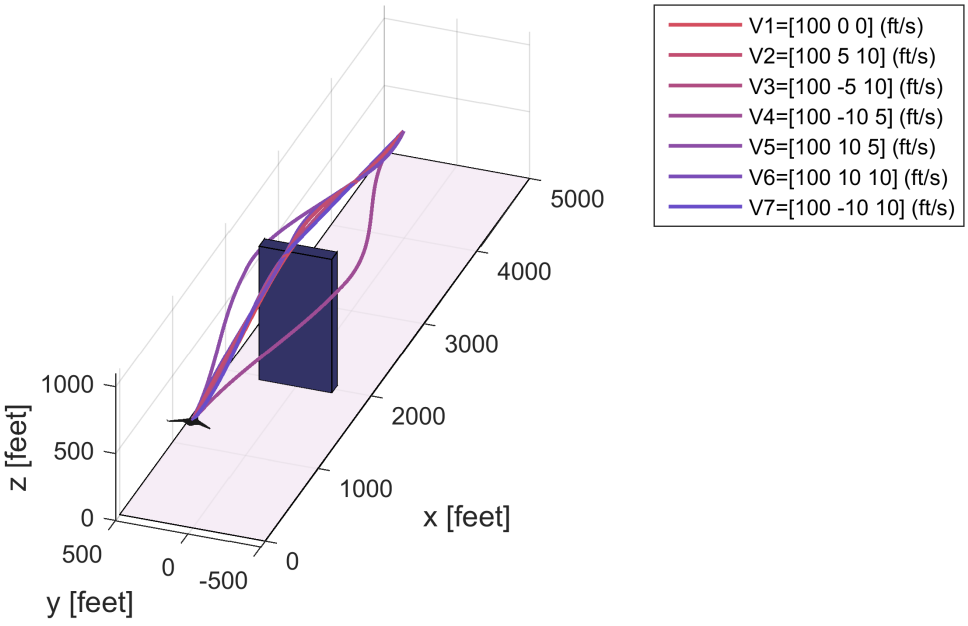



**Figure 19:** Optimal collision avoidance trajectory based on P-norm inequality constraint

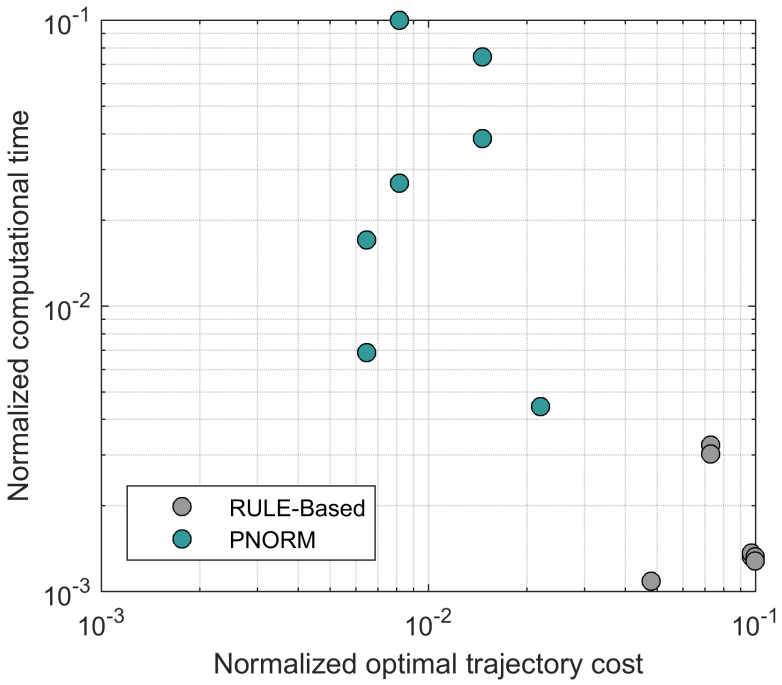



**Figure 20:** Optimal trajectory cost vs. Computational time

algorithm defines three one-dimensional constraints by the safe distance $r_s$ and information about the minimum and maximum points (i.e., $P_{xmin}$, $P_{ymax}$ and $P_{zmax}$) among the point cloud information from the sensor. Figure 21 illustrates the three inequality constraints, $D_1(z)$, $D_2(z)$, and $D_3(y)$.

From these three inequality constraints, the new collision avoidance algorithm selects one constraint. The selection of a constraint is based on the safe distance from the projected point information, $[y_p \;\; z_p]$. The projected point can be specified from a velocity vector or a vector between the next waypoint and the current position of an UAV. Using velocity vector is an appropriate concept when the acceleration impact is minimal. This algorithm using the projected velocity vector information is called Simplified Collision Avoidance Algorithm 1 (SCAA-1). However, if the magnitude of the initial acceleration is high, the SCAA-1 using the velocity vector, may not be adequate because the optimal avoidance trajectory also depends on the magnitude and direction of acceleration. In such case, vector information is determined by the current UAV position, and the target position would be a better approach to define

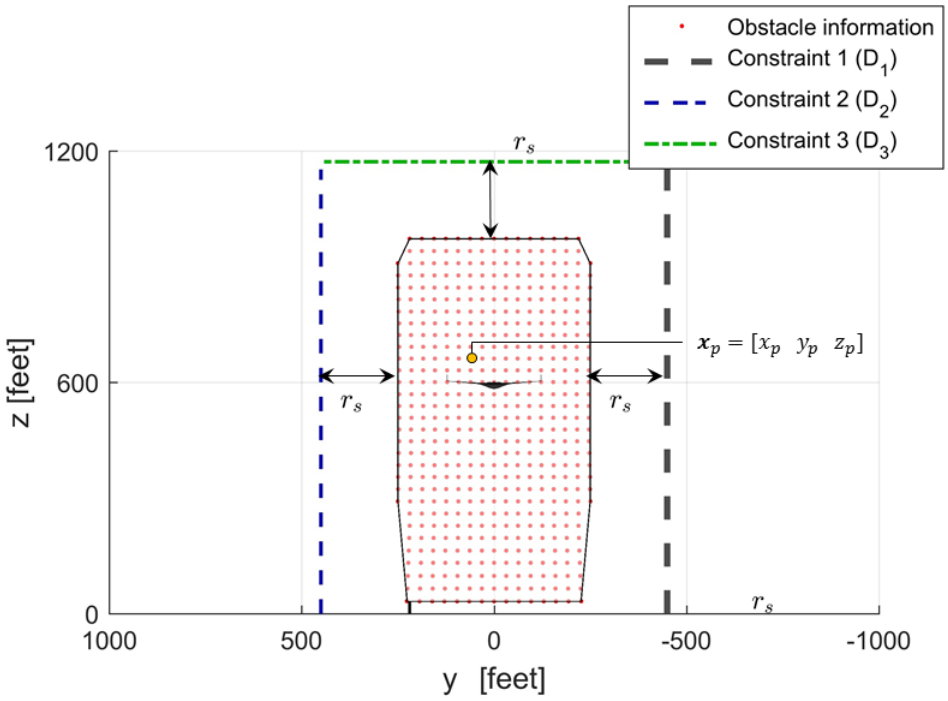



**Figure 21:** The one-dimensional inequality condition definition of the simplified optimal collision avoidance methods (SCAA–1 and SCAA–2)

the projected point, which selects the one-dimensional inequality constraint. This collision avoidance algorithm is called Simplified Collision Avoidance Algorithm 2 (SCAA-2).

**Table 11:** Optimal collision avoidance algorithm (SCAA–1 and SCAA–2)

| | | |
|---|---|---|
| Phase 1 | Path constraints | $\mathbf{x}(t_1) = [x(t_1) \quad free \quad free]^T$<br>$\mathbf{v}(t_1) = [u(t_1) \quad 0 \quad 0]^T$<br>$\mathbf{a}(t_1) = [a_x(t_1) \quad 0 \quad 0]^T$<br>$x_{t_1} = min\ P_x - r_s$<br>$z \geq z_{min}$<br>$\begin{cases} y \leq g_{ymin} + Mt_1 \\ -y \leq -g_{ymax} + Mt_2 \quad \mathbf{t} = [t_1 t_2 t_3],\ \text{M is large number} \\ -z \leq -g_{zmax} + Mt_3 \end{cases}$<br>$where,$<br>$D = [\lvert(min\ P_y - r_s) - y_p\rvert,\ \lvert(max\ P_y + r_s) - y_p\rvert,\ \lvert(max\ P_z + r_s) - z_p\rvert]$<br>$\mathbf{t} = [0\ 1\ 1], \quad if\ min\ D = \lvert(min\ P_y - r_s) - y_p\rvert$<br>$\mathbf{t} = [1\ 0\ 1], \quad if\ min\ D = \lvert(max\ P_y + r_s) - y_p\rvert$<br>$\mathbf{t} = [1\ 1\ 0], \quad if\ min\ D = \lvert(max\ P_z + r_s) - z_p\rvert$ |
| | Velocity constraints | $0 \leq u \leq u_{max}$<br>$v_{min} \leq v \leq v_{max}$<br>$w_{min} \leq w \leq w_{max}$<br>$V_{min} \leq \sqrt{u^2 + v^2 + w^2} \leq V_{max}$ |
| | Load factor constraints | $\lvert a_x \rvert < g_{xmax}$<br>$\lvert a_y \rvert < g_{ymax}$<br>$\lvert a_z + g \rvert < g_{zmax}$ |
| Phase 2 | Path constraints | $t_f = free$<br>$\mathbf{x}(t_f) = [x(t_f) \quad y(t_f) \quad z(t_f)]^T = \mathbf{x}_t$<br>$\mathbf{v}(t_f) = [u(t_f) \quad 0 \quad 0]$<br>$\mathbf{a}(t_f) = [0 \quad 0 \quad 0]$ |
| | Velocity constraints | $0 \leq u \leq u_{max}$<br>$v_{min} \leq v \leq v_{max}$<br>$w_{min} \leq w \leq w_{max}$<br>$V_{min} \leq \sqrt{u^2 + v^2 + w^2} \leq V_{max}$ |
| | Load factor constraints | $\lvert a_x \rvert < g_{xmax}$<br>$\lvert a_y \rvert < g_{ymax}$<br>$\lvert a_z + g \rvert < g_{zmax}$ |
| Corner conditions | | $\lambda^T(t_1^+) = \lambda^T(t_1^-)$<br>$H^T(t_1^+) = H^T(t_1^-)$<br>$H_u^T(t_1^+) = H_u^T(t_1^-)$ |

For the performance analysis of the two collision avoidance algorithms (SCAA-1 and SCAA-2), we execute the seven case experiments, described in 10. The next two figures depict the results of the optimal collision avoidance trajectories. The trajectory results show that the SCAA-1 optimal trajectories vary according to the initial velocity components because the one-dimensional inequality constraint is determined by the velocity direction. In contrast, the optimal collision avoidance trajectories generated by the SCAA-2 algorithm maintain almost the same trajectories because the initial position and the next target position have the same vector, which always selects the $D_3$ inequality constraint, specifically $\mathbf{t} = [1\ 1\ 0]$.

Figure 24 presents the results of the analysis for the optimal trajectory and required computational costs. The SCAA-1 and SCAA-2 methods require less computational time than the PNORM collision avoidance approach, and these two methods improve the optimal trajectory cost compared to the rule-based collision avoidance algorithm. From the experiment results, the SCAA-1 and SCAA-2 algorithms improve the computational runtime for trajectory optimization by decreasing the complexities of constraints, but the incremental complexities cause degradation of the required computational efficiency. These results show that the reduced complexity of an inequality constraint decreases computational burdens.

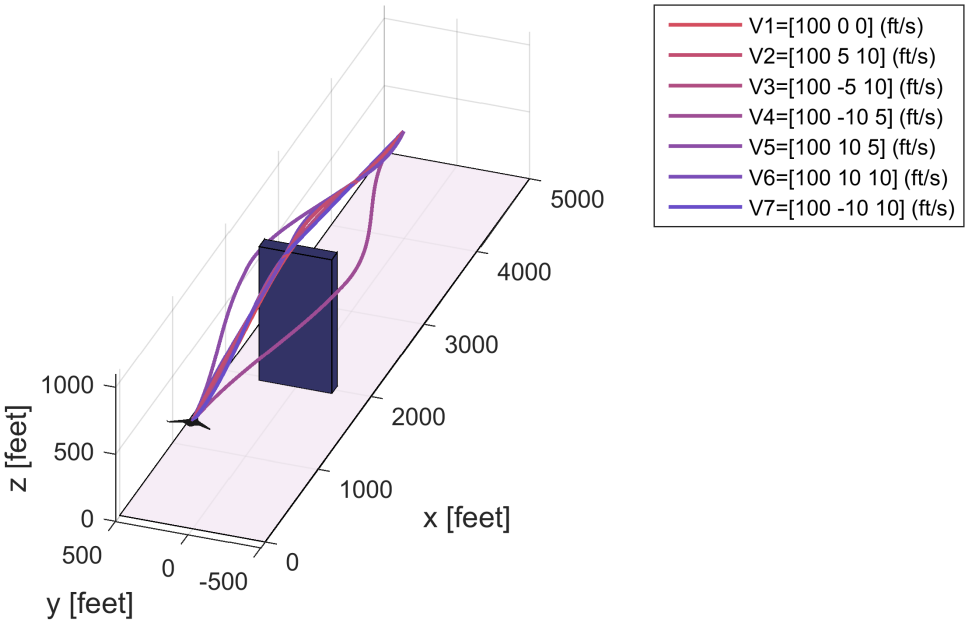



**Figure 22:** Optimal collision avoidance trajectory SCAA–1

The results of the previous case studies imply that relaxing constraints in the formulation of an optimal collision avoidance problem improves computation runtime. Therefore, to achieve more computational efficient algorithm, we suggest a simpler optimal collision avoidance algorithm (SCAA-3) shown in Table 12.

To be more specific, instead of using inequality constraints, the SCAA-3 algorithm entails a fixed terminal position as a safe position $\mathbf{x}_s$ to avoid an obstacle. The safe

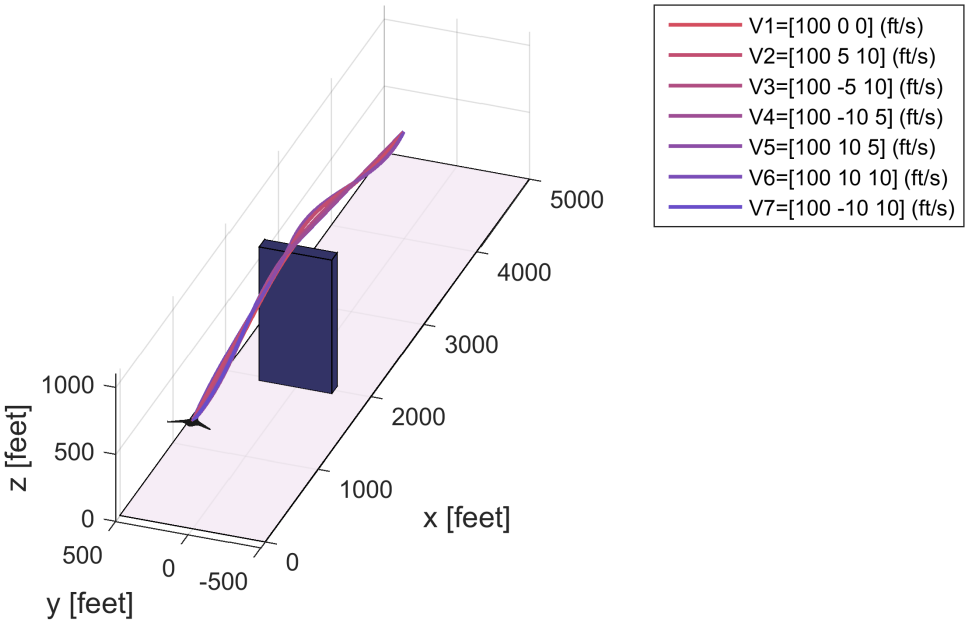



**Figure 23:** Optimal collision avoidance trajectory SCAA–2

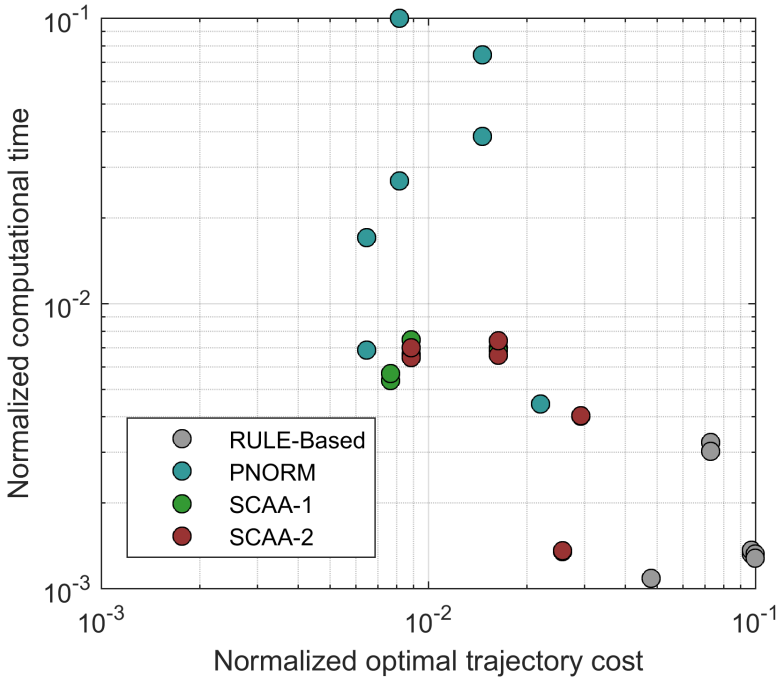



**Figure 24:** Optimal trajectory cost vs. Computational time

position $\mathbf{x}_s$ is defined as the projected point $\mathbf{x}_p$ closest to one of the three line equations $D_1(z)$, $D_2(z)$, and $D_3(y)$. The three lines are determined by the sensor information and the minimum required separation distance $r_s$, and the projected point $\mathbf{x}_p$ is defined by the vehicle velocity vector.

**Table 12:** Optimal collision avoidance algorithm (SCAA–3)

| | | |
|---|---|---|
| Phase 1 | Path constraints | $\mathbf{x}(t_1) = \mathbf{x}_s$<br>$\mathbf{v}(t_1) = [u(t_1) \quad 0 \quad 0]^T$<br>$\mathbf{a}(t_1) = [a_x(t_1) \quad 0 \quad 0]^T$<br>$z \geq z_{min}$<br>$D = [\lvert(min\ P_y - r_s) - y_p\rvert,\ \lvert(max\ P_y + r_s) - y_p\rvert,\ \lvert(max\ P_z + r_s) - z_p\rvert]$<br>$\begin{cases} \mathbf{x}_s = [x_{t1}\ D_m\ z_p], & if\ min\ D = \lvert(min\ P_y - r_s) - y_p\rvert\ then\ D_m = \lvert(min\ P_y - r_s) - y_p\rvert \\ \mathbf{x}_s = [x_{t1}\ D_m\ z_p], & if\ min\ D = \lvert(max\ P_y + r_s) - y_p\rvert\ then\ D_m = \lvert(max\ P_y + r_s) - y_p\rvert \\ \mathbf{x}_s = [x_{t1}\ y_p\ D_m], & if\ min\ D = \lvert(max\ P_z + r_s) - z_p\rvert\ then\ D_m = \lvert(max\ P_z + r_s) - z_p\rvert \end{cases}$<br>$where,$<br>$x_{t_1} = min\ P_x - r_s$ |
| | Velocity constraints | $0 \leq u \leq u_{max}$<br>$v_{min} \leq v \leq v_{max}$<br>$w_{min} \leq w \leq w_{max}$<br>$V_{min} \leq \sqrt{u^2 + v^2 + w^2} \leq V_{max}$ |
| | Load factor constraints | $\lvert a_x\rvert < g_{xmax}$<br>$\lvert a_y\rvert < g_{ymax}$<br>$\lvert a_z + g\rvert < g_{zmax}$ |
| Phase 2 | Path constraints | $t_f = free$<br>$\mathbf{x}(t_f) = [x(t_f) \quad y(t_f) \quad z(t_f)]^T = \mathbf{x}_t$<br>$\mathbf{v}(t_f) = [u(t_f) \quad 0 \quad 0]$<br>$\mathbf{a}(t_f) = [0 \quad 0 \quad 0]$ |
| | Velocity constraints | $0 \leq u \leq u_{max}$<br>$v_{min} \leq v \leq v_{max}$<br>$w_{min} \leq w \leq w_{max}$<br>$V_{min} \leq \sqrt{u^2 + v^2 + w^2} \leq V_{max}$ |
| | Load factor constraints | $\lvert a_x\rvert < g_{xmax}$<br>$\lvert a_y\rvert < g_{ymax}$<br>$\lvert a_z + g\rvert < g_{zmax}$ |
| Corner conditions | | $\lambda^T(t_1^+) = \lambda^T(t_1^-)$<br>$H^T(t_1^+) = H^T(t_1^-)$<br>$H_u^T(t_1^+) = H_u^T(t_1^-)$ |

The results of the SCAA-3 trajectories under the conditions in Table 10 are depicted in Figure 25. This approach applies the safe-position concept to the computationally efficient avoidance trajectory, which is similar to the rule-based collision avoidance algorithm. Unlike those of the rule-based approach, the computed trajectories of the SCAA-3 spread out according to the initial velocity vector since the safe position results from the point with the minimum distance from the line and the projected velocity vector. The comparison assessment with respect to the computational expense and the optimal trajectory are graphically summarized in Figure 25. The results show that the SCAA-3 method decreases the computational expense, but it does not decrease the expense as much as the one-dimensional optimal collision avoidance approaches, SCAA-1 and SCAA-2. However, from the computational

time perspective, the SCAA-3 algorithm exhibits little degradation and yields higher improvement than the rule-based collision avoidance approach.

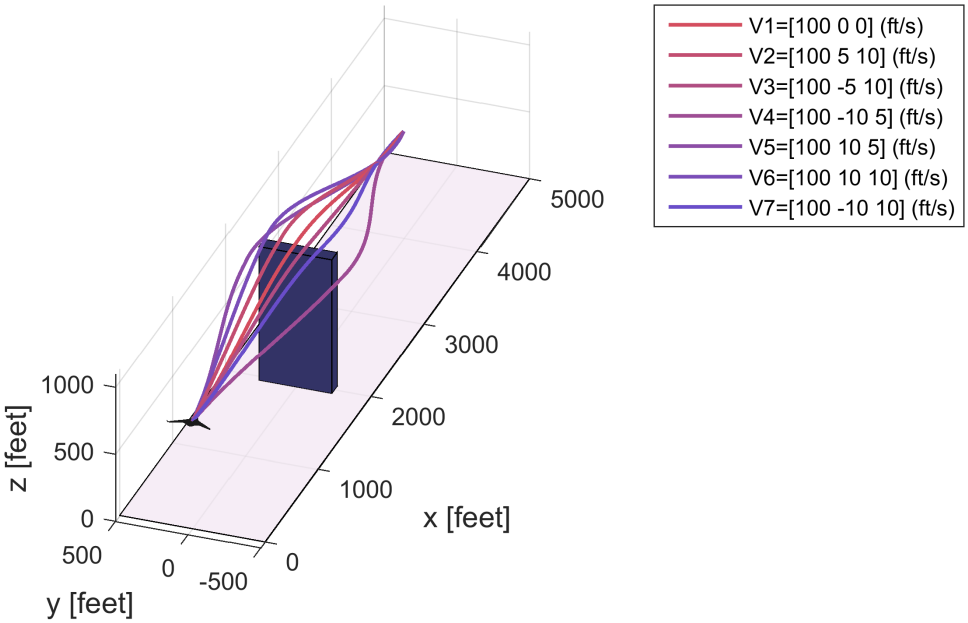



**Figure 25:** Optimal collision avoidance trajectory SCAA–3

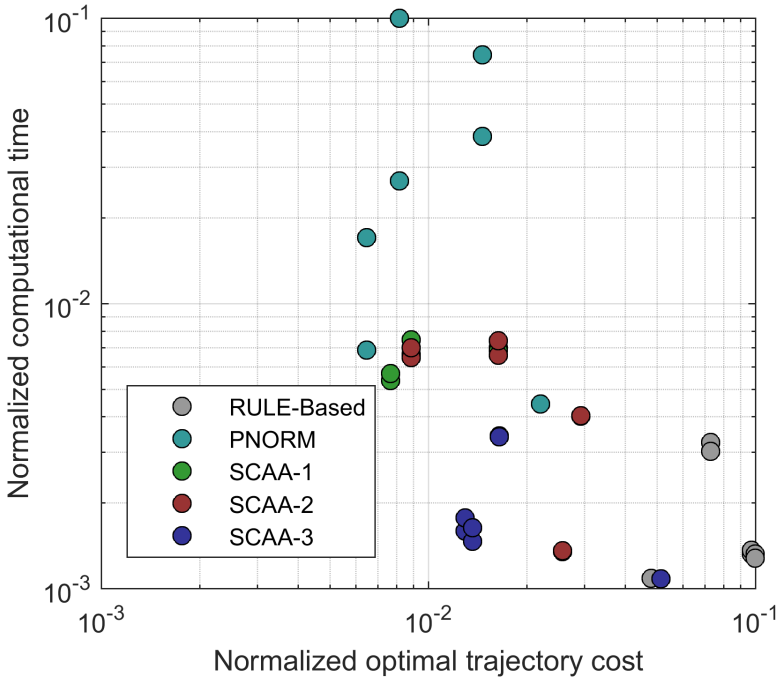



**Figure 26:** Optimal trajectory cost vs. Computational time

## *3.3 Hybrid collision avoidance methodology using machine learning*

In the previous section, the simplified optimal collision avoidance algorithms are discussed to achieve both low computational cost and low optimal trajectory cost. The results of the case experiments reveal that a dominant general solution does not exist that works for any algorithm, computationally and optimally because of the complexity level of an optimal collision trajectory problem. For instance, when the formulation for the collision avoidance algorithm has mathematically complex constraints like PNORM approach, the optimal trajectory will have a low optimal trajectory cost whereas solving optimal trajectory problem needs more computational expense. On the other hand, when the formulation of the collision avoidance algorithm has a simple formulation like SCAA-3, the computed optimal trajectory will have a large computational improvement, but it yields some degradation of the optimal trajectory cost. From these observations, we pose a following question: *How can a collision avoidance algorithm be formulated to maintain computational efficiency and the optimal trajectory with a low trajectory cost?*

To answer the research question, we need to observe the individual sample experiments in detail. For the examination of the best trajectory solution with the fast runtime and the low trajectory cost, we adopt the overall optimal cost function, which is shown in Equation 103 since this overall cost function aggregates two metrics (optimal trajectory cost and computation runtime) into one metric, this aggregated overall cost function explicitly enables the evaluation of each collision avoidance algorithm according to the given initial conditions.

$$\mathbf{J} = W_1\bar{\mathbf{J}}_{opt} + W_2\bar{\mathbf{J}}_{com}, \tag{103}$$

where $\bar{\mathbf{J}}_{opt}$ and $\bar{\mathbf{J}}_{com}$ are an optimal trajectory cost and a computational time cost, respectively. The terms $W_1$ and $W_2$ are the weights for the optimal trajectory cost and the computational time. It is assumed that the sum of the two weights is one

$(W_1 + W_2 = 1)$. The overall cost function with two weights is a flexible concept. That is to say, when a designer emphasizes the optimal trajectory cost, the weight $W_1$ increases, but when the designer wants to improve computational efficiency, the weight $W_2$ increases.

Using this overall cost function, we evaluate the best obstacle avoidance algorithm based on the sample experiments. The best obstacle avoidance algorithm is selected by the lowest overall cost under the given weights.

**Table 13:** Best collision avoidance strategy based on initial conditions

| Case | Velocity vector $[feet/sec]$ | Best alternative |
|---|---|---|
| 1 | [100 0 0] | PNORM |
| 2 | [100 5 10] | SCAA-3 |
| 3 | [100 -5 10] | SCAA-3 |
| 4 | [100 -10 5] | SCAA-1 |
| 5 | [100 10 5] | SCAA-1 |
| 6 | [100 10 10] | SCAA-3 |
| 7 | [100 -10 10] | SCAA-3 |

After evaluating the overall cost of the avoidance algorithms for the seven cases shown in Table 3.3, we identify the best strategy with the lowest overall cost. In the overall cost function, the weights of this evaluation are assumed as $W_1 = 0.5$ and $W_2 = 0.5$. The evaluation results present that while SCAA-3 was selected the most frequently, it is not always the best optimal collision strategy because any formulated method cannot be always the best in terms of the overall cost. In other words, the best avoidance strategy varies depending on the initial condition. Moreover, note that the best obstacle avoidance strategy can also be different depending on the two weights. The best avoidance method depends on different weight conditions; thus it is necessary to explore the general trend of the best avoidance method.

For this investigation of the weight variation effect, 100 sample trajectories of four avoidance algorithms (PNORM, SCAA-1, SCAA-2, and SCAA-3) are defined

under randomly selected initial/terminal conditions that satisfy flight envelope constraints. The terminal conditions are assumed as a level flight condition. We execute all collision avoidance algorithms and evaluate the optimal trajectory cost $\bar{\mathbf{J}}_{opt}$ and the computational runtime $\bar{\mathbf{J}}_{com}$. Then, we compute the overall cost function $\mathbf{J}$ as the two weights ($W_1$ and $W_2$) vary. Based on the results of the overall cost function, we specify the probability of the best avoidance strategy. The probability means the percentage of each method that has the lowest overall cost under the experiment results of 100 samples, and the given weights. For example, if SCAA-1 has 20 samples with the lowest overall cost under the given weight, the probability of the best strategy for SCAA-1 under the given weight is 20 percent. Figure 27 presents the probability of the best strategy according to different weight conditions. The results represent that when the weight for the computational time is high (low $W_1$), SCAA-3 yields outstanding performance. As the weight $W_1$ increases, we can observe the performance improvement of the obstacle avoidance strategy with one-dimensional constraint (SCAA-1 and SCAA-2). In the middle region of the weight $W_1$, these two methods are highly probable to be the best strategy. When the weight $W_1$ is around one, the obstacle avoidance algorithm (PNORM) with two-dimensional constraint improve. These results are expectable because the SCAA-1 is the simplest and fastest algorithm, but due to the simple constraint, it requires high optimal trajectory cost. On the other hand, the PNORM has the most complex constraint and slowest algorithm, but it yields low optimal trajectory cost due to the sophisticated constraint.

Based on the previous analysis, we can observe that any introduced obstacle avoidance method cannot provide the best obstacle avoidance performance with respect to overall cost function and is dependent on the weight, which means that the best obstacle avoidance method varies. From the two observations, if a-priori knowledge about the best collision avoidance algorithm in an input space, which is the variables'

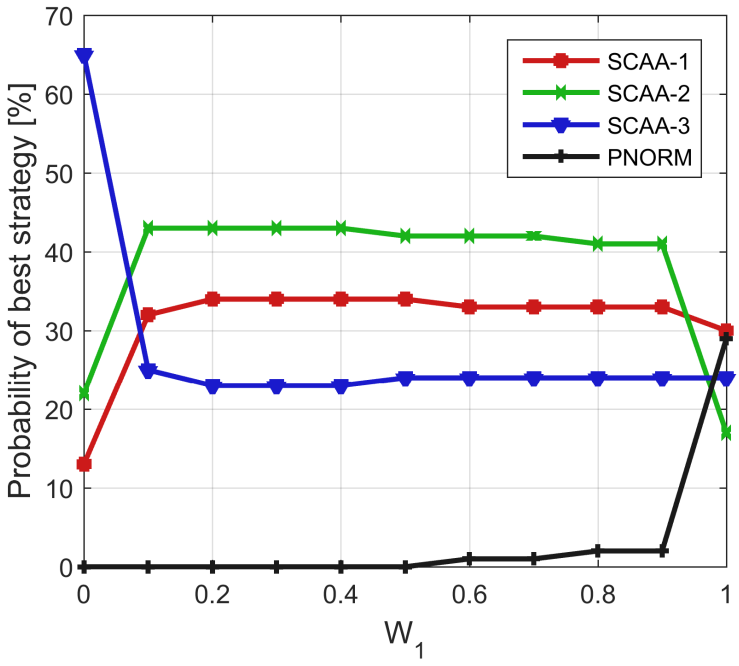



**Figure 27:** Overall trajectory cost of each avoidance method as weights ($W_1$ and $W_2$) vary

space associated to the optimal trajectory problem, can be obtained by a learning technique, the best collision avoidance algorithm among the pre-identified strategies can be selected during a real-time simulation. Consequently, this process selecting the best strategy improves the performance of the obstacle avoidance with respect to the optimal cost, and they provide computationally tractable solutions because the pre-identified strategies are determined by the simple prediction model. The best collision avoidance algorithm is identified from the overall cost function below 103 the given weights $W_1$ and $W_2$.

The simple prediction model can be defined by using diverse methods such as a fuzzy controller and a machine learning technique. The fuzzy logic controller has been applied to aircraft vortex flow control to adjust the fuzzy-rule [142]. The machine learning technique can be applied to the hybrid method through supervised learning methods. Because the machine learning algorithm is very flexible and has diverse approaches, this thesis focuses on applying the machine learning algorithm to the

hybrid method.

Accordingly, we suggest the hybrid methodology for an optimal collision avoidance. The hybrid methodology is a technique that selects the best collision avoidance strategy through a classification algorithm in pre-processing, which is one of the machine learning techniques. During a real-time simulation, the classification result provides the best avoidance mode based on input conditions of the optimal obstacle avoidance problem. In this section, a novel hybrid collision avoidance methodology using a machine learning will be discussed.

The proposed hybrid methodology for the optimal collision avoidance is based on the framework of the generic decision-making process [132] because the hybrid methodology as a data-driven approach is a top-down decision making process based on the pre-evaluated data. The generic decision-making steps as a top-down design decision support process is described in Figure 28. The main steps of the process entail six steps: establishing the need, defining the problem, establish value, generating feasible alternatives, evaluating alternatives, and making a decision.

The proposed framework of the hybrid collision avoidance methodology is conceptually represented in Figure 28. The first step is the definition of the problem. The second step is defining the input space. Then, objective functions (i.e., cost function) are identified, and an overall evaluation cost function is determined to merge multi-objective functions. The next step is building various alternatives for an optimal collision avoidance trajectory problem. From these collision avoidance alternatives, the design of experiments is conducted, and the overall cost function for all collision avoidance alternatives is evaluated from the experiment results. Then, the prediction model is specified by a machine learning algorithm. This prediction algorithm has a simple functional form that is computationally efficient in the on-line process. In the on-line process, the best collision avoidance algorithm is selected according to the results of the prediction model based on the vehicle conditions. The functions

corresponding to each step of the hybrid optimal collision avoidance methodology will be described in the subsections that follow.

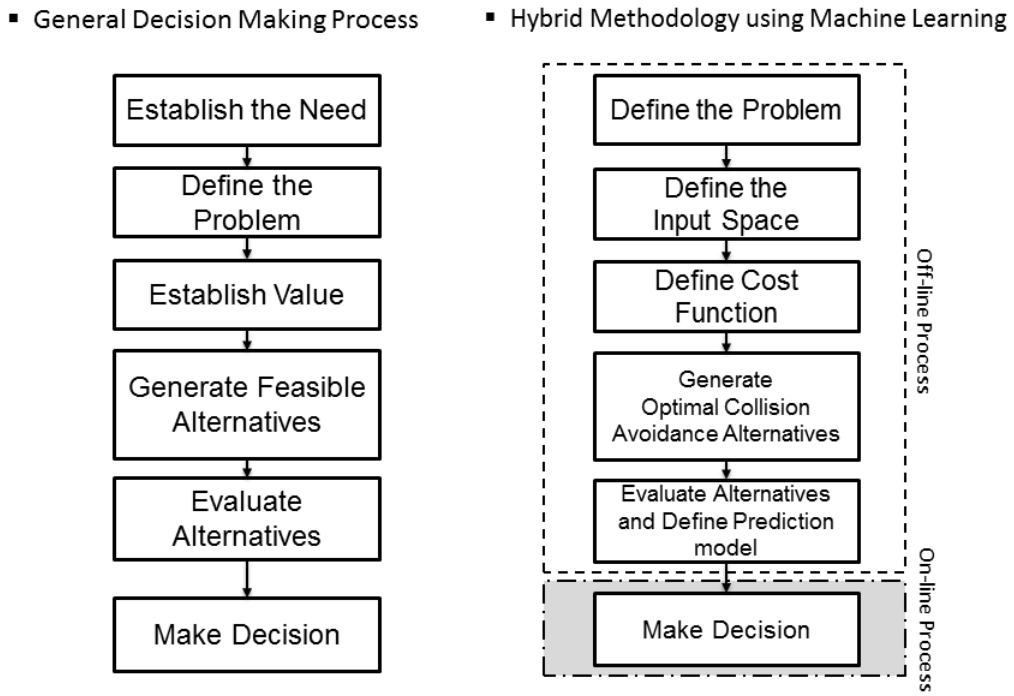



**Figure 28:** Framework of hybrid collision avoidance methodology using a machine learning technique

### 3.3.1 Step 1-2: Define the problem and the input space

The first step of the hybrid optimal collision avoidance methodology is defining a problem. During this step, we define a problem to be solved through a trajectory optimization in a collision avoidance domain. An example is an obstacle avoidance for terrain obstacles, such as a building, tree, or mountain. Other examples are avoiding a moving obstacle or a combination of moving and fixed obstacles. In this thesis, our interested problem is an obstacle avoidance problem with non-cooperative and fixed obstacles. Therefore, the problem for the hybrid method is a collision avoidance

problem under both a non-cooperative and a ground obstacle.

The second step is defining an input space. The input space indicates the space including all required input variables to solve on the optimal trajectory problem. In the case of an obstacle avoidance problem with a fixed obstacle, the variables of the input space can be vehicle states (position, velocity and acceleration), obstacle information (obstacle position and size of obstacle), or a target position and sensor specification (distance, azimuth, and elevation range). However, the input space has large dimensions that cause computational inefficient solution for a prediction model due to high dimensional space. To reduce the dimension of the input space, one can adopt a relative coordinate system. For instance, instead of expressing the vehicle position and obstacle position in the global coordinate system, a single relative coordinate system to describe both positions will reduce the dimension of the required input space.

The next step is specifying variables' ranges in the input space. In the optimal collision avoidance formulations we introduced, the variables' ranges can be determined. The relative positions between a vehicle and an obstacle position can be identified from the sensor specification since obstacle detection range is restricted by the sensor specifications, such as azimuth and elevation angles, and distance range.

The allowable velocity and acceleration ranges can be defined by a flight envelope and V-n diagram, which addresses the structural and aerodynamic maneuverability limitations [134][12]. The input range for the size of an obstacle can be determined by relevant statistical data about interested areas. Figure 29 summarizes the notional concept of defining the input space.

### 3.3.2 Step 3: Define the cost function

The third step is defining a cost function. In this section, we will discuss two cost functions (Optimal trajectory cost and computation runtime), and the aggregation

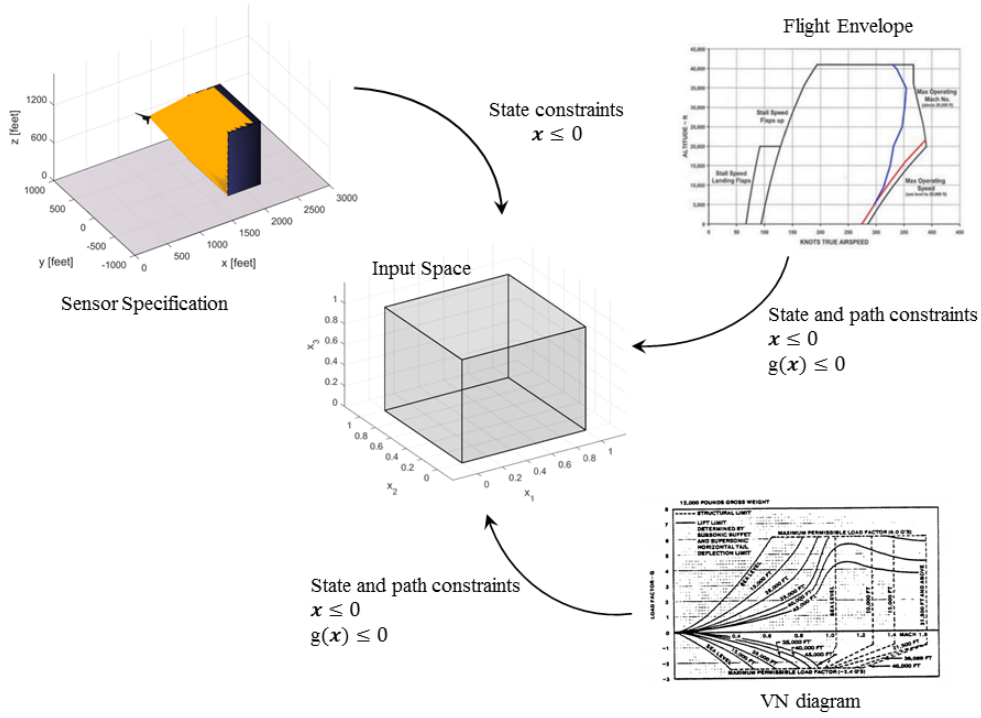



**Figure 29:** Notional diagram of designing an input space

of two cost functions.

In the trajectory optimization problem, we need to define an optimal trajectory cost function, that is also called performance index. This cost function can be formulated into diverse forms, such as time-optimal, energy optimal, or combination of the two cost functions. For instance, in a time-optimal problem, a cost function can be $J = t_f$. In an energy-optimal problem, a cost function can be $J = \int_{t_0}^{t_f} u^2$. In our collision avoidance problem, the cost function is defined by a minimal-energy problem shown in Equation 91.

In the trajectory optimization problem, we also consider an additional metric, computation runtime because the characterization of the UASNAS problem requires a large number of experiments resulting from diverse mission scenarios and UAV platforms; thus, the trajectory optimization should have a computationally efficient structure. Moreover, the previous sample experiments show that the computational

runtime to solve the optimal trajectory problem varies depending on its mathematical complexity. Therefore, the computational runtime is selected as a metric.

Next, we need a method to identify the best collision avoidance alternative. The defined optimization problem is a multi-objective problem that entails the optimal trajectory cost and the computational efficiency. Thus, we need to specify an aggregated objective function $J = g(w_1\bar{\mathbf{J}}_{opt}, \quad w_2\bar{\mathbf{J}}_{com})$. In the equation, $w_1$ and $w_2$ are weights, and $\bar{\mathbf{J}}_{opt}$ and $\bar{\mathbf{J}}_{com}$ are attributed costs, which are the computational runtime and the optimal trajectory cost, respectively. The function $g(\cdot)$ is a multi-objective function. For the evaluation multi-objective function, the well-known approach is the Pareto frontier method. The Pareto frontier adapts the weighted sum function shown in the following form.

$$\mathbf{J} = W_1\bar{\mathbf{J}}_{opt} + W_2\bar{\mathbf{J}}_{com}, \tag{104}$$

where $W_1$ and $W_2$ are weights, and $\bar{\mathbf{J}}_{opt}$ is a normalized optimal cost, and $\bar{\mathbf{J}}_{com}$ is a normalized computational time. The new cost function, which is named as an overall cost function, enables the evaluation for the best strategy in the step of evaluating alternatives.

### 3.3.3 Step 4: Generate optimal collision avoidance alternatives

This step is formulating multiple optimal collision avoidance alternatives. The optimal collision avoidance algorithm can be built by various mathematical formulation with different complexity of constraints. Note that other constraints, such as a dynamic constraints and path constraints except the constraint for the obstacle avoidance should be same. In addition, the cost function must be same since we will specify the best collision avoidance algorithm using the specified cost function. Assuming that the number of the formulated avoidance algorithms are $n$, an avoidance

alternative space is $\mathbf{A} = [A_1, \quad A_2, \quad ..., \quad A_n], \quad \mathbf{A} \in \mathbb{R}^n$. $\mathbf{A}$ is the avoidance alternatives vector, $n$ is the number of collision avoidance algorithms and $A_i$ indicates each avoidance algorithm.

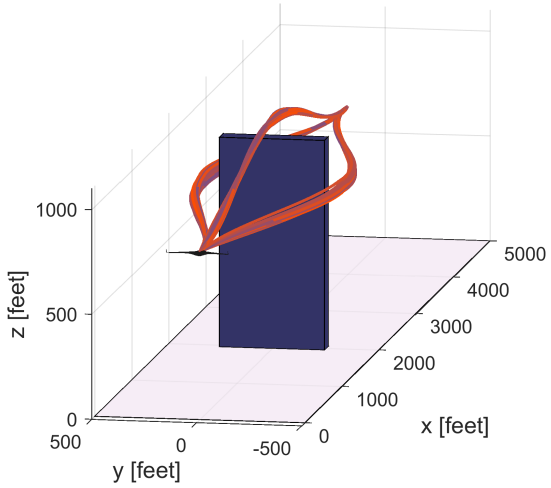
In this thesis, we have introduced four collision avoidance algorithms and one benchmark case. In the hybrid method, we will only consider the proposed four collision avoidance algorithms (PNORM, SCAA-1, SCAA-2, SCAA-3) described in Tables 10, 11, and 12 as alternatives in the hybrid method since the rule-based approach reveals worse performance resulting from the sample case studies. It means that $n$ is four. For the sample studies of the hybrid method, we generate 100 obstacle avoidance trajectories of individual methods. In the sample experiments, we fixed initial/boundary conditions described in Table 10 and randomly selected initial velocity conditions.

### 3.3.4 Step 5: Evaluate alternatives and define perdiction model

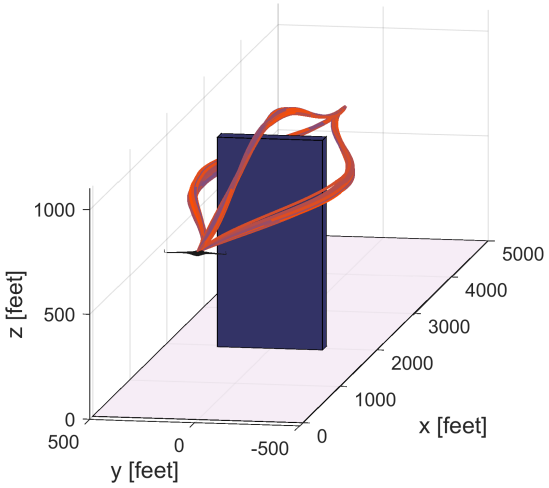
The most novel step in the hybrid methodology is the evaluation of alternatives and the generation of the prediction model. The objective of this step is to define the prediction model from the overall cost analysis of the alternatives through a machine learning technique. In the first step, we execute the design of experiments. In the design of experiments, a space-filling design method was chosen.

Next, we run the optimal collision avoidance problem based on the design of experiments and compute the overall cost function presented in Equation 104. Then, from the results of the overall cost, we can specify the best collision avoidance strategy. The best strategy is determined by solving a classification problem. The classification problem provides a prediction model that identifies the best one in on-line process.

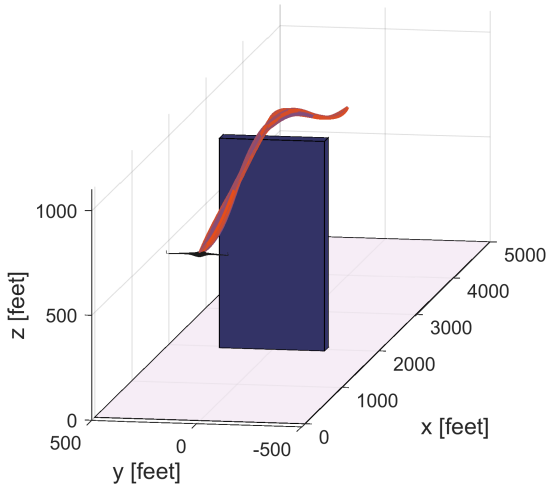
Figure 31 shows the notional concept of the evaluation for the alternatives and the definition of the prediction model. In the first step, we collect 100 sample cases and simulates these cases of each collision avoidance algorithm. The first figure shows the



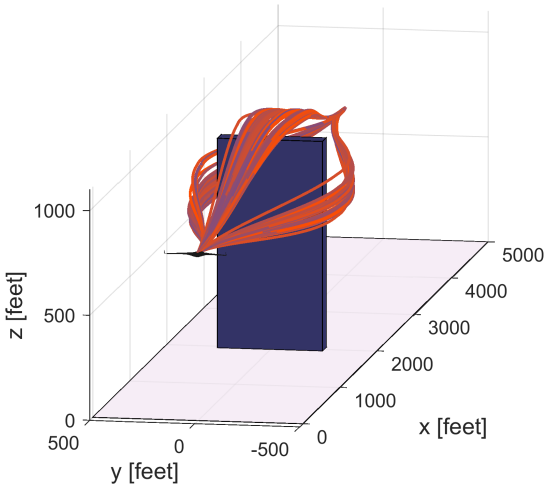

(a) Optimal collision avoidance algorithm PNORM




(b) Optimal collision avoidance algorithm SCAA-1




(c) Optimal collision avoidance algorithm SCAA-2




(d) Optimal collision avoidance algorithm SCAA-3

**Figure 30:** Optimal collision avoidance trajectories of each avoidance method

results of two metrics (optimal trajectory cost and computational runtime) of four alternatives; thus, each method has 100 results. The second graph is the results of the best strategy based on the overall cost function that is 100 data. In the figure, we presume the input space is two dimension. These 100 data set is classified by a machine learning technique. The last graph is the result of the classification. The classification problem can be specified through diverse classification algorithms such as k-nearest neighbor, Bayesian multi-class classification, neural network classification and ensemble learning, which are the kinds of supervised learning for classification problem in machine learning field [53][126]. Due to various learning techniques for a classification problem, it leads to a following question: *Which classification method is the best technique for the proposed hybrid collision avoidance methodology?* In the later subsection, we will discuss the details of a machine learning algorithm for a classification problem.

#### *3.3.4.1 Classification method*

The key idea of the hybrid optimal collision avoidance methodology is a classification algorithm, which is one of the machine learning techniques, since the class classification technique provides the best avoidance mode information according to the vehicle conditions and the detected obstacle information. Our problem is a multi-class classification problem since we have multiple alternatives. The representative multi-class classification algorithms are decision tree, K-nearest neighbor, Bayesian multi-class classification algorithm, multi-class classification using neural network and ensemble learning.

Table 14 summarizes the features of the representative learning techniques. The decision tree technique gives classification information based on the input variables through sorting out the inputs from the tree root to the edges node. This technique

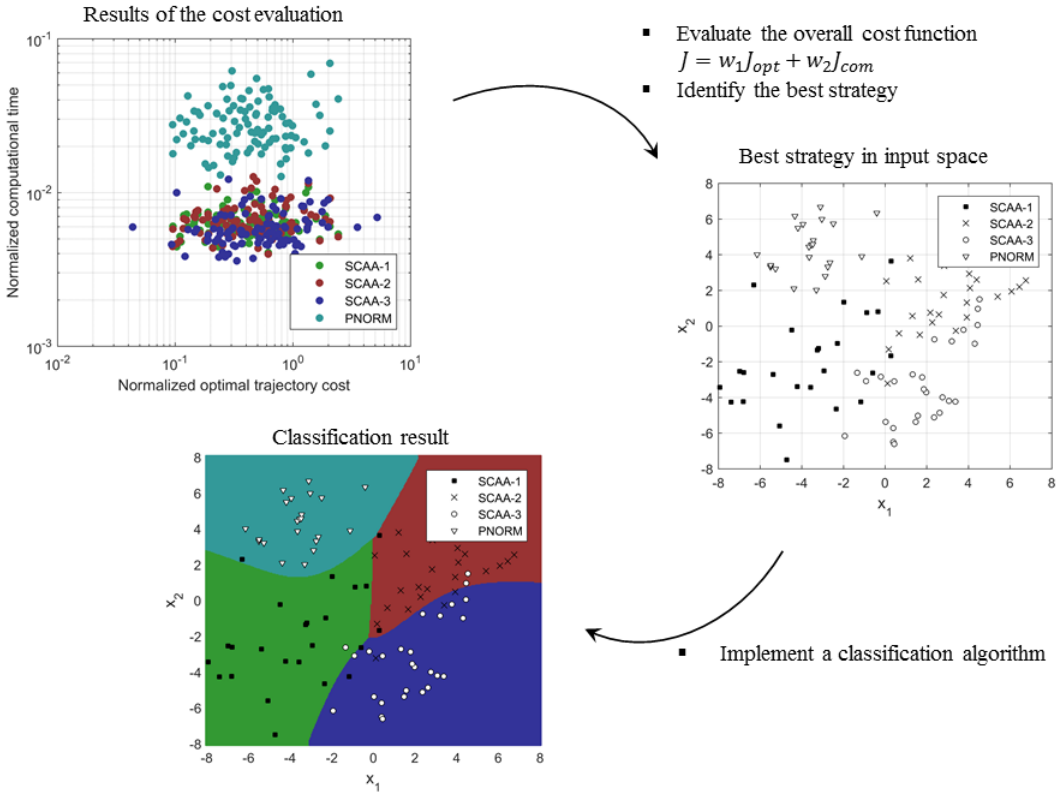



**Figure 31:** Notional concept of the evaluation for the alternatives and the definition of a prediction model

has rapid fitting and prediction, but the prediction accuracy rate is medium. K-nearest neighbor does not require any pre-processing to create a prediction model. This technique directly utilizes training data points for the identification of the multi-class classification. The benefit of the K-nearest neighbor algorithm is that it does not have the pre-processing structure to generate a prediction model. However, this technique acquires the classification information from an on-line process, so the required computational time is highly dependent on the number of training points. Another classification is Bayesian multi-class classification (Naive Bayes), which needs the process of estimating the probability distribution of each alternative (collision avoidance method) through Gaussian assumption or a Kernel estimator. This technique can give accurate classification results when a precise distribution estimation is achieved.

Neural network multi-class classification method computes the prediction model using neural network for the classification, which is also called a multi-layer logistic classification method. The other technique is the ensemble learning that uses combinations of multiple weak learners (i.e., a prediction model) to build a strong learner that is a combination of the weak learners. There are many introduced methods of the ensemble technique in the literature such as boosting, Adaboost, and ensemble neural network. The representative learning algorithms and characteristics are presented in Table 14. In this table, $*$ indicates that performance is dependent on the amount of the evaluation by the Kernel function. For candidates of the classification algorithms, we select neural network and ensemble neural network because they provide high accurate prediction result, fast prediction speed and low memory usage.

**Table 14:** Learning algorithms for multi-class classification[98]

| Algorithm | Predictive Accuracy | Fitting Speed | Prediction Speed | Memory Usage |
|---|---|---|---|---|
| Trees | Medium | Fast | Fast | Low |
| SVM | High | Medium | * | * |
| Naive Bayes(Bayesian Classification) | Medium | * | * | * |
| K-Nearest Neighbor | * | N/A | Medium | High |
| Neural Network | High | Low | Fast | Low |
| Esemble Neural Network | High | Low | Fast | Dependent on architecture |

Appendix A introduces the characteristics of the representative multi-classification algorithms such as Neural Network (NN), Ensemble Neural Network using Bagging (ESNN-Bagging),and Ensemble Neural network using Ordinary Least Square technique (ESNN-OLS). For the weight optimization process of the neural network and the ensemble neural network, we should specify optimal structure about the neural network and the ensemble learning with neural networks, such as a number of layers, a number of nodes, and a regularization parameter, and a number of neural networks. This optimal neural network structure makes a research question:*How can we define the optimal structure of a neural network and an ensemble neural network?*

For the optimization of the neural network optimization, we develop a new optimization process. The main idea of the optimization process includes two processes:

optimization for a single neural network, and optimization of an ensemble structure. These two main processes effectively reduce computational resources compared to the entire single and ensemble structure optimization. The detailed description will be discussed in the later section.

Figure 32 illustrates the proposed methodology that is the optimization scheme with a full-factorial design experiments. In the first step, we collect training data, $D = [(\mathbf{x}_1, y_1), (\mathbf{x}_2, y), \ \cdots, (\mathbf{x}_m, y_m)\ ]$ from the optimal trajectory problems. In the training data, $\mathbf{x}$ indicates input variables ($\mathbf{x} \in R^{m \times n}$), and $y_m$ is collision avoidance modes ($y \in R^{m \times k}$). $m$ is the number of training data, $n$ is the dimension of the input space, and $k$ is the number of the collision avoidance algorithms. Collecting the training data is executed by full-factorial experiments. The design variables of the full-factorial experiments are number of layers, number of nodes, regularization factors and number of neural networks. Note that the number of neural network is the design variable for an ensemble neural network structure in the second experiment (i.e., the optimization of the ensemble learning structure). Next, we solve an optimization problem to identify weights of a neural networks defined from the design of experiments. For the optimization process of the neural network weights, we implement gradient-based approach using a back-propagation method and adopt $K$-fold validation method that validates multiple rounds to decrease variability. In details, we partition the entire data into $K$ test sets. In each cross-validation process, we select one of the test sets and use the rest of the data sets as a training data. We perform $K$th optimization processes using different test data.

After executing all designs of the experiments, we sort the test results of all neural network structures based on the high prediction success rate. Among all neural networks, the neural network with the top prediction performance is selected as the single neural network for the hybrid methodology. In the optimization of the ensemble learning structure, we need to optimize the number of the neural networks.

For this work, we will perform additional full-factorial design about number of neural networks. In the full-factorial experiments, each neural network of the ensemble learning structures is defined based on the sorted results of the single neural network. The sorted results are used because if the design variables in an optimization process entails a full-factorial experiment with all variables (the number of layers, the number of nodes, a regularization factor, and the number of neural networks) simultaneously, the experiment will have computational issues resulting from a large number of experiment cases. Therefore, we use the outcomes of the single neural network for the optimization process of the ensemble learning to reduce the computational expense. For instance, if an experiment has three neural networks in an ensemble learning structure, the top three neural network with highest prediction rate will be selected. For the ensemble learning algorithm, we adopt the bagging and OLS technique described in Section A. After the full-factorial experiments, the ensemble structure will be determined based on the best prediction results. To be more specific, the number of neural networks for the bagging and OLS technique is specified.

### 3.3.5 Step 6: Make a decision

In the previous step, we introduced the optimization process for the single neural network and ensemble neural network prediction models, which specify the best obstacle avoidance alternative. Specifically, the prediction model solves the multi-class classification problem and gives the best solution. The last step is a decision making step that identifies the best option among a set of obstacle avoidance alternatives. In other words, the prediction model will be adopted to select the best alternative in the on-line process.

Figure 33 illustrates the block diagram of the novel hybrid optimal collision avoidance algorithm. The machine learning algorithm that includes a prediction model

resulting from multi-class classification determines the best optimal collision avoidance mode based on vehicle states and airborne sensor information through using the prediction model. This best avoidance mode information is updated to real-time optimal trajectory function so that the function can generate the best optimal collision avoidance trajectory according to the predicted best avoidance mode.

Figure 34 summaries the entire process of the hybrid methodology.

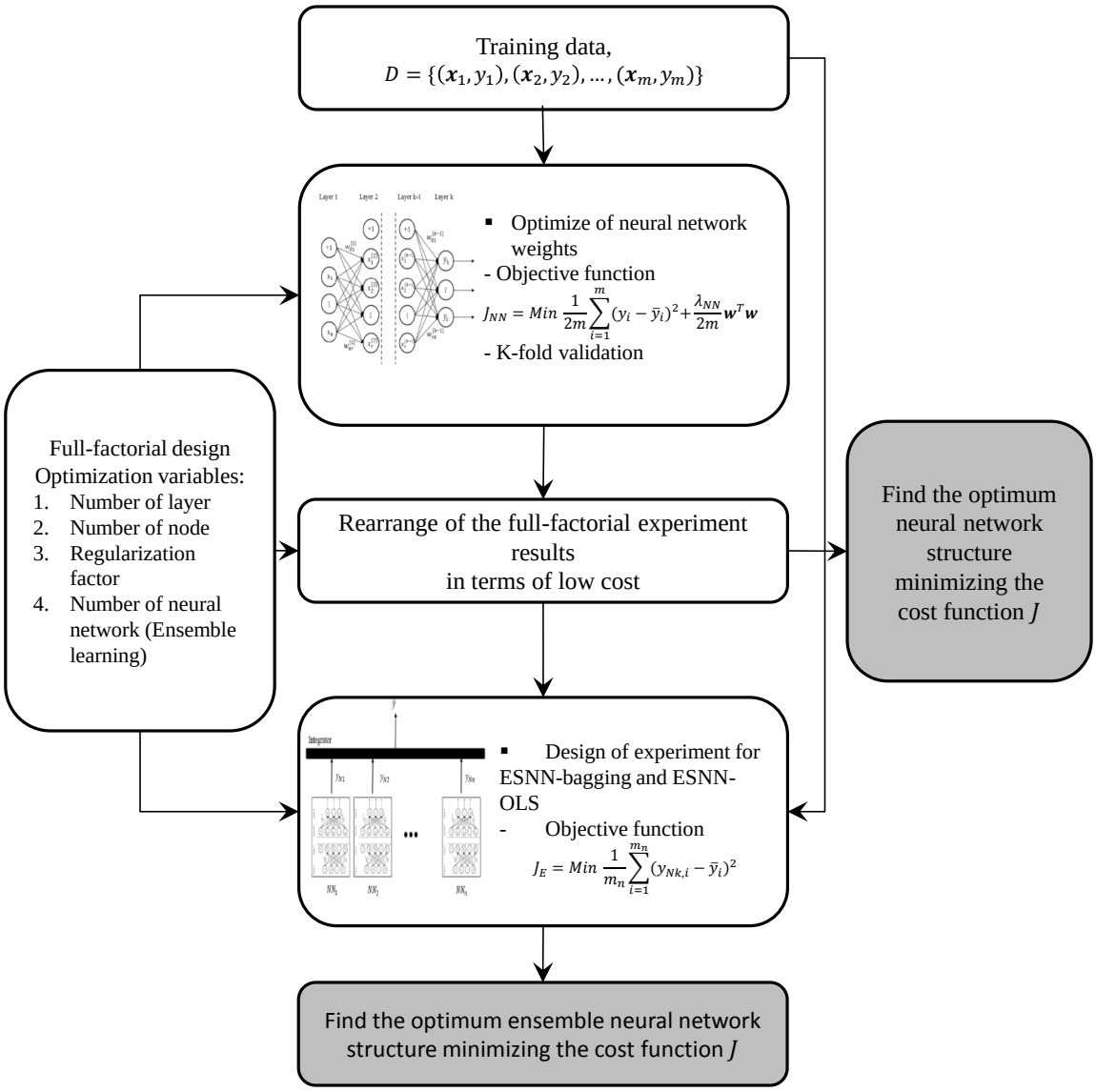



**Figure 32:** Flow diagram of the optimization of neural network and ensemble neural network structures

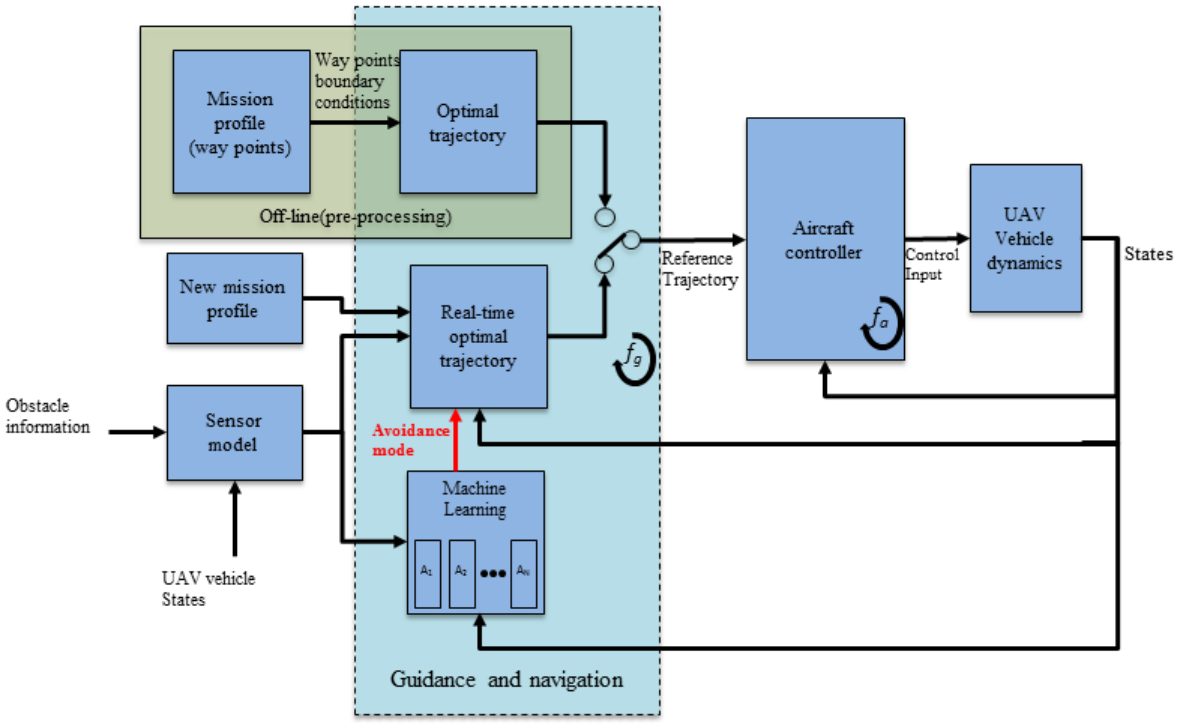



**Figure 33:** Framework of hybrid optimal collision avoidance algorithm

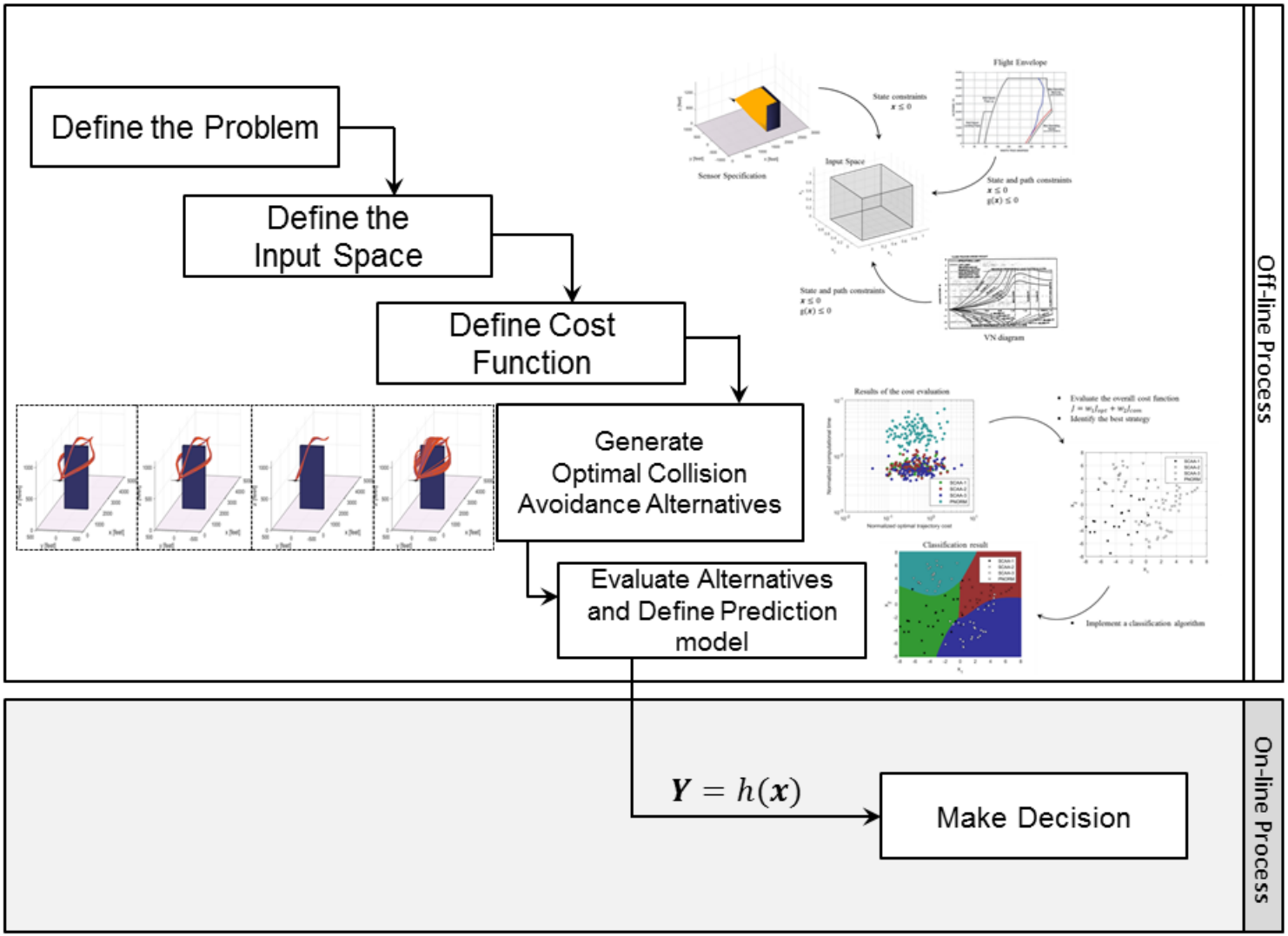



**Figure 34:** Framework of hybrid collision avoidance methodology using machine learning

## *3.4 Numerical simulation*

### 3.4.1 Performance of obstacle avoidance algorithms

In the previous section, we formulate following three hypotheses based on some observations from the sample case studies.

- Waypoint based collision avoidance algorithm (SCAA-1) will be a dominant solution with respect to the overall cost function, if the weight $W_1$ is low value.
- Collision avoidance algorithms with one-dimensional inequality constraints (SCAA-1, SCAA-2) will improve the overall cost if the weight $W_1$ is medium value.
- Collision avoidance algorithm with two-dimensional inequality constraint (PNORM) will improve the overall cost performance if the weight $W_1$ is high value.

For the validation of three hypotheses, we compute probability of the best strategy resulting from the design of the experiments for the hybrid approach that include 15,466 cases. The best strategies are judged from the lowest result of the overall cost function, which is shown in Equation 104. Figure 35 illustrates the results of the overall cost function. As expected, in the low value of the weight $W_1$, the collision avoidance algorithm SCAA-3 has outstanding performance because the mathematical complexity of the SCAA-3 is the simplest. That is to say, when $W_1$ is zero, approximately 67.75 [%] of the total experiment data is specified as the best collision avoidance algorithm. On the other hand, other obstacle avoidance algorithms SCAA-1, SCAA-2, and PNORM have relatively lower probability of the best strategy. This result clearly presents that the waypoint-based approach SCAA-3 is the dominant solution in the low value of the weight $W_1$.

In the middle value of the weight $W_1$, the collision avoidance algorithms SCAA-1, SCAA-2, and SCAA-3 show good performance. Between 30 [%] and 35 [%] among the total experiments are identified as the best strategy. On the other hand, the

two-dimensional obstacle algorithm PNORM has the lowest performance. This result supports the second hypothesis: one-dimensional collision avoidance algorithm (SCAA-1 and SCAA-2) will have high performance because the sum of the probability of the best strategy resulting from these two strategies have around 65 [%] when the weight $W_1$ is 0.5.

In the high value of the weight $W_1$, the performances of the SCAA-1 and SCAA-2 degrade, but the collision avoidance algorithm PNORM with two-dimensional inequality constraint improves the performance with respect to the overall cost function. Note that although the PNORM has the most sophisticated inequality constraint, it cannot have the best obstacle avoidance algorithm at the highest value of the weight $W_1$ because we impose the limitation of the iteration number to solve the optimal trajectory problem which achieves tractable computational time. Therefore, the result presents that the hypothesis 'Collision avoidance algorithm with two-dimensional inequality constraint (PNORM) will improve the overall cost performance when the weight $W_1$ is high value' is rejected.

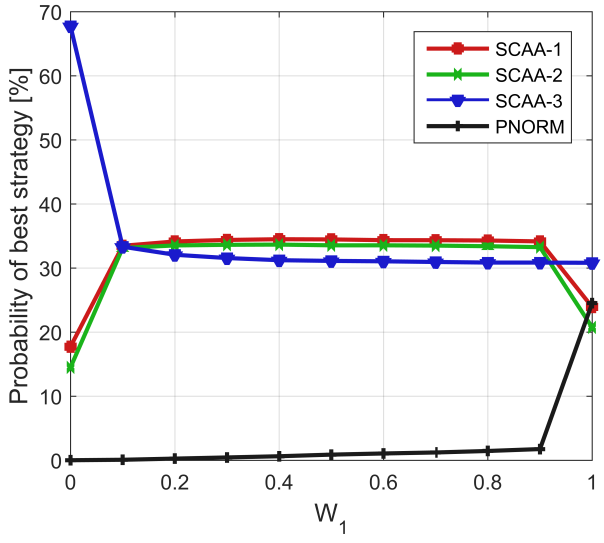
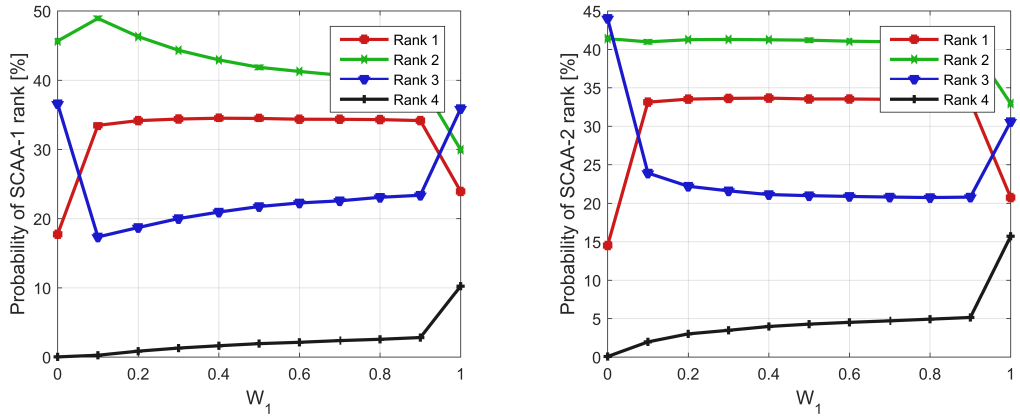



**Figure 35:** Overall cost analysis

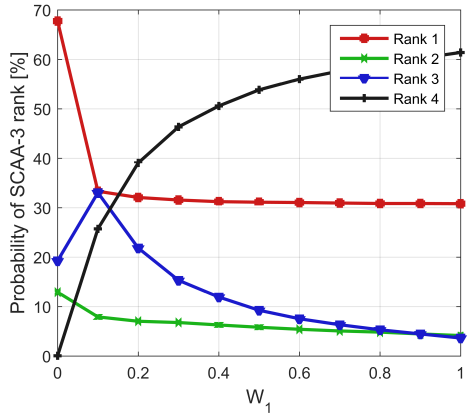
To characterize individual obstacle avoidance algorithm, we compute probability of ranks according to each algorithm. In Figure 36, four graphs show the results of rank probability according to four different obstacle avoidance algorithms. The visual inspection reveals that the SCAA-2 and SCAA-3 have similar trend. In the middle of the weight $W_1$, most cases around 70 [%] have first and second ranks. In the SCAA-3 around the low value of the weight $W_1$, most cases are ranked on the first. When the weight increases, the case with fourth rank increases, and the case with the first rank decreases simultaneously. In the PNORM method, when the weight is low, most cases have forth rank, but as the weight $W_1$ increases, the numbers of the first, second and third increase.



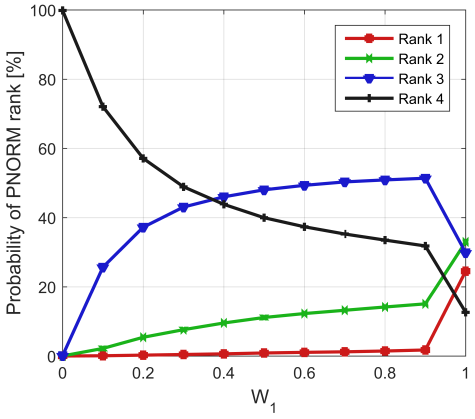

(a) SCAA-1




(b) SCAA-2




(c) SCAA-3




(d) PNORM

**Figure 36:** Rank probability of each collision avoidance algorithm

### 3.4.2 Learning classification algorithm for hybrid method

This section discusses numerical simulation results about learning a classification model by three different learning algorithms, which are a single neural network and two ensemble neural networks (bagging and ordinary least square methods). It also addresses prediction performance analysis of three hybrid methods.

The optimization process introduced in the previous section includes two designs of experiments (DOE). The first DOE is for the optimization of a single neural network. The second DOE is to define the number of neural networks with the optimal ensemble structures. Note that in the second DOE process, we utilize the results of the single neural network from the first DOE because of reducing the number of experiments through decreasing the size of variables.

The details of the experiments in the first DOE, the full-factorial experiments have three design variables: the number of node, the number of layer, and the regularization coefficient. Note that the variables (the number of node and the number of layer) are discrete variables, and regularization coefficient is a continuous variable.

The range of the number of nodes are defined as 1 to 50, and the range of the number of layer is from one to two. These two ranges are identified as initial sample experiments through observing learning performance according to these variables. The regularization coefficient is [0  1].

For the learning prediction model, we collect 15,466 data-set that entails input variables and output labels that indicate four alternatives (PNORM, SCAA-1, SCAA-2, and SCAA3). Among the data-set, 80 % of the data (11563) is utilized for training a neural network, and 20 % of the data (3903) is utilized in a test phase. Using this data-set, the full-factorial DOE is conducted to optimize all neural networks. To measure the performance of each neural network, the prediction success rate is computed.

The full-factorial experiment with 15,466 cases requires large computational resources because of high expense to solve the trajectory optimization problem. Therefore, in order to accelerate running experiments, we adopt the parallel computing resource offered by the PACE cluster that is an advanced computing environment provided by Georgia Institute of Technology.

In the full-factorial experiment, the results of the prediction rate are sorted according to the high prediction success rate. The neural network with the highest prediction rate is selected as a single neural network for the hybrid method. The sorted result shows that the best neural network has two layers, zero regularization coefficient. Each layer (first and second layer) has 46 and 24 nodes, respectively.

Based on the sorted full-factorial DOE results, another full-factorial DOE is executed to specify the structures of the two ensemble neural networks. This second DOE includes one design variable, which is the number of neural network.

The variable, a number of neural network, varies from 3 to 50. This range is identified based on the feasible computational time since the higher number of neural network causes higher online computational time to calculate prediction solution.

The neural network structure of each ensemble structure is defined from the results of the first full-factorial experiments. For instance, once the number of neural network in an ensemble learning is five, we select five neural network structure with the top-five prediction rate from the previous full-factorial experiments.

Figure 37 presents the results of two ensemble learning algorithms (Bagging, and OLS) as the number of neural networks increase. Visual inspection of the results reveals that the performance of both ensemble methods improves in the low number of neural network as the number of neural network increases, but after reaching the maximum performance, the prediction rate of both algorithms decrease. Based on this result, we select the number of neural networks with the maximum performance for two ensemble structures. The bagging ensemble learning method has 10 neural

networks, and the OLS ensemble learning method has 18 neural networks. Table 15 summarizes the results of the optimization for a single neural network and two ensemble learning methods (Bagging and OLS).

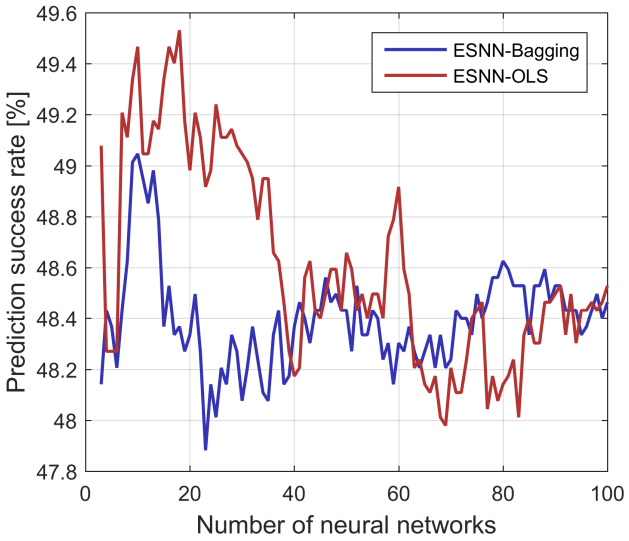



**Figure 37:** Overall cost analysis

Using the optimized the single neural network and two ensemble neural networks, we analyze the performance of three classification algorithms. The performance is measured by the error cost ($J_{err} = J_{opt} - J_{actual}$). In details, the misclassified data is collected from the test results. The collected data is evaluated by the error cost $J_{err}$. Figure 38 shows the error cost of all obstacle avoidance methods including the proposed hybrid methods. In the figure, X axis indicates the number of data, which implies the classification performance. The number of data tells misclassified data. In other words, higher number of data indicates worse classification performance. Table 16 summarizes the percentage of the best strategy. Among seven obstacle algorithms (PNORM, SCAA-1, SCAA-2, SCAA-3, HYBRID-NN, HYBRID-ESNN-Bagging, HYBRID-ESNN-OLS), the three hybrid algorithms show the outstanding performance, and the worst performance is PNORM algorithm. In general, compared to four obstacle avoidance algorithms (PNORM, SCAA-1, SCAA-2, SCAA-3),

**Table 15:** Results of the full-factorial design of experiments

| Rank | Number of layer | Num. of nodes at 1st layer | Num. of nodes at 2nd layer | Regularization factor | Single NN | ESNN-Bagging | ESNN-OLS |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 46 | 24 | 0 | o | o | o |
| 2 | 2 | 48 | 28 | 0.3 | | o | o |
| 3 | 2 | 44 | 38 | 0.2 | | o | o |
| 4 | 2 | 46 | 32 | 0.9 | | o | o |
| 5 | 1 | 38 | 0 | 0.8 | | o | o |
| 6 | 2 | 44 | 46 | 0.2 | | o | o |
| 7 | 1 | 50 | 0 | 0.5 | | o | o |
| 8 | 2 | 50 | 40 | 0 | | o | o |
| 9 | 2 | 50 | 48 | 1 | | o | o |
| 10 | 2 | 34 | 26 | 0.5 | | o | o |
| 11 | 1 | 50 | 0 | 0.6 | | | o |
| 12 | 2 | 48 | 48 | 0.8 | | | o |
| 13 | 2 | 26 | 48 | 1 | | | o |
| 14 | 2 | 40 | 4 | 0.5 | | | o |
| 15 | 2 | 38 | 40 | 0.9 | | | o |
| 16 | 1 | 38 | 0 | 0.1 | | | o |
| 17 | 2 | 46 | 36 | 0.2 | | | o |
| 18 | 2 | 44 | 40 | 0 | | | o |

the proposed hybrid methods has higher classification performance and the hybrid methods present similar performance.

**Table 16:** Rate of the best method

| Algorithm | Rate of the best method [%] |
|---|---|
| PNORM | 0.87 |
| SCAA-1 | 34.48 |
| SCAA-2 | 33.54 |
| SCAA-3 | 31.11 |
| HYBRID-NN | 48.64 |
| HYBRID-Bagging | 49.16 |
| HYBRID-OLS | 49 |

Figure 39 is the probability density function of the cost for all test cases. This result shows that the hybrid methods using a neural network and two ensemble learning are higher probability in the low error cost region compared to others while these hybrid methods are lower probability in the high error cost region. These trends indicates that the hybrid approach provides better avoidance trajectory through selecting one of alternatives (PNORM, SCAA-1, SCAA-2, SCAA-3).

To sum up, from the experiment results, the proposed hybrid methods provide

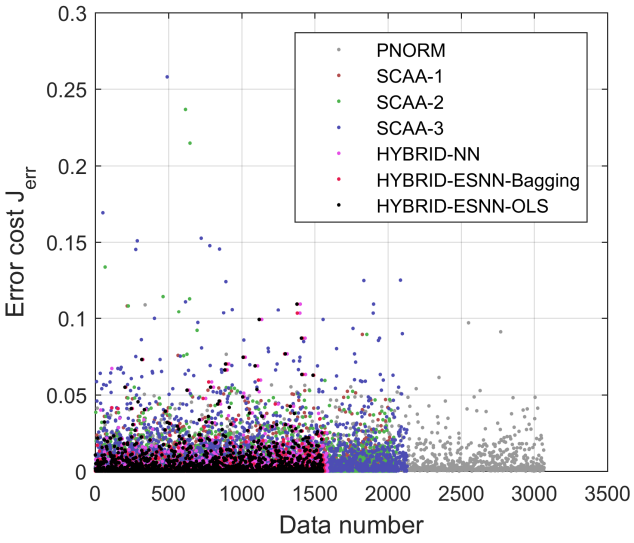



**Figure 38:** Error cost analysis

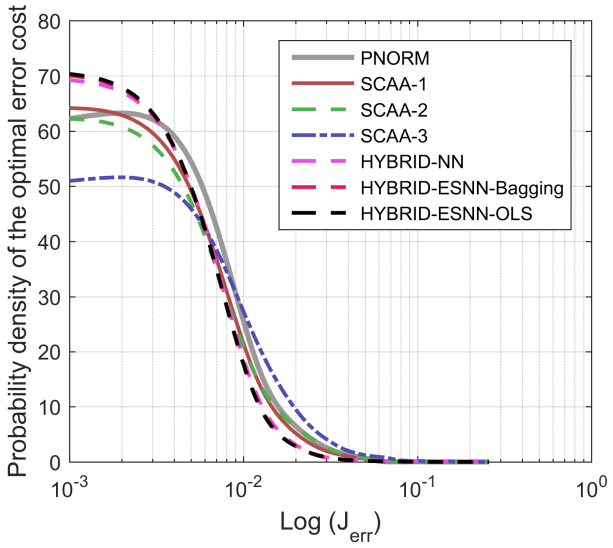



**Figure 39:** Error cost analysis

a better solution. The hybrid method with ensemble learning methods are better performances compared to the hybrid with a single neural network. Nevertheless, the performance difference of the three classifications is minimal in the hybrid method. Moreover, the hybrid with a single neural network present more computationally efficient than other two approaches shown in Table 17. Hence, as a part of the hybrid method, we select the single neural network because of its high classification performance and low computational burden.

**Table 17:** Computation runtime

| Algorithm | Computational time [sec] |
|---|---|
| HYBRID-NN | 0.0012 |
| HYBRID-Bagging | 0.0039 |
| HYBRID-OLS | 0.0048 |

## *3.5 Conclusion*

This chapter describes the optimal collision avoidance algorithm based on the observation of the existing avoidance algorithms. The formulated collision algorithms solve a two-phase optimal trajectory problem with dynamic constraints, a cost function, path constraints, event constraints, and link constraint. The sample case studies of four optimal collision avoidance algorithms present that the mathematical complexity of the trajectory optimization problem affects a computational expense and an optimal trajectory cost. To be more specific, the PNORM, which has a two-dimensional inequality constraint, generates an avoidance trajectory with a low cost, but the required computational time to solve the PNORM is high. On the other hand, the SCAA-3, which has the simplest mathematical formula, requires a low computational expense, but generates a high cost trajectory.

Based on the sample case studies, the best collision avoidance algorithm is dependent on initial conditions and obstacle sensor information. This observation leads to the hybrid collision avoidance algorithm that can select the best collision avoidance algorithm based on the prediction model. The prediction model is determined in the pre-processing through a multi-class classification algorithm that is a supervised learning technique in a machine learning domain. The hybrid collision avoidance algorithm is demonstrated by a numerical simulation. The numerical simulation results present that the proposed hybrid method shows more outstanding performance than the conventional optimal collision avoidance algorithms (PNORM, SCAA-1, SCAA-2, and SCAA3).

The hybrid collision avoidance algorithm is a highly flexible structure. That is, different collision avoidance algorithm can be implemented with the same cost function and dynamic constraints. In the machine learning part, diverse classification algorithms can be applied to improve the classification performance that allows the classification function to select a better strategy.

# CHAPTER IV

# COLLISION AVOIDANCE ALGORITHM IN AN URBAN ENVIRONMENT

There are numerous studies investigating collision avoidance concepts and techniques that can be generally divided into five groups: stochastic methods, road map methods, potential field approaches, geometric methods and optimization based methods. Stochastic methods effectively search nonconvex high-dimensional spaces for global or local obstacle avoidance paths based on the environment as perceived by airborne sensors. The rapid random tree (RRT) [90][92] is a popular method of this kind. Road map methods use visual graph characteristics and path-planning algorithms to partition collision-free paths and create a piecewise linear path or curved path using smoothing techniques [58]. Doebbler et al. have proposed a heuristic approach for optimal path planning that is for General Aviation class aircraft [41]. The fundamental concept of artificial potential field methods is the creation of force map where a waypoint generates an attractive force and obstacles generate repulsive forces [25]. Based on the resulting force map an algorithm generates an optimal collision-free path.

Optimization-based and geometric methods are closely related, and have presented a range of promising solutions. For instance, Chakravarthy and Ghose proposed a collision cone technique to avoid a moving obstacle with irregular shape in two-dimensional space [29]. Watanabe et al. extended this approach with a minimal-effort optimization framework in three-dimensional space [150]. Another example is that of Schouwenaars et al. [143] who suggested mixed-integer linear programming (MILP) that incorporates binary constraints based on the area information outside

of an obstacle. However, this MILP approach is a two-dimensional approach that has limitations in three-dimensional space. Yoshiaki et al. [88] expanded more precise MILP algorithm for three-dimensional collision avoidance problem that includes two phases: the construction of coarse cost map and detail trajectory optimization. These MILP frameworks have been employed for real-time collision avoidance problem in diverse platforms [100]. However, solving MILP is commonly implemented via the branch and bound method which is computational expensive [145] [100].

Moon et al. [105] suggested a rule-based collision avoidance approach that defines an optimal avoidance trajectory resulting from a safe position based on airborne sensor information and a flight envelope protection function. However, when an unmanned aircraft detects multiple obstacles at the same time they are interpreted as a single obstacle, even when there is sufficient space to fly safely between them. As a result highly energy-inefficient obstacle avoidance trajectories may be observed. Kang et al. [80] expanded this framework to generate more efficient trajectories through a global path searching function that is optimized by external sources having obstacle information. Compared to a local-path optimization only, global-path searching and local-path optimization provides more effective trajectories. However, this concept is vulnerable to loss-link or surveillance interruption scenarios where the unmanned aircraft cannot access external obstacle information sources, and is essentially reduced to the same local-optimization approach it seeks to improve upon. In this paper we address the question of how an optimal collision avoidance algorithm can produce highly energy-efficient trajectories in a multiple obstacle environment while relying solely on on-board sensors and capabilities.

The work here presented proposes a two-layer obstacle collision avoidance algorithm that incorporates a global-path optimization and a local-path optimization. In the global-path optimization, based on the detected obstacle information from an airborne sensor, an on-board system identifies a number of obstacles through a clustering

technique. This clustering problem solves a distance-based constraint optimization problem that includes minimum-separation distance between pairs of adjacent obstacles. Then, the system detects a cluster that is a potential threat. In the local-path optimization we employ a multi-phase optimal obstacle avoidance problem to avoid the cluster.

For the online trajectory optimization we employ a model predictive control (MPC) scheme that produces computationally feasible solutions. That is, solving actual nonlinear constrained dynamics trajectory optimization is an NP-hard problem, but MPC solves approximated linear dynamics and simplified constraints, and regularly updates the results of the optimal trajectory in a real-time manner. This MPC approach provides a computationally tractable solution for online optimal collision avoidance problem and has been successfully implemented in the past [105][80][97].

We hypothesize that our two-layer collision avoidance algorithm yields more energy efficient trajectories without incurring in prohibitive computational burden relative to the single-layer approach. In the remainder of the paper we first present the formulation and implementation of our algorithm, and then test the hypothesis via direct comparison of numerical simulations.

## *4.1 New path planning architecture using a learning algorithm*

### 4.1.1 Two-level algorithm concept

The multi-obstacle avoidance problem is inherently challenging. In general the mathematical complexity computational expense of an optimal trajectory solution grows quickly with the explicit treatment of multiple obstacle constraints. It follows that most optimal trajectory frameworks in recent work do not explicitly consider multiple obstacles scenarios [140][105][80]. Instead, existing algorithms employ the collision avoidance framework for one obstacle regardless of the number of downstream trajectory threats, producing inefficient trajectories whenever safe flight between obstacles

is feasible. To address this major shortcoming the new algorithm here proposed features a two-layer structure accommodating online global-path and local-path optimization processes, illustrated in Figure 40. The global-path optimization uses on-board sensor data to efficiently resolve multiple downstream obstacles based on their relative location and separation. It then specifies a potential threat based on the identified multiple clusters and vehicle state information.

The local path optimizer produces an optimal collision avoidance trajectory using the selected global-path trajectory. To do so it solves a multi-phase optimal trajectory problem based on vehicle dynamics, constraints, obstacle information, and mission waypoints. Two fundamental phases are defined for this problem: obstacle avoidance and recovery. In the obstacle avoidance phase the terminal trajectory point is an intermediate waypoint that satisfies three linear constraints around the obstacle, one above and one on either side, guaranteeing a minimum safe separation distance between the aircraft and the obstacle. In the recovery phase the trajectory takes the aircraft from the intermediate waypoint to a prescribed target, or final waypoint. The avoidance path is continuously optimized based on updated airborne sensor information and vehicle states.

### 4.1.2 Two-level algorithm in the guidance, navigation, and control architecture

The guidance navigation and control architecture employed in this work, illustrated in Figure 41, is consistent with typical constructs in current practice. Optimal trajectory generation realized with guidance and navigation elements, and application of aircraft controller to the flight dynamics ensures adherence to said trajectory. In our two-level concept guidance and navigation have an off-line block where the trajectory is defined a-priori in accordance with a prescribed mission profile, pre-defined waypoints, and available obstacle information (if applicable or available). The on-line counterpart regularly updates the trajectory based on the current vehicle state and sensor data

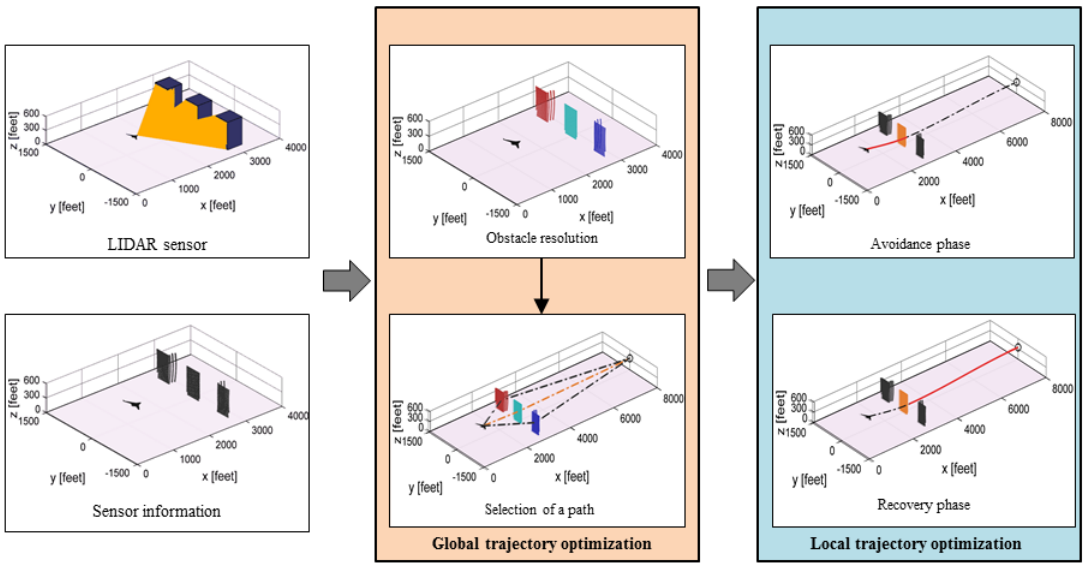



**Figure 40:** Concept formulation and flow of the two-layer collision avoidance algorithm

to compensate for perturbations or the avoidance of unknown obstacles. The global and local components of the proposed trajectory optimization concept are realized in this region of the architecture. As shown in Figure 41 sensor data corresponding to detected obstacles in the operating environment are passed onto the global block where distinct obstacles are resolved, selects a potential threat. The local component then solves for the optimal trajectory based on the specified potential threat. The aircraft controller generates input commands to follow the optimal trajectory, and applies them to the vehicle flight dynamics. The aircraft controller loop monitors vehicle states and updates control inputs with feedback rate $f_a$. The guidance and navigation loop updates the optimal trajectory with feedback rate $f_g$ given aircraft states and the current global path approximation.

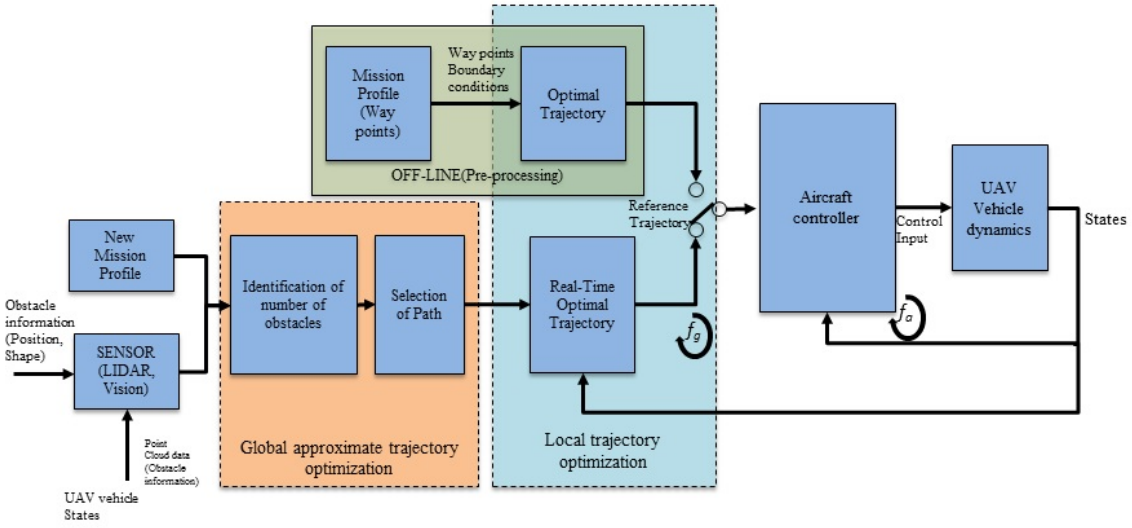



**Figure 41:** New path planning architecture using a machine learning algorithm

### 4.1.3 Global trajectory optimization

#### *4.1.3.1 Obstacle resolution*

The first step in the identification of an optimal global trajectory is the resolution of multiple obstacles ahead of the aircraft's current flight path. This function entails a suitable characterization of obstacles based on their location relative to the aircraft and each other, as detected by the on-board sensor. We adopt a general sensor model that captures the fundamental mechanism underlying many such instruments. The model features an array of distance-measuring rays originating at the sensor, uniformly structured over the horizontal and vertical field of view. We assume the field of view is symmetric about the x-axis in the vehicle coordinate system $G_V$ and forward looking. The distance to an external body along any ray is known whenever the ray intersects a surface of that body, unless it exceeds a detection range specified in the model. The relative angular placement of all sensor rays within the field of view is prescribed so that the angle of each the ray in the vehicle coordinate system is known. Accordingly, with measured distances along known angles in $G_V$ the sensor data array provides the location of all detection points. Sensor capabilities may be defined by

adjusting the vertical and horizontal field of view, density of distance-measuring rays, and the detection range. We assume that instrument error and signal return time are negligible, so that the model is deterministic and instantaneous. We also assume the detection range is isometric so that no variation exists along vertical or horizontal field of view directions.

Obstacle resolution is hence realized as an operation on the sensor data array where separate objects must be characterized and distinguished. To this end we consider the vast body of work in machine learning, pattern recognition, and data mining. Techniques in this domain can be broadly categorized as *supervised* and *unsupervised*. In the former a required a-priori data set is used as the basis upon which some underlying relationship is inferred and applied to a posterior data set. This approach is central in many techniques for set membership (classification), mapping, and anomaly detection. Unsupervised techniques do not require an a-priori data set, and instead establish high-level data structure and properties from those at a lower-level. The most popular and well-known class of unsupervised techniques is clustering, where pairwise similarity or proximity between data points is evaluated and used as the basis to construct clusters. Clusters comprise high-level data structure, whereas data point proximity is lower-level. Given that sensor data is comprised of the location of all detection points, we assert that clustering is an ideal technique to resolve multiple individual obstacles.

In general clustering techniques are appealing due to their simplicity and scalability for a broad spectrum of applications. There are also many known shortcomings, but continuing work to address them has resulted in a rich pool of algorithms. The $k$-means algorithm, one of the most popular and classical clustering methods, partitions a data set onto $k$ clusters by sequentially assigning points to clusters based on their proximity and updating the cluster definitions to reflect said point assignments. Point proximity to a cluster can be evaluated with different measures of distance

in the parameter space, and relative to different points (e.g., cluster centroid, point closest to the centroid, nearest/farthest point, etc.), which gives rise to the myriad of variations of the method observed in the literature.

The main drawback of this clustering technique rests in the requirement to specify the number of clusters $k$ a-priori. Some approaches have been proposed to circumvent this issue, for instance, by evaluating some top-level partition quality metric for different values of $k$ and selecting that for which said metric is optimal (see for example Ref. [118]). These approaches often fail to provide sufficiently tractable results and are computationally expensive. Jia et al. posit that if the point cloud of each group has a non-convex shape these clustering algorithms yield results that only guarantee a local minimum [71].

For the obstacle avoidance trajectory problem the number of clusters $k$ cannot be user-specified. Moreover, resolution of clusters as separate objects must be pursuant of an explicitly defined minimum separation distance. Ester et al. [45] proposed DBSCAN, a density-based spatial clustering algorithm. In DBSCAN the number of clusters are not defined a-priori but rather result form the clustering process itself, driven by the grouping of points that meet a user-defined minimum density. The latter is established by two parameters that define the $\epsilon$-*neighborhood* of a point $p$: the minimum number of points *minPts* that must be within a distance $\epsilon$ from point $p$. These required user-defined algorithm parameters are lower-level data attributes that are more intuitive, meaningful, and practical than emergent clustering characteristics such as number of clusters. Point density also provides an ideal mechanism to resolve adjacent clusters while guaranteeing that a minimum safety distance between them exists. Specifically for the current application of obstacle resolution, $\epsilon$ can be used to explicitly set the minimum distance between obstacles beyond which they are resolved as separate objects.

Another approach that does not require prescription of the number of clusters is

spectral clustering. This technique is based on algebraic graph theory where a graph partitioning problem is solved from similarity characteristics. The graph partitioning problem can be solved with diverse methods: ratio cut, normalized cut, minimum cut or graph cut using an optimization function. These methods as traditional partitioning techniques are NP-hard problems [71]. A more computationally attractive alternative is found in the NJW algorithm, proposed by Ng. et al [113], while providing a classical spectral clustering method. In this approach the number of clusters $k$ can be determined from the top $k$ eigenvalues of the Laplacian matrix $\mathbf{L}$, generated from the affinity matrix $\bar{A}$ where each of its elements is defined with the Gaussian function

$$\bar{A}_{ij} = exp(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}) \tag{105}$$

The vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ denote data points, or vertices in the graph-theoretic sense, and $\sigma$ is a hyper parameter. Despite its implementation across a variety of applications the NJW algorithm presents a major drawback, namely, that identification of the top $k$ eigenvalues is only tractable whenever they are distinctly greater than all remaining eigenvalues starting with $k+1$. In other words, a sharp drop past the $k^{th}$ eigenvalue is a clear and tractable indication of $k$. However, such a sharp delineation across eigenvalues is not always observed, and a more gradual progression of diminishing eigenvalues is common. In that case determining the number $k$ of clusters does not have a tractable standard, and the top $k$ eivenvectors do not always produce correct clustering results [71]. To solve this shortcoming, the number of cluster $k$ can be determined from k-block diagonality of the Laplacian matrix $\mathbf{L}$ [49]. Once $k$ is known, clusters are resolved with any algorithm requiring $k$ as an input, such as $k$-means. As described, this approach does not feature a characteristic distance parameter to tune the algorithm and set the minimum inter-obstacle distance for resolution as separate clusters. To address this gap we utilize a distance based adjacency matrix $A$ in lieu of the affinity matrix $\bar{A}$ in Eq. 105. An adjacency matrix is binary and denotes pairwise

connectivity of nodes with a "1", and "0" otherwise. We introduce a characteristics distance $d$ as the basis of distance-based adjacency and define elements in $A$ as follows:

$$\begin{aligned} A_{ij} &= 1, \quad if \, \|\mathbf{x}_i - \mathbf{x}_j\| \leq d \\ A_{ij} &= 0, \quad if \, \|\mathbf{x}_i - \mathbf{x}_j\| > d \end{aligned} \tag{106}$$

The overall approach is summarized in Algorithm 2. Caution should be exercised whenever noise is present in the data because it degrades k-block diagonality [49]. For applications where data features high signal-to-noise ratio, or in the extreme case of completely deterministic data such as that produced by our deterministic sensor model, concerns associated with k-block diagonality may be ignored.

**Algorithm 2** Spectral clustering

Inputs: point cloud information $\mathbf{P} \in \mathbb{R}^{n \times d}$, distance constraint between obstacles $D_o$
(1) Distance-based adjacent matrix $A_{ij} \in \mathbb{R}^{n \times n}$, $d = D_o$
(2) Degree matrix $D_{ii} = \sum_j A_{ij}$ if $i = j$, otherwise $D_{ij} = 0$
(3) Unnormalized Laplacian matrix $L = D - A$
(4) Solve for $k$ with $k$-block diagonality: $rank(L) = n$ - $k$
(5) Treat each row of $L$ as a point in $\mathbb{R}^k$, and cluster into $k$ clusters via $k$-means
Outputs: Clustering result $C_1$, $C_2$, ... , $C_k$

We denote the distance $D_o$ as the user-defined minimum observed distance between obstacles for resolution via sensor data clustering. Accordingly, in our implementation of the two clustering approaches here discussed the characteristic distance parameter, $\epsilon$ for DBSCAN and $d$ for spectral clustering with k-block diagonality, are set to guarantee that sensed obstacles that appear separated by a distance $D_o$ or more are resolved as separate objects.

Each of the clustering algorithms is implemented in a two-dimensional domain where the sensor data has been collapsed onto the $yz$ plane of the vehicle coordinate system. Eliminating obstacle depth affords significant computational efficiency in two ways. First, it expedites runtime of the clustering function. Second, the need to

resolve obstacles based on depth (along the longitudinal $x$ axis of the vehicle coordinate system), and the need to incorporate this information in a more complex global trajectory optimization, is circumvented altogether. As a result obstacles partially or totally blocked by nearer obstacles are not resolved as separate objects. This information reduction does not compromise the optimality of the global trajectory selection nor does it degrade the efficacy of the local trajectory optimization or the overall collision avoidance capability. On the contrary, even without depth data in obstacle resolution clustering, the proposed approach is easily handles downstream obstacles hidden by nearer ones, and provides energy efficient trajectories as illustrated by results in Section 4.2.3.

#### *4.1.3.2 Selection of a path*

The second step in global trajectory optimization is to construct global path alternatives commensurate with the obstacle information produced in the previous step. We note however that there exists an infinite number of feasible trajectories that maintain a minimum safety distance from all obstacles. We propose a tractable approach whereby a finite set of approximate global trajectories are defined as an ordered sequence of waypoints $[x_0, x_{ik}, x_t]$. $x_0$ is the current aircraft position, $x_{ik}$ is an intermediate waypoint associated with the $k^{th}$ resolved obstacle, and $x_t$ is a posterior target location. For the visualization purpose, let's assume that the intermediate waypoint is generated as the cluster average, or centroid, of the obstacle:

$$\mathbf{x}_{ik} = \frac{1}{n_k} \sum \mathbf{P}_k, \tag{107}$$

where $\mathbf{P}_k$ is the array of obstacle sensor data for the $k^{th}$ cluster, and $n_k$ is the number of points in the $k^{th}$ cluster.

For instance, in Figure 42 three obstacles are individually resolved and three intermediate waypoints, $x_{i1}$, $x_{i2}$, and $x_{i3}$ are identified at the centroid of their respective

sensor data cluster. The three possible approximate trajectories are generated by connecting the current vehicle position, $x_0$, an intermediate point $x_{ik}$, and the target position $x_t$: $\mathbf{x}_0 - \mathbf{x}_{i1} - \mathbf{x}_t$, $\mathbf{x}_0 - \mathbf{x}_{i2} - \mathbf{x}_t$, and $\mathbf{x}_0 - \mathbf{x}_{i3} - \mathbf{x}_t$.

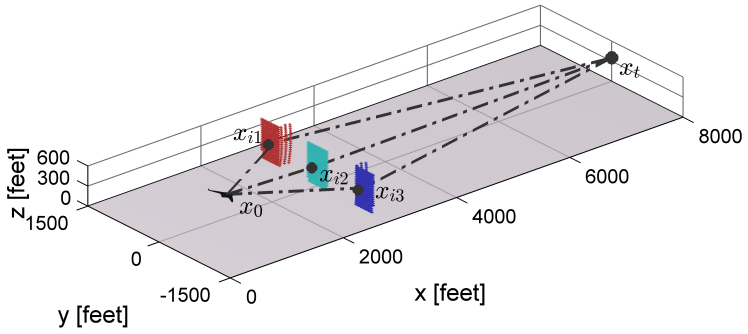



**Figure 42:** Global path optimization

The next step is identifying one potential path among a potential path. The path is selected by individual cluster information and vehicle velocity vector $v = [v_x \ v_y \ v_z]$. To be more specific, we can project each point cluster onto each plane of which normal vector is parallel to $x$-direction ($v_x$) of velocity vector. The details of defining a project plane is described in Figure 44. It then projects the velocity vector onto individual plane. For instance, in the case of Figure 43, we projects the velocity vector onto three planes (plane 1, plane 2, and plane 3). The projected velocity vector judges a potential threat. In other words, if the projected vector on a plane is inside of the plane boundary, that cluster is identifies as a potential threat. In the example, the cluster on the plane 2 is potential threat. In some cases, it does not have any potential threat. This case solves one phase optimal trajectory problem in the local trajectory optimization.

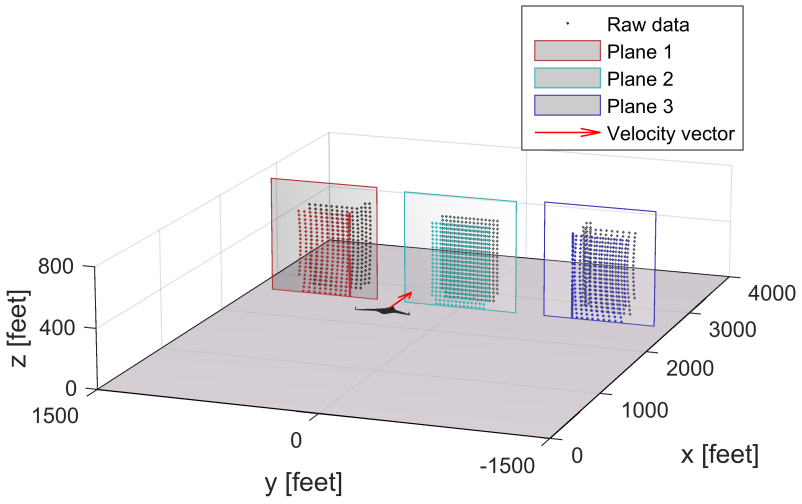



**Figure 43:** Identification of a potential threat

### 4.1.4 Local trajectory optimization

#### *4.1.4.1 General multi-phase optimal trajectory problem*

The purpose of the local trajectory optimization is to avoid the obstacle resolved with cluster $\tilde{C}$ and then reach the target position $\mathbf{x}_t$. We recognize that obstacle avoidance and recovery towards a prescribed target encompass the two fundamental phases of collision avoidance, and that a multi-phase approach to trajectory optimization is therefore well suited for this problem.

In general the multi-phase problem divides the trajectory into $n$ phases or segments ($p \in 0, 1, 2, ..., n$) and sequentially solves for the optimal trajectory in each phase $p$. The formulation is predicated on the definition of a performance index or cost function, dynamic constraints, path constraints, event constraints, and link constraints. The cost function can be written as

$$min \;\; J(\bar{\mathbf{x}}^{(p)}, \mathbf{u}^{(p)}, t) = \sum_{p=1}^{n} \Phi^{(p)}(\bar{\mathbf{x}}_f^{(p)}, t_f^{(p)}) + \sum_{p=1}^{n} \int_{t_0^{(p)}}^{t_f^{(p)}} L^{(p)}(\bar{\mathbf{x}}^{(p)}, \mathbf{u}^{(p)}, t) dt, \qquad (108)$$

where the superscript $^{p}$ denotes the $p^{th}$ phase, $\bar{\mathbf{x}}$ is a state vector, $\mathbf{u}$ is a control input vector, $t$ is time, $t_0$ is initial time, and $t_f$ is terminal time. $\Phi$ and $L$ are terminal and transient costs, also called Mayer and Lagrange costs in the optimal control theory context. Inclusion of transient and terminal costs represents the most general formulation of the cost function. However, for subclasses of optimal trajectory problems only transient or terminal costs will suffice.

Vehicle state equations are captured as dynamic constraints, typically expressed in the form:

$$\frac{d\bar{\mathbf{x}}^{(p)}}{dt} = f^{(p)}(\bar{\mathbf{x}}^{(p)}, \mathbf{u}^{(p)}, t). \tag{109}$$

Path constraints are algebraic inequalities that capture vehicle flight performance limitations such as velocity, acceleration, and thrust limits. These constraints can be represented as follows

$$\mathbf{c}_{min}^{(p)} \leq \mathbf{c}^{(p)}(\bar{\mathbf{x}}^{(p)}, \mathbf{u}^{(p)}, t) \leq \mathbf{c}_{max}^{(p)}. \tag{110}$$

Event constraints establish the conditions that must be satisfied at the final time $t_f$ of each phase $p$, and are generally written in the form

$$\mathbf{E}_{min}^{(p)} \leq \mathbf{E}^{(p)}(\bar{\mathbf{x}}^{(p)}, \mathbf{u}^{(p)}, t) \leq \mathbf{E}_{max}^{(p)}. \tag{111}$$

Phase link constraints ensure continuous state transition between phases. For the link $s$ between phases $p-1$ and $p$, the constraints are expressed as:

$$\chi^{(s)}(\bar{\mathbf{x}}_f^{(p-1)}, \bar{\mathbf{x}}_0^{(p)}) = \bar{\mathbf{x}}^{(p-1)}(t_f) - \bar{\mathbf{x}}^{(p)}(t_0) = 0, \quad (p = s+1) \tag{112}$$

#### *4.1.4.2 Two-phase trajectory optimization framework*

To formulate the local-trajectory optimization we suggest a two-phase approach based on the multi-phase optimal trajectory framework. The first phase ($p = 1$) is an avoidance phase and the second phase ($p = 2$) is a recovery phase. The cost function and constraints are defined accordingly.

Trajectory solutions can be time-optimal, effort- or energy-optimal, or hybrid by combining the two. Other performance index formulations linked to the vehicle dynamics are certainly possible, although time and energy are very commonly used as the basis. The time-based approach minimizes trajectory duration, say in a hostile operating environment, and has been employed in a number of studies [140][105][80]. Minimal-time trajectories however impose high energy requirements associated with aggressive maneuvers and accelerations, which in turn strain energy management requirements and can degrade mission feasibility. On the other hand a minimal-energy solution emphasizes mission energy management and efficiency, and is not driven by time or duration limitations. For this study we adopt an energy-based performance index to assess effort-efficiency improvements attainable with the proposed collision avoidance concept. The performance index is defined as:

$$J = \frac{1}{2}\sum_{p=1}^{n=2}\int_{t_0^{(p)}}^{t_f^{(p)}} \mathbf{u}^{(p)T}\mathbf{W}^{(p)}\mathbf{u}^{(p)}dt, \tag{113}$$

where phase $p$ (here limited to $p = 1, 2$) is denoted by the superscript, $t_0$ and $t_f$ are initial and final times respectively, $\mathbf{u}^{(p)} \in \mathbb{R}^3$ is the acceleration command input vector $[u_{cx}^{(p)} \quad u_{cy}^{(p)} \quad u_{cz}^{(p)}]^T$, and $\mathbf{W}^{(p)}$ is a $3 \times 3$ weighting matrix presumed to be the same in both phases. The $1/2$ factor is inherent in the transient cost function $L$ and is simply placed outside the summation in Eq 113 above. The cost function here adopted is based on the notion that work expended follows the commanded acceleration, squared to ensure positive quantities, and has been used frequently in prior optimal trajectory problems (see for instance Ref. [15]). In the present formulation the effort associated with the trajectory is entirely captured in the transient cost expression, and a terminal cost component is not necessary.

In general constraint complexity has a significant effect on the runtime performance of the online obstacle avoidance algorithm, so the choice of constraints and their inherent sophistication must be considered carefully. This is particularly relevant

for dynamic constraints where simplified flight dynamics can be adopted whenever possible instead of complex non-linear alternatives. Moon and Kang [105][80] propose first-order acceleration dynamic equations for simplified kinematics of an optimal trajectory problem. We employ this first-order approximation for dynamic constraints to achieve a computationally and dynamically feasible solution. This choice for low order dynamics constraints is a practical one, driven by the present focus to demonstrate the proposed two-layer approach in relation to the classical one-layer alternative. An implementation of our method with higher order dynamics is immediately feasible, but increased runtime should be expected as a result.

The simplified kinematic constraints are as follows:

$$\begin{bmatrix} \dot{\mathbf{x}}^{(p)} \\ \dot{\mathbf{v}}^{(p)} \\ \dot{\mathbf{a}}^{(p)} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{0}}_{3\times3} & \tilde{\mathbf{I}}_{3\times3} & \tilde{\mathbf{0}}_{3\times3} \\ \tilde{\mathbf{0}}_{6\times6} & & \tilde{\mathbf{I}}_{3\times3} \\ & & \tilde{\mathbf{A}} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(p)} \\ \mathbf{v}^{(p)} \\ \mathbf{a}^{(p)} \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{0}}_{6\times3} \\ \tilde{\mathbf{B}} \end{bmatrix} \mathbf{u}^{(p)}, \tag{114}$$

where $\mathbf{x}^{(p)}$ is the position vector $[x^{(p)} y^{(p)} z^{(p)}]^T$, $\mathbf{v}^{(p)}$ is the velocity vector $[u^{(p)} v^{(p)} w^{(p)}]^T$, $\mathbf{a}$ is the acceleration vector $[a_x^{(p)} \quad a_y^{(p)} \quad a_z^{(p)}]^T$, and $\mathbf{u}^{(p)}$ is the acceleration command $[u_{cx}^{(p)} \quad u_{cy}^{(p)} \quad u_{cz}^{(p)}]^T$. These states are expressed in a navigation coordinate system. $\tilde{\mathbf{I}}_{3\times3}$ is a $3 \times 3$ identity matrix, and $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{0}}_{n\times m}$ are defined as

$$\begin{aligned} \tilde{\mathbf{A}} &= \begin{bmatrix} -\frac{1}{\tau_x} & 0 & 0 \\ 0 & -\frac{1}{\tau_y} & 0 \\ 0 & 0 & -\frac{1}{\tau_z} \end{bmatrix}, \\ \tilde{\mathbf{B}} &= \begin{bmatrix} \frac{\kappa_x}{\tau_x} & 0 & 0 \\ 0 & \frac{\kappa_y}{\tau_y} & 0 \\ 0 & 0 & \frac{\kappa_z}{\tau_z} \end{bmatrix}, \\ \tilde{\mathbf{0}}_{n\times n} &= \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}. \end{aligned} \tag{115}$$

$\tau_x$, $\tau_y$, and $\tau_z$ are time constants with respect to the three acceleration state variables $[a_x^{(p)}\ a_y^{(p)}\ a_z^{(p)}]^T$. $\kappa_x$, $\kappa_y$, and $\kappa_z$ are the gains of the first order approximation. These parameters are identified based on the least square technique from an aircraft dynamic model.

Event constraints in the first phase prescribe the terminal vehicle states that guarantee obstacle avoidance, and are therefore fundamental for the entire collision avoidance trajectory problem. In our approach prescription of this terminal state is predicated on two conditions. First, the aircraft is in unaccelerated level flight. This flight condition has been commonly used in the past, for instance by Moon and Kang [105]. It offers a simple and tractable state that is justifiable in most cases when the target point after successful collision avoidance prescribes an altitude and velocity comparable to that of the initial condition. We adopt this condition but note that it is simply suggested here, and that other conditions may also be considered. Second, to enforce a safety distance from any avoided obstacle we introduce the variable ($r_s$). Three linear inequality constraints are defined at a distance $r_s$ around the obstacle cluster $\tilde{C}$, one to the left, one to the right, and one above, as shown in Fig 44.

Accordingly, we define the terminal state as

$$\begin{aligned}
&\mathbf{x}^{(1)}(t_f) = [x^{(1)}(t_f) \quad free \quad free]^T, \quad (x^{(1)}(t_f) = min\, \tilde{C}_x - r_s) \\
&\mathbf{v}^{(1)}(t_f) = [u^{(1)}(t_f) \quad 0 \quad 0]^T \qquad\qquad , \\
&\mathbf{a}^{(1)}(t_f) = [0 \quad 0 \quad 0]^T,
\end{aligned} \tag{116}$$

where $\tilde{C}_x$ is $x$ position information of $\tilde{C}$. Note that the safe position along the $x$ axis $x^{(1)}(t_f)$ is prescribed according to $r_s$ too. The $y$ and $z$ positions are free variables that are only restricted by the following inequality constraints, pursuant of the safe distance from the obstacle for the avoidance trajectory:

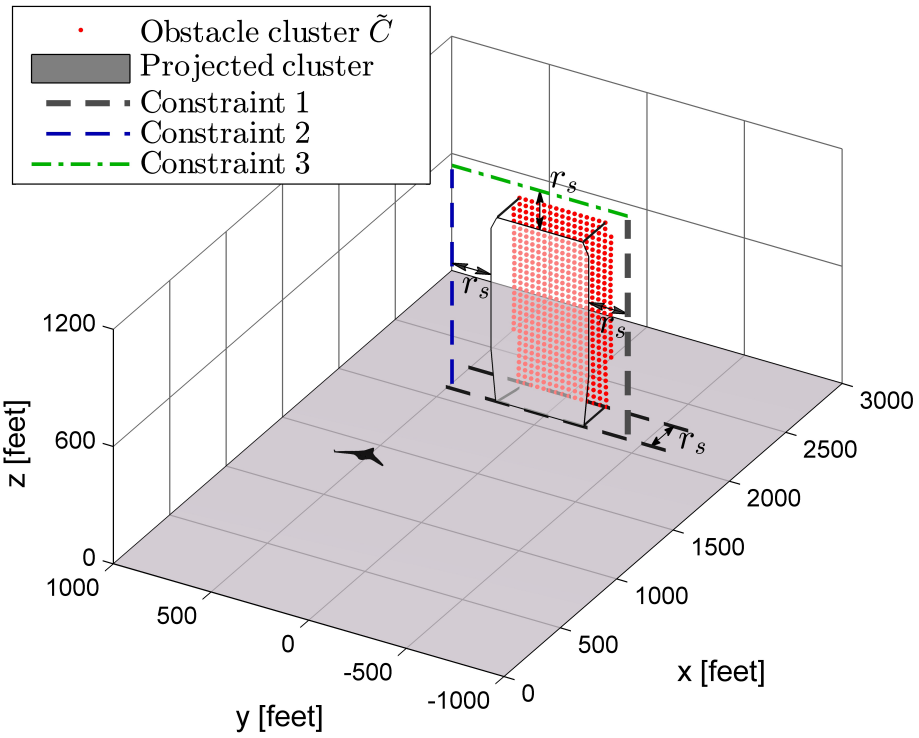



**Figure 44:** Notional depiction of the three obstacle avoidance trajectory event constraints at a safety distance $r_s$ around the obstacle cluster $\tilde{C}$

$$\begin{cases} y \leq \tilde{C}_{ymin} + Mt_1 \\ y \geq \tilde{C}_{ymax} - Mt_2 \\ z \geq \tilde{C}_{zmax} - Mt_3 \end{cases} \tag{117}$$

$$\mathbf{t} = [t_1 \quad t_2 \quad t_3], t_i \in [1, 0]$$

$$\text{M is a large number}$$

$\tilde{C}_{ymin}$ and $\tilde{C}_{ymax}$ are minimum and maximum $y$ position, and $\tilde{C}_{zmax}$ is the maximum $z$ position, in the point cloud $\tilde{C}$. The binary vector $\mathbf{t}$ identifies which of the three obstacle avoidance constraints is active with vector element $t_i = 0$, and $t_{j \neq i} = 1$ otherwise. Identification of the active constraint, and therefore definition of the binary vector $\mathbf{t}$, is solved for as follows:

$$
\begin{aligned}
&D = [\left|(min\ \tilde{C}_y - r_s) - y_p\right|,\ \left|(max\ \tilde{C}_y + r_s) - y_p\right|,\ \left|(max\ \tilde{C}_z + r_s) - z_p\right|] \\
&\mathbf{t} = [0\ 1\ 1], \quad if\ \ min\ D = \left|(min\ \tilde{C}_y - r_s) - y_p\right| \\
&\mathbf{t} = [1\ 0\ 1], \quad if\ \ min\ D = \left|(max\ \tilde{C}_y + r_s) - y_p\right| \\
&\mathbf{t} = [1\ 1\ 0], \quad if\ \ min\ D = \left|(max\ \tilde{C}_z + r_s) - z_p\right|.
\end{aligned}
\tag{118}
$$

Here, $y_p$ and $z_p$ are the vehicle position in $y$ and $z$ projected with the current velocity vector $\mathbf{v}_0^{(1)}(t)$ onto a plane perpendicular to the velocity vector at a downrange distance $x^{(1)}(t_f)$.

Event constraints for the second phase dictate the terminal state at the target position, for which we recommend unaccelerated level flight, as follows

$$
\begin{aligned}
&t_f^{(2)} = free \\
&\mathbf{x}^{(2)}(t_f) = \left[x^{(2)}(t_f)\ \ y^{(2)}(t_f)\ \ z^{(2)}(t_f)\right]^T \\
&\mathbf{v}^{(2)}(t_f) = [v_x^{(2)}(t_f)\ \ 0\ \ 0] \\
&\mathbf{a}^{(2)}(t_f) = [0\ \ 0\ \ 0]
\end{aligned}
\tag{119}
$$

Path and link constraints are defined equally for the avoidance and recovery phases. Path constraints curtail vehicle dynamic performance and include limits on velocity and acceleration, keeping the vehicle within the flight envelope to prevent unfeasible maneuvers. Path constraints are defined as follows

$$
\begin{aligned}
&0 \leq u \leq u_{max}, \\
&v_{min} \leq v \leq v_{max}, \\
&w_{min} \leq w \leq w_{max}, \\
&V_{min} \leq \sqrt{u^2 + v^2 + w^2} \leq V_{max}.
\end{aligned}
\tag{120}
$$

$u_{min}$, $v_{min}$, and $w_{min}$ are the low speed limits, and $u_{max}$, $v_{max}$, and $w_{max}$ are the high

speed limits. The acceleration constraints are stated as simple load factor constraints:

$$\begin{aligned} |a_x| &< g_{xmax} \\ |a_y| &< g_{ymax} \\ |a_z + g| &< g_{zmax} \end{aligned} \tag{121}$$

$g_{xmax}$, $g_{ymax}$, and $g_{zmax}$ are maximum allowable accelerations. We also adopt a minimum altitude constraint as a path constraint to guarantee flight above some operational minimum to prevent possible collision with other ground assets.

$$z \geq z_{min} \tag{122}$$

The last constraint is the linkage constraint at the phase transition point to enforce state continuity across phases:

$$\bar{\mathbf{x}}^{(p-1)}(t_f) = \bar{\mathbf{x}}^{(p)}(t_0) \tag{123}$$

Since the proposed framework of the optimal collision avoidance problem has two phases, in the phase linkage constraint (Equation 123) $p$ is two. The formulated framework of the multi-phase optimal trajectory problem is solved by the Gauss Pseudospectral method (GPM) provided by the open-source software GPOPS. This Gauss Pseudospectral technique was developed by Benson [19][20], and advanced and validated by empirical cases studies from Huntinton et al [62][61][63]. This GPM technique employs an orthogonal collocation method based on the Legendre-Gauss points. The GPOPS software provides MATLAB interface with non-linear programming problem solver SNOPT [54][55].

## *4.2 Numerical simulation*

### 4.2.1 Simulation of unmanned aircraft dynamics, controller, and sensor

For the numerical simulation we assume a small electric fixed-wing unmanned aircraft, and adopt the Aerosonde specification taken directly from the literature [154] [18].

Table 22 summarizes airframe parameters, along with those assumed for the airborne sensor and collision avoidance scheme. For simplicity the vehicle equations of motion assume a point mass model as reported in previous work [94][102][30]. For the aircraft tracking controller, we employ standard feedback control structure. The on-board sensor is modeled as a generic light detection and ranging instrument, or lidar, with no instrument uncertainties.

**Table 18:** Description of UAV parameters

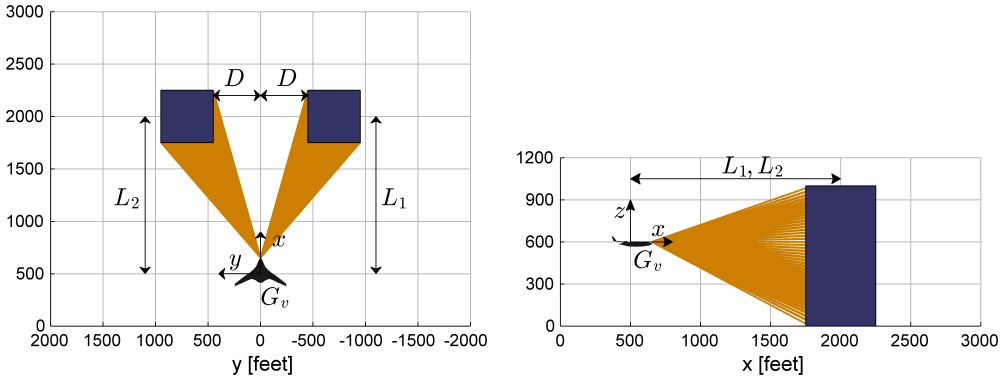
| | UAV parameters | Variable | Value | Unit |
|---|---|---|---|---|
| Vehicle parameter | Weight | $w$ | 29.76 | $[lb]$ |
| | Planform area | $S$ | 6.1 | $[ft^2]$ |
| | Area swept out by the propeller | $S_p$ | 0.1348 | $[ft^2]$ |
| | Propeller aerodynamic coefficient | $C_p$ | 1 | |
| | Efficiency constant of the motor | $K_m$ | 8 | |
| | Lift coefficient at zero angle of attack | $C_{L0}$ | 0.28 | |
| | Lift curve slope | $C_{L\alpha}$ | 3.45 | |
| | Aspect ratio | $AR$ | 10.7 | |
| | Span efficiency | $e$ | 0.9 | |
| | Zero-lift drag coefficient | $C_{D0}$ | 0.03 | |
| Sensor parameter | Distance range | $S_r$ | 2000 | $[ft]$ |
| | Azimuth range | $AZ$ | 60 | $[deg]$ |
| | Elevation range | $EL$ | 45 | $[deg]$ |
| | Sensor resolution | $\theta_{sen}$ | 2 | $[deg]$ |
| Collision avoidance parameter | Minimum distance constraint between obstacles | $D_o$ | 150 | $[ft]$ |
| | Minimum separation distance between an obstacle and a vehicle | $r_s$ | 70 | $[ft]$ |
| | Minimum altitude | $z_{min}$ | 200 | $[ft]$ |
| | Maximum acceleration | $g_{max}$ | 3 | $[ft/s^2]$ |
| Other parameters | Update rate of guidance and navigation loop | $f_g$ | 1 | $[Hz]$ |
| | Update rate of aircraft control loop | $f_a$ | 10 | $[Hz]$ |

### 4.2.2 Comparative analysis of clustering algorithms for obstacle resolution

We compare the DBSCAN and spectral clustering for the present application in terms of prediction success rate and computational time. These figures of merit are evaluated over a design of experiments based on the setup illustrated in Figure 45 where two identical, generic, cuboid obstacles lay ahead of the aircraft flight path. The $x$, $y$, and $z$ axes shown are the vehicle coordinate system $G_V$. The sensor data collection is assumed instantaneous, and the clustering algorithms executed at that instant only so that there is no trajectory optimization or flight simulation, only clustering evaluation at the instant depicted. The distance $2D$ between obstacles, referenced to the origin of the $y$ axis, is sampled as follows: $D = [20 \;\; 40 \;\; 60 \;\; 80 \;\; 100]$.

Similarly, the down-range distance between the unmanned aircraft and each of the two obstacles is varied with the $x$ axis location of the obstacle centroid as follows: $L1, L2 = [1100 \quad 1200 \quad 1300 \quad 1400 \quad 1500]$. We combine the three independent factor variations into a full factorial design of experiment with 125 samples. The UAV sensor parameters and the minimum separation distance between obstacles $D_o$ are described in Table 22. The parameter $D_o$ is mapped to the characteristic distance in the clustering algorithm of choice, so that obstacles are resolved as separate objects only if the distance between them is greater than $D_o$. Note that to allow safe flight between resolved obstacles the distance between them must be at least twice the minimum separation distance between aircraft and any obstacle $r_s$.

$$D_o \geq 2r_s \tag{124}$$




(a) Top view (b) Side view

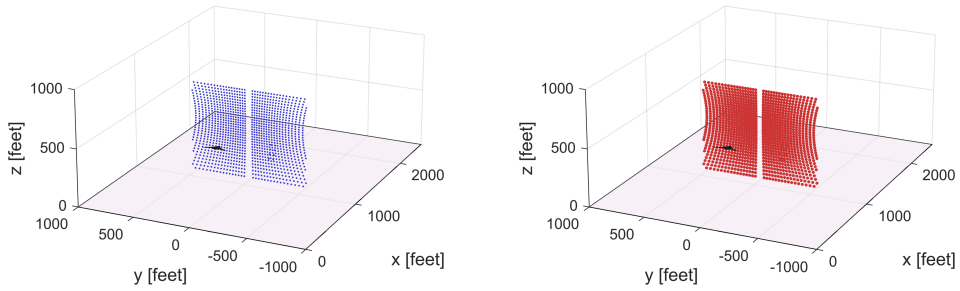
**Figure 45:** Simulated experiment setup for comparative analysis of clustering algorithms. (Aircraft not shown to scale)

This experiment was conducted in a commercially available desktop computer with a 3.40GHz Intel(R) Core(TM) i7-2600 processor and a 8GB RAM. Results of the clustering experiments are summarized in Table 19 and show that the both algorithms accurately resolve clusters with a 100% success rate. That is, when the gap between obstacles is larger than $D_o$ (i.e., $2D \geq D_o$), both algorithms exactly resolve two clusters regardless of variation of the distance $L_1$ and $L_2$.

In terms of computational efficiency, DBSCAN is faster and features lower variance across experiments. We attribute greater runtime mean and variance of spectral clustering in part to the use of $k$-means with random initialization for initial center points. Figures 46 to 49 illustrate four representative cases in the evaluation of clustering algorithms. Based on the results obtained we opt for DBSCAN as the preferred clustering algorithm in the global-path optimization.

**Table 19:** Comparison assessment results of the clustering algorithms

| | DBSCAN | | Spectral method | |
|---|---|---|---|---|
| | Prediction success rate (%) | Computational time (sec) | Prediction success rate (%) | Computational time (sec) |
| Average | 100 | 0.1120 | 100 | 0.3412 |
| Standard deviation | 0 | 0.0268 | 0 | 0.1920 |



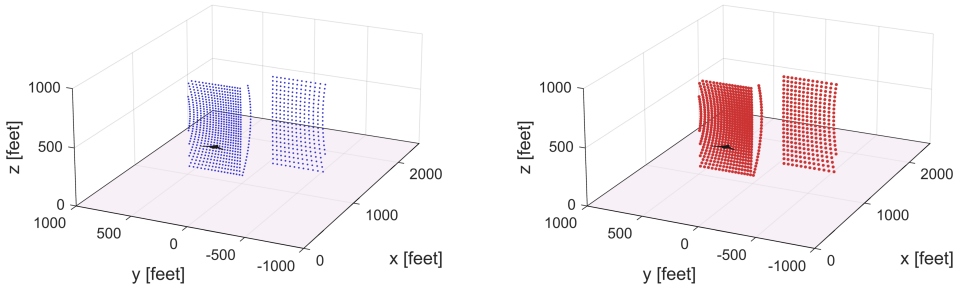

(a) Raw data

(b) Clustering results

**Figure 46:** Clustering results ($D = 20ft$, $L_1 = 1100ft$, $L_2 = 1100ft$)

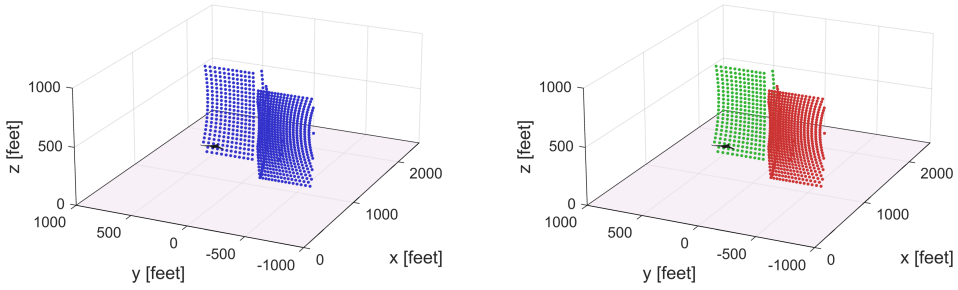
### 4.2.3 Assessment of two-layer collision avoidance with numerical simulation

To test the hypothesis that our two-layer collision avoidance algorithm yields more energy efficient trajectories without incurring in prohibitive computational burden relative to the single-layer approach we conduct an experiment with numerical simulations. The experiment is comprised of three use cases designed to be relevant and representative for UAS applications of interest, and to capture critical arrangements of relative obstacle placement in terms of on-board sensing and optimal avoidance



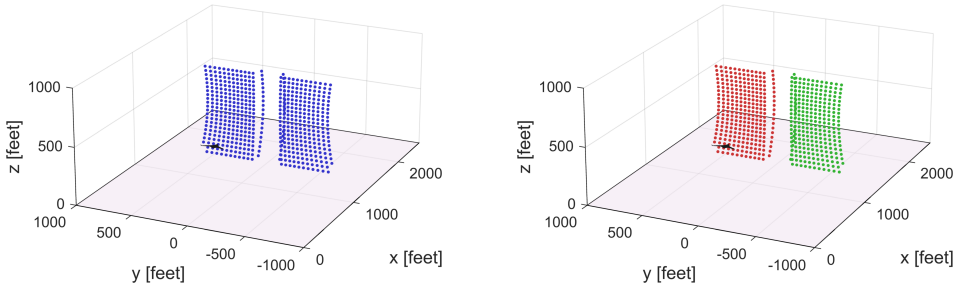

(a) Raw data (b) Clustering results

**Figure 47:** Clustering results ($D = 40ft$, $L_1 = 1100ft$, $L_2 = 1500ft$)




(a) Raw data (b) Clustering results

**Figure 48:** Clustering results ($D = 100ft$, $L_1 = 1500ft$, $L_2 = 1100ft$)




(a) Raw data (b) Clustering results

**Figure 49:** Clustering results ($D = 100ft$, $L_1 = 1500ft$, $L_2 = 1500ft$)

trajectory execution. Simulations for the three use cases are ran with both methods using the same initial conditions, set to steady level flight at 600 feet altitude with velocity 100 feet/sec.

The first use case is illustrated in Figure 50 with the trajectory simulation results of the two collision avoidance methods. The obstacle-free trajectory between the initial and final points is shown in red, and the executed collision avoidance trajectory with each method is shown in dotted blue. The first scenario features two obstacles of comparable altitude (∼1,000 ft) and footprint, located at the same downrange distance (∼4,000 ft), and separated by a distance (∼250 ft) greater than twice the safety distance defined for aircraft-to-obstacle separation. This initial scenario is intended to characterize the benefits of safe flight between adjacent obstacles. Results show that with the one-layer obstacle avoidance algorithm the aircraft maneuvers left around the left obstacle, since that route is more energy efficient than flying over the two obstacles or around the right obstacle. As expected, in this simulation the aircraft treats both obstacles as a single object and executes the most energy-efficient trajectory to avoid it and meet the final point. In contrast, the result of two-layer obstacle avoidance algorithm shows that the aircraft flies a trajectory through the gap between the two obstacles, with only a minor course correction too small to be appreciated in the figure. Again, the result is reasonably expected as the gap between the two obstacles exceeds the minimum separation, allowing the global optimizer to resolve them as two separate objects, and for the local optimizer to produce an energy-optimal avoidance trajectory between the two.

The second use case, depicted in Figure 51, builds upon the first one and is designed to test cases when there is a third obstacle immediately behind the gap between the first two obstacles. As before the one-layer algorithm avoids the first two obstacles by maneuvering to the left of the left obstacle, and then keeping to the left of the third obstacle in the back. The two-layer algorithm executes an avoidance

trajectory between the first two obstacles, and then around the right of the third obstacle behind them.

The third use case, depicted in Figure 52, reverses the obstacle arrangement, presenting first a single obstacle whose footprint and orientation are intentionally chosen to significantly block the two obstacles behind it. This scenario is intended to test the case where information about downrange obstacles cannot be acquired until a closer obstacle is cleared. Moreover, it tests the feasibility and efficacy of the proposed method with regards to our choice to cluster sensor data in a $2-D$ domain, that is, collapsed to the $xy$ plane, so that obstacles are not resolved based on depth data. Results shows that the two obstacle avoidance algorithms have the same initial obstacle avoidance trajectory to avoid the first obstacle. After avoiding the first obstacle the aircraft detects two downstream obstacles and from that point on the two algorithms produce different avoidance trajectory solutions. In the one-layer obstacle avoidance trajectory the aircraft flies above the two obstacles because it computes the energy-efficient avoidance trajectory resolving the two obstacles as a single object. The two-layer algorithm resolves two separate obstacles and executes an energy optimal trajectory through the gap between them.

We quantitatively assess performance of both algorithms on the three scenarios by estimating the work done by the aircraft, or conversely the energy required to execute each trajectory. We adopt the classical energy formulation for flight performance where the work performed by the aircraft, i.e. the energy expended, is estimated as thrust (force) applied over distance, where it is assumed that the thrust and velocity vectors are aligned so that thrust is always on the direction of motion. In this formulation there is no work associated with lift as it is always perpendicular to the trajectory. The work associated with drag is always an energy loss. Weight work corresponds to the change in potential energy. The work associated with the imbalance between trust, drag, and the component of weight aligned to them, resulitng in

longitudinal accelerations, corresponds to the change in kinetic energy. The latter is often adjusted via thrust input. Accordingly,

$$\begin{aligned} T &= D + W sin\gamma + F \\ T ds &= D ds + W ds sin\gamma + F ds \\ W_T &= W_D + W_W + W_F \\ W_T &= W_D + \Delta PE + \Delta KE \end{aligned} \tag{125}$$

Here $\gamma$ is the flight path angle, $F$ is the thrust-controlled force imbalance between thrust, drag, and the weight component aligned to the flight path, $ds$ is an infinitesimal segment along the flight trajectory, and $W_T$ denotes the work associated with force $T$ as is also the case for forces $D$, $W$, and $F$. Note that for cases where $\gamma = 0$ the work associated with weight is zero, and there is no change in potential energy $\Delta PE = 0$. Similarly, for the case of null imbalance force $F = 0$, the work associated with $F$ is zero and there is no change in kinetic energy $\Delta KE = 0$.

Recognizing that thrust, and the work it performs over $ds$, equate to changes in kinetic and potential energy while overcoming drag, then the work performed (or energy expended) in executing a trajectory is given by

$$W_T = \int_c T ds, \tag{126}$$

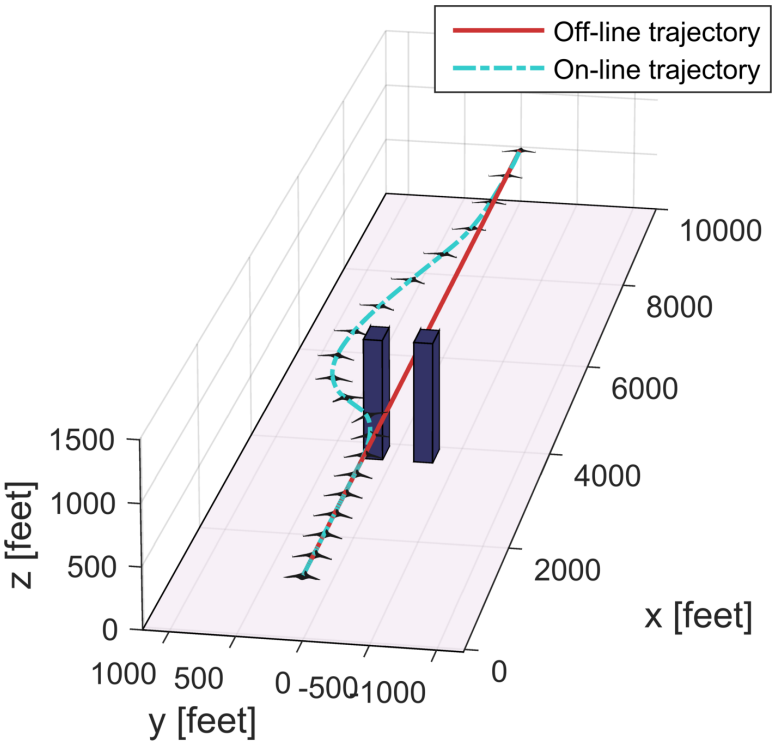
where $T$ is thrust, $c$ is an avoidance trajectory, and $ds$ is the length of a discrete segment along the avoidance trajectory.

Table 20 summarizes the measurement results of each method under the three different scenarios. Results clearly indicate the proposed method requires less energy than the one-layer collision avoidance method. It is worth noting too that both algorithms establish energy-optimal trajectories around resolved obstacles, so that all energy-efficiency benefits observed can only be attributed to the resolution of separate objects from in the proposed two-layer approach.

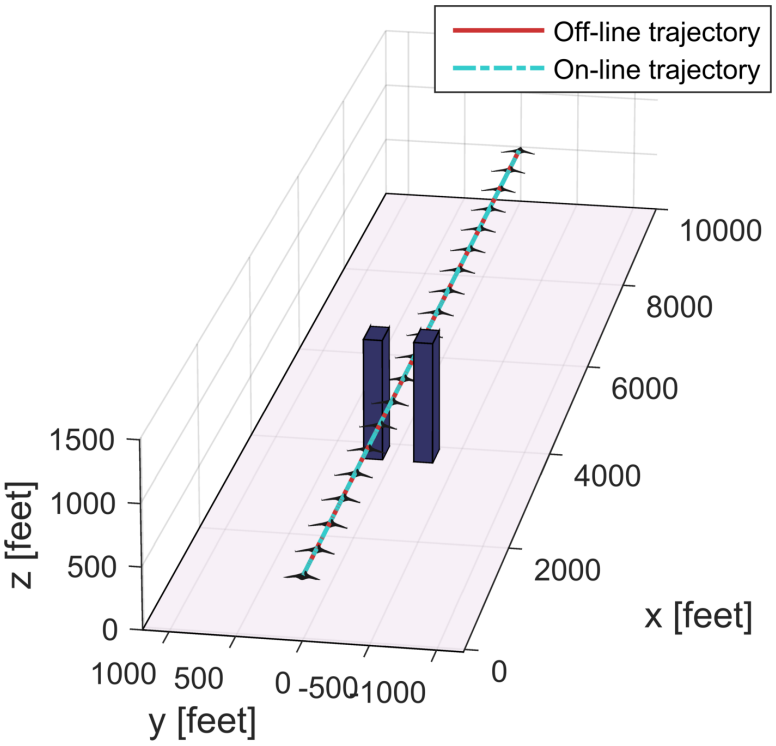
Results clearly show that the proposed two-layer approach generates more energy-efficient obstacle avoidance trajectory. The inclusion of the global trajectory layer results in additional computational expense where the most demanding step is the solution of the clustering problem. Based on our runtime assessments of DBSCAN presented in Table 19 we find that this cost is acceptably low in comparison to the typical runtime of the local trajectory optimization. Additional runtime benefits may be attained via programming improvements or parallel computing techniques.

**Table 20:** Summary of numerical simulation

| Scenario | One-layer structure, W1 (lb ft) $W_1$ | Two-layer structure (lb ft) $W_2$ | Energy difference (lb ft) $W_1 - W_2$ |
|---|---|---|---|
| 1 | 28,830 | 18,695 | 10,135 |
| 2 | 28,830 | 19,075 | 9,755 |
| 3 | 20,888 | 19,763 | 1,125 |

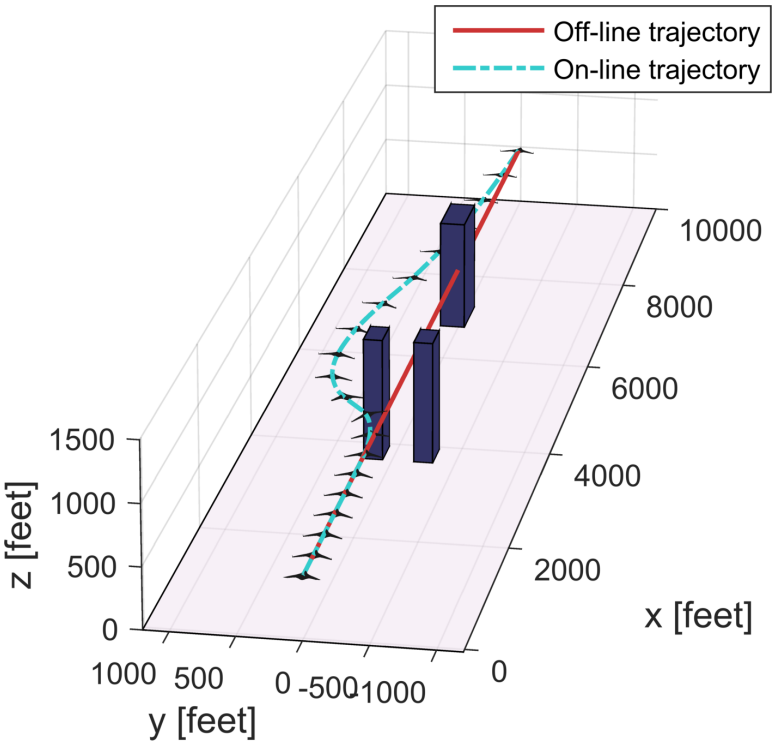



(a) One-layer obstacle avoidance algorithm

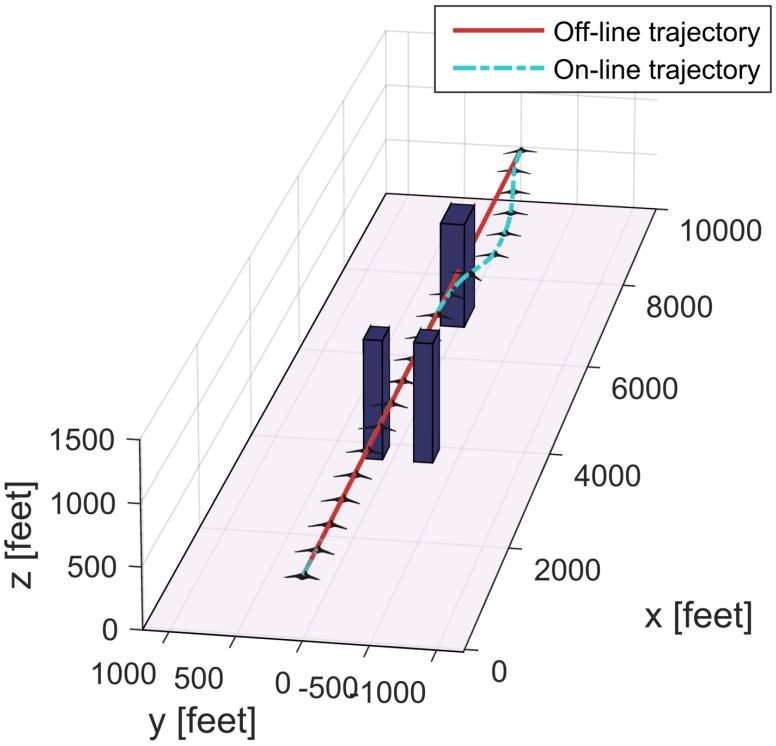



(b) Two-layer obstacle avoidance algorithm

**Figure 50:** Numerical simulation results of obstacle avoidance algorithms in the first scenario

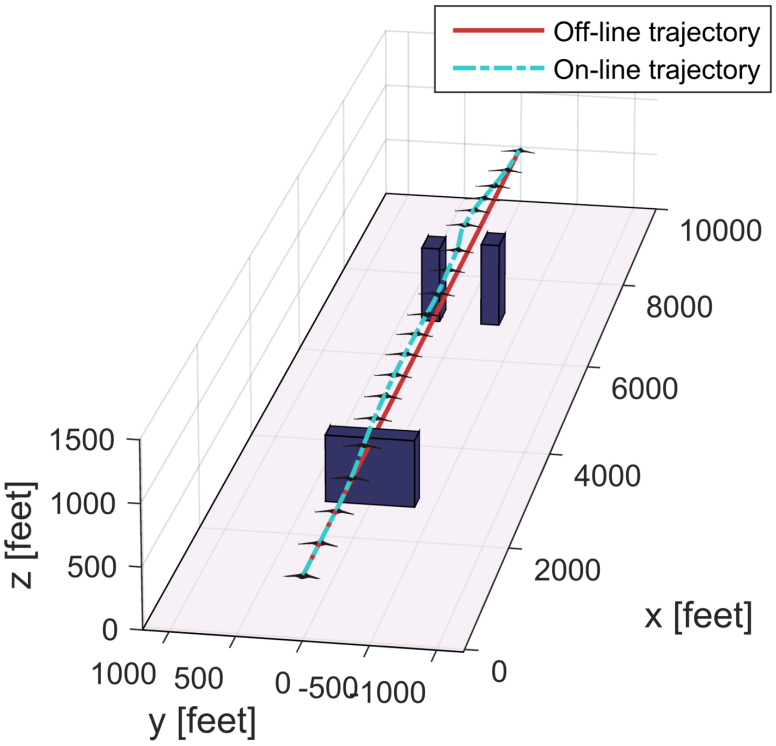



(a) One-layer obstacle avoidance algorithm

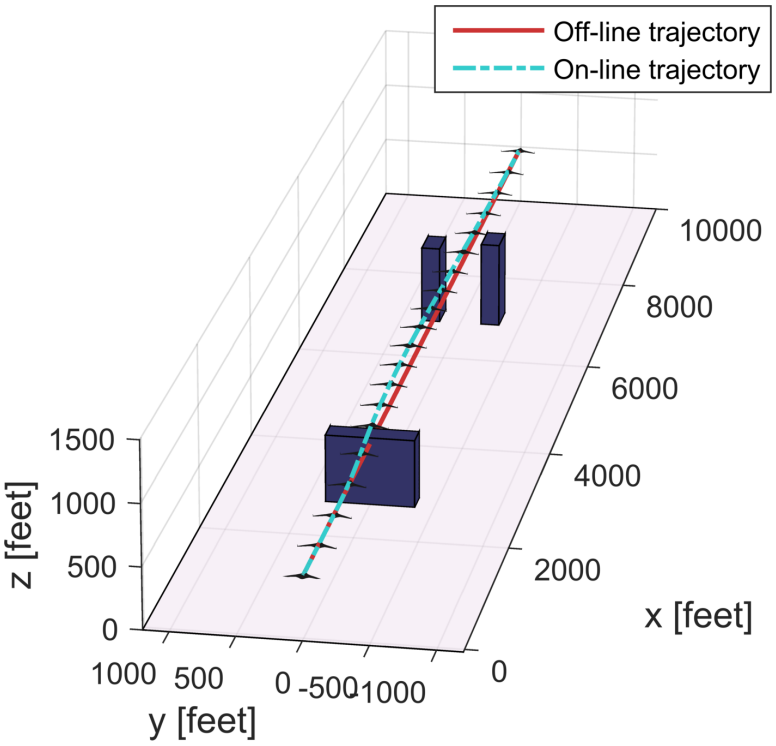



(b) Two-layer obstacle avoidance algorithm

**Figure 51:** Numerical simulation results of obstacle avoidance algorithms in the second scenario




(a) One-layer obstacle avoidance algorithm




(b) Two-layer obstacle avoidance algorithm

**Figure 52:** Numerical simulation results of obstacle avoidance algorithms in the third scenario

## *4.3 Conclusion*

Many existing collision avoidance concepts using on-board sensor data fail to generate efficient trajectories because, even though multiple objects may be detected, the collision avoidance algorithm treats them as a single conglomerate regardless of their number and relative location. Resulting avoidance trajectories are therefore optimal only in the context of a large avoidance region, and fail to exploit free space between obstacles that may allow for more efficient solutions. The approach proposed in this paper features a two-layer multi-phase optimal collision avoidance algorithm that entails global- and local-path optimization. The purpose of the global-path optimization is to resolve multiple obstacles as separate objects, given they satisfy a separation minimum, with sensor data clustering. The global-path optimization also identifies a finite and tractable set of candidate approximate trajectories using the cluster data of resolved objects, and determines the approximate optimal path via the identification of a potential threat based on a velocity vector. We identify two candidate clustering techniques that allow for the parametric specification of cluster separation minimum for obstacle resolution: DBSCAN and spectral clustering. A design of simulation experiments is executed evaluate both techniques in terms of prediction success rate and computational runtime. Although both methods are found to have 100% rates for the simulations performed, DBSCAN featured lower runtime mean and variance, thus emerging as the recommended clustering technique for the proposed concept.

Numerical simulations with status-quo one-layer collision avoidance and the proposed two-layer concept are conducted to compare their performance in terms of energy-efficiency of executed trajectories. Three relevant use cases are utilized to examine multi-obstacle placements of interest, and assess the feasibility of the proposed concept, and test the main hypothesis of this work. Results indicate that trajectory solutions with the two-layer avoidance method provide tangible energy efficiency benefits while incurring in very small additional computational cost.

Collision avoidance with on-board sensors is a relatively new and quickly evolving field where many areas of additional work are warranted. Regarding the work here reported we argue in favor of a broader benchmarking study where a large set of multi-obstacle scenarios is utilized, potentially with hardware-in the loop real-time simulations. Moreover, this two-layer collision avoidance algorithm has a flexible structure. In other words, the local trajectory optimization can be realized with any of a myriad of obstacle avoidance formulations. That is, in the local-trajectory optimization, the collision avoidance algorithm SCAA-1 is solved and demonstrated, but instead of the SCAA-1, the PNORM, SCAA-2, or SCAA3 can be implemented to generate the optimal multi-obstacle avoidance trajectory.

We also believe that extension to a probabilistic study is a natural next step so that uncertainties associated with instrumentation, on-board computer runtime, and flight dynamics and controls governing trajectory execution, are treated explicitly. Ultimately, the work here presented outlines a new concept that offers the potential of significant benefits for the rapidly growing field of UAS applications and the increasing need to develop safety-assuring technologies.

# CHAPTER V

# CONSTRUCTION OF REALISTIC URBAN ENVIRONMENTS

The construction of a realistic three-dimensional urban environment has gained attention from diverse fields such as virtual city tourism, environment monitoring, a rescue mission and a surveillance mission. The realistic urban environment is a critical component to understand the UASNAS integration problem because many operation concepts such package delivery and drone ambulance have to be planned to execute a mission in an urban area. Therefore, to analyze an urban operation problem, a modeling and simulation environment requires a capability of generating a realistic urban environment. The realistic urban model needs to be generated rapidly and automatically since the UASNAS problem requires to explore diverse mission scenarios with different UAS platforms in various urban areas.

In the obstacle avoidance problem in the UASNAS domain, many researchers have built urban models to evaluate their obstacle avoidance algorithms [157][136][153][59][56]. Most existing urban models were created under assumptions that a building is a cuboid, cylinder or simple shape structure. In addition, they failed to describe the relationship between a generated artificial obstacle environment and the realistic urban environment. Stastny et al. have generated a sophisticated urban environment to test their proposed obstacle avoidance algorithm, but they do not explain about the details of a urban modeling process [136].

In the computer science field, considerable researches have been conducted about

building/urban modeling methods. The traditional approach to constructing an urban model is based on aerial images and other data sources. This image- and data-based approach is a computational expensive and time consuming manual process. To overcome the computational issues of the manual process, many urban modeling methodologies have recently relied on airborne LiDAR (Light Detection and Ranging) data that collects high fidelity information with centimeter precision. The drawbacks of LiDAR data include uncertainties from inertial navigation errors, sensor noise and reflection errors of some surfaces. These uncertainties degrade the precision quality of an urban model. On the other hand, LiDAR data can be collected quickly and at low cost. Because the LiDAR information entails point cloud information, a user can easily handle the point information to construct a urban model. Due to these advantages, many urban modeling techniques have been recently developed [75][60][158][159]. You et al.[158] suggested automatic process using LiDAR information. This approach reconstructs LiDAR information through re-sampling, hole filling, and tessellation. Then, the reconstructed information is applied to a classification algorithm to judge either building or bare-land. The classified result is refined by building primitives, and then the refined information is optimized by fitting and filtering techniques. Based on the automatic process suggested by the You et al., Hu et al. [60] suggested an advanced urban modeling method combining airborne LiDAR data and aerial imagery information. This additional imagery information allows precise edge detection and improves computational complexity. Zhou et al. [159] introduce a building modeling method that includes a classification algorithm to distinguish between vegetation area and building area, a roof generation algorithm from boundary detection, and also creates polygon meshes to construct a building model.

These urban modeling methods in the computer science domain may not be applicable to an urban modeling of an obstacle avoidance problem since these methods

include detailed building/vegetarian models as unnecessary information.

In addition, the building model with polygon shape or mesh-type urban environment cannot be implemented on the introduced flight simulation model because the simulation environment assumes that obstacle is a cuboid shape for a simplification of a complex urban environment. Furthermore, our obstacle avoidance scenarios do not consider near-ground operations. In other words, a building model does not require specific ground models such as trees or other objects around ground area. These observations and assumptions lead to the following research question: *How can an urban model be constructed to describe a realistic urban environment for UASNAS obstacle avoidance problem?*

To answer this research question, this paper proposes a rapid and data-driven urban modeling approach using LiDAR data. The proposed approach has five steps: resampling/refining data, classification, principal component analysis, grid generation, and urban modeling. In the following subsection, the details of each step and the results of case studies will be introduced.

## *5.1 Data-driven grid-based urban modeling*

Data-driven urban modeling method using LiDAR information is inherently challenging. In general, LiDAR information has error sources due to the GPS/INS error, inherent signal noise, and others. These error sources degrade the accuracy of an urban model. Moreover, the data-driven urban modeling method to tackle the UASNAS problem requires a rapid and realistic urban model with appropriate fidelity since diverse mission scenarios with various UAS platforms should be explored to fully understand the UASNAS problem. Lastly, in the UAS simulation model developed in this thesis, obstacles were assumed as a cuboid shape to simplify an actual building shape. This assumption enables us to accelerate numerical simulation time that allows the execution of more experiment cases. To satisfy the obstacle assumptions

from the developed simulation environment, the urban model must be constructed by a composite cuboid block.

To satisfy the objectives and the constraints, we propose a rapid, data-driven and grid-based urban modeling method. Figure 53 illustrates the proposed urban modeling approach that entails five steps. The first step includes collecting LiDAR data, resampling and refining original data to decrease the size of LiDAR information. The second step is solving a clustering problem for the identification of individual building components from unlabeled LiDAR data. The third step specifies the identification of the principal directions of each building to define the rotational angle of each classified building in terms of a global coordinate system. The fourth step is grid generation that defines the fidelity of the identified buildings. The last step is urban generation that entails defining width, length, and height of all buildings. The following subsections describe the details of each step.

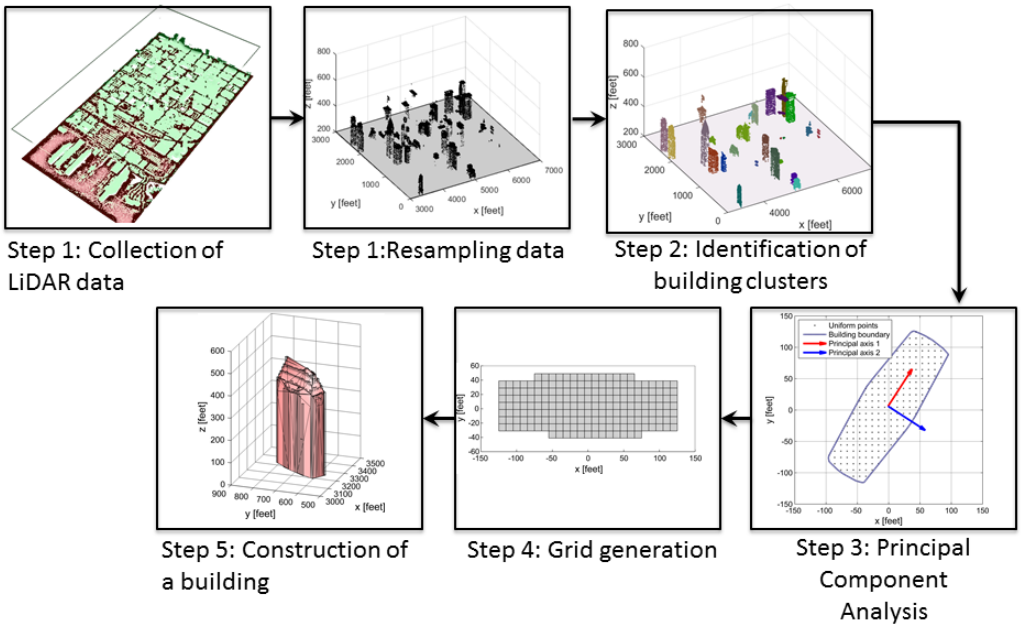



**Figure 53:** A rapid, data-driven and grid-based urban modeling method

### 5.1.1 Collecting/Resampling/refining LiDAR data

For a rapid and data-driven approach, we use LiDAR information collected by airborne LiDAR sensor that includes point cloud information about ground objects. The urban LiDAR data is provided from multiple organizations. Table 21 summarizes the list of LiDAR resource websites [66][115][46][33][119]. The LiDAR data in this paper is mostly from Open Topography and PAMAP LiDAR Elevation Data. The Open Topography [115] is a web-based free open source that has a large database with high-resolution topology data. PAMAP is digital base maps of Pennsylvania, which is managed by the Pennsylvania Department of Conservation and Natural Resources, and Bureau of Topographic and Geologic survey.

**Table 21:** Airborne LiDAR Resource

| Name | Website address |
|---|---|
| United States Interagency Elevation Inventory (USIEI) | https: //coast.noaa.gov/inventory |
| Open topology | http: //www.opentopography.org |
| USGS Earth Explore | http: //earthexplorer.usgs.gov |
| PAMAP LiDAR Elevation Data | http: //www.dcnr.state.pa.us/topogeo/pamap/lidar/index.htm |
| Indiana Spatial Data Portal | http: //gis.iu.edu/datasetInfo/statewide/in_2011.php |

Before collecting the LiDAR data from the introduced digital resources, the area of interest was selected from Google Earth. The raw LiDAR data of the selected region was collected by digital database introduced in Table 21. However, this raw LiDAR data cannot be directly readable; thus, we use an open-source post-processing program called Rapidlasso tool [125]. This software has diverse functions such as converting raw LiDAR data into a variety of formats, filtering LiDAR data, and checking the quality of the LiDAR data. Through using this software, the collected raw data can easily be converted into the readable format of the raw LiDAR data. Figure 54 illustrates the detailed steps of collecting point cloud information.

We select an example area that is downtown San Diego. The size of the selected area is approximately 6000 by 3000 feet (length and breath). The collected LiDAR information has a 2,508,951 point cloud that is not computationally tractable for

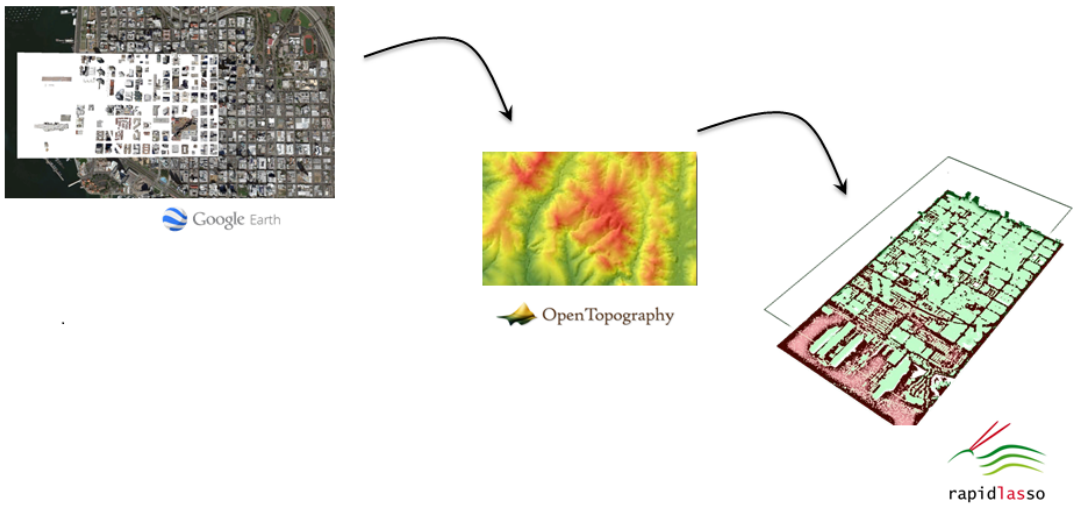



**Figure 54:** Collection of LiDAR data

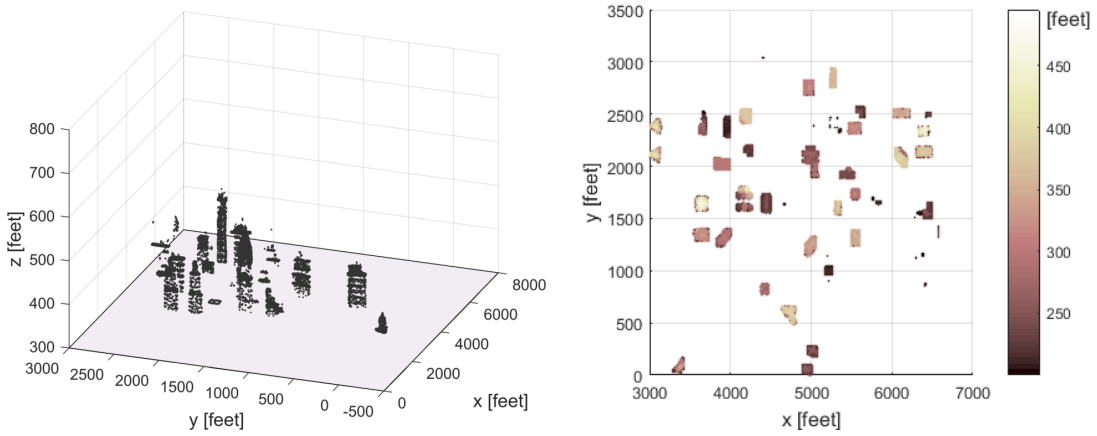
data analysis and visualization purposes. To improve the computational issues, we randomly resample the collected point cloud data. As a result, the resampled data was reduced by 50 percent (1,254,476 points) of the original data.

An altitude constraint is also considered because our UAS mission scenarios do not include near-ground operations. In other words, all the point cloud below the altitude constraint is eliminated. This altitude constraint can minimize noise impact generated by vegetation and complex ground facilities. Therefore, this concept helps individual buildings to be identified from point cloud information. Moreover, this constraint is beneficial for computational efficiency because we can reduce the amount of point cloud information.

In our example model, we assume that the altitude constraint is 200 feet because according to UAS NASA project document, 'Unmanned Aircraft System (UAS) Traffic Management (UTM)', they proposed an UAS operation airspace for low-altitude drones between 200 feet and 500 feet [11]. They defined the airspace above 500 feet that shares with manned aircraft as Integrated airspace.

The result of the resampling technique and the altitude constraint is that the final size of the point cloud shrinks to 67,049 points presented in Figure 55, which is

approximately 2.67 percent of the initial LiDAR information.




(a) Resampled LiDAR point cloud in three-dimensional space

(b) Resampled LiDAR point cloud as viewed from above

**Figure 55:** Result of resampled point cloud in a densed urban example

### 5.1.2 Identification of building clusters

The next step is specifying individual building information from the resampled point cloud that resulted from the previous step. Since the resampled point cloud does not include any labels to indicate different objects, we apply a technique to identify individual objects using a distance or similarity metric. The identification of building clusters can be applied by unsupervised learning clustering algorithms. Clustering algorithms have been actively researched in the commuter science community; thus, many clustering techniques exist, but we skip the further explanation about the overview of clustering techniques because we elaborate diverse types of clustering techniques in the two-layer collision avoidance algorithm section. In this section, the density-based spatial clustering of applications with noise (DBSCAN) method suggested by Ester et al. [45], which computes maximum group defined by density-connected points is employed. The robustness of the DBSCAN technique is controlled by considering the maximum radius of a neighborhood $\epsilon$ and the minimum number of points $p$ in the group to satisfy the maximum radius. The point density approach is

an ideal structure to group adjacent points and can also eliminates noise data. This technique enables us to solve the non-linear clustering problem and provide a robust solution against uncertainties. Therefore, we implement the DBSCAN technique on the clustering problem from resampled LiDAR point cloud shown in Figure 55.

Figure 5.1.2 is the clustering result from the DBSCAN technique. For the DBSCAN technique, the maximum radius of a neighborhood $\epsilon$ and the minimum number of points $p$ were chosen to be 50 feet and 50 points, respectively. These parameters are defined by observing characteristics based on the parameter variation to eliminate noise points and precisely collect buildings. From these parameters, the identified number of the clusters is 26. In other words, the identified number of buildings in the given urban area is 26.

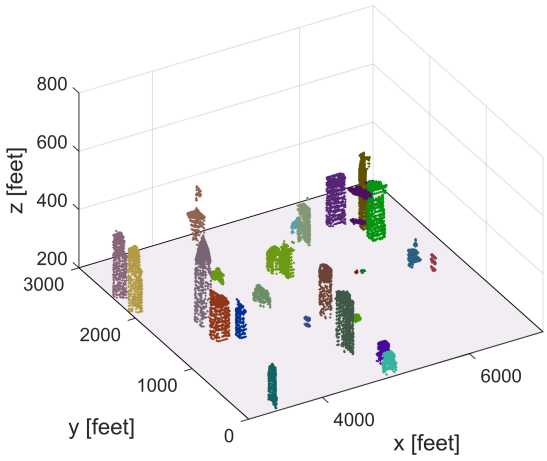



**Figure 56:** Clustering result of the DBSCAN technique

### 5.1.3 Identification of rotational angle and construction of a building

Based on the clustering results ($\mathbf{C} = [C_1,\ C_2,\ \cdots\ , C_k]$), we need to characterize an individual building cluster. In order to capture the building features, we also need to specify the rotation angle of each cluster information in the global coordinate system. The rotational angle can be utilized to more precisely characterize building clusters.

Specifying the rotational angle of each cluster can be defined by Principal Component Analysis (PCA) that provides most sensitive axes from given point cloud information. In the mathematical context, the PCA technique is an orthogonal linear transformation method that changes the original coordinate system into a new orthogonal coordinate system with the highest variance. This PCA technique has been widely implemented in various fields such as pattern recognition, compressing data structure, and reduction of dimensions minimizing the loss of the data information [52] [23] [13]. This paper will briefly discuss an overall concept of the principal component analysis based on the reference book written by Jolliffe [77].

It is assumed that data $D$ are $D = \{\mathbf{x}_1,\ \mathbf{x}_2,\ \cdots,\ \mathbf{x}_m\}$. $\mathbf{x}_i$ indicates $i$th observation in $n$ dimensional space $\mathbf{x} \in \mathbb{R}^{n \times m}$. The matrix formulation of the data can be written by

$$D = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{127}$$

From the given data, we can compute the mean and covariance according to the following equations:

$$\mu = E[\mathbf{x}] \tag{128}$$

$$\Sigma = E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T] \tag{129}$$

The covariance matrix can be represented by a linear transformation equation

$$\Sigma = \mathbf{A}\Lambda\mathbf{A}^T, \tag{130}$$

where $\mathbf{A}$ is the orthogonal linear transformation matrix that entails eigenvectors, and $\Lambda$ that is diagonal matrix includes eigenvalues $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$. The covariance $\Sigma$

can also be written, $\Sigma = \lambda_1 a_1^T a + \lambda_2 a_2^T a_2 + \cdots + \lambda_n a_n^T a_n$. Reducing the dimension of the given data can be defined by introducing a transformation in a latent space. We select a new transformation matrix $\mathbf{A}_k = \{a_1, a_2, \ \cdots \ , a_k\} \in \mathbb{R}^{n \times k}$. The $k$ eigenvectors, which is $k \leq m$, are specified from the first $k$ largest eigenvalues, $\lambda_1 \geq \lambda_2 \geq \ \cdots \ \geq 0$. The physical meaning of this process is that we select $k$ dimensional latent space with $k$ large variance. The covariance matrix can be represented by a linear transformation equation

$$\nu = A_k^T \mathbf{X}, \tag{131}$$

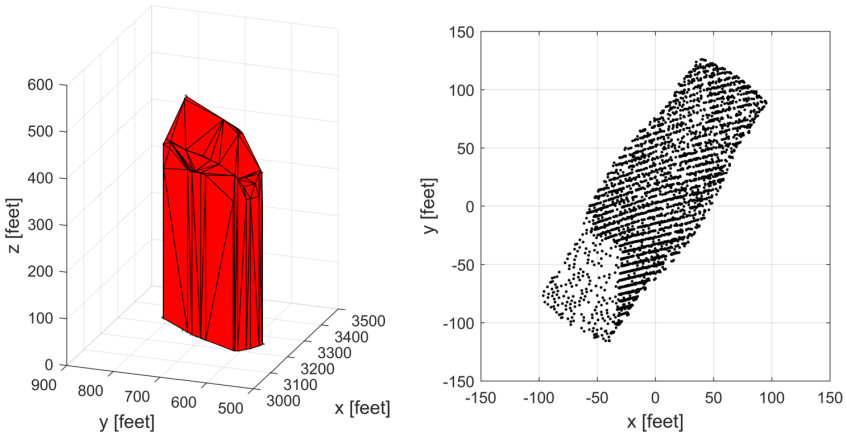
where the $k$ orthogonal transformation matrix is $A_k \in \mathbb{R}^{k \times n}$ and the result of the latent variable is $\nu \in \mathbb{R}^{n \times k}$. The inverse transformation onto the original space can be defined as

$$\mathbf{X} = A_k \nu. \tag{132}$$

Based on the PCA technique, we introduce transformation of the point cloud onto the principal component axes through an example case. In the example case, the PCA problem is formulated in two-dimensional space because the PCA result yields the rotational angle of each building cluster in terms of z-axis in a building coordinate system. As the example, we select one of the clusters resulting from the clustering algorithm. Figure 57 is the point cloud example of a cluster that is Manchester Grand Hyatt San Diego building. Although the actual hotel has two separated tall section, the results of the previous process divides into two clusters. Note that the connection part of the hotel is low height that is excluded due to the altitude constraint in the resampling/filtering phase

Using the point cloud, all point cloud are projected onto a ground plane presented in Figure 57(b), $\bar{\mathbf{x}} \in R^2$. Next, we compute the corresponding eigenvalues and eigenvectors. The physical meaning of computing eigenvalues indicate the sensitivity of the point cloud information that describes the variance of the data. The eigenvectors

describe the direction of the vectors. The computed two eigenvectors are shown in Figure 58.




(a) Three dimensional building example (Manchester Grand Hyatt San Diego)

(b) Raw point cloud information

**Figure 57:** Example of point cloud

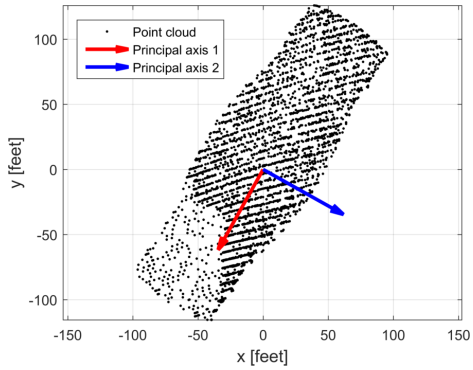
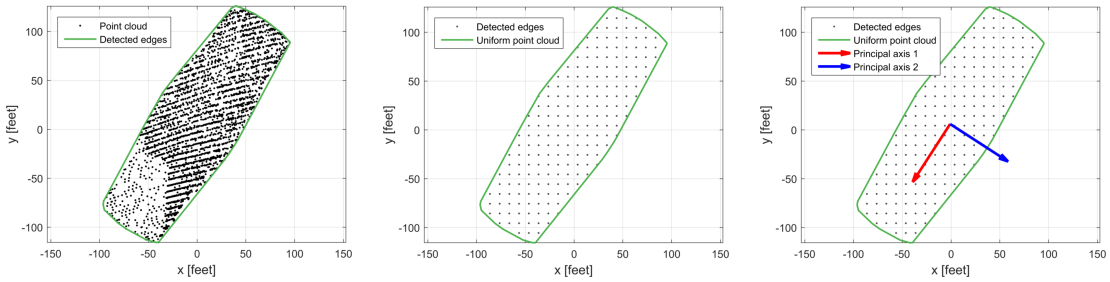



**Figure 58:** Problem of the PCA using raw LiDAR information

The visual inspection of the result reveals that the PCA algorithm estimates the principal coordinate that has the most variation of the point cloud, but the computed two principal axes are slightly shifted from the axes with the most variance. The reason of the shift is that the given point cloud information has some irregular patterns. The point cloud on the bottom of the left side is more sparse than the point cloud in the top of the right side. This irregular pattern may cause the misalignment result of the actual principal axes. Therefore, we introduce additional steps to mitigate

this problem. The main idea of the additional steps is using uniformly spread point cloud data instead of the irregular point cloud information to minimize bias resulting from the irregular point density. The additional steps are edge detection, addition of uniformly spread point cloud, and analysis of the principal axes. Figure 59 describes the results of the suggested PCA process. The result presents that the estimated principal axes detects the principal components more precisely. From this result, we can conclude that the modified PCA algorithm helps reduce biased-rotation caused by the irregular point density.




(a) Edge detection (b) Including uniformly spreaded point cloud (c) Analysis of the principal axes

**Figure 59:** Modified PCA approach

### 5.1.4 Grid generation

From the principal component result, we transform the point cloud data into the principal axes using Equation 131. The transformed point data are presented in Figure 60.

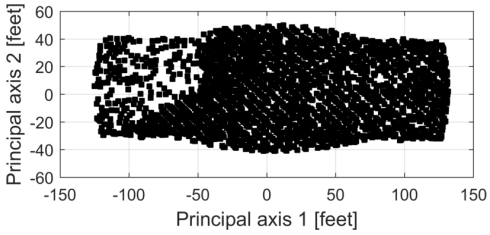
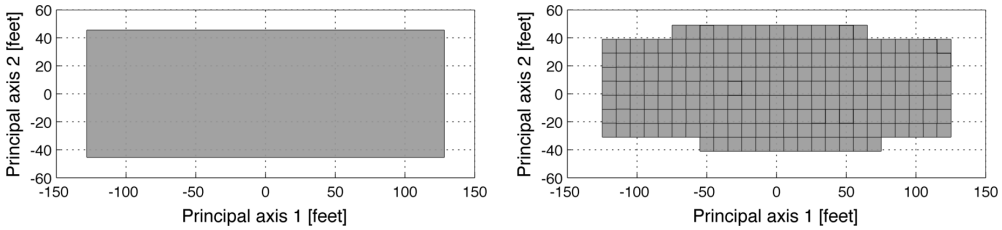



**Figure 60:** Point cloud data in principal coordinate system

The next step is to characterize a composite cuboid building configuration. It identifies width, length, and height based on the transformed point cloud information. In this step, we introduce a grid-generation to adjust the fidelity of a building. The grid generation controls the resolution of building details. Figure 61 shows examples of the grid-generation.

In the example of a single grid case presented in Figure 61(a), we calculate width and length directly from the point cloud information in the principal component domain. In the multiple grid example, the entire building area is discretized by grids with 10 [feet] by 10 [feet] (width and length) and identified a grid based on the point cloud information in the principal component domain.




(a) One grid generation (b) Multiple Grid generation

**Figure 61:** Grid generation results

Next, we need to define the height of the given cloud to define the height of

individual grid. In other words, defining the height of the cloud specifies the height of a building in the one grid case. In the multiple grid case, the height of the cloud is the height of that particular part of the building.

For example, if we suppose that we have $m$ grid, the point cloud in a grid can be written $\mathbf{P}_c = [\mathbf{P}_{c1}, \mathbf{P}_{c2}, \ \cdots \ , \mathbf{P}_{cm}]$, $(\mathbf{P}_c = \mathbb{R}^{m\times 3})$, where $\mathbf{P}_{ci} = [x_{ci}, y_{ci}, z_{ci}]$. $c$ is a cluster index and $i$ is the index for grid. If the number of point cloud in a $k$th cluster is $n$, the height can be evaluated as

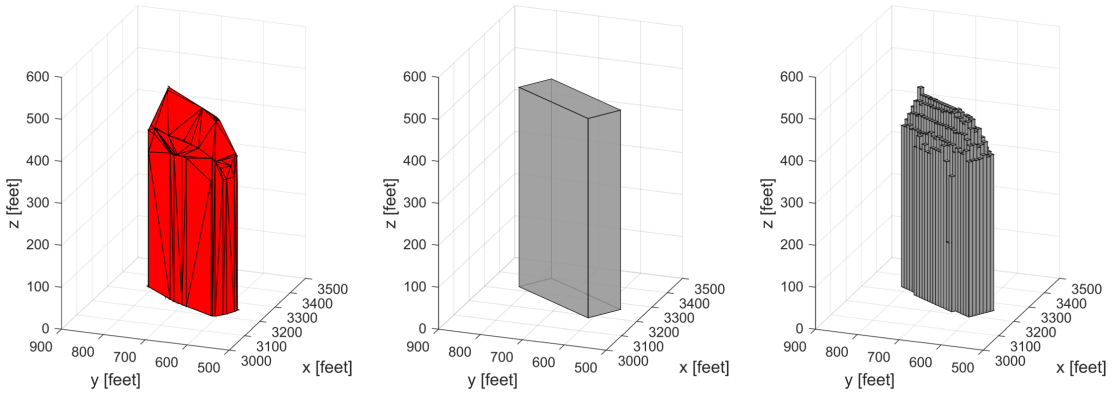
$$z_{ck} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_{cki}, \tag{133}$$

where $z_{ck}$ is the height of $k$th grid in $c$th cluster, $n$ is the number of point cloud inside of each grid, and $\mathbf{z}_{cki}$ is the height vector of $i$th point in the $k$th grid of $c$th cluster.

Based on the results of length, width, and height, the cuboid can be fully constructed. After generating a cuboid, it requires a coordinate transform from the principal axes system into the global coordinate system. The rotational angle can be easily computed by an eigenvector of the first principal component. Then, we can transform the cuboid shape onto the global coordinate system using Euler angle transformation. Figure 62 shows example results of a building modeling. Figure 62(a) is the mesh grid of the building resulting from the raw point cloud that provides approximated building shape, Figure 62(b) is the urban modeling result using the single grid approach, and Figure 62(c) is the result of the multiple grid approach. The example studies show that higher number of grid points lead to more detailed building model. To generate an entire urban environment, this process is repeated until we build the building models of all clusters.

### 5.1.5 Examples of urban construction

This section discusses example studies of an introduced urban modeling methodology. The quantitative evaluation of the urban modeling is not feasible since we do not

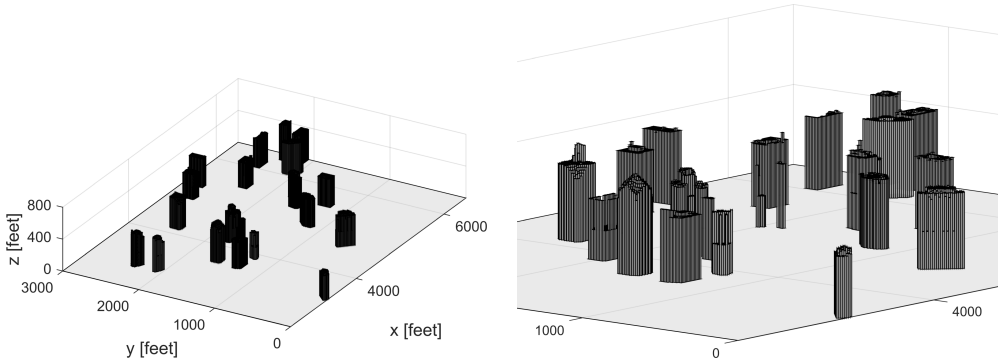



(a) Example of a building (b) Result of a building model using one grid (c) Result of a building model using multiple grids

**Figure 62:** Grid generation results in three dimensional space

have the actual measurements of the selected cities. Many papers have also performed qualitative analysis instead of quantitative evaluation about modeling accuracy because of the lack of actual measurements [60][146]. For the comparison of different fidelities, we select single-grid and multiple-grid approaches. The first example is downtown of San Diego as a dense urban area. Figures 63 and 64 are the results of the urban modeling by multiple- and single-grid approaches. The both approaches can successfully recognize the buildings that are higher than the altitude constraint. The second example is downtown San Diego in the different region as a sparse urban environment. Figures 65 and 66 present the results of the two urban modeling approaches about the sparse region. The results also show that the proposed modeling method successfully captures the size and the location of the buildings.

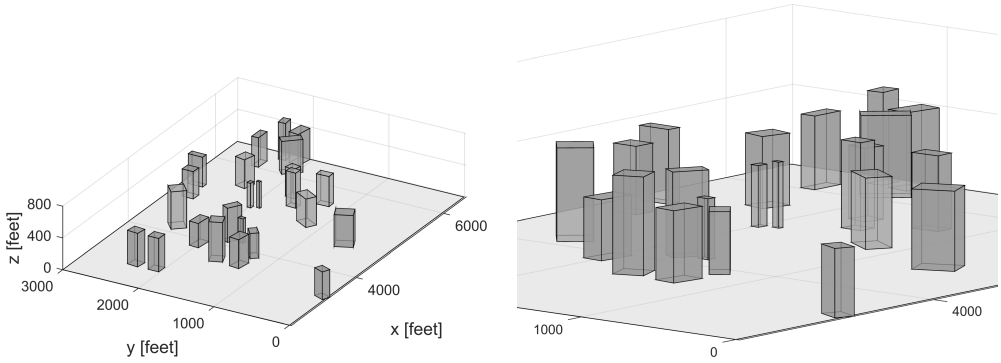
Using the proposed urban modeling methodology, we generate eight different cities shown in Figure 67. The results show that the buildings of all eight cities are successfully detected and characterized precisely. The results also show that this technique could generate an urban model in a rapid and automatic manner.




(a) Result of dense urban modeling with multiple grid

(b) Close-up view about result of dense urban modeling with multiple grid

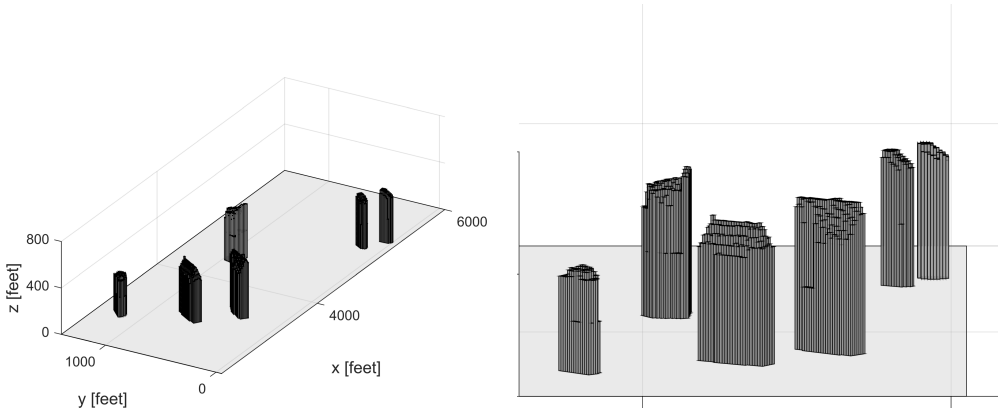
**Figure 63:** Example of a dense urban modeling with multiple grid




(a) Result of dense urban modeling with one grid

(b) Close-up view about the result of dense urban modeling with one grid

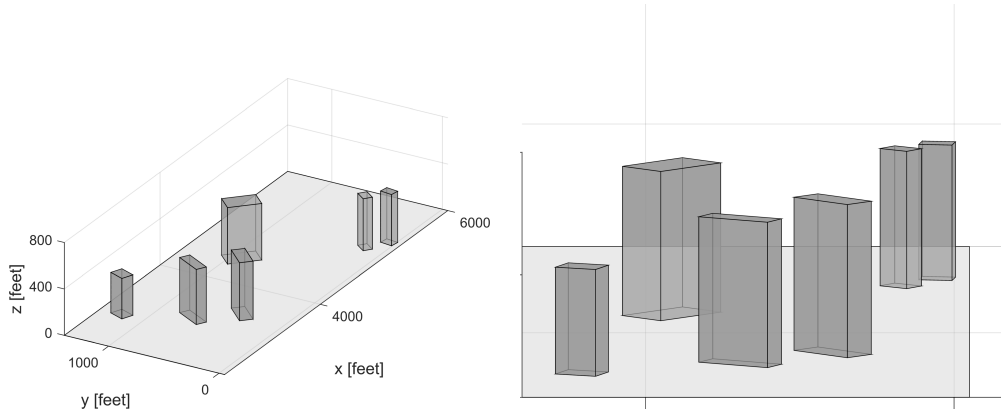
**Figure 64:** Example of a dense urban modeling with single grid




(a) Result of sparse urban modeling with multiple grid

(b) Close-up view about result of sparse urban modeling with multiple grid

**Figure 65:** Example of a sparse urban modeling with multiple grid




(a) Result of sparse urban modeling with one grid

(b) Close-up view about result of sparse urban modeling with one grid

**Figure 66:** Example of a sparse urban modeling with single grid

## *5.2 Conclusion*

This chapter introduces the data-driven rapid and automatic urban modeling technique which allows us to explore diverse collision avoidance scenarios in different urban environments. The suggested method utilizes airborne LiDAR data and has six steps: collection of LiDAR data, resampling data, identification of building clusters, PCA analysis, grid generation, and construction of a urban model. Unlike urban modeling techniques in computer science domain, this proposed methodology is more tractable for obstacle avoidance problems in urban operations because of its rapid process and appropriate level of fidelity. Experiment results show that the suggested urban modeling precisely detects all buildings and accurately constructs all the detected buildings.

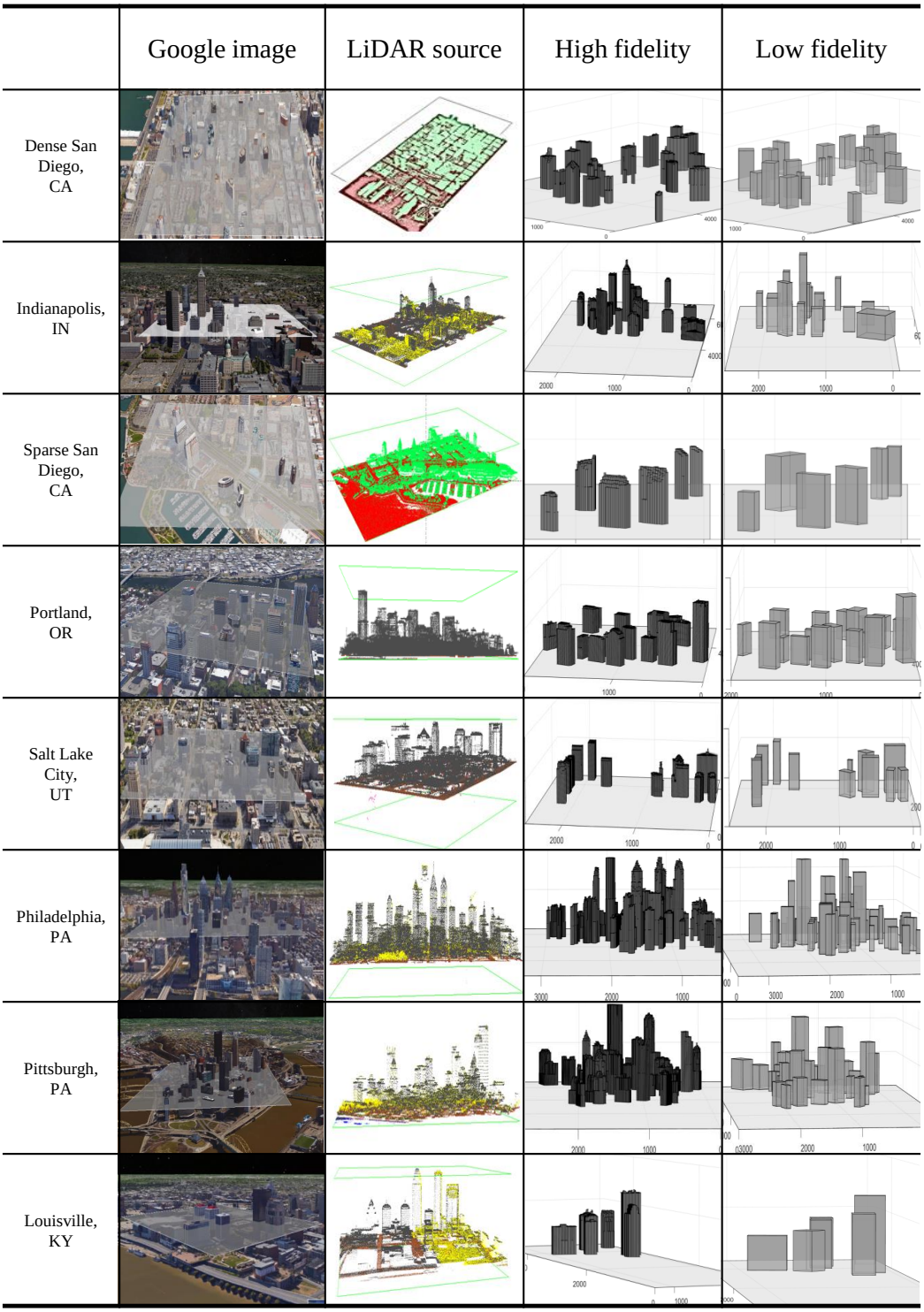



**Figure 67:** Examples of realistic urban environments

# CHAPTER VI

# SYSTEM OF SYSTEMS LEVEL INTEGRATION EXPERIMENT

This chapter describes the characterization of coupling and cross-coupling impacts with different system/sub-system components through a virtual experiment using the developed UAS modeling and simulation environment. For the characterization, we will first introduce potential experimental UAS scenarios, explain the design of experiments, and then analyze the experiment results. The potential scenario section introduces possible scenarios that can be explored to characterize the UASNAS problem through the introduced modeling and simulation environment. This section also introduces one representative scenario that we will examine as an realistic UAS problem. The following section will cover the experiment design, to discuss variable selections/designs and to introduce the final experiment set-up. The final section will be a discussion of the experiment results and illustrate crucial observations associated with coupling and cross-coupling impacts. The system of systems level experiments and understanding of coupling/cross-coupling effects in the collision avoidance problem is associated with Objective 2 of this thesis:

- Objective 2: This thesis aims to quantitatively characterize collision avoidance as a critical element of separation assurance in terms of system behavior across different levels of abstraction and multiple disciplines.

## 6.1 Potential experiment scenarios

The developed modeling and simulation environment includes models of a system, sub-systems, and the environment. The system model includes UAV flight dynamics, and the sub-systems model includes a sensor model, a flight controller, and a collision avoidance algorithm. The environment model is an urban model built from the proposed rapid, data-driven and grid-based urban modeling methodology. This system of systems modeling and simulation environment allows the exploration of diverse experimental scenarios and even observe coupling/cross-coupling effects. This section will discuss potential environmental scenarios and introduce a representative experimental scenario that will be explored through the introduced modeling and simulation environment.

The first potential scenario is observing the interactions between UAV characteristics and sensor performance relevant to the coupling problem between a system and a subsystem. The sensor capability and UAV maneuverability performance highly affect the obstacle avoidance performance as a critical factor to safely avoid an obstacle. The FAA literature [47] also poses a question "What is the required sensor coverage (distance range, azimuth range and elevation range) to avoid a fixed obstacle and a moving obstacle?" As an example of the coupling problem, a UAV with low maneuverability may require a better sensor system that is capable of having larger aperture area and longer detection range. Lower maneuverability requires an earlier maneuver to completely avoid an upcoming obstacle. On the other hand, a UAV with high maneuverability may need less stringent sensor requirements than the UAV with low maneuverability because high maneuverability enables a quick and agile maneuver to avoid an obstacle. Regarding the interactions between the sensor system and the vehicle maneuverability performance, possible research questions are as follows:

- What are the sensitivities of collision avoidance safety and energy consumption to execute an entire mission with respect to sensor parameters and UAV

maneuver characteristics?

- What is the strongest interaction between the variables related to vehicle performance and the sensor parameters?

The second potential scenario is the exploration of the interaction between a sensor system and a collision avoidance algorithm associated with the interactions between subsystem components. The collision avoidance algorithm can have an impact on avoidance characteristics depending on sensor specifications, such as distance range and the field of view (i.e., field of vision). For example, a more relaxed collision avoidance algorithm may require higher capabilities of the sensor system since the relaxed collision avoidance algorithm generates an avoidance trajectory that operates in close proximity to an obstacle. On the other hand, a more restrictive collision avoidance algorithm may be less sensitive to the sensor performance since the collision avoidance algorithm is likely to produce trajectories with a higher perception of safety. Because of these relationships, it is necessary to characterize the interaction between a sensor system and a collision avoidance algorithm. The following research questions are associated with the coupling effects between a sensor system and an obstacle avoidance algorithm:

- What are the sensitivities of mission safety and required energy with regard to sensor specification and different obstacle avoidance algorithms?

- What is the interaction between parameters of a collision avoidance algorithm, such as safe distance and minimum separation distance, and sensor parameters (field of view and detection range)?

The third possible experimental scenario is examining the impact of an urban environment with different building density levels. The level of the building density may significantly impact the obstacle avoidance characteristics depending on the features of the collision avoidance algorithm, the sensor system, and the types of UAV

platforms. To be more specific, if an urban environment has sparse obstacle density, flying through obstacles will be done in a more energy efficient manner, but if an urban environment has a high density of obstacles, avoiding the entire dense region would be a more practical way to reduce energy consumption and to circumvent an intense maneuver that minimizes collision possibilities. This possible experiment can lead to the following questions:

- What are the sensitivities/interactions of safety and energy consumption with respect to the characteristics of a collision avoidance algorithm and urban density level?
- What are the sensitivities/interactions of safety and energy consumption with regard to UAV maneuverability and the level of urban density?
- What are the sensitivities/interactions of safety and energy consumption in terms of the performance of a sensor system and the level of urban density?

Among the introduced potential scenarios, this thesis will explore the coupling effects between the sensor performance and the collision avoidance algorithm as a representative problem.

## *6.2 Characterization of an urban environment*

This section discusses the characteristics of diverse cities and defines a representative urban scenario. This characterization of cities enables an understanding of the density level of the different cities and provides the density level of the representative urban scenario.

From an obstacle avoidance perspective, the characteristics of the cities may be depicted by how dense the urban environment is. This density description can be defined by information about the building population level of the selected urban environment. Density level of an urban environment can be represented by diverse

metrics such as the number of buildings per unit area, or the area/volume obstacle density of a given urban environment. To describe the density level of an urban environment, this thesis utilizes three different metrics. Before introducing the metrics, it is necessary to first introduce the definition of an urban airspace that indicates available airspace around a given city. This urban airspace will then allow for the assessment of the density level of a given urban environment.

Figure 68 illustrates the definition of the urban airspace. The urban airspace in XY space is specified by a safety margin the rightmost and leftmost of the given buildings. Additional margin space on each side from the rightmost and leftmost buildings is also considered in the XY axis. The urban airspace in the Z axis is defined by the minimum altitude constraint, the maximum height of a given urban environment, and the safety margin. The minimum altitude constraint is considered because UAVs are not operated near the ground environment, which has complex ground structures, such as transmission towers and elevated highways. The maximum height of the urban airspace is defined by the sum of the safety margin and the height of the tallest building in the given urban airspace area. In this thesis, the safety margin space is assumed to be 50 $[feet]$. From this airspace definition, the urban airspaces of eight different cities are defined in Figure 69. In the figures, the blue cuboid box indicates the urban airspace. A visual inspection shows that the defined urban airspaces have different features with respect to the area, the volume, and the height of the airspaces.

Based on the definition of the urban airspace, the urban density level is computed. For the measurement of the density level, three metrics are applied. The first metric is two-dimensional airspace ratio $\rho_{2D}$, which is the occupied area by buildings in the two-dimensional airspace. This metric can represent the population level with respect to the two-dimensional space. The occupied building area $A_{occ}$ is the total building area on the cross-sectional plane at the minimum altitude. The two-dimensional

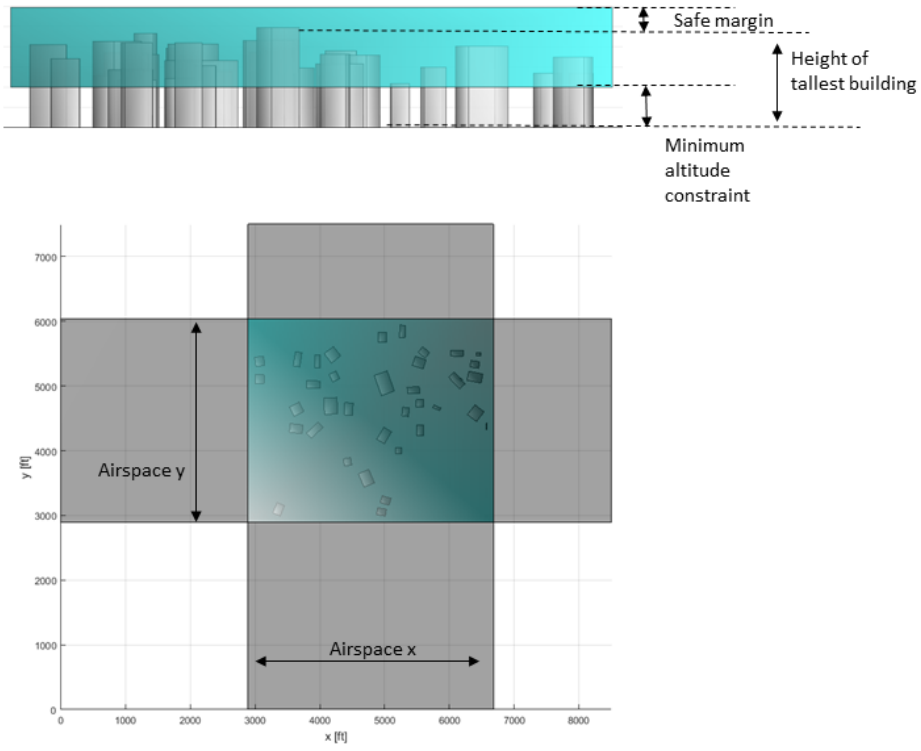
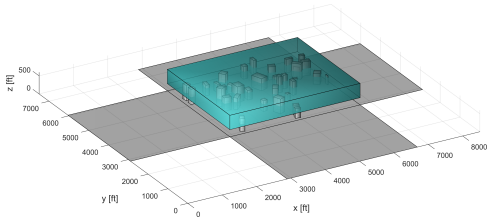
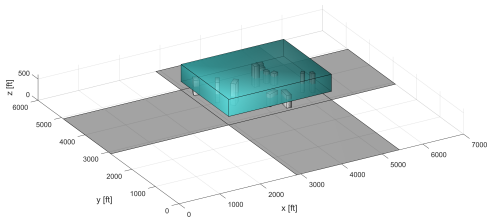


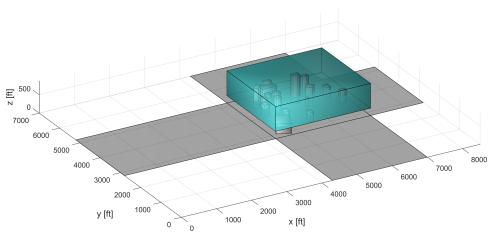

**Figure 68:** Definition of urban airspace (Example : Dense San Diego)



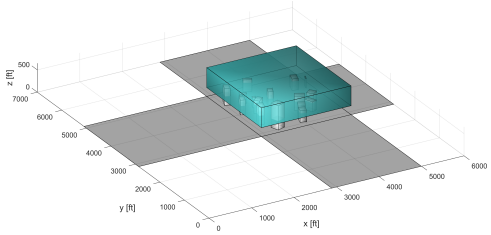

(a) Dense San Diego, CA



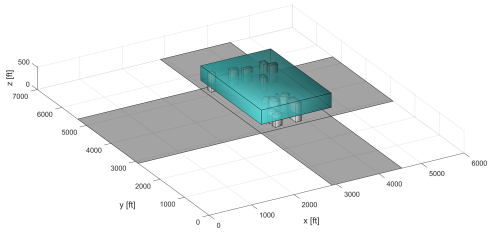

(b) Sparse San Diego, CA



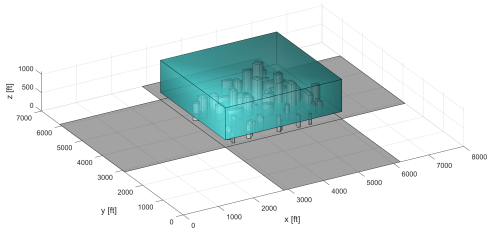

(c) Indianapolis, IN



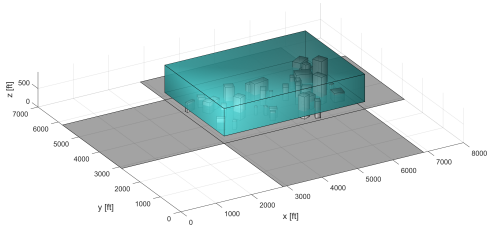

(d) Portland, OR



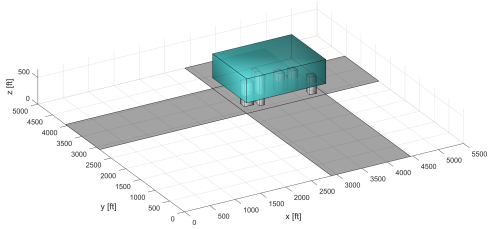

(e) Salt Lake City, UT




(f) Philadelphia, PA




(g) Pittsburgh, PA




(h) Louisville, KY

**Figure 69:** Available airspace of eight representative cities

airspace ratio can be mathematically written as

$$\rho_{2D} = \frac{A_{occ}}{X_{airspace} Y_{airspace}}. \tag{134}$$

The two-dimensional density cannot fully describe the urban density level since the UAV flies in three-dimensional space. Therefore, this thesis suggests another metric, three-dimensional airspace ratio $\rho_{3D}$, which is the occupied volume by buildings in a given urban airspace. In other words, it is the ratio of the buildings' volume $V_{occ}$ within the given urban airspace to the entire urban airspace volume. $Z_{airspace}$ is the height between the minimum altitude constraint and the maximum altitude of the given urban airspace. This three-dimensional airspace ratio can be mathematically written as

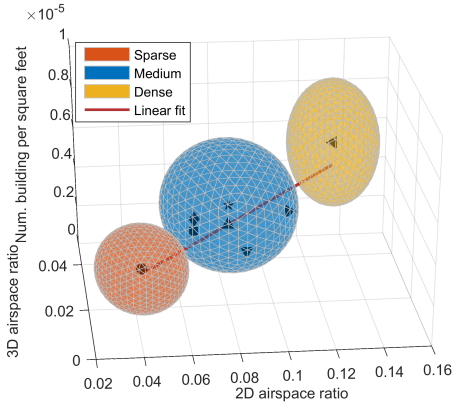
$$\rho_{3D} = \frac{V_{occ}}{X_{airspace} Y_{airspace} Z_{airspace}}. \tag{135}$$

Although these two metrics can give density from a two- and three-dimensional space perspective, they cannot provide the sense of the number of buildings. Knowing the number of building can provide an idea of how complex airspace routes are in the given urban environment. Hence, another metric that will be considered is the number of buildings per square foot in the two-dimensional urban airspace. This metric simply illustrates how many buildings exist in the given urban airspace.

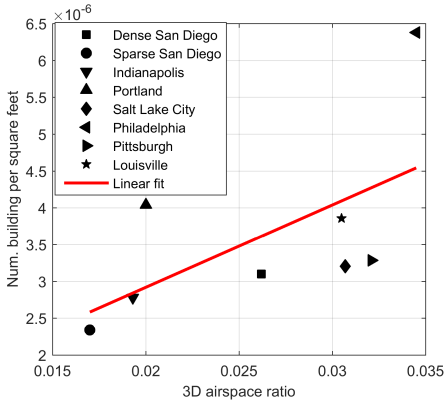
Based on the definitions of three metrics, one can evaluate and compare the density level of the eight different cities. Figure 70 summarizes the comparison results. Figure 70(a) is a three-dimensional graph to visualize the three metrics for the eight cities, and the remaining graphs are the projected results onto each plane (XY, XZ, and YZ planes). Visual inspection of the results reveals a linear trend. This linear trend implies that a city with a higher 2D/3D airspace ratio is likely to have a higher number of buildings per square foot. Among the eight cities, the densest city is Philadelphia, while the sparsest city is Louisville.

All the cities are categorized into three groups, which are sparse, medium, and

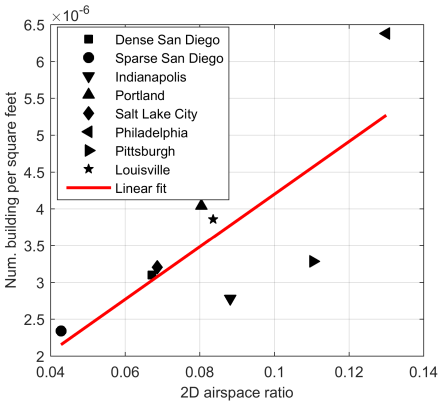
dense with respect to the density level, in order to characterize the density trend of all cities more easily according to the similarity feature. The results presented in 70(a) show that the sparse group includes Louisville only, the dense group consists of Philadelphia only, and the rest are in the medium density group.



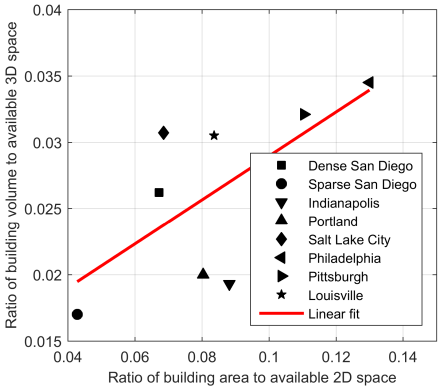

(a) 3D plot




(b) 3D ratio vs. number of buildings per square foot




(c) 2D ratio vs. number of buildings per square foot




(d) Building ratio in 2D vs. building ratio in 3D

**Figure 70:** Analysis results of urban environment

This section discussed the overall characteristics of the eight different cities according to urban complexity. To characterize the urban density level, three different metrics were introduced. From the analysis results using the metrics, the eight cities were divided into three groups (sparse, medium, and dense) for categorization purposes. Because of the limited LiDAR resources of cities, this categorization yields a statistically weak conclusion, but it still provides a meaningful classification. To

achieve a statistically meaningful trend, more urban LiDAR data is necessary.

The analysis outcomes of the urban density are applicable to the problem of random city generation with different density levels. In general, creating a realistic urban environment using actual urban information is a highly challenging problem because of the difficulty of collecting urban information (e.g., LiDAR and geometry data). The density analysis of different buildings enables us to generalize the urban density model. This generalized urban model is potentially implementable to create an artificial but realistic urban environment without any actual urban information.

Based on the density studies of the eight different cities, the dense San Diego urban area shown in Figure 71 was selected as a representative scenario because it has a medium level of density.

**Figure 71:** Representative urban scenario (San Diego) - Google Earth image

## 6.3 System of systems level experimental design

The objective of the integrated experiment is to observe coupling and cross-coupling effects between a system and a system, a system and a subsystem, and a subsystem and a subsystem. The representative scenario of the integrated experiment (i.e., system of systems level experiment) is defined as the exploration of coupling and cross-coupling effects between sensor and GNC parameters. With regard to exploring these coupling and cross-coupling effects, the following fundamental questions can be posed:

- Question 1: Which cases are the infeasible cases (collision cases)? What are the key drivers which result in collision situations?
- Question 2: What are the sensitivity of the design variables (sensor parameters and GNC parameters) with respect to the minimum distance and energy?
- Question 3: What are the interactions between sensor and GNC parameters?

These questions will be answered by the outcomes of the integrated experiment. Before discussing the design process of the experiments, the experimental environment will be introduced to provide the entire simulation structure. The experimental environment includes aircraft dynamics, an aircraft controller, a GNC algorithm, a sensor model, and an urban environment. The aircraft dynamics are represented by the point mass model described in Section 2.1.1, and the aircraft controller is an application of the statistical gain-scheduling method described in Section 2.2. The GNC algorithm employs the suggested Hybrid method illustrated in Section 3.3. The urban environment model is constructed by the rapid, data-driven, and grid-based urban modeling technique stated in Chapter 5.

The on-board sensor is defined as the general light detection and ranging equipment with no uncertainties. The vehicle, sensor, collision avoidance, and other parameters are summarized in Table 22. The urban model is assumed to be the downtown of

San Diego with a single grid approach shown in Figure 64 since a single grid approach is the simplest urban model, which will help accelerate the experiments.

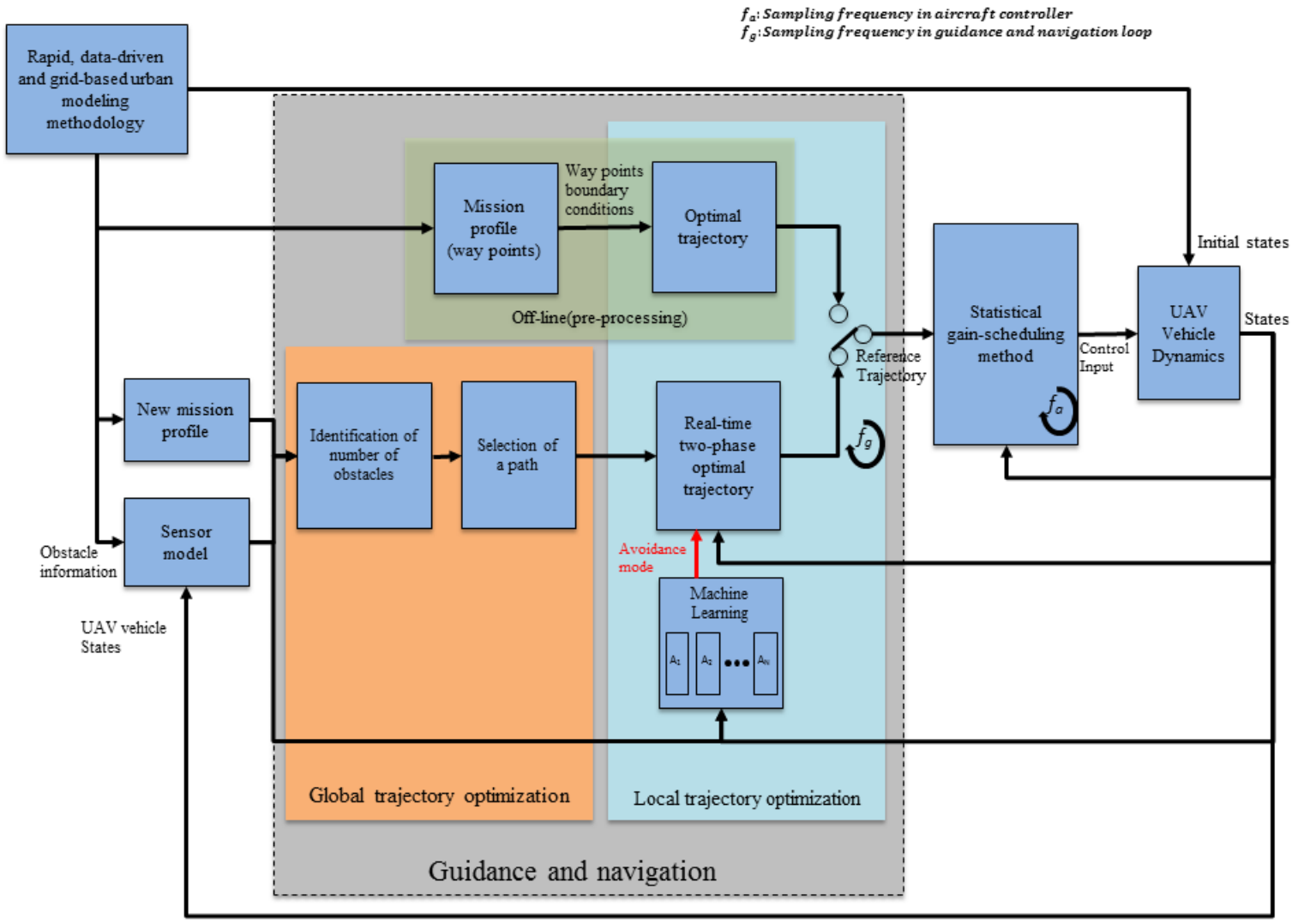



**Figure 72:** Block diagram of UAV flight simulation

**Table 22:** UAV parameters of the integrated experiment

| | UAV parameters | Variable | Value | Unit |
|---|---|---|---|---|
| Vehicle parameters | Weight | $w$ | 29.76 | $[lb]$ |
| | Planform area | $S$ | 6.1 | $[ft^2]$ |
| | Area swept out by the propeller | $S_p$ | 0.1348 | $[ft^2]$ |
| | Propeller aerodynamic coefficient | $C_p$ | 1 | |
| | Efficiency constant of the motor | $K_m$ | 8 | |
| | Lift coefficient at zero angle of attack | $C_{L0}$ | 0.28 | |
| | Lift curve slope | $C_{L\alpha}$ | 3.45 | |
| | Aspect ratio | $AR$ | 10.7 | |
| | Span efficiency | $e$ | 0.9 | |
| | Zero-lift drag coefficient | $C_{D0}$ | 0.03 | |
| Sensor parameters | Sensor resolution | $\theta_{sen}$ | 1 | $[deg]$ |
| Collision avoidance parameters | Minimum altitude | $z_{min}$ | 200 | $[ft]$ |
| | Maximum acceleration | $g_{max}$ | 3 | $[ft/s^2]$ |
| Other parameters | Update rate of guidance and navigation loop | $f_g$ | 1 | $[Hz]$ |
| | Update rate of aircraft control loop | $f_a$ | 10 | $[Hz]$ |

The experimental setup requires a reasonable amount of computational resources and must have a large enough number of experiment cases to generalize the outcomes from the experiments. However, due to the large number of variables, the integrated experiment may require such a large number of experiments that there is no computationally feasible solution. Therefore, the experiment should tailor the number of variables to achieve a reasonable number of experiments, which enables the sensitivity and interaction analysis. The issue of the large number of experiments will be addressed through an example study and provide a solution to achieve a reasonable number of experiments.

To examine this issue of experiment size, the design of experiments described in Table 23 will be assumed. The experiment considers several key design variables such as sensor parameters, a certain aircraft model, guidance and navigation parameters, and initial/terminal conditions. Since the experiment provides the answers to the three questions related to sensitivities and interactions between the sensor and the GNC parameters, the experiment fixes the aircraft model and varies the sensor and GNC parameters. The sensor parameters include a range of azimuth and elevation angles, as well as distance. The guidance and navigation have two parameters, safe distance and separation distance, which are parameters in global- and local- trajectory

optimizers. For the exploration to observe at least quadratic responses, each of these variables needs four factors. Although the initial/terminal conditions exist as an infinite number of conditions, we select 10 factors for simplicity. Based on these assumptions and variable definitions, the result of the total number of DOE cases is approximately $10^9$, which is too many to be computationally suitable. Therefore, to reach a reasonable number of experiments, it is necessary to strategically select experiment cases. In this subsection, the detailed process of creating the experimental design will be discussed.

**Table 23:** Example of infeasible DOE

| | Types of variable | Variable name | Factors |
|---|---|---|---|
| Input DOE variables | Sensor parameter | AZ | 4 |
| | | EL | 4 |
| | | Distance | 4 |
| | Aircraft | | 1 |
| | Guidance and navigation | Safe distance | 4 |
| | | Separation distance | 4 |
| | Initial position | | 10 |
| | Initial velocity | | 10 |
| | Initial acceleration | | 10 |
| | Terminal position | | 10 |
| | Terminal velocity | | 10 |
| | Terminal acceleration | | 10 |
| | | Total number of cases | 1.024E+09 |

### 6.3.1 Defining initial trajectories

Possible initial trajectories exist in the infinite number of trajectories according to the initial/terminal conditions that may cause an increase in the computational expense. To fully explore the obstacle avoidance problem in a given urban environment, the initial trajectories must be representative scenarios, not trivial cases that have no obstacles along the trajectory. To generate representative trajectories and avoid trivial cases, the most challenging trajectories that are a subset of the representative trajectories are selected. This section elaborates on the details of defining the

representative trajectories and specifying the most challenging trajectories.

To decrease the number of initial trajectories, this thesis makes several assumptions. First, the initial flight condition is assumed to be level flight, and then ten initial/terminal positions are defined that are equivalently spaced in the X and Y axis. It is also assumed that the initial altitude is half of the height of the tallest building in the given environment (Dense San Diego), which is 350 $[feet]$, which satisfies the minimum altitude constraint of 200 $[feet]$. This altitude requirement is determined by the concept of air traffic management for low-altitude drones performed by NASA [1].

The initial/terminal positions are located farther from the urban airspace because this gap from the urban airspace prevents aggressive maneuvers near the initial/terminal conditions due to close proximity to an obstacle. The initial velocity selected is the normal cruise speed of the Aerosonde UAV, 70 $[feet/sec]$. The vectors of the initial velocities are assumed to be parallel to the X and Y axis, which can be written as $[1 \quad 0 \quad 0]$ and $[0 \quad 1 \quad 0]$ in the global coordinate system. Figure 73 illustrates the initial/terminal conditions.

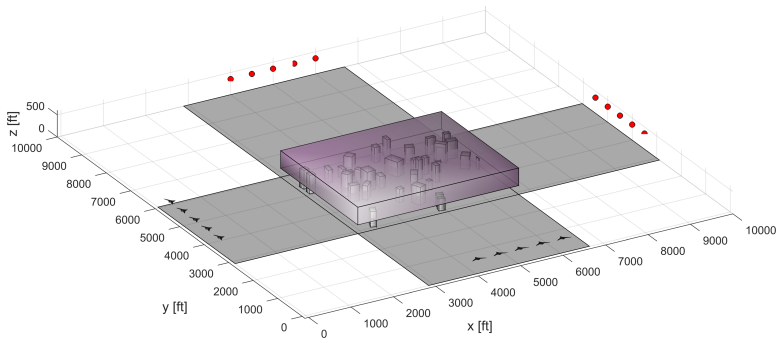
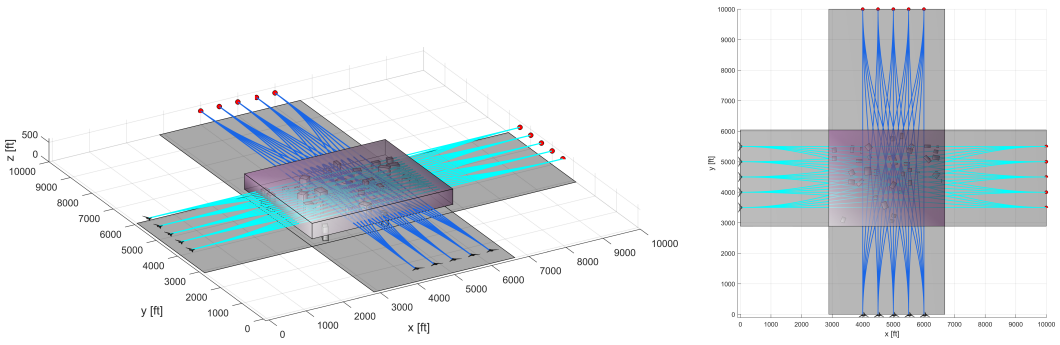



**Figure 73:** Initial/terminal conditions

Based on the ten initial/terminal conditions, vehicle dynamics, and constraints described in Section 4.1.4, one solves the trajectory optimization problem to determine initial trajectories. The trajectory optimization problem utilized by the optimization

framework was introduced in Chapter 3. The designed full-factorial trajectory optimization problem with respect to X and Y axis has 50 experimental cases. The results of the initial trajectories are shown in Figure 74.




(a) 3D view (b) Top view

**Figure 74:** Initial and terminal conditions

Among 50 initial trajectories, the ten most challenging trajectories will be selected. To evaluate the challenge level of a trajectory, the obstacle ratio along that trajectory is calculated. The obstacle ratio is the ratio of obstacle area to the area of an entire initial trajectory. The area of the initial trajectory is the area defined by a margin on both sides of the initial trajectory because it is necessary to consider the size of an aircraft. The area of the obstacle is the obstacle area within the area of the initial trajectory in an urban airspace. Figure 75 depicts an example area. In the figure, $A$ indicates the area of an initial trajectory, and $B$ is the area of obstacles along the trajectory. The obstacle ratio along the initial trajectory is $\rho = A/B$.

The obstacle ratios of 50 initial trajectories are evaluated and presented as a histogram in Figure 76. The trajectories on the right side, with a higher obstacle ratio, have more obstacles along the trajectory compared to those on the left side of the figure.

Using the calculated obstacle ratios, one can select ten initial trajectories with the highest obstacle ratios, which implies that they are the most challenging trajectories. The selected initial trajectories are depicted in Figure 77. This approach of

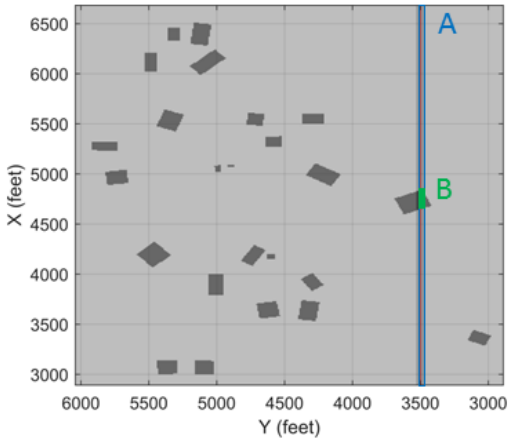



**Figure 75:** Example of computing the obstacle ratio along an initial trajectory

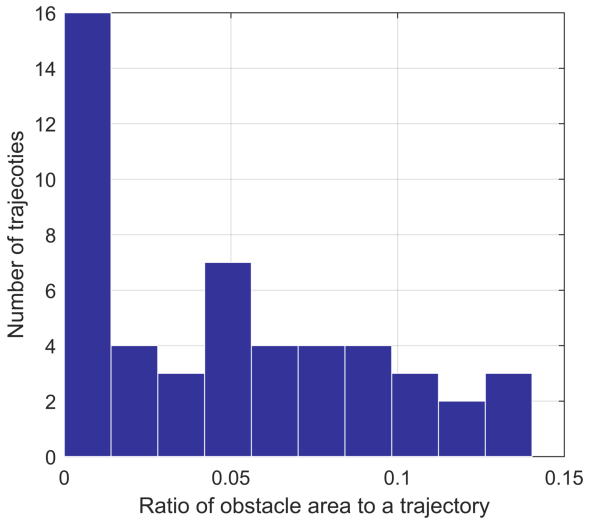
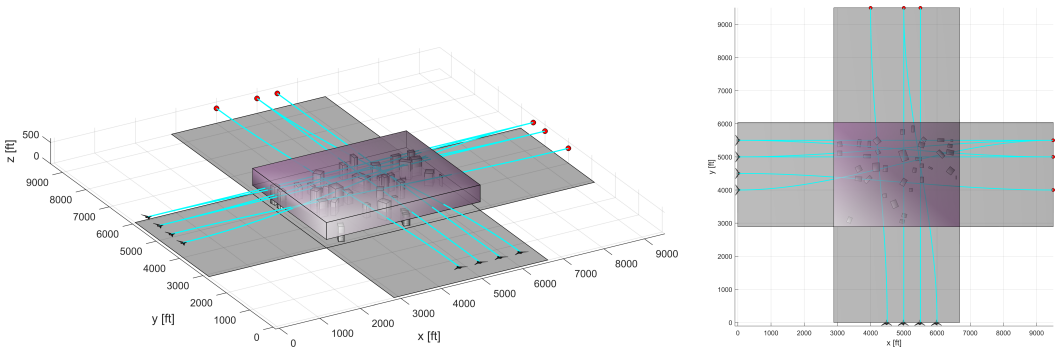



**Figure 76:** Distribution of the obstacle ratio

selecting ten initial trajectories avoids the computational expense of including trivial trajectories, which have zero or few obstacles, in the experiment.




(a) 3D view (b) Top view

**Figure 77:** Selected ten initial trajectories

### 6.3.2 Defining sensor variables

This section describes a defining experiment of the sensor parameters (distance range, ranges of azimuth and elevation angle). The experimental design of the sensor variables is determined by the representative LiDAR sensor specifications. Table 24 summarizes commercially available LiDAR sensors provided by Phoenix Aerial Systems [2]. The characteristics of these sensors are that a sensor with a longer range is likely to have a small range of elevation angle while the sensors with a shorter range have a larger field of view.

**Table 24:** Specifications of representative LiDAR sensors

| | VLP-16 | HDL32E | LUX4 | LUX 8 | VUX-1 High Accuracy | VUX-1 UAV | VUX-1 LongRange | Min | Max |
|---|---|---|---|---|---|---|---|---|---|
| Range(m) | 120 | 120 | 200 | 200 | 400 | 920 | 1350 | 120 | 1350 |
| Range (feet) | 393.6 | 393.6 | 656 | 656 | 1312 | 3017.6 | 4428 | 393.6 | 4428 |
| AZ(deg) | 360 | 360 | -60 to 50 | -60 to 50 | 355 | 330 | 330 | 330 | 360 |
| EL(deg) | -15 to 15 | -30 to 10 | 3.2 | 6.4 | Single layer | Single layer | Single layer | 15 | 20 |

Based on the LiDAR specifications, the sensor variables are designed. Table 25 represents the ranges and factors of each variable. The sensor parameters have four factors in order to get a third order response.

**Table 25:** Experiment design of sensor parameters

| Range | Min. value | Max. value | Factors |
|---|---|---|---|
| Distance | 400 (feet) | 1300 (feet) | (400 700 1000 1300) |
| AZ | -80 (deg) | 80 (deg) | (100 120 140 160) |
| EL | -35 (deg) | 35 (deg) | (5 15 25 35) |

### 6.3.3 Defining a design of experiments for guidance and navigation parameters

In the proposed two-level obstacle avoidance algorithm, two parameters, which are safe-distance $r_s$ and the minimum separation distance between buildings $D_o$, are critical variables to determine the performance of obstacle avoidance. Intuitively, with a higher value of safe-distance, a UAV selects an avoidance trajectory with a greater distance from a building. It means that the avoidance trajectory may have a higher perception of safety. When there is a higher value of minimum separation distance, multiple obstacles can be identified as one obstacle since the clustering algorithm in the global-trajectory optimization recognizes individual clusters based on the distance criterion. Therefore, a higher minimum separation distance results in a more restrictive trajectory. Therefore, these two parameters may significantly impact the obstacle avoidance performance. The selection of these two parameters is also a challenging problem because if high values of these two variables are chosen to enhance the perception of safety, the results of the avoidance trajectory may not fly between buildings. On the other hand, if we choose very small values, many crashes can happen because of the proximity to the obstacles. To avoid these problems, the two variables are defined from the observations of the given urban environment (San Diego). The main idea of defining two variables is that these variables are determined from the average gap between the buildings in the given urban environment. To be more specific, we assess the distribution of the distance between the buildings that is computed in the two-dimensional urban map. For example, we project the San Diego map onto the ground plane shown in Figure 78 and evaluate the minimum distances

between the two buildings. In the figure, each individual rectangular box describes a building.

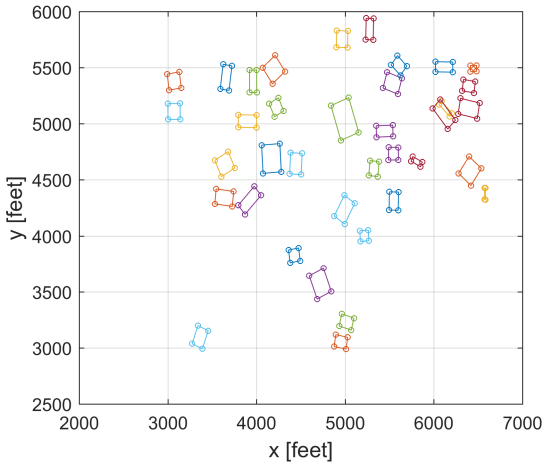



**Figure 78:** Building map in two-dimensional space

Figure 79 illustrates the distribution of the distance between the buildings. The mean $\mu_d$ of the distribution is 57.5 $[feet]$, and the standard deviation $\sigma_d$ is 27.5 $[feet]$. From this analysis, the safe distance is defined as 30, 57.5 and 85 $[feet]$ that is respectively $\mu_d - \sigma_d$, $\mu_d$, and $\mu_d + \sigma_d$. The minimum separation distance is defined as two times the safe-distance, which are thus 60, 115, and 170 $[feet]$ because in the two-level algorithm, the minimum separation distance should satisfy the condition, $2r_s \leq D_o$.

### 6.3.4 Summary of the design of experiments

Table 26 summarizes the design of experiments based on the introduced assumptions and designed experiments. The total number of experiments is 3840, which is much more favorable to computational expense than the approximately one billion cases required in the initial experiment design.

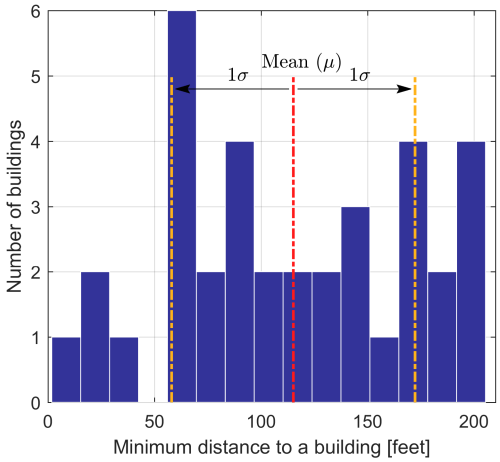



**Figure 79:** Distribution of the distance between buildings

**Table 26:** Summary of integrated experiment

| | Types of variable | Variable name | Factors | Subfactor | Min | Max | DOE |
|---|---|---|---|---|---|---|---|
| Input DOE variables | Sensor parameter | AZ | 4 | | 20 | 80 | 20 40 60 80 |
| | | EL | 4 | | 5 | 35 | 5 15 25 35 |
| | | Distance | 4 | | 400 | 1300 | 400 700 1000 1300 |
| | Aircraft | | 1 | | | | |
| | Guidance and navigation | Safe distance | 6 | 3 | 50 | 150 | (30 57.5 85) |
| | | Separation distance | | 3 | 100 | 200 | (60 115 170) |
| | Initial trajectory | | 10 | | | | The most challenging routes |
| | Total DOE | | 3840 | | | | |

## *6.4 Results of the system of systems level experiments 1*

The result of the experimental design yields 3840 cases with ten different initial trajectories. This number of experiments is still large enough to be infeasible to run a single desktop machine. To mitigate this computational issue, the designed experiments were executed using the parallel computing resource that is provided by the PACE (Partnership for an Advanced Computing Environment) cluster at the Georgia Institute of Technology.

Figure 80 shows the results of the avoidance trajectories with ten different initial trajectories. Each trajectory includes 384 cases according to the different experiment parameters: sensor parameters and GNC parameters described in Table 26.

A visual inspection of the ten results shows that the avoidance trajectories have high variability depending on the initial trajectories. Some trajectories present high variability. Other trajectories have low variability. These trends imply that the experiment may have a highly nonlinear trend.

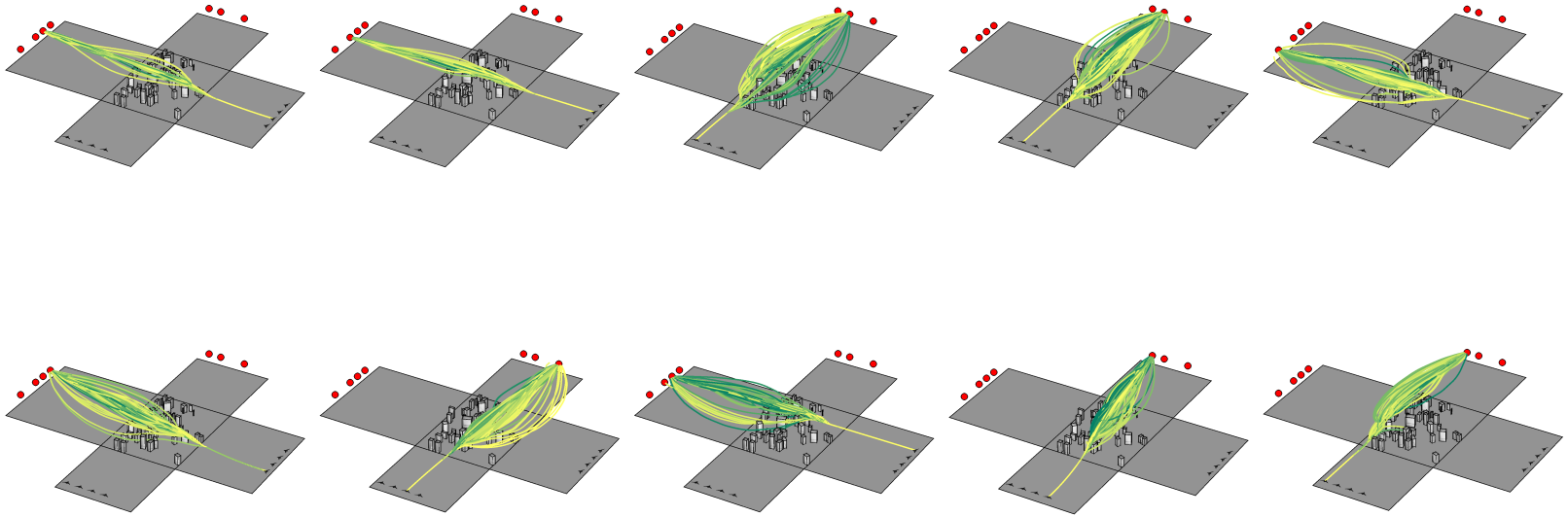


**Figure 80:** Results of integrated experiment

Based on the experiment results, one can analyze infeasible and feasible cases. The infeasible cases identify the main drivers of the collision situation. The feasible case studies characterize the sensitivities and interactions between the sensor parameters and GNC parameters.

The analysis of the infeasible avoidance cases will be discussed to answer the following questions: "What are the infeasible cases? What are the key factors that yield a collision situation?" mentioned in Section 6.1. To answer the questions, the histogram results are analyzed.

To investigate the infeasible case study, it is necessary to count the number of non-collision cases and collision cases according to different sensor/GNC parameters. To be more specific, a parameter of each variable in the sensor system includes 960 cases with ten trajectories. In other words, each trajectory has 96 cases depending on the design variables. The safe distance parameters [60 115 170] (feet) have [1920 1280 640] experimental cases, and the parameters of the minimum separation distance [60 115 170] (feet) have [640 1280 1920] experiment cases. If one of the ten trajectories in a design variable has a collision case, it specifies the collision case. To be a non-collision case, the results of the ten trajectories in the design variable should not be collided. Figure 81 shows the distribution of the non-collision cases. The results show that the major drivers of achieving a non-collision maneuver are a sensor elevation angle and range. When these two parameters are designed to be larger than 15 degrees and 700 feet, the avoidance performance can be significantly improved. The graph reveals that the sensor azimuth angle is relatively insensitive because of the small variation of collision cases. The parameters of safe distance and minimum separation distance present a linear trend in the collision cases, but they do not imply any trend related to avoidance performance since these two variables are dependent on each other, as described in Equation 124. In other words, the minimum separation distance must satisfy the condition, which is two times larger than the safe distance. To check the

sensitivity of two variables (safe distance and minimum separation distance), the heat map that counts the number of collision and non-collision cases is plotted in Figure 82. It is important to note that the boxes in this figure which have zeros in them violate the condition that the minimum separation distance must be greater than or equal to double the safe distance, so they do not count. With that in mind, this heat map shows that these two parameters are insensitive because all combinations of the safe distance and minimum separation distance have a similar number of collision and non-collision.

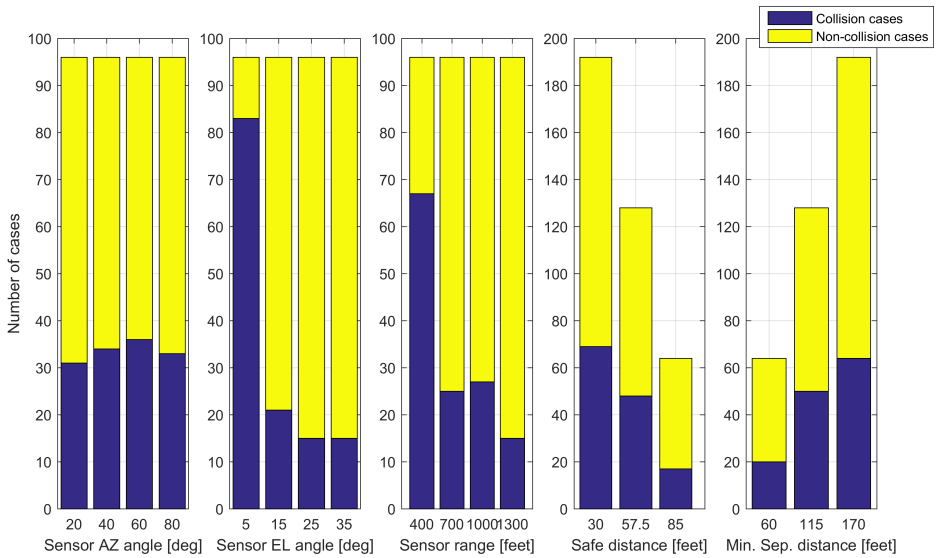
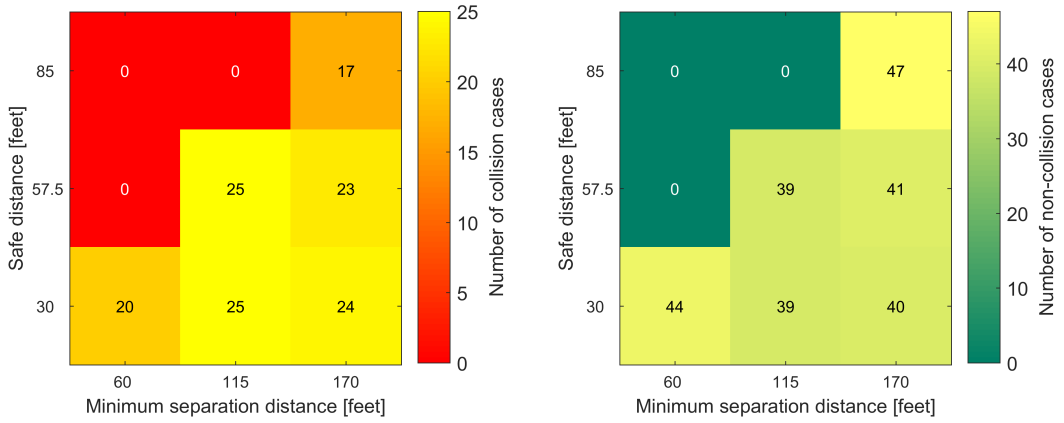



**Figure 81:** Non-collision distribution of design variables

To characterize the performance of the obstacle avoidance trajectory, two metrics (minimum distance and energy consumption) are considered. The minimum distance indicates the perception of the safety and the energy consumption describes the actual energy consumption to perform the entire mission. The minimum distance, which indicates the safety level, is computed by the minimum distance between the UAV and obstacles along the entire trajectory. The work of an aircraft is calculated from the classical energy formulation described in Equations 125 and 126. Figure 83 shows the results of the two metrics from all avoidance trajectories except the collision cases.




(a) Collision case (b) Non-collision case

**Figure 82:** Heat map of safe distance and minimum separation distance

The results indicate that whereas most cases require low energy, some cases require high energy.

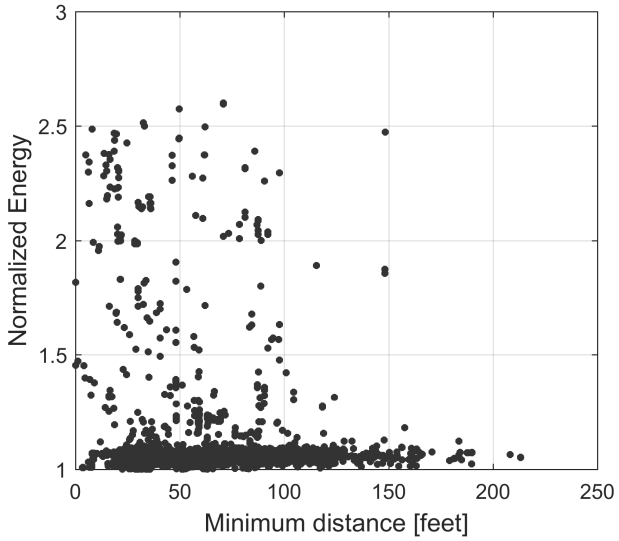
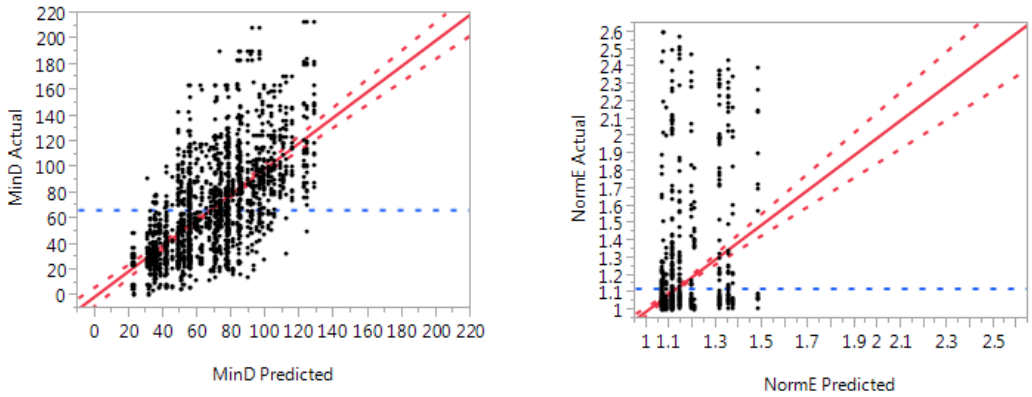



**Figure 83:** Results of collision avoidance

Next, we discuss feasible case studies on sensitivity and interactions associated with the question "What is the sensitivity of design variables (sensor and GNC parameters) with respect to minimum distance and energy?". This study can be implemented using various techniques, such as surrogate modeling, Analysis of Variance

(ANOVA), and screening tests. Sensitivity analysis using surrogate modeling is a typical approach for an early design phase. Lamoureux et al. have applied the kriging surrogate modeling technique to their sensitivity analysis for the early phase analysis of the health indicator of an aircraft engine's pumping unit [110]. However, this surrogate modeling approach is not a suitable technique for the sensitivity analysis of the feasible cases for this thesis because the response of the two metrics are highly nonlinear. Figure 84 shows the results of surrogate models using the second order response surface modeling technique [110]. The graph shows the actual by predicted plot that indicates the quality of the prediction model. It is obvious that the regression model cannot capture the actual response. Table 27 summarizes the quality of the regression model using the common metrics. The results evidently illustrate that the generated surrogate model cannot represent the actual response. Consequently, the sensitivity analysis requires another approach without any prior models.




(a) Minimum distance (b) Normalized energy

**Figure 84:** Issues of surrogate models

**Table 27:** Summary of surrogate models

| | Min distance (feet) | Normalized energy |
|---|---|---|
| RSquare | 0.4292 | 0.1214 |
| RSquare Adj | 0.4261 | 0.1197 |
| Root Mean Square Error | 29.2526 | 0.2337 |

Figure 85 shows the response changes according to parameter variations in one

trajectory among ten initial trajectories that are for observing the non-linearity of the response. Table 28 summarizes the parameter variations. The result shows that the combination of a poor sensor and a restrictive GNC parameter requires more energy to avoid obstacles. A better sensor leads to more energy efficient trajectories, and a more restrictive GNC parameter results in a farther distance from the obstacles. However, this result of the local sensitivity is only acceptable in this initial trajectory because of the highly nonlinear response. To explore the trajectory variations on the different levels of required energy, see Figure 86. The majority of the trajectories are in the lower energy region, but a few trajectories are in the higher energy region. The actual trajectory response indicates that the energy efficient trajectories have monotonic maneuvers, but the trajectories with the high energy have more aggressive maneuvers. From these two results, the responses of the avoidance trajectories are highly non-linear and chaotic.

**Table 28:** Parameter definition of a local sensitivity analysis

| | Sensor | | GNC | | |
|---|---|---|---|---|---|
| | AZ [degree] | EL[degree] | Range [feet] | Safe distance [feet] | Min. Sep. distance [feet] |
| Parameter 1 | 20 | 15 | 400 | 30 | 60 |
| Parameter 2 | 20 | 15 | 1300 | 30 | 60 |
| Parameter 3 | 80 | 35 | 1300 | 30 | 60 |
| Parameter 4 | 80 | 35 | 1300 | 30 | 170 |
| Parameter 5 | 80 | 35 | 1300 | 85 | 170 |

Using the experiment data, one way to perform the sensitivity analysis is a partition analysis. Partition analysis is one of the multivariable analysis techniques. The partition method recursively partitions to generate a decision tree. When data have input variables $\mathbf{X}$ and an output variable $Y$, the partition process splits the tree structure based on the groupings of X. The grouping of X is judged by fitting the result to the output Y. This partition process is typically repeated until the final tree structure reaches the desired fit. The partition technique is a powerful technique because it does not require any prior models like a response surface model to explore the relationship between the input variables X, the results of the decision tree are

very intuitive, and it can also handle high-dimensional problems easily. The partition analysis is provided by the statistical software JMP.

Because of the highly nonlinear response, we employ the partition analysis with respect to three different views, risk averse, risk taken, and risk nominal. The risk averse approaches mean that the worst case scenarios are always considered. The risk taken approach utilizes low risk results. The risk nominal approach takes the medium risk of the entire response. Therefore, the risk averse approach evaluates the minimum value of the minimum distance among the ten trajectory results because increasing the safe distance is more desirable in terms of the perception of safety. The risk averse approach also assesses the maximum value of the energy result among the ten trajectories since lower energy consumption means better avoidance trajectory in terms of the energy perspective.

Using the same reasoning, the risk taken approach calculates the two metrics by the maximum value of the minimum distance and the minimum value of the energy consumption. The nominal concept uses the median value of the two metrics.

Figure 88 is the partitioning result by the risk averse approach. The result reveals that the major improving factor is a safe distance greater than 57.5 [feet]. The second factor is a sensor range that is greater than 700 [feet]. The results of the risk nominal approach are shown in Figure 89. The analysis results illustrate that the biggest hitters for improving the minimum distance are a safe distance and a sensor range that are greater than 57.5 [feet] and 1000 [feet], respectively. Figure 90 presents the analysis of the risk taken approach. The major variables enhancing the perception of the safety are a sensor range, sensor elevation angle, and safe distance that are higher than 1000 [feet], 15[degree], and 57.5 [feet], respectively. From the minimum distance analysis with respect to the three different perspectives, the safe distance is the common variable that includes all three partition analyses. Therefore, the safe distance is a critical variable that enables higher safety perception.

Next, we discuss the results of the partition analysis of the energy consumption. In the risk averse approach, the safe distance and sensor elevation angle are identified as major contributors. These variables are lower than 57.5 [feet] and lower than 15 [degrees]. The risk nominal analysis also shows that the major variables minimizing energy consumption are the same variables as the conservative approaches. The risk taken approach reveals that the major factors are the safe distance and sensor elevation angles that are lower than 85 [feet] and 25 [degrees]. The energy improvements in the optimistic results are minimal. In sum, safe distance and sensor elevation angle are key variables to enhance the energy efficiency.

Based on the partition analysis, there is a trade-off because a higher perception of safety requires a larger safe distance, but a lower energy needs a smaller safe distance. Using different risk perspectives, the partition analysis allows a decision maker to analyze the sensitivity.

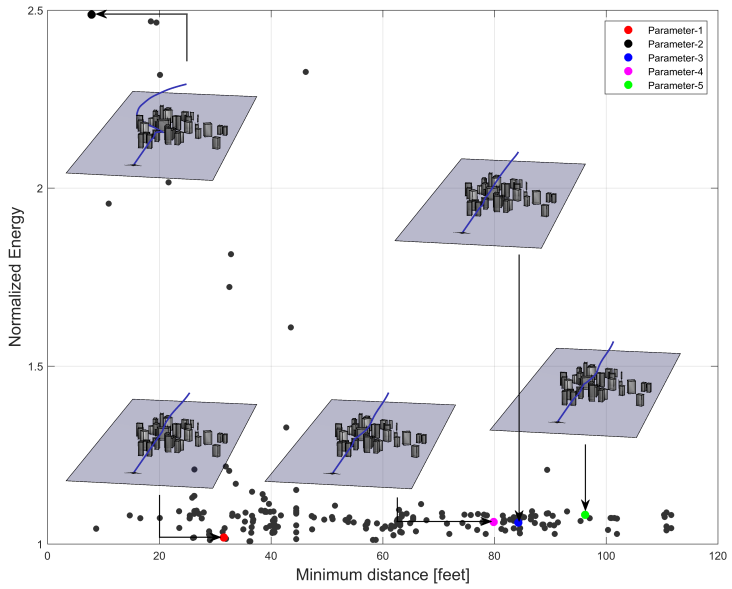



**Figure 85:** Local sensitivity analysis

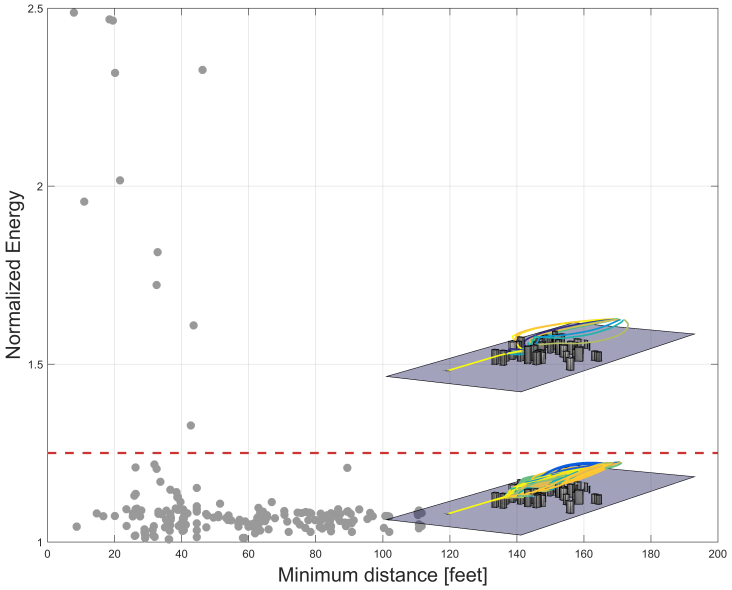



**Figure 86:** Trajectory variation according to two different levels of energy consumption

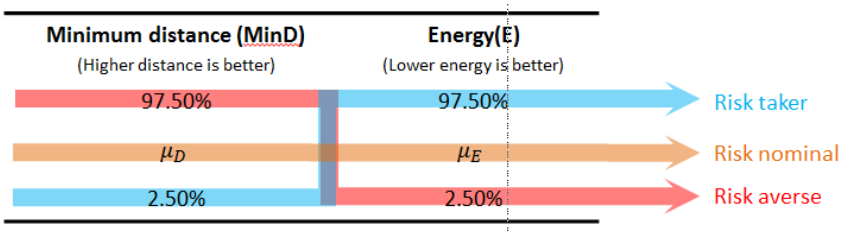



**Figure 87:** Risk definition for a partition analysis

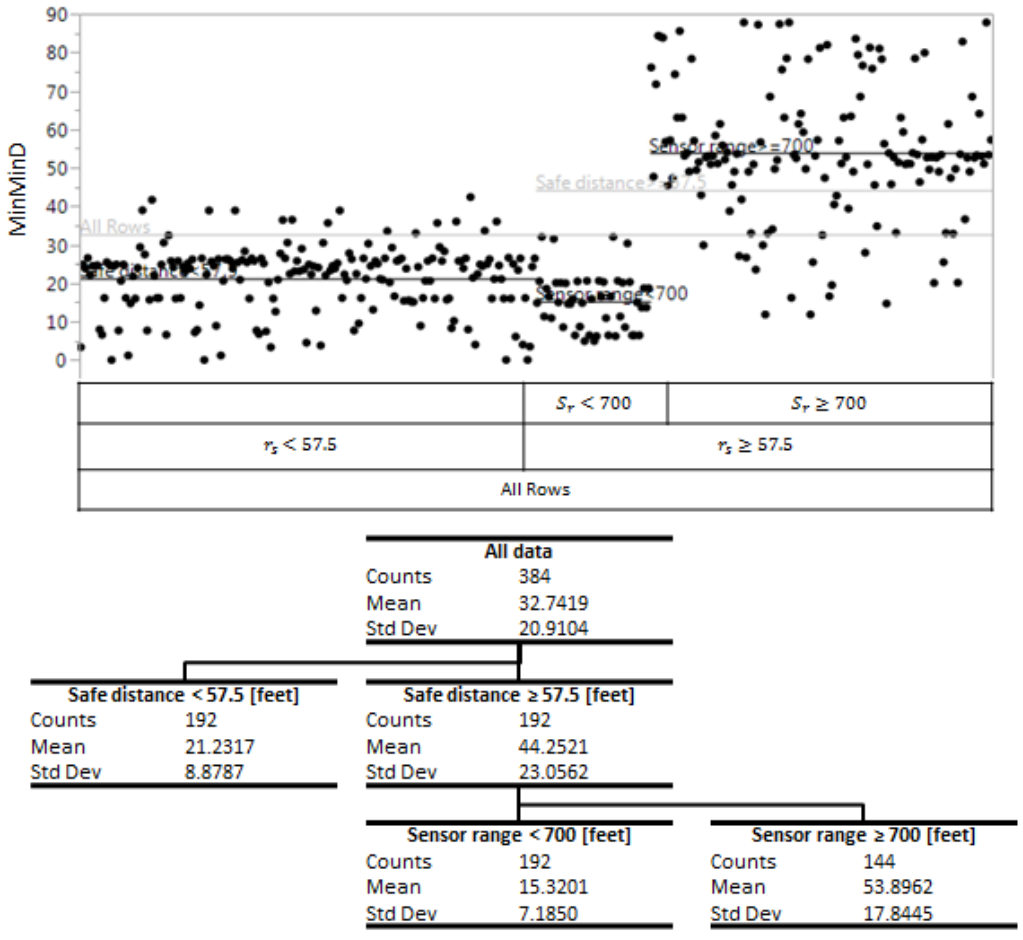



**Figure 88:** Partition analysis of minimum distance (Risk averse approach)

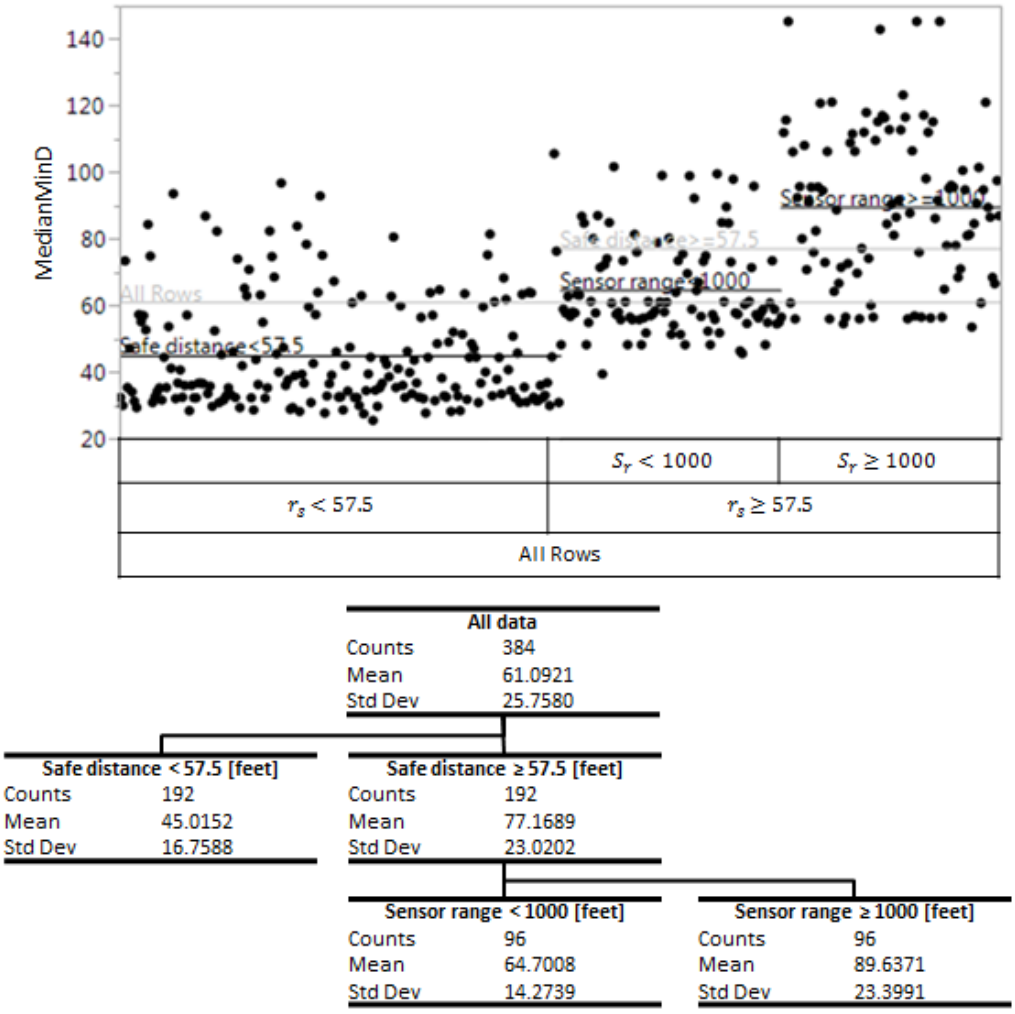



**Figure 89:** Partition analysis of minimum distance (Risk nominal approach)

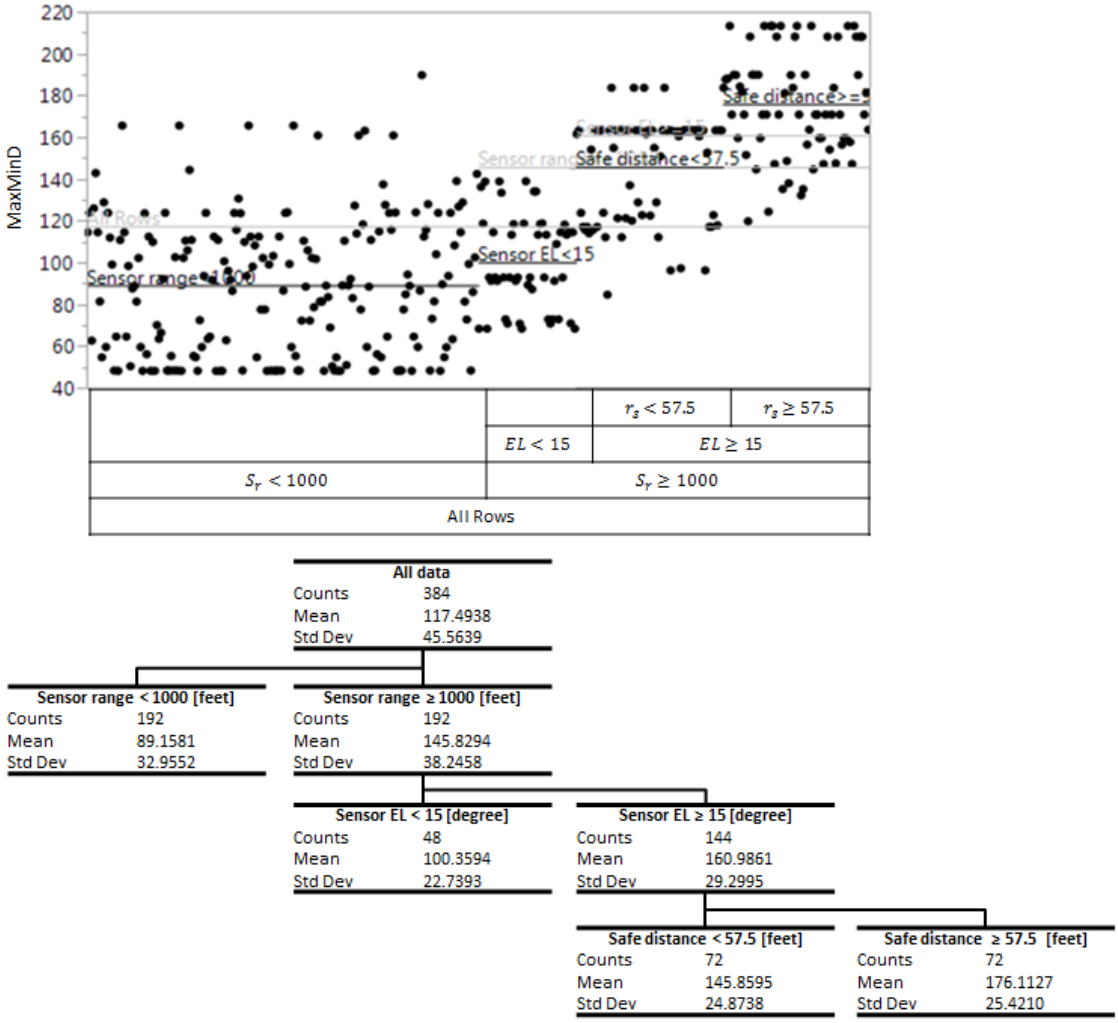



**Figure 90:** Partition analysis of minimum distance (Risk taken approach)

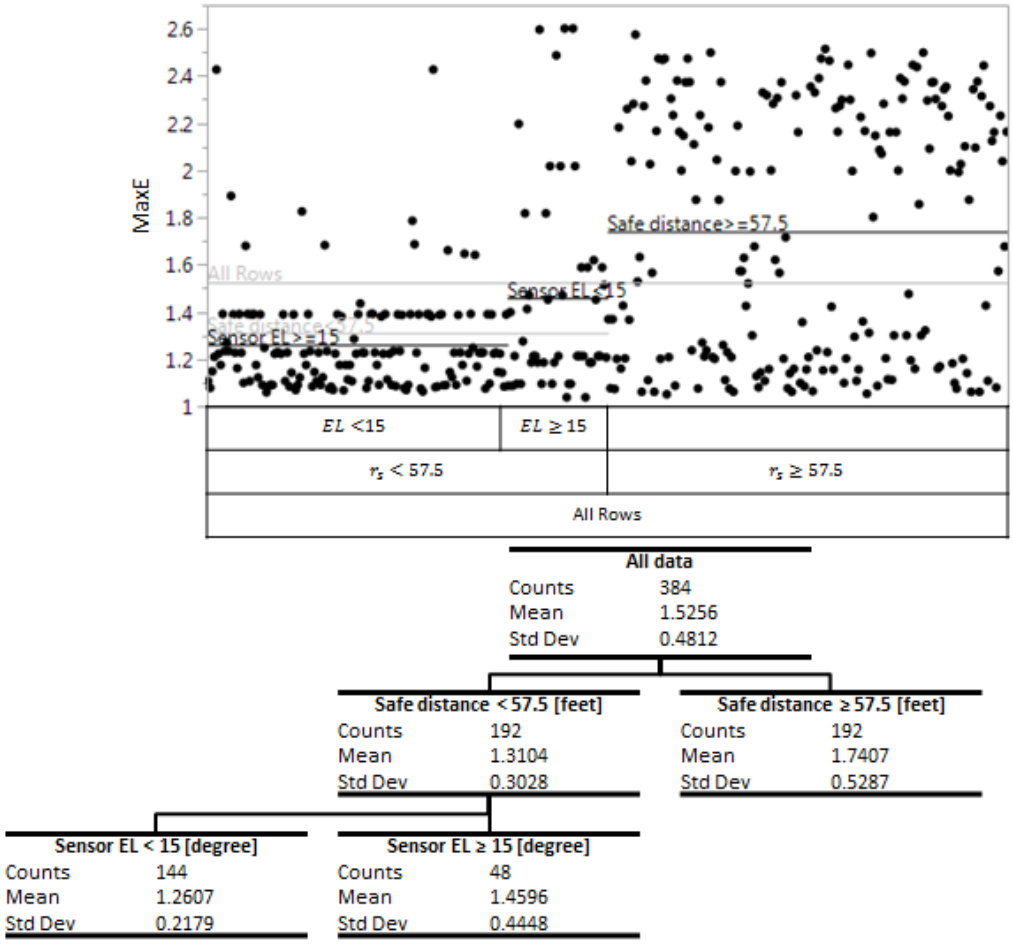



**Figure 91:** Partition analysis of energy consumption (Risk averse approach)

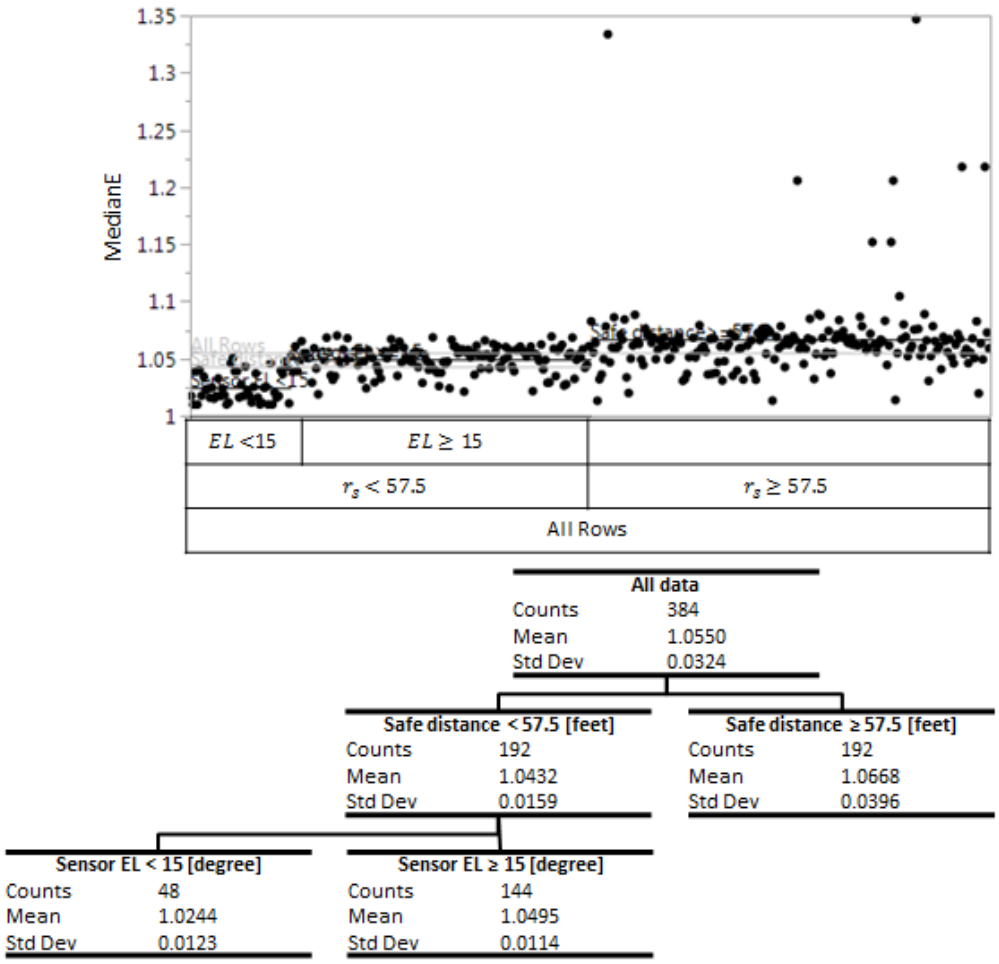



**Figure 92:** Partition analysis of energy consumption (Risk nominal approach)

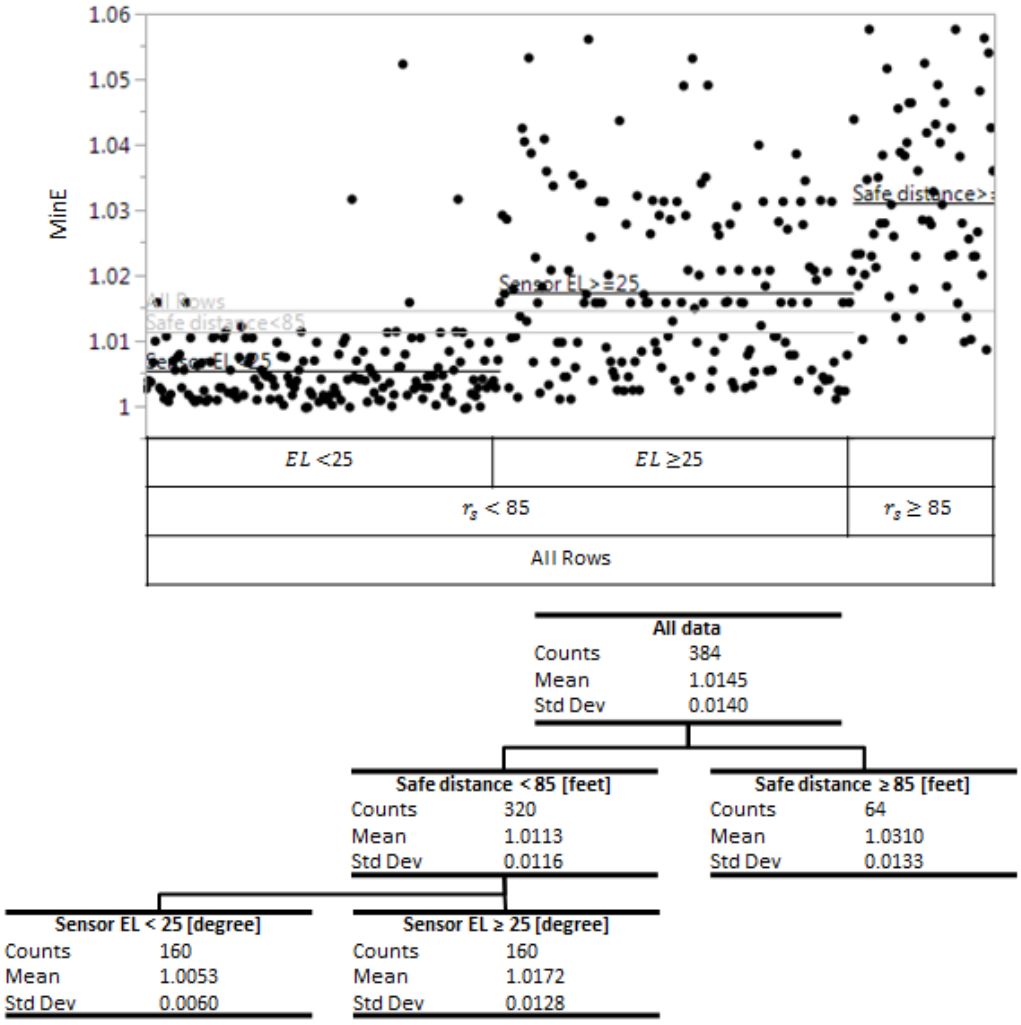



**Figure 93:** Partition analysis of energy consumption (Risk taken approach)

## *6.5 Results of the system of systems level experiments 2*

The results of the first integrated experiments show that the given problem has a highly non-linear trend, making it difficult to observe the sensitivity or interactions of sensor and GNC parameters. Moreover, the results of the partition analysis do not provide a statistically strong conclusion to answer the two questions: What is the sensitivity of design variables (sensor and GNC parameters) with respect to the minimum distance and energy?, and what is the interaction between sensor and GNC parameters?

To answer those questions, an experiment should be redefined to reduce the variability of the response. To reduce the variability, the design of experiments is redesigned by fixing one trajectory and selecting two of the GNC parameters (the safe distance $r_s$ and the minimum separation distance, $D_o$) that are restrictive and relaxed and two types of sensors that are a good sensor and a poor sensor. The relaxed GNC parameter generates an avoidance trajectory and allows a UAV to fly near an obstacle, and the restrictive GNC parameter generates an avoidance trajectory with a higher perception of the safety. Figure 94 illustrates the experiment concept (full-factorial design of experiments). The red dots indicate experiment points. Table 29 summarizes the definitions of GNC/sensor parameters.

**Table 29:** Parameters of the redesigned Experiment

| | Sensor parameter | | |
|---|---|---|---|
| Variable | AZ(deg) | EL(deg) | Range (feet) |
| Poor sensor | 20 | 15 | 700 |
| Good sensor | 80 | 35 | 1300 |
| | GNC parameter | | |
| Variable | Safe Distance (feet) | Min.Sep Distance(feet) | |
| Relaxed GNC | 30 | 60 | |
| Restrictive GNC | 85 | 170 | |

To obtain more statistically meaningful results, the designed experiments are repeated by perturbing the initial positions of a UAV in Y and Z direction. There

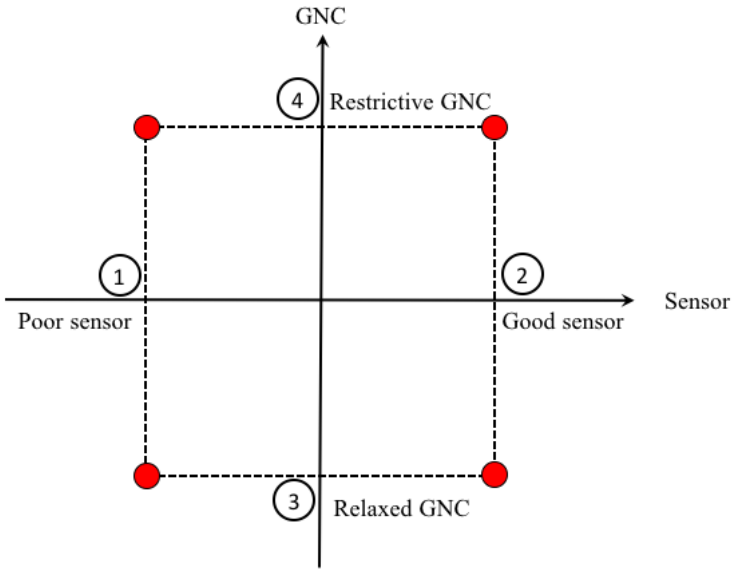



**Figure 94:** Concept of the redesigned experiment

are 200 repetitions for each of the four designed experiments. Therefore, the total number of cases is 800. The experiment outcomes are characterized based on the two metrics, minimum distance and energy consumption.

In Figure 29, the numbers (①, ②, ③, ④) indicate the analysis order. The first and second experiments fix the sensor parameters to a poor or good sensor and then vary the GNC parameters. These experiments enable the sensitivity analysis of the GNC parameters according to the fixed sensor. The third and fourth experiments fix the GNC parameters to either relaxed or restrictive and then change the sensor performance. The experiments also explain the sensitivity of the sensor parameter according to the defined GNC parameter.

The result of the experiment ① (Fixed poor sensor and varied GNC parameters) is shown in Figure 95 with respect to the minimum distance and the normalized energy. The trajectory response in terms of the minimum distance shows that the avoidance trajectories with restrictive GNC keep a farther distance from an obstacle because a higher safe distance in the GNC generates a higher perception of safety.

The results of both cases with regard to the energy are widely spread. Because of the wide spread trend, both avoidance trajectories have high variability. To observe the distribution, the histograms are plotted in Figure 96. The histogram results evidently indicate that the restrictive GNC improves safety distance from an obstacle because of the higher safe distance. However, the restrictive GNC worsens the required energy. The reason is that the poor sensor, which has the narrow field of views, detects an obstacle too late and the high safe distance generates a more aggressive avoidance trajectory.

The second experiment (2) (Fixed good sensor and varied GNC parameters) is depicted in Figure 97. The results of the minimum distance represent that more restrictive GNC parameters improve the safety perception. The result of the energy consumption reveals that both responses have low energy consumption. Figure 98 illustrates the distribution of both results. The histogram shows that a more restrictive GNC enhances the perception of safety, but it also yields a small decrease of the energy efficiency.

The third experiment (3) (Fixed relaxed GNC and varied sensor performance) is a comparison analysis of the impacts of the sensor capability according to different GNC parameters. The experiment result is depicted in Figure 99. The results show that a better sensor provides better energy efficiency since a better sensor allows earlier trajectory generation to avoid upcoming obstacles, and the better sensor has a little improvement of perception safety. To compare both good and poor sensors impacts on the relaxed GNC parameters, the histograms are plotted in Figure 100. The histogram results illustrate that a better sensor provides a better safety perception, and a little improvement of the energy consumption.

The last experiment (4) is the comparison study of the sensor capability when the GNC parameters are defined as a restrictive concept. The experiment results shown in Figure 101 represent that the good sensor system provides lower variability

with respect to the energy consumption and generates energy efficient trajectories. The results of the trajectories also demonstrate that the avoidance trajectories have relatively less variability. The results also indicate that a better sensor yields a higher perception of safety. Figure 102 is the histogram analysis to observe the distribution changes. Similar to the previous observation, a better sensor results in a farther distance from an obstacle since the mean value of the minimum distance shifts to the right according to the sensor improvements, and the better sensor also provides more energy efficient trajectories because the mean of the distribution moves towards lower energy.

The previous analysis allows us to explore the response changes according to different sensor/GNC parameters. However, these results still do not provide the answers about the sensitivity and interactions between the variables. Therefore, for the sensitivity and interaction analyses, the previous histogram results are used to generate the contour surface plot shown in Figure 103. It illustrates the averages and the quantile values (2.5%, 25%, 75%, 97.5%) of the two metrics. Using this contour surface result, this thesis will present some key features and discuss the sensitivity and interactions using the interaction profiler later this section.

The result shows that the restrictive GNC and good sensor generate more space from an obstacle. In other words, a more restrictive GNC and better sensor systems allow a better perception of safety. Another observation is that when the GNC parameter with the poor sensor was put in the restrictive concept, the variability of the minimum distance still increases. The reason is that due to the limited performance of the sensor, the UAV detects the obstacle too late and the avoidance trajectory also generates a more aggressive avoidance trajectory caused by the restrictive GNC parameter.

In the case of the energy consumption, the better sensor system creates energy efficient trajectories and decreases the variability of the energy consumption. The

quantile response shows that as the sensor capability increases, the variability improvements reach the point of diminishing returns. In Figure 103, the contour surface has the mean value located outside of the distribution between 25% and 75% because the actual response is skewed in the distribution.

Based on the results of the contour surface graph, this thesis will discuss the sensitivity and interaction using an interaction profiler. To discuss them with respect to different safety perspectives, three different terminologies like the previous partition analysis are defined in Table 30. A risk taker pursues an optimistic approach that always considers the best case scenario. Therefore, the risk taker approach uses 97.5 % response in the minimum distance and 2.5 % response in the energy consumption. On the other hand, a risk averse approach is totally opposite to the risk taker approach and evaluates the worst case scenario. Thus, the risk averse approach utilizes 2.5 % response in the minimum distance and 97.5 % response in the energy metric. The last approach is risk nominal, which applies the mean values ($\mu_D$, $\mu_E$) of both responses.

**Table 30:** Definition of risk profiles

| Risk profile | Minimum distance<br>Higher distance is better | Energy consumption<br>Lower energy is better |
|---|---|---|
| Risk Taker | 97.50% | 2.50% |
| Risk Averse | 2.50% | 97.50% |
| Risk Nominal | $\mu_D$ | $\mu_E$ |

The first analysis of sensitivities and interactions will be in terms of the risk nominal perspective. Figures 104 and 105 are interaction profiles. The profile of the minimum distance displays weak interactions between the sensor and GNC parameters, as indicated by the similar slopes of the two curves. Moreover, to achieve a higher perception of safety, it requires a better sensor and more restrictive GNC parameters. The result of the energy profile reveals that the poor sensor and restrictive GNC are more sensitive, and for the energy efficient trajectories, a good sensor is required regardless of the impact of GNC parameters because the impact of the GNC

parameter in the good sensor is significantly small. Because of the slope difference, it shows a strong interaction between the two variables.

Second is the analysis of the results from a risk taker perspective. Figures 106 and 107 are the interaction profiler of the risk taker view. The outcome of the minimum distance shows that the interaction between the two variables is weak. The improvement of the perception of safety requires a better sensor and more restrictive GNC parameters. The response of the minimum energy shows that the good sensor system and restrictive GNC parameters are not relatively sensitive compared to the poor/relaxed GNC. Because of the low energy variation regardless of the parameter variations, the impact on the energy consumption is minimal.

The final perspective to analyze is the risk averse one. Figures 108 and 109 are the interaction profiler. The result of the minimum distance indicates that a good sensor and restrictive GNC parameters are more sensitive than a poor sensor and relaxed GNC parameters. It also reveals that the good sensor and restrictive GNC parameters yield a higher perception of safety. The energy response indicates that the sensor parameters are more sensitive than the GNC parameters. The good sensor generates energy efficient trajectories regardless of GNC parameters.

In sum, this thesis has analyzed the interactions and sensitivities based on three different safety concepts. Depending on the different safety perspectives, the sensitivities and interactions show different results. However, there is a consensus result that the good sensor system provides a higher perception of safety regardless of the types of GNC parameters. This sensitivity and interaction analysis allows one to characterize the impact of the sensor system and GNC parameters.

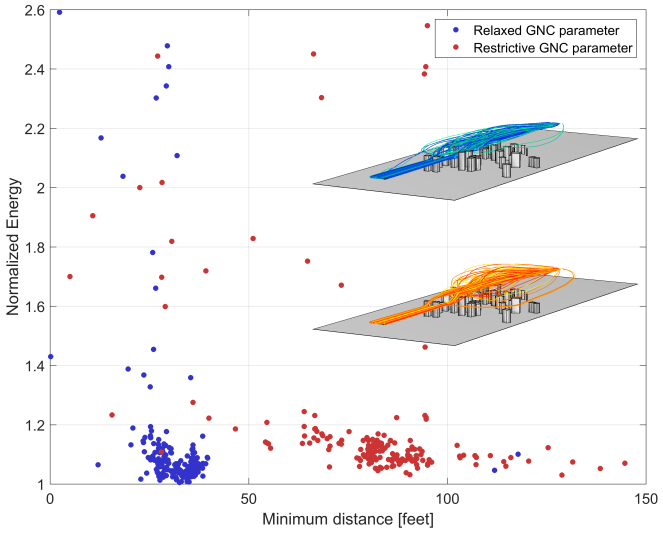



**Figure 95:** Experiment results (Fixed poor sensor and varied GNC performance)

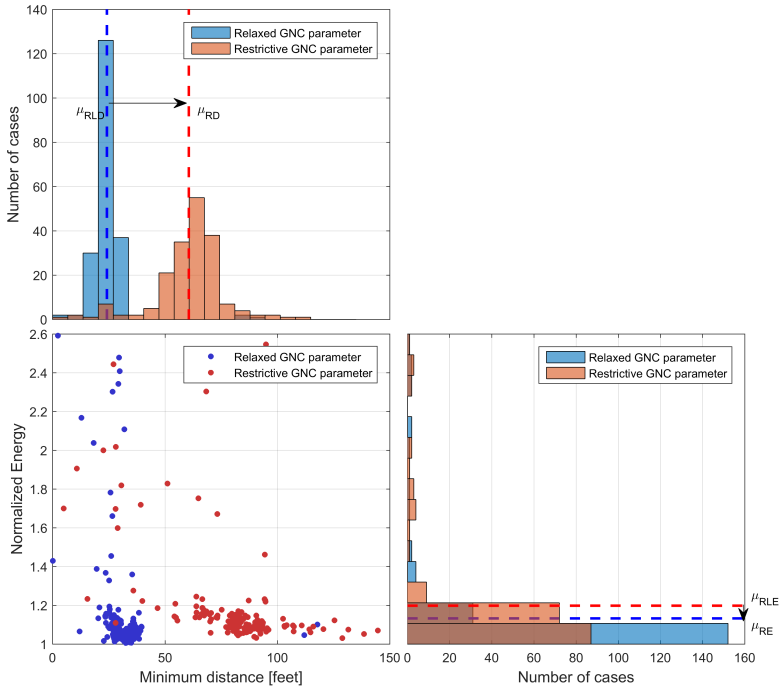



**Figure 96:** Distribution analysis (Fixed poor sensor and varied GNC performance)

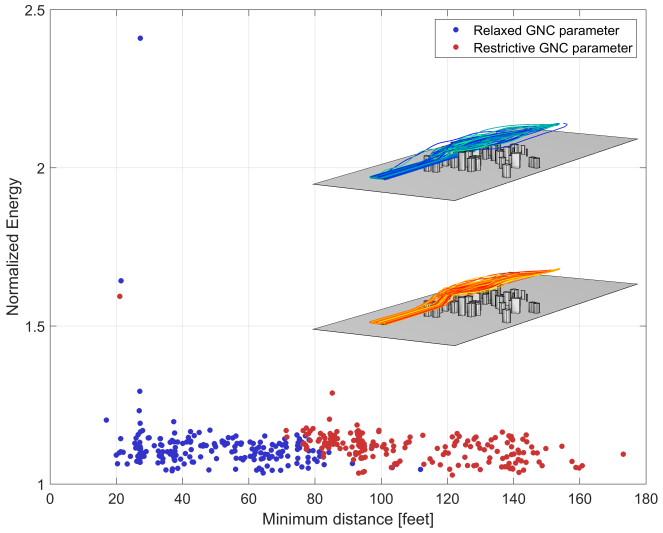



**Figure 97:** Experiment results (Fixed good sensor and varied GNC performance)

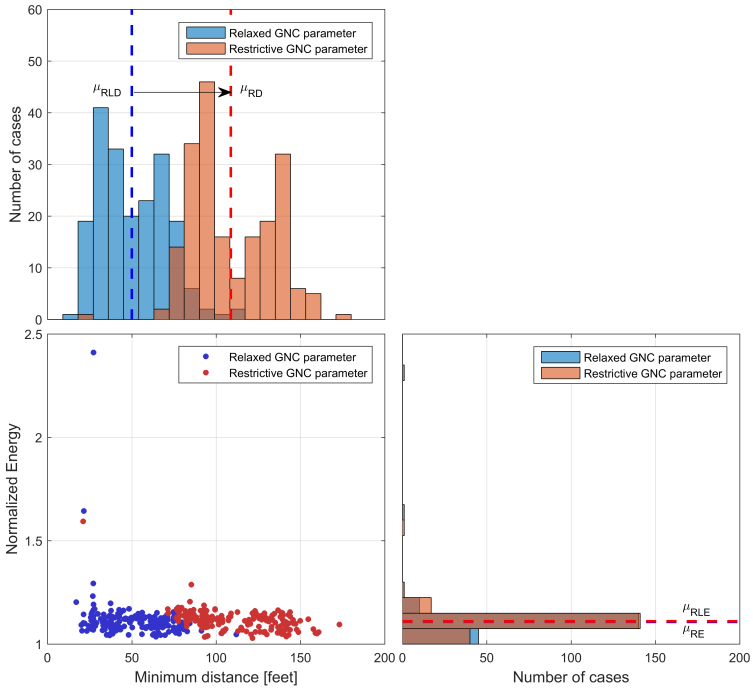



**Figure 98:** Distribution analysis (Fixed good sensor and varied GNC performance)

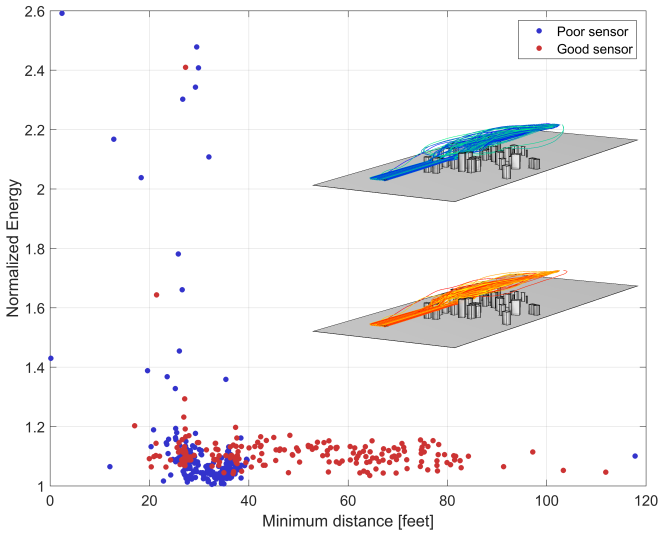



**Figure 99:** Experiment results (Fixed relaxed GNC and varied sensor performance)

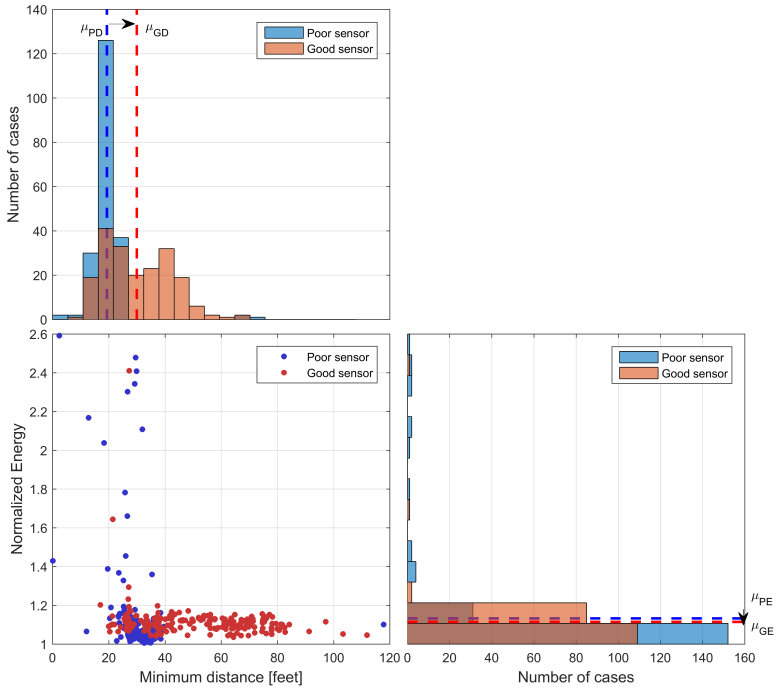



**Figure 100:** Distribution analysis (Fixed relaxed GNC and varied sensor performance)

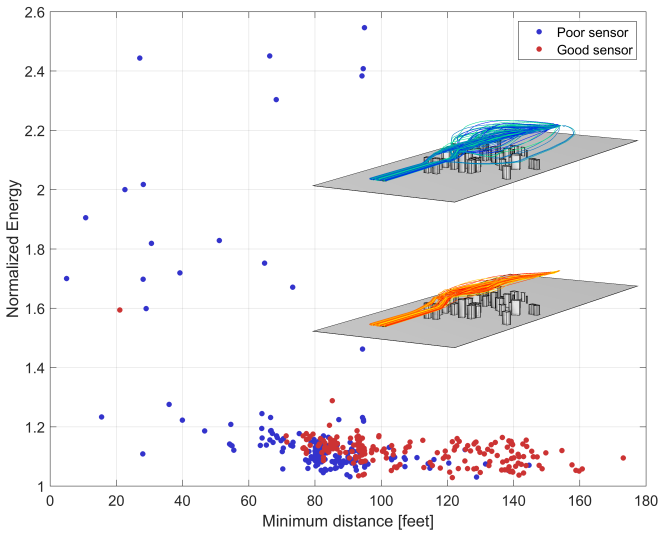



**Figure 101:** Experiment results (Fixed restrictive GNC and varied sensor performance)

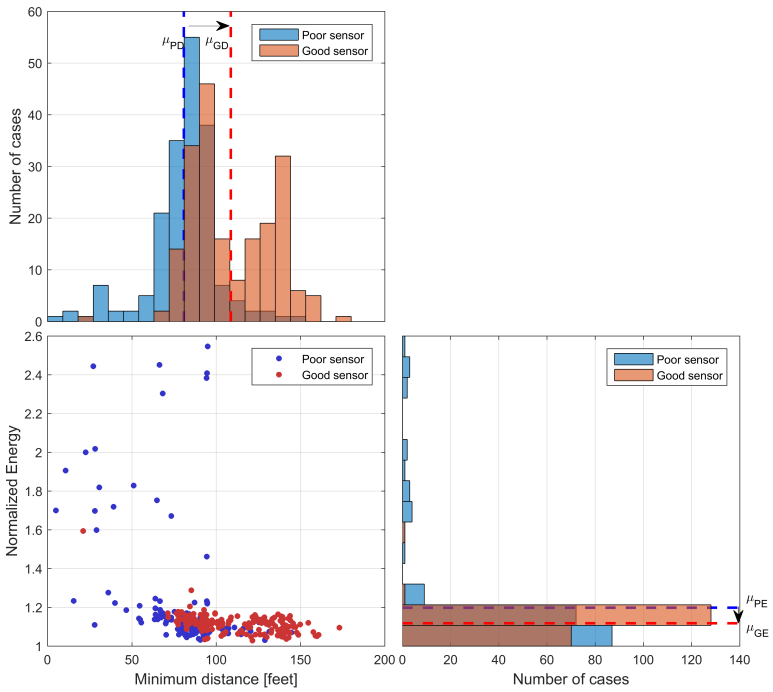
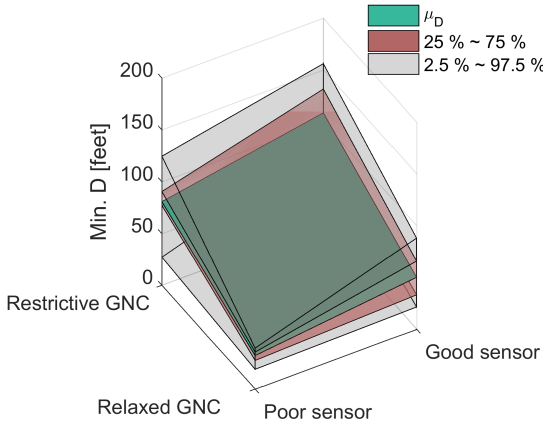
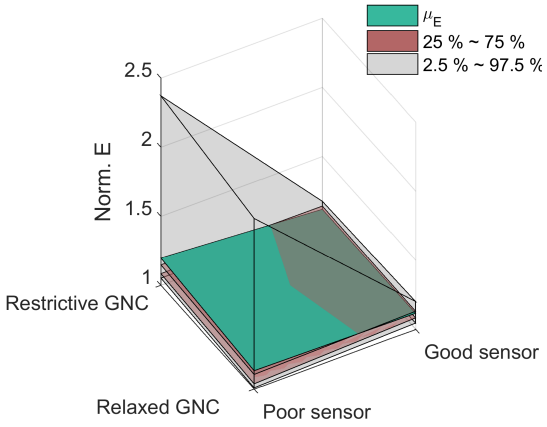


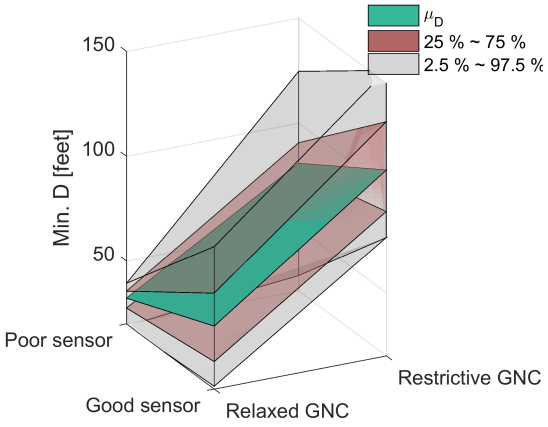

**Figure 102:** Distribution analysis (Fixed restrictive GNC and varied sensor performance)



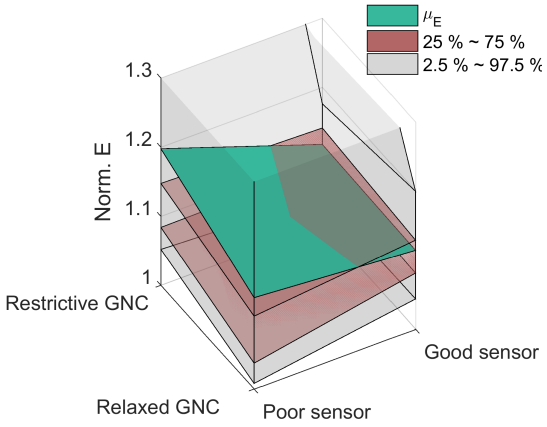

(a) Contour surface plot of the minimum distance




(b) Contour surface plot of the energy consumption




(c) Contour surface plot of the minimum distance (Zoom In)




(d) Contour surface plot of the energy consumption (Zoom In)

**Figure 103:** Contour surface plot

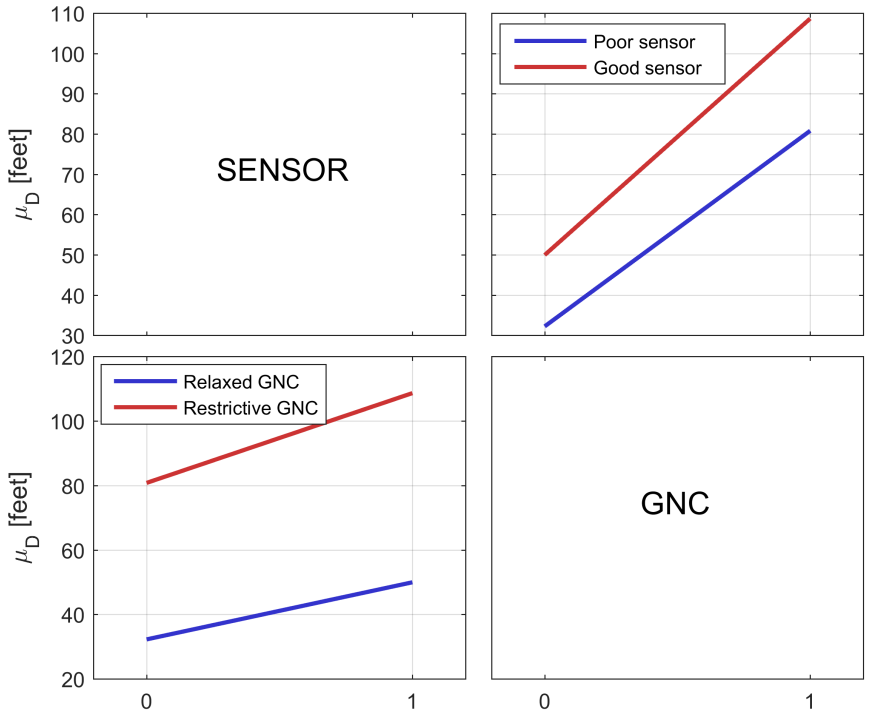



**Figure 104:** Interaction profiles (Risk nominal) of the minimum distance

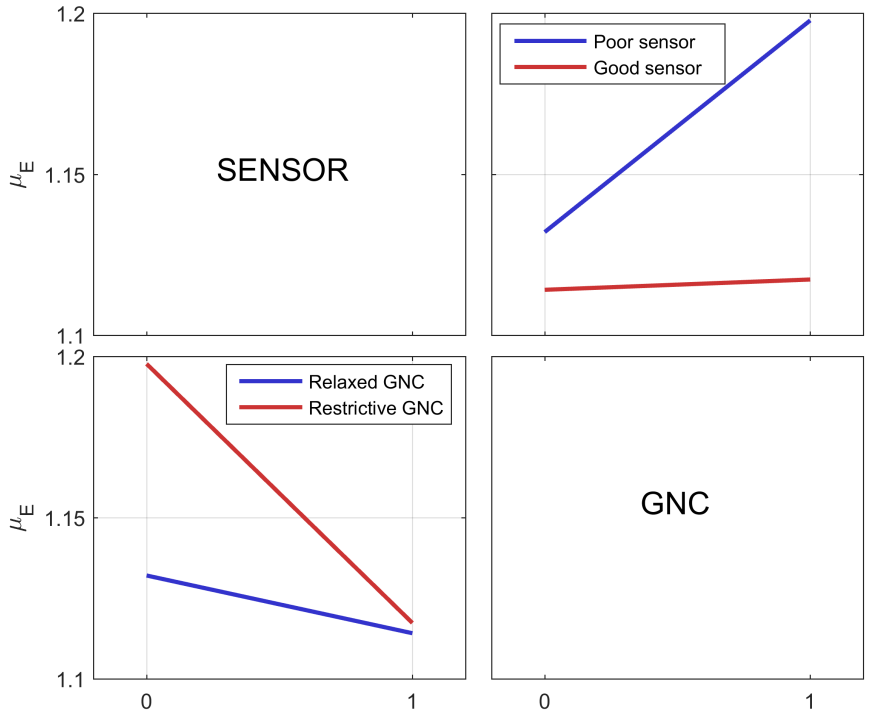



**Figure 105:** Interaction profiles (Risk nominal) of the energy consumption

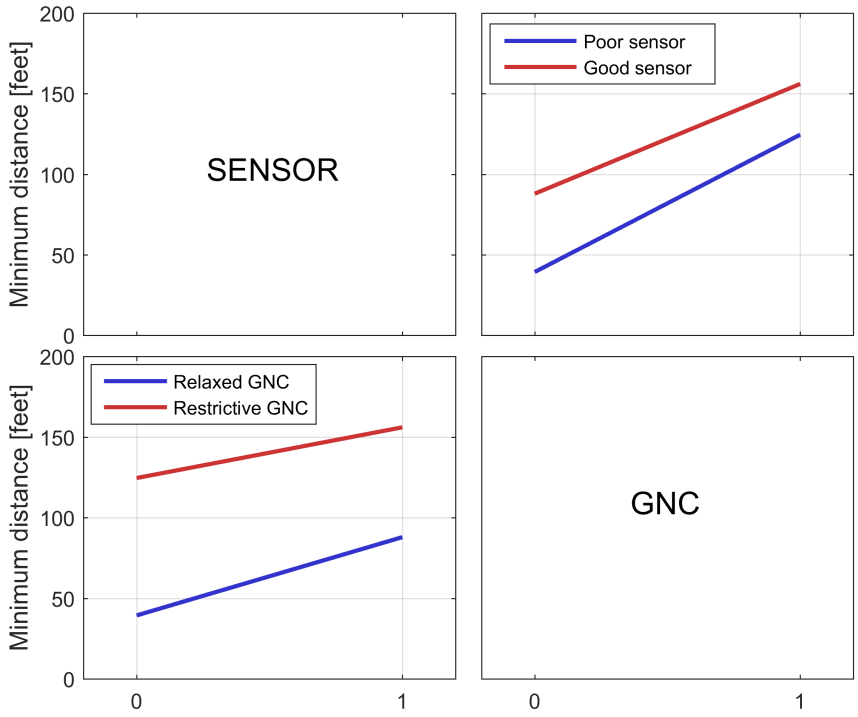



**Figure 106:** Interaction profiles (Risk taker) of the minimum distance

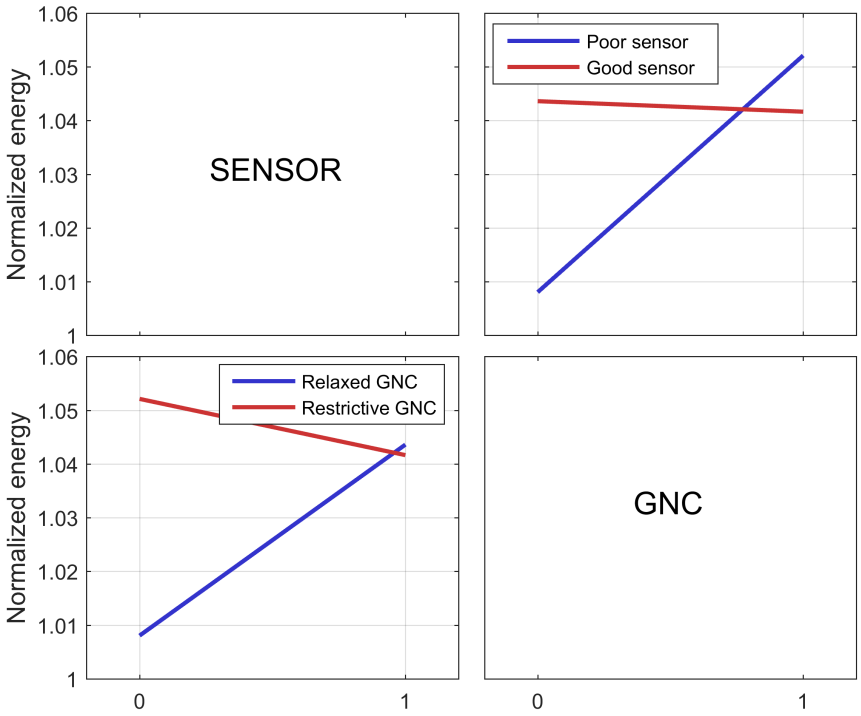



**Figure 107:** Interaction profiles (Risk taker) of the energy consumption

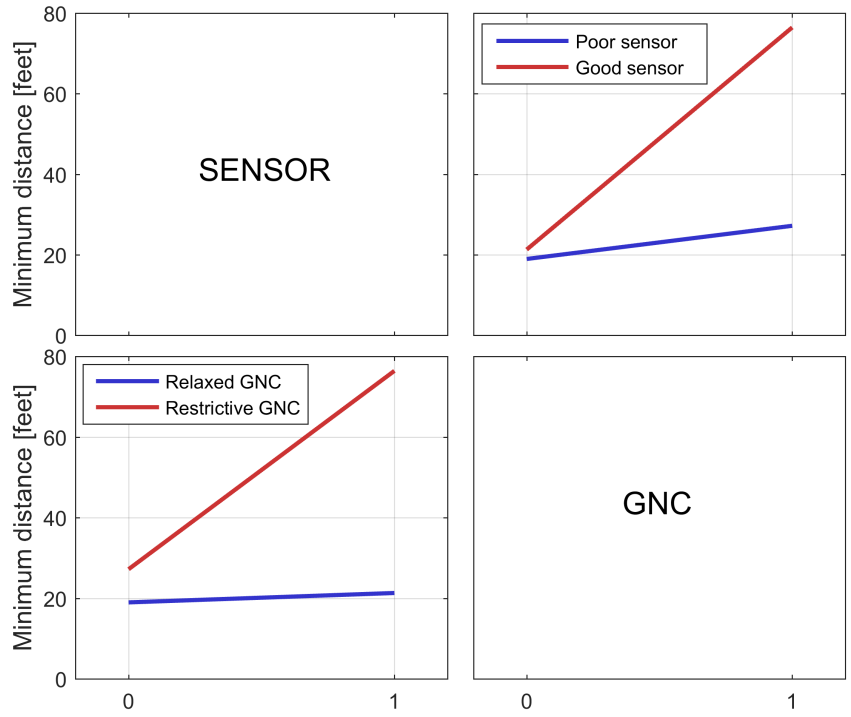



**Figure 108:** Interaction profiles (Risk averse) of the minimum distance

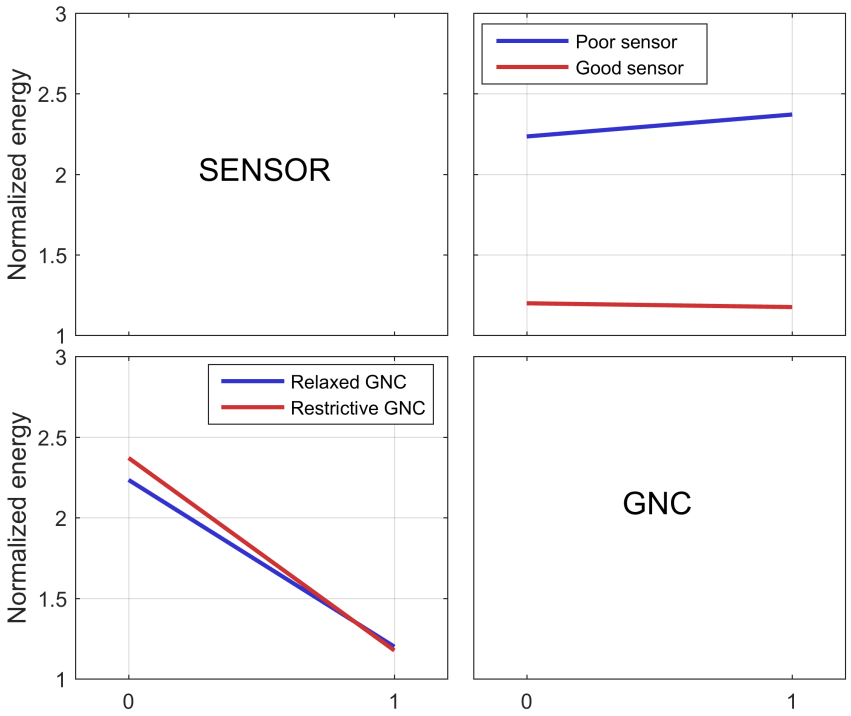



**Figure 109:** Interaction profiles (Risk averse) of the energy consumption

## *6.6 Conclusion*

In this chapter, the second objective, "to quantitatively characterize collision avoidance as a critical element of separation assurance in terms of system behavior across levels of abstraction and multiple disciplines", is addressed by exploring integrated experiments. The integrated experiment is designed to observe interactions/sensitivities of the sensor system and GNC parameters, which are examples of coupling between sub-system components.

To select an urban operating environment, eight different cities are analyzed with respect to the level of the obstacle densities. This density level is characterized by three metrics: two-dimensional airspace ratio, three-dimensional airspace ratio, and the number of buildings per square foot in two-dimensional space. The result of the urban characterization reveals a linear trend. In other words, a city with a higher two- and three-dimensional airspace ratio is likely to have more buildings per square foot. Based on the urban density analysis, San Diego downtown is selected as a representative UAS problem.

The experiment design is a challenging problem because of the high number of design variables that required an infeasible experiments with respect to the computational expense. Therefore, some assumptions such as initial flight conditions are defined, and the initial trajectories are identified by selecting the most challenging trajectories which include more obstacles along the initial trajectory. This results in a total number of 3840 experiments, which is still a high number. To accelerate the computation, the parallel computing technique using the PACE cluster is implemented.

The results of the integrated experiments are analyzed with respect to infeasible/feasible cases. The exploration of the infeasible cases demonstrates that the sensor elevation angle and sensor range are the main drivers to achieve a lower collision probability. For the feasible study, we characterize the avoidance trajectories by the

minimum distance and the energy. The minimum distance implies the perception of safety, and the energy indicates the avoidance trajectory efficiency. The study of the feasible cases reveals that the problem is a highly non-linear problem where conventional sensitivity/interaction analysis approaches are not applicable. Therefore, the partition analysis is employed and analyzed with regard to three aspects: risk taker, risk nominal, risk averse. The result of the partition analysis illustrates that the safe distance is a crucial variable to increase the perception of safety and the energy efficiency.

A new experiment with the reduced number of variables and repeated experiments is designed to extract a more statistically meaningful trend. The reduced number of variables is aimed at executing more repeated experiments with limited computing resources. The experiment results are also analyzed according to three different perspectives, risk taker, risk nominal, and risk averse. As a result, the analysis shows that to improve the perception of safety, a more restrictive GNC parameter is required, and that more energy efficient avoidance routes require higher sensor performance.

In conclusion, this chapter addresses the integrated modeling and simulation environment, introduces potential scenarios, and explores a canonical problem to answer the main research questions that are the second objective of this thesis.

# CHAPTER VII

# CONCLUSION AND FUTURE WORK

## *7.1 Summary of thesis objectives and contributions*

This dissertation provides insights on a new flight simulation environment to address the integration of unmanned aircraft systems into the national airspace system. Specifically, the new flight simulation environment developed in this thesis enables the efficient examination of collision avoidance during fully autonomous flight within an urban environment populated with fixed obstacles for which there is not a surveillance data source. The new flight simulation environment provides improvements over the existing conventional flight simulation scheme which typically includes, for high fidelity simulations, a high order flight dynamics model, a filtering technique, and a detailed sensor uncertainty model. This high fidelity model degrades the computational efficiency that limits fast-simulation environment. On the other hand, when the conventional flight simulation environment has low fidelity, it entails trajectory based on the simulation environment with a point mass assumption that cannot capture coupling that may exist between a system, its subsystems, and the operational system of systems.. To capture these important coupling effects, this dissertation defines two main objectives:

- OBJECTIVE 1: to study and develop improvements in modeling and simulation of fully integrated UAS to address current gaps and enable systems analysis across levels of abstraction and multiple disciplines
- OBJECTIVE 2: to quantitatively characterize collision avoidance as a critical element of separation assurance in terms of system behavior across levels of abstraction and multiple disciplines

To address the first objective, this thesis introduces improvements to the existing modeling and simulation environment paradigm in three respects: aircraft controller, obstacle avoidance algorithm in the guidance, navigation, and control, and realistic urban modeling area.

In the aircraft controller, the statistical gain-scheduling method is introduced to improve the conventional gain-scheduling techniques with respect to the computational efficiency. In the conventional gain-scheduling technique, the mechanism of the online scheduling is computationally unfavorable. To improve this online computational issue, the thesis suggests the surrogate model based gain-scheduling method. The hypotheses in this regard are as follows:

- HYPOTHESIS 1) The proposed statistic gain-scheduling method enables computation runtime improvements without a loss of accuracy or fidelity and also has a comparable stability performance

- HYPOTHESIS 2) The stability (gain-margin and phase margin) of the gain-scheduling using surrogate modeling will be as good as the stability of the conventional gain-scheduling methods using nearest neighbor interpolant and linear interpolant

To demonstrate the first hypothesis, this thesis measures offline and online computational times. The experiment results show that the proposed statistical gain-scheduling method minimally increases offline computational time but improves the online computational time significantly. To test the second hypothesis, the deviation of the multivariate gain and phase margin analysis is measured and compared with other conventional scheduling methods. The experiment results show that the proposed gain-scheduling method provides the smallest deviation from the optimal solution. These two results are evidence in support of the two hypotheses to improve the conventional aircraft control scheme.

In the obstacle avoidance algorithm, four different optimal collision avoidance algorithms are introduced to improve the benchmark collision avoidance algorithm. The benchmark case may have a high trajectory cost when the next waypoint is not considered in the trajectory optimization formulation. The formulation of the trajectory optimization should produce a computationally fast solution to improve the entire simulation performance. To meet those requirements (computation time, optimal trajectory cost), the four different optimal collision avoidance algorithms (Proposition 1 − Proposition 4) are introduced with the multi-phase optimal trajectory formulation and different obstacle avoidance constraints. The performance of the four obstacle avoidance algorithms depends on the weighting assigned to computational cost ($W1$) and trajectory cost ($W2$) respectively. The following hypotheses are formulated based on the sample case studies.

- HYPOTHESIS 3) Waypoint based collision avoidance problem (SCAA-3) will be a dominant solution with respect to the overall cost function when the weight of $W1$ has low value.

- HYPOTHESIS 4) One-dimensional constraints (SCAA-1, SCAA-2) will improve the performance of the collision avoidance with respect to the overall cost function with the medium value of $W1$.

- HYPOTHESIS 5) Two-dimensional constraints (PNORM) will improve the performance with respect to the overall cost function with high $W1$ value.

The experiment results reveal that the SCAA-3 collision avoidance inequality yields outstanding performance when $W1$ has lower value. The SCAA-1 and SCAA-2 collision avoidance inequalities have better performance when the weight $W1$ has a middle value. When the $W1$ has a high value, the performance of the PNORM approach is improved, but it is not the best method because of the iteration limitation in the trajectory optimization function.

The hybrid collision avoidance method using a machine learning technique was also formulated to improve the computational efficiency as well as the optimal cost. The hybrid collision avoidance algorithm selects the best collision avoidance method among the formulated alternatives through a multi-class classification algorithm. The demonstration results show that the proposed hybrid method has better performance than other conventional collision avoidance algorithms.

For the multi-obstacle avoidance problem, this thesis proposed a two-layer obstacle avoidance algorithm to improve the drawback of the conventional trajectory optimization where the inability to discern to adjacent obstacles results in energy inefficient trajectories. The trajectory energy benefits of the proposed are stated in the following hypothesis:

- HYPOTHESIS 6) Two-layer collision-free obstacle avoidance algorithm that includes global- and local- path optimization structure generates more energy efficient avoidance trajectory when facing multiple obstacles

The two-layer structure features a global- and a local- trajectory optimization process. In the global optimization, a UAV detects and identifies the number of objects based on the distance-based clustering algorithm, and then specifies a potential upcoming threat. During the local optimization, this technique solves the trajectory optimization algorithm for the resolved obstacle that minimizes an energy consumption. The experiment results demonstrate that the proposed two-layer collision avoidance algorithm generates more energy efficient avoidance trajectory.

In the realistic urban modeling, this thesis proposes a rapid, data-driven, and grid-based urban modeling methodology. For the exploration of the UAS obstacle avoidance problem in an urban operation, the rapid and realistic urban environment is a key component because of diverse urban mission scenarios. The proposed methodology includes six steps: collection of LiDAR data, data resampling, identification of

building clusters, principal component analysis, grid generation, and construction of a building. The proposed methodology was successfully demonstrated by constructing eight different urban models with two different levels of fidelity. This environment construction also allows for additional analysis of urban environments in terms of attributes relevant to the collision avoidance problem, such as obstacle density and average inter-obstacle distance.

The last part of this thesis is a system-of-systems experiment that integrates all the capabilities and improvements above mentioned, and addresses the second objective. A sensitivity and interaction analysis between a sensor system and a GNC system for fixed-obstacle collision avoidance comprises the core application of this experimental demonstration. Two key metrics are measured: the energy required for the collision avoidance trajectory (normalized by the obstacle-free trajectory energy), and the proximity to fixed obstacles as a proxy for perceived safety. The initial experiment results show a highly non-linear response that is not able to present accurate sensitivities and interactions pursuant of traditional techniques based on analysis of variance. Never the less, the sensor and GNC characteristics associated with favorable performance and unfeasible designs (collision cases) are delineated using discriminant analysis. In addition, to further resolve the sensitivities and interactions relevant to energy efficiency and proximity to obstacles, a new experiment was designed by reducing the number of variables and repeating the experiments with small perturbations in the initial conditions of the trajectories. The redesigned experiment leads to the sensitivities and interaction analyses through a response surface model approach. The results of the sensitivity/interaction analyses characterize the energy consumption and the perception of a safety with respect to different three risk aspects: risk taker, risk nominal, and risk averse.

The formulated UAV flight simulation environment and analysis methodology can provide the rapid system of systems analysis in the collision avoidance problem

in an urban operation. Many UAV flight simulation environments have been developed to address different problems associated with a collision avoidance problem in an urban environment. Stastny et al. introduces UAV collision avoidance simulation environment in a realistic urban environment that includes six DOF dynamics models, adaptive neural network for a UAV controller, potential field algorithm to generate collision avoidance trajectory against moving and fixed obstacles [137][135]. However, this paper does not provide enough details of the urban model and sensor model. Shanmugavel et al. simulates collision avoidance problem in a three dimensional fixed obstacle environment and uses Dubins path algorithm to avoid obstacles [133], but this paper is limited to the path planning problem to avoid obstacle. Orr et al. have developed the framework for UAV collision avoidance algorithm in a realistic urban environment [116]. The vehicle model has high fidelity six DOF models and a realistic urban model that is the Fort Benning Georgia McKenna Military Operations on Urban Terrain (MOUT) Site. This model includes a wind gust model and also assesses the impact of the wind gust in the urban model. However, this flight simulation model does not entail collision avoidance algorithm and sensor model. Cybyk et al. have developed the UAV simulation environment with a wind gust model using CFD analysis and six DOF high fidelity models [31], but this simulation environment does not include a collision avoidance algorithm to avoid building obstacles. Therefore, the existing simulation environments in the literatures do not provide enough details about a system and subsystem level component. To fill the lacking details, the introduced UAV flight simulation environment in this dissertation explains the modeling information of system and subsystem components such as the statistical gain-scheduling method, the guidance and navigation, the sensor model and the urban modeling methodology that enables us to explore the interactions and sensitivities with different levels of abstraction.

## *7.2 Recommendations for further research*

To analyze and explore the UAS integration problem in an urban environment within the scope of this dissertation, flight dynamics, an aircraft controller, collision avoidance algorithm, and realistic urban models were developed based on conventional algorithms/models. The developed algorithms and models are computationally attractive solutions that enable us to examine diverse UASNAS problems. There are, however, several key issues or needed researches on modeling of system/subsystem components, designing integrated experiment and experiment scenarios. The following researches are the recommended future researches.

- The validation of the vehicle parameters and controller is the further research topic. The current vehicle dynamics model and controller is not determined by the actual flight data because of the lack of the validation process. Therefore, the avoidance behavior of a simulated UAV model may be different characteristics from a actual UAV platform. Thus, future work must consider the validation process to achieve more realistic vehicle behavior.

- The generalization of a sensor model is one of the needed research fields. The actual LiDAR sensor is a probabilistic model that delivers different performance depending on the distance and azimuth/elevation angle of a detected object. To achieve the even sensor performance in the entire sensor operating region, a multi-sensor fusion technique is commonly applied. However, this thesis assumed that a LiDAR sensor model is deterministic that does not consider any probabilistic models or a sensor fusion concept. To make the sensor model more realistic, it requires a probabilistic sensor model, or the same deterministic model that can represent a sensor fusion concept.

- In the hybrid collision avoidance algorithm, a neural network and ensemble neural network are employed to solve a multi-class classification problem. As

an alternative, a fuzzy logic controller is also implementable to specify the best strategy. d system would provide better experiments results and more general trends.

- The impact of various obstacle avoidance techniques is also one of the further research topics. Depending on the different obstacle avoidance algorithms, the performance of the collision avoidance could be varied. For instance, some algorithms may have outstanding performance in a sparse urban environment, but others may have better performance in a dense urban environment. Therefore, multiple collision avoidance algorithms should be studies to fully explore the UASNAS collision avoidance problem.

- The analysis of the impact of different types of UAS platforms should be more explored. This research only focused on exploring the interaction of sensor capabilities and a collision avoidance algorithm. However, the collision avoidance performance also depends on the vehicle maneuverability. Hence, the integrated experiment with different types of vehicles is a key UAS research area.

- The exploration of an urban environment with a different density level is also a further research topic. The different urban environment with different obstacle configurations may affect an obstacle avoidance capability depending on different sensor systems or UAV platforms. In the UAV design perspectives, this research enables an aircraft designer to design sensors and platform systems to fly safely in a given urban environment without any collisions. In the regulation/operational perspectives, this research can also provide the insights of the interaction about the UAV platforms and sensor systems that enable decision makers to understand the relationship between an urban density and different system/subsystem components.

- The characterization of UAS collision avoidance trajectories in urban environments using chaos theory is a potential research topic. The results of the system of systems level experiment show that the avoidance trajectories in an urban operation present highly non-linear behaviors. These non-linear behavior leads to a challenging problem for the global sensitivity and interaction analysis. Therefore, the chaos theory whereby the simulation environment here developed may help infer the underlying vector field from which key statistical properties may be extracted using chaotic systems methods.

- In the urban operation, a UAV must maintain flight stability and maneuverability under the variation of weather conditions. Because of geometrically complex terrain and urban environment, the wind gust can significantly influence the performance of flight stability or maneuverability. Orr et al. have researched the framework for the wind effect of the path planning for UAV operation in an urban environment and observe the impact of the light breeze [116]. The result clearly illustrates that the tracking performance of the UAV is significantly affected by the wind gust profile. Therefore, the study of the wind gust impact in a complex urban area is critical to reduce a collision risk. In order to address the wind gust impact in an urban area, the time-varying airflow environment must be included to assess the vehicle stability and maneuver performance. Cybyk et al. capture 3D unsteady airflow in an urban environment through using CFD analysis from a Large Eddy Simulation model [31]. Using this 3D airflow model, this paper constructs a UAV flight simulation model to explore coupled interactions between the UAV maneuverability characteristic and the wind gust influence. This paper also analyzes a mission feasibility for the intelligence, surveillance and reconnaissance (ISR) mission performance under the wind gust. The 3D unsteady wind gust model can be applied to the interaction/sensitivity analysis between systems, subsystems, or a system and

a subsystem using the formulated simulation model shown in Figure 111. The wind gust profile can be computed by the Large Eddy simulation model using the urban environment. The result of the wind gust profile produces force disturbances to the UAV dynamics model. This updated UAV flight simulation environment enables us to explore the impact of the wind gust profile.

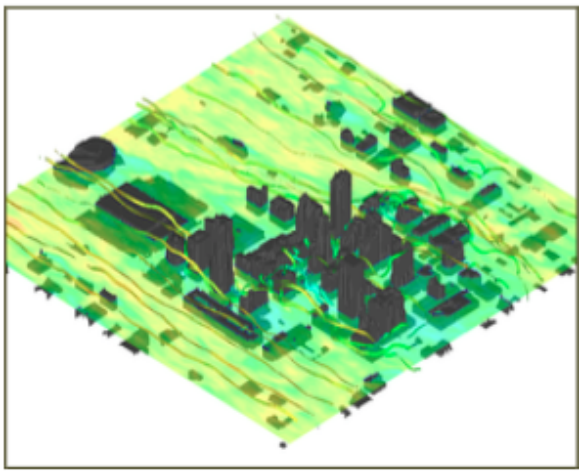


**Figure 110:** Wind gust profile of Oklahoma city [31]

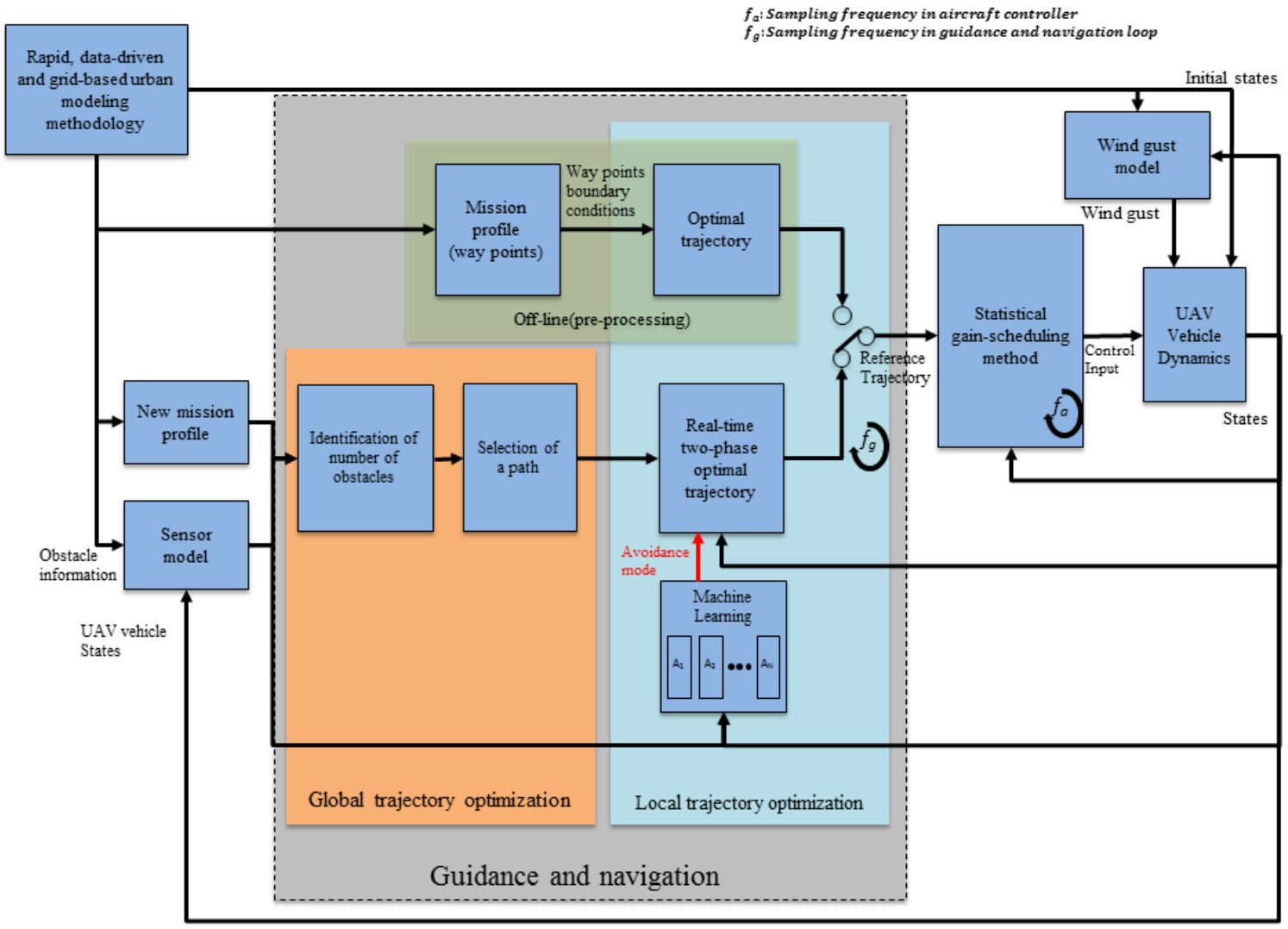



**Figure 111:** Block diagram of UAV flight simulation with a wind gust model

- The thesis addresses collision avoidance problems against ground fixed obstacles. In the urban operation, an UAV may meet fixed obstacles as well as moving obstacles such as an unmanned aircraft system, a bird, and a commercial airplane. To explore more realistic urban operation problem, moving and fixed obstacles must be studied. In order to incorporate moving and fixed obstacles, the optimal collision avoidance algorithm should be reformulated since the current formulated collision avoidance algorithm does not consider the obstacle's moving direction.

# APPENDIX A

# MULTI-CLASS CLASSIFICATION LEARNING ALGORITHM

## *A.1 Neural network*

To shift a traditional information process based on a rule-based approach motivated from biological nervous systems, the neural network (i.e, artificial neural network) has been introduced. The neural network has a highly complex structure with a large number of elements called neurons or node. These nodes are interconnected and produce approximation functions based on input and output data. Compared to other machine-learning techniques such as linear regression, a support vector machine and a decision tree, these approximation functions yield highly accurate predictions. Therefore, neural network techniques have implemented in diverse fields such as pattern recognition, medical domains for diagnostic, prognostic tasks, and stock market forecasts [22][42][111]. In the aerospace community, the artificial neural network technique has been applied for aircraft system identification [86]. The neural network is also powerful to solve a classification problem. Therefore, the neural network for a classification problem is a suitable technique for a multi-class classification of the introduced hybrid collision algorithm. This section introduces the mathematical formulation of the neural network.

It is assumed that the labeled data are $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2),\ \ldots\ , (\mathbf{x}_m, y_m)\}$. $\mathbf{x}_i$ is the $i$th input vector in $n$ dimensional space ($\mathbf{x} \in \mathbb{R}^{(m \times n)}$). $y_i$ is the class information of the $i$th input vector $\mathbf{x}$, which is ($\mathbb{R}^{(m \times 1)}$). We also assume that $y$ has $c$ classes. Figure 112 illustrates a typical neural network structure with $k$ layers. The neural network structure with $k$ layers includes an input layer, hidden layers, and an output

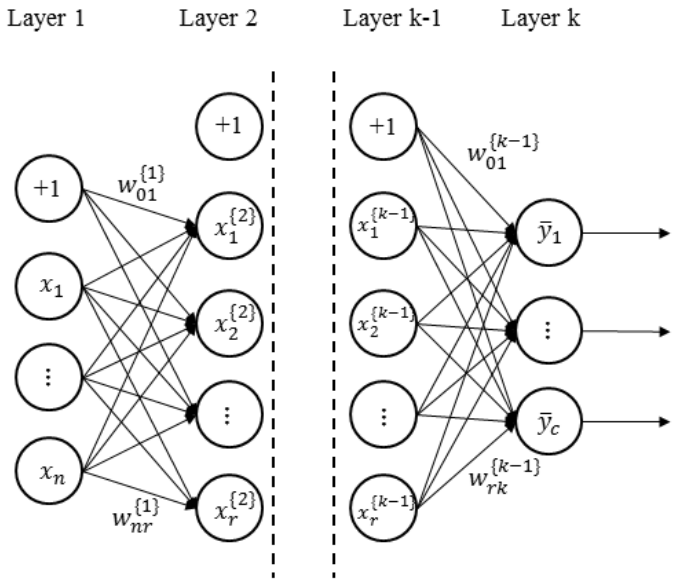



**Figure 112:** Neural Network structure

layer. The first layer of the neural network is the input layer, which has $m + 1$ perceptrons with a constant term for a bias effect. The hidden layers are layers from the second to the $k - 1$th layers in the neural network structure. Each layer in the hidden structure has $r$ perceptrons and a constant term. The output layer is the last $k$th layer, which provides prediction results. The outputs of each layer can be computed from the perceptrons' information of the previous layer, which involves weights $\mathbf{w}$ and a non-linear function.

To optimize this neural network structure, a typical optimization technique is a feedforward algorithm that computes perceptron information based on the previous layer information: weights $\mathbf{w}$, and perceptrons $\mathbf{x}^{\{k\}}$. This computation process is continuously repeated until the outcomes of the last perceptrons are specified. That is, perceptron results, $\mathbf{x}^{\{n\}}$, in the $n$th layer are defined by a weighted summation of the previous perceptrons, $\mathbf{x}^{\{n-1\}}$. Then we evaluate a non-linear function, $h_w(\cdot)$, and an activation function. Notable activation functions are the hyperbolic tangent function, the sigmoid function, the hard limiter function, and the ramp function.

These processes are repeated until the perceptrons on the last layer are estimated. The mathematical expression of the feedforward algorithm is as follows,

$$\begin{aligned}
\mathbf{x}^{\{1\}} &= \mathbf{x} \\
\mathbf{x}^{\{2\}} &= h_w(\mathbf{w}^{\{1\}T}\mathbf{x}) \\
\mathbf{x}^{\{3\}} &= h_w(\mathbf{w}^{\{2\}T}\mathbf{x}^{\{2\}}) \\
&\vdots \\
\mathbf{x}^{\{k-1\}} &= h_w(\mathbf{w}^{\{k-2\}T}\mathbf{x}^{\{k-2\}}) \\
\bar{\mathbf{y}} = \mathbf{x}^{\{k\}} &= h_w(\mathbf{w}^{\{k-1\}T}\mathbf{x}^{\{k-1\}}),
\end{aligned} \tag{136}$$

where $\mathbf{w}^{\{n\}}$ is a weight vector on the $n$th layer. The activation function $h_w$ is assumed to be the sigmoid function:

$$h_w(\mathbf{x}) = \frac{1}{1 + e^{-w^T\mathbf{x}}} \tag{137}$$

For the optimization of the neural network output $\bar{y}$, we define an objective function that minimizes the mean-square error:

$$E = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \bar{y}_i)^2. \tag{138}$$

This neural network prediction model has an overfitting problem. For instance, when the number of training data is small, this cost function is likely to lead to an overfitted prediction model. To mitigate this overfitting problem, we can employ various techniques: early stopping during the optimization process, curvature-driven smoothing, and averaging over several plausible networks through the Bayesian approach [22] [43]. Among several techniques for the overfitting issue, a popular approach is a regularization method, which includes an additional penalty function in the cost function [74]. The error cost function with the regularization term can be

$$E = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \bar{y}_i)^2 + \frac{\lambda_{NN}}{2m}\mathbf{w}^T\mathbf{w}, \tag{139}$$

where the first term is the prediction error, and the second term is the regularization term. $\bar{y}_i$ is the prediction results, $\lambda_{NN}$ is the weight for the regularization factor (i.e.,

hyper-parameter), and $\mathbf{w}$ is the weight vector in the neural network. Based on the error cost function, we can reconstruct an objective function for a neural-network optimization that minimizes the error function:

$$Min \;\; E = Min \;\; \frac{1}{2m}\sum_{i=1}^{m}(y_i - \bar{y}_i)^2 + \frac{\lambda_{NN}}{2m}\mathbf{w}^T\mathbf{w} \tag{140}$$

For the optimization, gradient information can be computed by a mathematically simple and computationally efficient back-propagation method that provides gradient information [91]. Gradient information with respect to weights in the neural network by the backpropagation method is

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{w}^{\{k-1\}}} &= \frac{\partial E}{\partial h_w(\mathbf{w}^{\{k-1\}T}\mathbf{x}^{\{k-1\}})}\frac{\partial h_w(\mathbf{w}^{\{k-1\}T}\mathbf{x}^{\{k-1\}})}{\partial \mathbf{w}^{\{k-1\}}} \\
\frac{\partial E}{\partial \mathbf{w}^{\{k-2\}}} &= \frac{\partial E}{\partial h_w(\mathbf{w}^{\{k-1\}T}\mathbf{x}^{\{k-1\}})}\frac{\partial h_w(\mathbf{w}^{\{k-1\}T}\mathbf{x}^{\{k-1\}})}{\partial h_w(\mathbf{w}^{\{k-2\}T}\mathbf{x}^{\{k-2\}})}\frac{\partial h_w(\mathbf{w}^{\{k-2\}T}\mathbf{x}^{\{k-2\}})}{\partial \mathbf{w}^{\{k-2\}}} \\
&\vdots \\
\frac{\partial E}{\partial \mathbf{w}^{\{1\}}} &= \frac{\partial E}{\partial h_w(\mathbf{w}^{\{k-1\}T}\mathbf{x}^{\{k-1\}})}\frac{\partial h_w(\mathbf{w}^{\{k-1\}T}\mathbf{x}^{\{k-1\}})}{\partial h_w(\mathbf{w}^{\{k-2\}T}\mathbf{x}^{\{k-2\}})}\cdots\frac{\partial h_w(\mathbf{w}^{\{1\}T}\mathbf{x}^{\{1\}})}{\partial \mathbf{w}^{\{1\}}},
\end{aligned} \tag{141}$$

For example, the gradient of the $(k-1)$th layer of the error objective function with the regularization term is

$$\begin{aligned}
\frac{\partial E}{\partial w_0^{\{k-1\}}} &= \frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y}_i)x_i^{\{k-1\}}, && r = 0 \\
\frac{\partial E}{\partial w_r^{\{k-1\}}} &= \frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y}_i)x_i^{\{k-1\}} + \frac{\lambda}{2m}\mathbf{w}_r^{\{k-1\}}, && r \geq 1.
\end{aligned} \tag{142}$$

When $r$ in the hidden layer is zero, a perceptron as a constant term does not include a regularization term, but other perceptrons ($r \neq 0$) in the hidden layer do. This gradient information by the backpropagation technique can be applied to the optimization process of the neural network.

For a comparative study of the two neural networks without regularization and with regularization, we perform sample case studies for a multi-class classification problem. A data set of the sample problem has random numbers with three classes

that are assumed to be a two-dimensional Gaussian distribution:

$$P(x|\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)) \tag{143}$$

$\Sigma$ is a covariance matrix, $\mu$ represents mean values, and $n$ is the number of dimensions ($n = 2$). Through these Gaussian assumptions, we generate three-class training data sets summarized in Table 31.

**Table 31:** Parameters of Gaussian random numbers with three classes

| Class | Variable name | Value |
|---|---|---|
| 1 | mean | $\mu_1 = \begin{bmatrix} -3 \\ -2 \end{bmatrix}$ |
| | covariance | $\Sigma_1 = \begin{bmatrix} 6 & 3 \\ 3 & 6 \end{bmatrix}$ |
| 2 | mean | $\mu_2 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$ |
| | covariance | $\Sigma_2 = \begin{bmatrix} 5 & 2 \\ 2 & 4 \end{bmatrix}$ |
| 3 | mean | $\mu_3 = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$ [2 -3] |
| | covariance | $\Sigma_3 = \begin{bmatrix} 3 & 2 \\ 2 & 5 \end{bmatrix}$ |

Training data for the optimization process comprise 80 % of the data, and test data for the validation process comprise 20 % of the data. Figure 113 shows the training data set with three classes from the definition of three Gaussian random numbers with two dimensions. We assume that in a neural network prediction model, the number of hidden layers is one, and the number of perceptrons (nodes) on the hidden layer is 100. Based on this neural network structure, we optimize two neural networks without regularization and with regularization using the backpropagation method. We assumed that the regularization parameter of the neural network with regularization is one. Figure 114 is the classification results using the two neural networks. The first graph is the neural network without regularization, which has a 73.33 % success rate for the test data. The classification results without the regularization

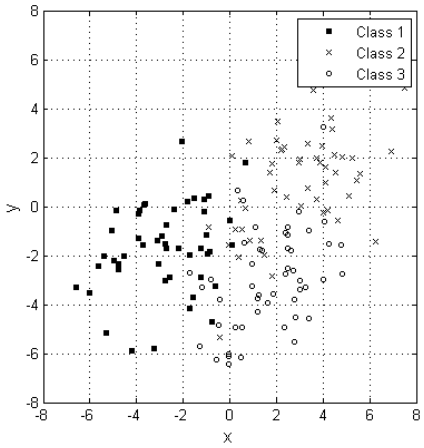
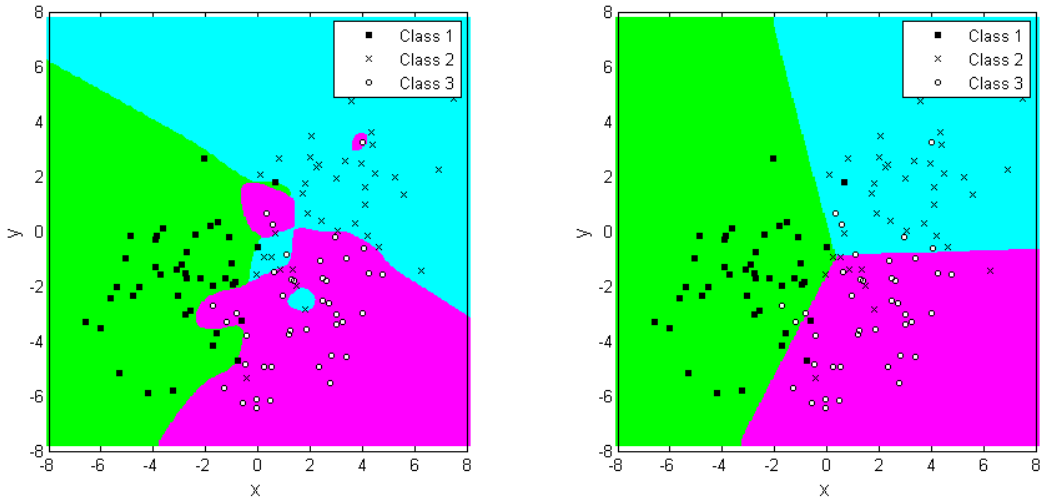



**Figure 113:** Training data

term appears to be overfitted because the borders of the three classifications have a highly non-linear pattern. The second graph presents the results of the neural network with regularization, which shows a 76.66 % prediction success rate for the test data. Because of the regularization term, unlike the neural network model without the regularization term, the training results do not show a non-linear pattern. From these results, we can observe that the neural network with regularization does not lead to an overfitting problem for classification.

## *A.2 Ensemble learning*

Classification algorithm using a neural network has more powerful method to have a prediction model for non-linear classification problems than other learning algorithms such as decision trees and k-nearest neighbors. In addition, constructing a neural network is easier than support vector machine (SVM) [43]. However, when training data includes a small number of samples and highly non-linear classification problems, we might encounter overfitting problems or incomplete training issues [72]. Non-linear classification problems with small training data in high dimension space can easily produce overfitted prediction results. To prevent this overfitting, we can add a regularization parameter with a high regularization weight, but this




(a) Neural network without regularization

(b) Neural network with regularization

**Figure 114:** Comparison results of the neural network without regularization and the neural network with regularization

regularization term with high weights can yield underfitting problems. To solve these overfitting and incompleted training problems, researchers have suggested ensemble learning techniques. The ensemble learning techniques combine weak learners to create a strong learner. That is, these ensemble techniques build multiple prediction models. The multiple prediction models can provide more accurate prediction results through a combination of the prediction results from the multiple weak learners. Therefore, the ensemble learning approach improves generalization performance through the combination of the multiple prediction models. The classical example of the ensemble learning method is a simple average, a majority, and weighted sum of prediction models. The challenge part of the ensemble learning method is training individual neural network and integrating all neural networks to create the ensemble prediction model. According to Islam, training techniques can be generalized as three approaches: independent training, simultaneous training, and sequential training [67].

The independent training method optimizes individual neural networks to minimize a residual error. The optimized individual neural networks are combined to

construct a giant neural network through using weighted sum of each learner or vote system. The drawback of this ensemble technique is a non-existing structure that checks the effects of interactions between neural networks in a training phase. The notable examples of the independent training method are bootstrap aggregation (bagging) and ordinary least square (OLS) method. The bagging technique creates multiple neural network learners based on a randomly selected subset of the training set [26]. The integrated ensemble structure of the bagging technique is built by adding an equal weight to each learner. That is, the final prediction result is resulted from the average of each learner's outcome. However, this bagging technique does not entails the prediction performance of the individual neural network model because the bagging technique only averages the outputs of multiple neural networks to produce the ensemble prediction result without examining the performance of each neural network. In order to handle this issue, Jia. et al [72] introduces OLS technique to reflect the performance of each neural network model in ensemble model. This OLS technique trains multiple individual neural networks. Then, these trained neural networks are integrated through a weighted summation process. These weights for the integrated ensemble model are specified by the least square method. The OLS technique yields better performances than the bagging technique, which averages the outcomes from all neural networks since the OLS technique considers the individual performances of neural networks through the weights.

In the sequential training method, the ensemble structure are sequentially trained to minimize the error cost function. The benefit of this process is to avoid correlation between a new neural network and previously trained neural network. The representative sequential training methods are sequential bagging and boosting [68].

In the independent and sequential training method, the correlation and interaction effects of individual network cannot be included in the ensemble model when we construct the ensemble learning structure. For instance, once all trained neural

network models are exactly same, the ensemble model would not generate a more accurate prediction model than a single neural network. To address this problem, Liu. et al suggested a negative correlation method to design the ensemble network [95] [96] because it trains multiple neural networks simultaneously to achieve better the ensemble neural network model through learning various network parts/aspects. Learning other aspects from different neural network models is implemented through adding a penalty term including a correlation parameter.

In the training perspectives for the three training methods, the sequential training method is not possible to parallelize the training process due to training dependency. This difficulty of the parallelization yields high computational runtime. The simultaneous training method is required to train all design variables that cannot use the training results of a single neural network. In other words, this simultaneous training technique requires high computational resource. Therefore, this paper introduces the details of the independent ensemble learning techniques.

### A.2.1 Neural network ensemble using Ordinary Least Square (OLS)

This section overviews the ensemble learning method using ordinary least square (OLS) proposed by Jia. et al [72]. The OLS ensemble technique yields better results for a classification problem through optimizing multiple neural networks than a bagging ensemble technique. The optimization of multiple neural networks is executed by least square technique to minimize the sum of errors of the individual neural networks. Figure 115 illustrates the typical architecture of the OLS ensemble learning method using neural networks. Sub-classification structure includes multiple weak learners ($NN_1$, $NN_2$, ..., $NN_n$), which are composed by neural networks. The individual neural network is separately trained through the standard optimization process of the neural network using the gradient based approach. In this optimization process, the backprogation algorithm produces a gradient information.

To address more details about OLS ensemble algorithm, we assume we have a data set, $D = \{(\mathbf{x}_1, y_1),\ (\mathbf{x}_2, y_2),\ \ \ldots\ ,\ \ (\mathbf{x}_m, y_m)\}$. $\mathbf{x}_i$ indicates input vectors in $n$ dimension space ($\mathbf{x} \in \mathbb{R}^{(m \times n)}$), and $m$ is the number of training data. $i$ indicates $i$th data point. $y_i$, ($\mathbb{R}^{(m \times 1)}$), is a label information of the $i$th input vector $\mathbf{x}$.

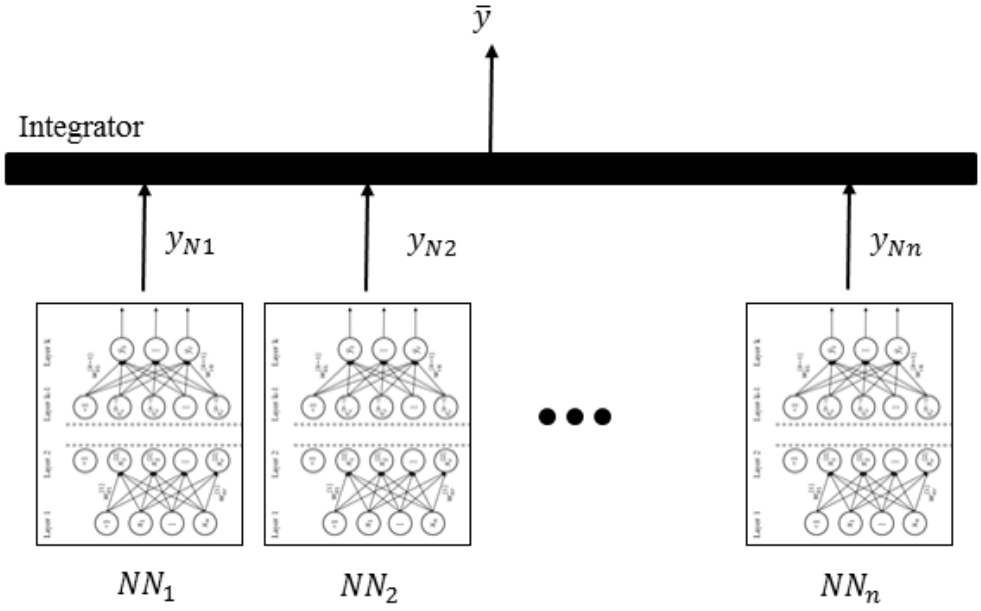



**Figure 115:** Ensemble architecture with neural network

The objective function for the optimization of each neural network is

$$Min \quad E_{Nk} = Min \quad \frac{1}{2m}\sum_{i=1}^{m}(y_{Nk,i} - \bar{y}_i)^2 + \frac{\lambda_{NN}}{2m}\mathbf{w}_{Nk}^T\mathbf{w}_{Nk}, \tag{144}$$

where $E_{Nk}$ is the error function of $k$th neural network, the first term in the right side of the equation is a sum of a residual error, and the second term is a regularization term. $\mathbf{w}_{Nk}$ is a weight vector of $k$th neural network. The standard gradient approach using a backpropagation technique optimizes individual neural network. The outputs from the optimization process are denoted by $Y_{N1}, Y_{N2}, ..., Y_{Nn}$, which is called sub-classifier. The outputs of these sub-classifiers are combined into an ensemble neural network. The ensemble neural network is optimized by weighted least square method to minimize the total residual error from all sub-classifiers. We assume the prediction

model is a function of weights and the outputs of sub-classifiers,

$$\bar{Y} = w_1 Y_{N1} + w_2 Y_{N2} + ... + w_n Y_{Nn}, \tag{145}$$

where $Y_{Nn}$ is the output vector of $n$th neural network and $\bar{Y}$ is the predicted output vector from the sub-classifiers. $w_1$, $w_2$, $\cdots$ and $w_n$ are the weights of each neural network model. The sum of weights is one ($\sum_i^n w_i = 1$). The optimal weights $w_i$ can be determined by linear regression model using ordinary least square method that can be assumed as

$$\bar{Y} = c_0 + c_{N1} Y_{N1} + c_{N2} Y_{N2} + ... + c_{Nn} Y_{Nn} + \epsilon, \quad \epsilon \sim N(0, \sigma^2), \tag{146}$$

where $\epsilon$ is an uncertainty term, and $c_{N1}$, $c_{N2}$ and $c_{Nn}$ are the parameters of the partial regression and $c_0$ is a constant. These regression parameters can be identified from a maximum likelihood estimation. For the maximum likelihood estimation, we define a residual error function,

$$E_r = \sum_{i=1}^{m} (y_i - \bar{y}_i)^2. \tag{147}$$

The objective function, which minimizes the residual error function, can be rewritten by the regression model expression from the equation 146:

$$Min \quad E_r = \sum_{i=1}^{m} (y_i - \{c_0 + c_{N1} y_{N1} + c_{N2} y_{N2} + ... + c_{Nn} y_{Nn}\})^2 \tag{148}$$

The optimized candidate parameters minimizing this residual function can be identified by the first derivatives with respect to the parameters of the partial regression. The following partial differential equations can be represented,

$$\begin{aligned}
\tfrac{\partial E_r}{\partial c_{N1}} &= \textstyle\sum_{i=1}^{m} (y_i - \{c_0 + c_{N1} y_{N1} + c_{N2} y_{N2} + ... + c_{Nn} y_{Nn}\}) y_{N1i} = 0 \\
\tfrac{\partial E_r}{\partial c_{N2}} &= \textstyle\sum_{i=1}^{m} (y_i - \{c_0 + c_{N1} y_{N1} + c_{N2} y_{N2} + ... + c_{Nn} y_{Nn}\}) y_{N2i} = 0 \\
&... \\
\tfrac{\partial E_r}{\partial c_{Nn}} &= \textstyle\sum_{i=1}^{m} (y_i - \{c_0 + c_{N1} y_{N1} + c_{N2} y_{N2} + ... + c_{Nn} y_{Nn}\}) y_{Nni} = 0 \\
\tfrac{\partial E_r}{\partial c_{N0}} &= \textstyle\sum_{i=1}^{m} (y_i - \{c_0 + c_{N1} y_{N1} + c_{N2} y_{N2} + ... + c_{Nn} y_{Nn}\}) = 0.
\end{aligned} \tag{149}$$

To address simple mathematical form, we can rewrite the previous partial differential equations by the following matrix expression,

$$\mathbf{Ax} = \mathbf{Y}. \tag{150}$$

In the matrix formulation, the definitions of the matrix $\mathbf{A}$ and the vector $\mathbf{x}$ are

$$\mathbf{A} = \begin{bmatrix} 1 & y_{N11} & y_{N21} & \cdots & y_{Nn1} \\ 1 & y_{N12} & y_{N22} & \cdots & y_{Nn2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & y_{N1m} & y_{N2m} & \cdots & y_{Nnm} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_{N1} \\ c_{N2} \\ \vdots \\ c_{Nn}. \end{bmatrix} \tag{151}$$

Once $\mathbf{A}^T$ is multiplied in both sides, we can obtain the following equation:

$$\mathbf{A^T Ax} = \mathbf{A^T Y}. \tag{152}$$

From the above equation, if $A^T A$ is invertible, the regression vector $\mathbf{x}$ can be written as

$$\mathbf{x} = (\mathbf{A^T A})^{-1} \mathbf{A^T Y}. \tag{153}$$

The regression vector $\mathbf{x}$ yields the optimal weights $w_i$ according to the following equation:

$$w_1 = \frac{c_{n1}}{\sum_{i=1}^{n} c_{ni}}, \;\; w_2 = \frac{c_{n2}}{\sum_{i=1}^{n} c_{ni}}, \;\; \cdots, \;\; w_n = \frac{c_{nn}}{\sum_{i=1}^{n} c_{ni}}. \tag{154}$$

Therefore, the prediction formula described in the equation 145 achieves the optimal prediction model by the optimized weights shown in the equation 154.

To observe the performance of ensemble learning techniques, we execute case studies on the two ensemble techniques: the ensemble learning method using the bagging technique and the OLS ensemble learning. For the performance analysis of the two ensemble techniques we generates sample data described in Table 31 that is same as the experiment set-up in the previous section. Figure 116 is the sample data

including training and test data. To observe the effect of the two ensemble learning algorithms, we assume that the number of the hidden layer is one, the number of perceptrons in the hidden layer is 200, and the number of weak learners is five. Based on this neural network structure, the individual learners are optimized by the gradient approach through the backpropagation method. In the OLS ensemble model, all weak learners use the same training data for the optimization. In the bagging ensemble model, each weak learner uses different training data by splitting all training data into groups so that its number is same as the number of the weak learners. In this manner, the ensemble learning model can avoid a correlation problem. These individual weak learners, which OLS and bagging ensemble learners, are integrated into the ensemble learners by the ordinary least square method and the average of all outputs, respectively. Figures 117 and 118 are the experiment results of the two ensemble learners. In the Figure 117, the first five graphs are the test results of each prediction model by the weak learners, and the last graph is the result of the ensemble learner, which is combined by the five weak learners through the mean of all prediction results. The visual inspection of the classification results present that the ensemble learning techniques has shaped from the weak learners. Therefore, the ensemble learning result has the most smooth boundaries. Unlike OLS weak learners, the bagging has more diverse learners because of the splitting training data. 32 is the summary of the classification results. The classification results represent that both independent ensemble learning techniques improve the classification performance of the single neural network. The results also present that the ensemble learning with OLS has slightly better classification performance than the ensemble learning with the bagging technique.

**Table 32:** Experiment results of the two ensemble methods : OLS and Bagging

| Classification method | Name | Classification result (%) |
|---|---|---|
| OLS | Average performance of the five weak learners | 76 |
| | Ensemble learner | 83.33 |
| Bagging | Average performance of the five weak learners | 78.668 |
| | Ensemble learner | 80 |

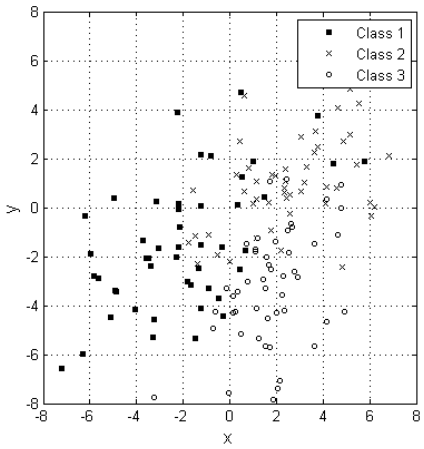
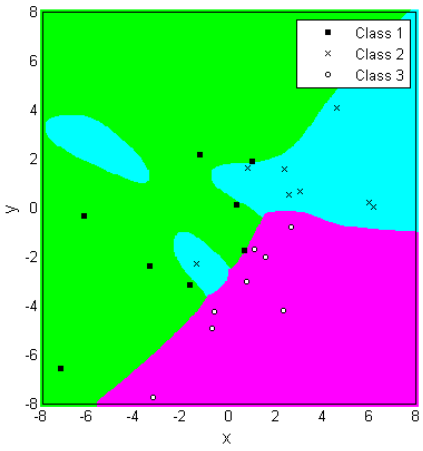
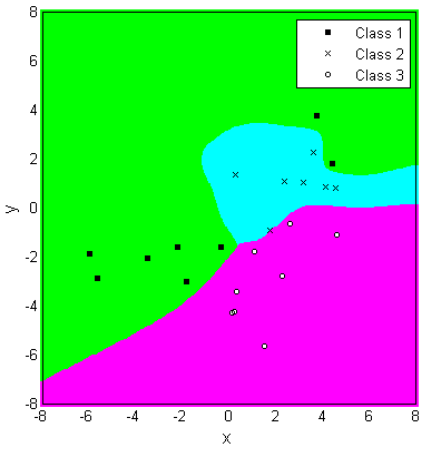


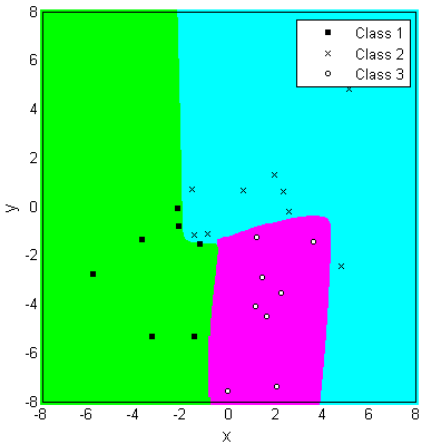

**Figure 116:** Training data



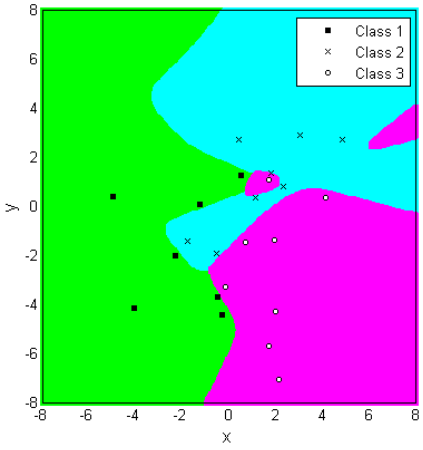

(a) First weak learner



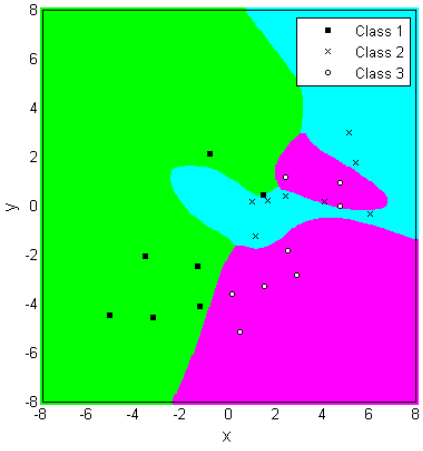

(b) Second weak learner



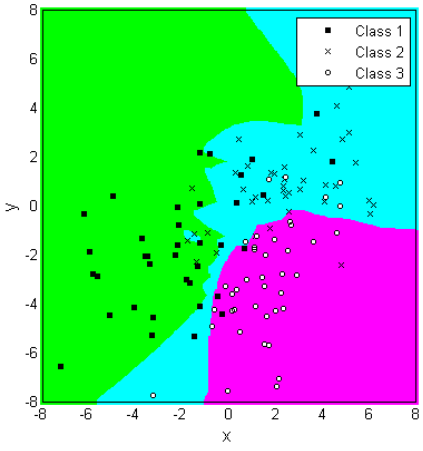

(c) Third weak learnern




(d) Fourth weak learner

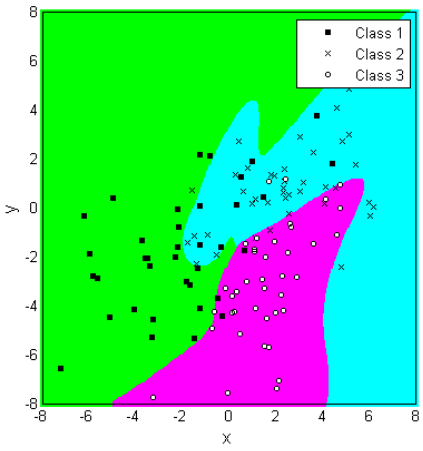



(e) Fifth weak learner

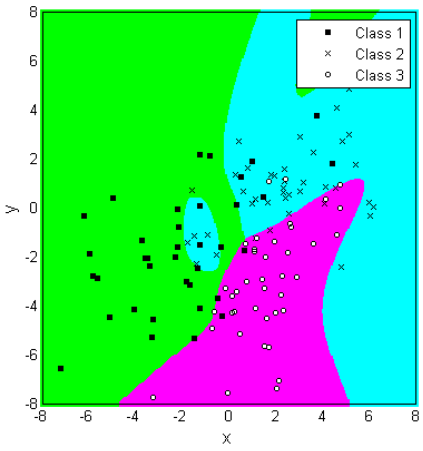



(f) Ensemble learning model

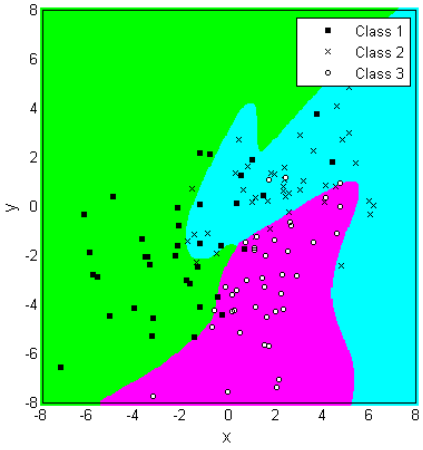
**Figure 117:** Ensemble learning method using the bagging technique



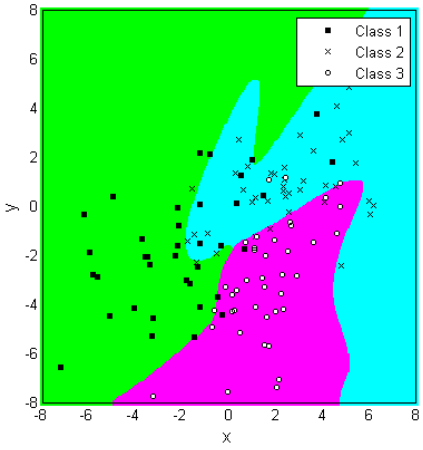

(a) First weak learner



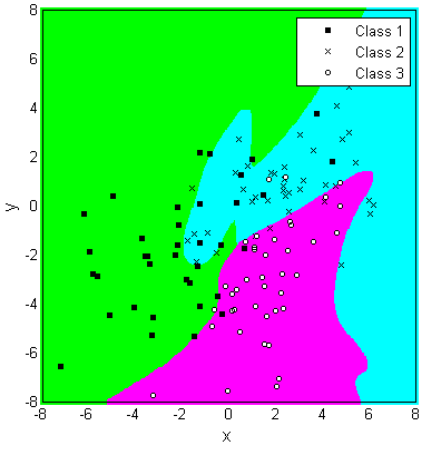

(b) Second weak learner



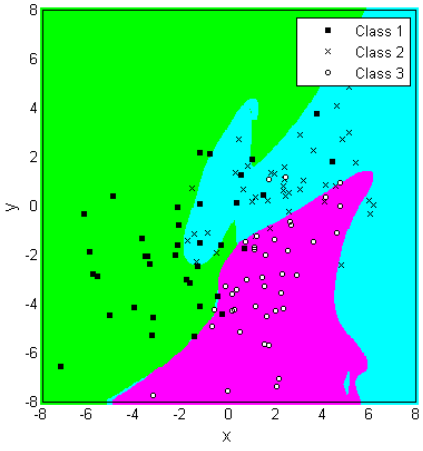

(c) Third weak learner




(d) Fourth weak learner




(e) Fifth weak learner




(f) Ensemble learning model

**Figure 118:** Ensemble learning method using the ordinary least square

# Bibliography


[1] "http://utm.arc.nasa.gov/index.shtml."

[2] "http://www.phoenix-aerial.com/information/lidar-comparison."

[3] *NASA Systems Engineering handbook*. NASA, 2007. NASA/SP-2007-6105.

[4] "Unmanned systems roadmap 2007 - 2032," tech. rep., Department of Defense, 2007.

[5] "Certification of authorization or waiver," 2008.

[6] "Federal actions needed to ensure safety and expand their potential uses within the national airspace system," tech. rep., United States Government Accountability Office, 2008. GAO-08-511.

[7] "NextGen UAS research development and demonstration roadmap," tech. rep., Next Generation Air Transportation System, 2012.

[8] "Amazon prime air project: Online retailer investing in unmanned drones to deliver goods," *ABC*, 2013.

[9] "Unmanned aircraft system(UAS) service demand 2015 - 2035, U.S department of transportation, research and innovative technology administration," tech. rep., Research and Innovative Technology Administration, 2013. DOT-VNTSC-DoD-13-01.

[10] "FAA faces significant barriers to safely integrate unmanned aircraft systems into the national airspace system," tech. rep., FAA, 2014. AV-2014-061.

[11] "UTM: Air traffic management for low-altitude drones," tech. rep., NASA, 2015.

[12] Afbca, E., *USAF Test Pilot School. Performance Phase Textbook*. Information for the Defense Community, 1986.

[13] Alexander, I. and Raiko, T., "Practical approaches to principal component analysis in the presence of missing values," *The Journal of Machine Learning Research*, vol. 11, pp. 1957 – 2000, 2000.

[14] Anderson, J., *Aircraft Performance & Design*. McGraw-Hill, 2010. ISBN-10: 0070019711.

[15] Anderson, M., Sverdrup, J., Lopez, J., and Evers, J., "A comparison of trajectory determination approaches for small UAV's," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, (Keystone, Colorado), 2006.

[16] Balandat, M., *Constrained Robust Optimal Trajectory Tracking: Model Predictive Control Approaches*. PhD thesis, Technische Universitt Darmstadt, 2010.

[17] Barfield, F., "Autonomous collision avoidance. the technical requirements," in *National Aerospace and Electronics Conference*, (Dayton, OH), pp. 808 – 813, IEEE, 2000.

[18] Beard, R. W. and McLain, T. W., *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[19] Benson, D., *A Gauss Pseudospectral Transcription for Optimal Control, Ph.D. Thesis, Dept. of Aeronautics and Astronautics*. PhD thesis, Massachusetts Institute of Technology, 2004.

[20] Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V., "Direct trajectory optimization and costate estimation via an orthogonal collocation method," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 6, pp. 1435 – 1440, 2006.

[21] BilGoerzen, C., Kong, Z., and Mettler, B., "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent Robot System*, vol. 57, pp. 65 – 100, 2010.

[22] Bishop, C. M., *Neural networks for pattern recognition*. Oxford university press, 1995.

[23] Bishop, C. M., *Pattern recognition and machine learning*. Springer, 2006.

[24] Brandt-Pollmann, U., Winkler, R., Sager, S., Moslener, U., and Schlder, J. P., "Numerical solution of optimal control problems with constant control delays," tech. rep., CER-ETH - Center of Economic Research (CER-ETH), 2006.

[25] Brooks, R. A., "A robust layered control system for a mobile robot," *IEEE Robotics and Automation*, vol. 2, no. 1, pp. 14 – 23, 1986.

[26] Brown, G., "Ensemble learning," in *Encyclopedia of Machine Learning*, Springer, 2010.

[27] Bryson, A. E. and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation and Control*. CRC Press, 1975. ISBN-10: 0891162283.

[28] Chakraborty, A., Seiler, P., and Balas, G. J., "Applications of linear and nonlinear robustness analysis techniques to the F/A-18 flight control laws," in *AIAA Conference on Guidance, Navigation, and Control*, (Chicago, Illinois), August 2009.

[29] Chakravarthy, A. and Ghose, D., "Obstacle avoidance in a dynamic environment: A collision cone approach," *Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 28, no. 5, pp. 562 – 574, 1998.

[30] Choi, Y., Jimenez, H., and Mavris, D., "Statistical gain-scheduling method for aircraft flight simulation," *Aerospace Science and Technology*, vol. 46, pp. 493 – 505, 2015.

[31] Cybyk, B. Z., McGrath, B. E., Frey, T. M., Drewry, D. G., Keane, J. F., and Patnaik., G., "Unsteady airflows and their impact on small unmanned air systems in urban environments," *Journal of Aerospace Information Systems*, vol. 11, no. 4, pp. 178 – 194, 2014.

[32] Dalamagkidis, K., Valavanis, K. P., and Piegl, L. A., "On unmanned aircraft systems issues, challenges and operational restrictions preventing integration into the national airspace system," *Progress in Aerospace Sciences*, vol. 44, no. 7, pp. 503–519, 2008.

[33] Data, P. L. E., "http://www.dcnr.state.pa.us/topogeo/pamap."

[34] Ding, X., Schild, A., Egerstedt, M., and Lunze, J., "Real-time optimal feedback control of switched autonomous systems," in *IFAC Conference on Analysis and Design of Hybrid Systems*, pp. 108 – 113, 2009.

[35] DoD, "Department of defense dictionary of military and associated terms," tech. rep., Department of Defense, 2010. Joint Publication 1-02.

[36] DoD, "Department of defense final report to congress on access to national airspace for unmanned aircraft systems," tech. rep., Department of Defense, 2010.

[37] DoD, "Final report to congress on access to national airspace for unmanned aircraft system, under secretary of defense(acquisition, technology and logistics)," tech. rep., Department of Defense, 2010.

[38] DoD, "Defense acquisition guidebook," tech. rep., Department of Defense, 2011.

[39] DoD, "Unmanned aircraft system airspace integration plan," tech. rep., Department Of Defense, 2011.

[40] DoD, "Unmanned systems integrated roadmap FY2011," tech. rep., Department Of Defense, 2013. 11-S-3613.

[41] Doebbler, J., Gesting, P., and Valasek., J., "Real-time path planning and terrain obstacle avoidance for general aviation aircraft," in *In AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. 15 – 18, 2005.

[42] Dreiseitl, S. and Ohno-Machado, L., "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5, pp. 352 – 359, 2002.

[43] Dreiseitl, S. and Ohno-Machado, L., "Logistic regression and artificial neural network classification models : a methodology review," *Journal of Biomedical Informatics*, vol. 35, pp. 352 – 359, 2002.

[44] Elliot, D., "DHL testing delivery drones," *CBSNEWS*, 2013.

[45] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., eds., *A density-based algorithm for discovering clusters in large spatial databases with noise*, vol. 96, International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996.

[46] Explore, U. E., "http://earthexplorer.usgs.gov."

[47] FAA, "Literature review on detect, sense, and avoid technology for unmanned aircraft systems," tech. rep., U.S Department of Transportation, Federal Aviation Administration, 2009. DOT/FAA/AE-08/41.

[48] FAA, "Integration of civil unmanned aircraft systems(uas) in the national airspace system(nas) roadmap," tech. rep., U.S Department of Transportation(Federal Aviation Administration), 2013.

[49] Feng, J., Lin, Z., Xu, H., and Yan, S., "Robust subspace segmentation with block-diagonal prior," pp. 3818 – 3825, 2014.

[50] Gertler, J., "U.S unmanned aerial systems," tech. rep., CRS Report for Congress, 2012.

[51] Geyer, C. M., Singh, S., and Chamberlain, L. J., "Avoiding collisions between aircraft: State of the art and requirements for UAVs operating in civilian airspace," tech. rep., Carnegie Mellon University,, Pittsburgh, PA, CMU-RI-TR-08-03 2008.

[52] Ghosh, S., Rancourt, D., and Mavris, D. N., "Principal component analysis assisted surrogate modeling (pca-sm) of correlated loads for uncertainty analysis of design load envelopes," in *16th AIAA/ISSMO Multidisciplinary analysis and optimization conference*, 2016.

[53] Gibbs, M. N., *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

[54] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," tech. rep., SIAM Review, 2005.

[55] Gill, P. E., Murray, W., and Saunders, M. A., *User's Guide for SNOPT Version 7: Software for Large Scale Nonlinear Programming*. Stanford University, 2006.

[56] Griffiths, S., Saunders, J., Curtis, A., Barber, B., McLain, T., and

Beard, R., "Obstacle and terrain avoidance for miniature aerial vehicles," *In Advances in Unmanned Aerial Vehicles*, 2007. Springer Netherlands.

[57] Hampton, M. E., "FAA faces significant barriers to safely integrate unmanned aircraft systems into the national airspace system," tech. rep., Federal Aviation Administration (FAA), 2014. AV-2014-061.

[58] Ho, Y.-J. and Liu, J.-S., "Collision-free curvature-bounded smooth path planning using composite bezier curve based on voronoi diagram," in *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on*, pp. 463 – 468, IEEE, 2009.

[59] Hrabar, S., "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," in *IROS 2008*, pp. 807 – 814, IEEE, 2008.

[60] Hu, J., You, S., Neumann, U., and Park, K. K., "Building modeling from lidar and aerial imagery," in *." In Proceedings of ASPRS*, 2004.

[61] Huntington, G. T., Benson, D. A., How, J. P., N. Kanizay, C. L. D., and Rao, A. V., "Computation of boundary controls using a gauss pseudospectral method," in *2007 Astrodynamics Specialist Conference*, (Mackinac Island, Michigan), 2007.

[62] Huntington, G. T., Benson, D. A., and Rao, A. V., "Design of optimal tetrahedral spacecraft formations," *Journal of the Astronautical Sciences*, vol. 55, no. 2, pp. 141 – 169, 2007.

[63] Huntington, G. T. and Rao, A. V., "Optimal reconfiguration of spacecraft formations using the gauss pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 3, pp. 689 – 698, 2008.

[64] Huntington, G. T., *Advancement and analysis of Gauss pseudospectral transcription for optimal control problems*. PhD thesis, Massachusetts Institute of Technology, 2007.

[65] Ibarrondo, F. and Sanz-Aránguez, P., "Integrated versus two-loop guidance-autopilot for a dual control missile with high-order aerodynamic model," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 2015.

[66] Inventory, U. S. I. E., "http://coast.noaa.gov."

[67] Islam, M. M., Yao, X., and Murase, K., "A constructive algorithm for training cooperative neural network ensembles," *Neural Networks, IEEE Transactions on*, vol. 14, no. 4, pp. 820 – 834, 2003.

[68] Islam, M. M., Yao, X., Nirjon, S. S., Islam, M. A., and Murase, K., "Bagging and boosting negatively correlated neural networks," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 3, pp. 771 – 784, 2008.

[69] Jang, J., Annaswamy, A., and Lavretsky, E., "Adaptive control of time-varying systems with gain-scheduling," in *American Control Conference*, (Seattle, WA), pp. 3416 – 3421, June 2008. ISSN:0743-1619.

[70] Jenie, Y. I., van Kampen, E.-J., de Visser, C. C., and Chu, Q.-P., "Selective velocity obstacle method for cooperative autonomous collision avoidance system for UAVs," in *AIAA Guidance, Navigation, and Control*, (Boston, USA), AIAA, 2013.

[71] Jia, H., Ding, S., Xu, X., and Nie, R., "The latest research progress on spectral clustering," *Neural Computing and Applications*, vol. 24, pp. 1477–1486, 2014.

[72] Jia, W., Zhao, D., Tang, Y., Hu, C., , and Zhao, Y., "An optimized classification algorithm by neural network ensemble based on pls and ols," *Mathematical Problems in Engineering*, 2014.

[73] Jianqiao, Y., Li, L., Hongxia, Z., and Chengdong, X., "Robust gain-scheduled controller design for air defense missile," in *25th Chinese Control Conference*, (Harbin, China), 2006.

[74] Jin, Y., Okabe, T., and Sendhoff, B., "Neural network regularization and ensembling using multi-objective evolutionary algorithms," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1, pp. 1 – 8, IEEE, 2004.

[75] Jinhui, H., Youand, S., and Neumann, U., "Approaches to large-scale urban modeling," *Computer Graphics and Applications*, vol. 23, no. 6, 2003.

[76] Johnson, C., "UAS integration in the NAS project," tech. rep., The National Aeronautics and Space Administration(NASA), 2010.

[77] Jolliffe, I., *Principal component analysis*. John Wiley & Sons, 2002.

[78] J.R., W., "UAV roundup 2011," tech. rep., American Institute of Aeronautics and Astronautics, 2011.

[79] Kahale, E., Castillo, P., and Bestaoui, Y., "Minimum time reference trajectory generation for an autonomous quadrotor," in *International Conference on Unmanned Aircraft Systems(ICUAS)*, (Orlando, FL), pp. 126 – 133, IEEE, 2014.

[80] Kang, K., , and Prasad., J. V. R., "Development and flight test evaluations of an autonomous obstacle avoidance system for a rotary-wing UAV," *Unmanned Systems*, vol. 1, no. 1, pp. 3 – 19, 2013.

[81] Kang, K., *Online optimal obstacle avoidance for rotary-wing autonomous unmanned aerial vehicles*. PhD thesis, Georgia Institute of Technology, 2012.

[82] Katsuhiko, O., *Modern Control Engineering*. Prentice Hall, fourth ed., 2002.

[83] Keller, H. B., "Numerical solution of two point boundary value problems," in *Society for Industrial and Applied Mathematics*, 1976.

[84] Kim, N. and Calise, A. J., "Neural network based adaptive output feedback augmentation of existing controllers," *Aerospace Science and Technology*, vol. 12, no. 3, pp. 248 – 255, 2008.

[85] Kirk, D. E., *Optimal Control Theory: An Introduction*. Dover, 1998. ISBN-10: 0486434842.

[86] Kirkpatrick, K., Jr, J. M., and Valasek, J., "Aircraft system identification using artificial neural networks.," in *In 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013.

[87] Knisely, N., Urcinas, A., Jimenez, H., and Mavris, D., "Unmanned aircraft systems integration into the national airspace system - tradeoff analysis for en route transit operations," in *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, (Indianapolis, Indiana), AIAA, 2012. AIAA 2012-5425.

[88] Kuwata, Y. and How, J., "Three dimensional receding horizon control for UAVs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Providence, Rhode Island), 2004.

[89] Lacher, A. R., Maroney, D. R., and Zeitlin, A. D., "Unmanned aircraft collision avoidancetechnology assessment and evaluation methods," in *The*

*7th Air Traffic Managemtent Research & Development Seminar*, (Barcelona, Spain), 2007.

[90] LaValle, S. M., *Planning algorithms*. Cambridge university press, 2006.

[91] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R., "Efficient backprop," *Neural networks: Tricks of the trade*, pp. 9 – 48, 2012.

[92] Levine, D., Luders, B., and How, J. P., "Information-rich path planning with general constraints using rapidly-exploring random trees," in *AIAA Infotech Aerospace Conference*, 2010.

[93] Liberzon, D., *Calculus of Variations and Optimal Control Theory*. Princeton University Press, 2012. ISBN: 9780691151878.

[94] Lin, W., "Distributed UAV formation control using differential game approach," *Aerospace Science and Technology*, vol. 35, pp. 54 – 62, 2014.

[95] Liu, Y. and Yao, X., "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, no. 10, pp. 1399 – 1404, 1999.

[96] Liu, Y. and Yao, X., "Simultaneous training of negatively correlated neural networks in an ensemble," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 29, no. 6, pp. 716 – 725, 1999.

[97] Maniatopoulos, S., Panagou, D., and Kyriakopoulos, K. J., "Model predictive control for the navigation of a nonholonomic vehicle with field-of-view constraints," in *American Control Conference (ACC)* (IEEE, ed.), pp. 3967 – 3972, 2013.

[98] Mathworks, *Supervised learning algorithms*, 2014.

[99] Mcfadyen, A., Durand-Petiteville, A., and Mejias, L., "Decision strategies for automated visual collision avoidance," in *International Conference on Unmanned Aircraft Systems(ICUAS)*, (Orlando, FL), IEEE, 2014.

[100] Mellinger, D., Kushleyev, A., and Kumar, V., "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *IEEE International Conference* (Robotics and (ICRA), A., eds.), pp. 477 – 483, 2012.

[101] Melnyk, R. V., *A framework for analyzing unmanned aircraft system integration into the national airspace system using a target level of safety approach*. PhD thesis, Georgia Institute of Technology, 2013.

[102] Menon, P. K., Sweriduk, G. D., and Sridhar, B., "Optimal strategies for free-flight air traffic conflict resolution," *Journal of Guidance Control Dynamics*, vol. 22, no. 2, pp. 202 – 211, 1999.

[103] Mercer, J., Prevt, T., Jacoby, R., Globus, A., and Homola, J., "Studying nextgen concepts with the multi-aircraft control system," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, (Honolulu, Hawaii), AIAA, 2008. AIAA 2008-7026.

[104] Miotto, P., Paduano, J. D., Feron, E., and Burken, J. J., "Modern fixed structure control design part ii: Automated gain scheduling," in *AIAA Conference on Guidance, Navigation, and Control*, (New Orleans), August 1997.

[105] Moon, J. and Prasad, J., "Minimum-time approach to obstacle avoidance constrained by envelope protection for autonomous uavs," in *Proceedings of AHS 65th Annual Forum*, (Grapevine, TX), 2009.

[106] Moon, J., *Mission-Based Guidance System Design for Autonomous UAVs*. PhD thesis, Georgia Institute of Technology, 2009.

[107] Morelli, E., "Global nonlinear parametric modelling with application to f-16 aerodynamics," in *IEEE American Control Conference*, vol. 2, pp. 997 – 1001, 1998.

[108] Morelli, E., "In-flight system identification," *AIAA*, pp. 238 – 247, 1998.

[109] Murphy, J. and Kim, S. K., "Live virtual constructive distributed test environment characterization report," tech. rep., NASA, 2013. UAS-04.05.0001.01.

[110] Myers, R. H. and Montgomery, D. C., *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley, second ed., 2011.

[111] Naeini, M. P., Taremian, H., and Hashemi, H. B., "Stock market value prediction using neural networks," in *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on*, pp. 132 – 136, IEEE, 2010.

[112] NASA, "Unmanned aircraft system gap analysis for national airspace system access," tech. rep., National Aeronautics and Space Administration, 2011. Ver. 1.1.

[113] Ng, A. Y., Jordan, M. I., and Weiss, Y., "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing system*, vol. 2, pp. 849 – 856, 2002.

[114] Nichols, R., Reichert, R., and Rugh, W., "Gain scheduling for H-infinity controllers: a flight control example," *IEEE Control Systems Technology*, vol. 1, no. 2, pp. 69 – 79, 1993.

[115] Opentopography, "http://www.opentopography.org."

[116] Orr, M. W., Rasmussen, S. J., Karni, E. D., and Blake, W. B., "Framework for developing and evaluating mav control algorithms in a realistic urban setting," in *American Control Conference*, pp. 4096 – 4101, IEEE, 2005.

[117] Park, J. and Kim, Y., "Obstacle detection and collision avoidance of quadrotor UAV using depth map of stereo vision," in *AIAA Guidance, Navigation, and Control*, (Boston, MA), AIAA, 2013. AIAA 2013-4994.

[118] Pham, D. T., Dimov, S. S., and Nguyen, C. D., "Selection of k in k-means clustering," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 219, no. 1, pp. 103 – 119, 2005.

[119] Portal, I. S. D., "http://gis.iu.edu/datasetinfo."

[120] Prevt, T. and Mercer, J., "MACS: A simulation platform for todays and tomorrows air traffic operations," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, AIAA, 2007. AIAA 2007-6556.

[121] Rademakers, N., Akmeliawati, R., Hill, R., Bil, C., and Nijmeijer, H., "Modelling and gain scheduled control of a tailless fighter," in *5th Asian Control Conference*, vol. 1, (Melbourne, Victoria, Australia), pp. 374 – 382, July 2004.

[122] Rao, A. V., *Extension of the Computational Singular Perturbation Method to Optimal Control*. PhD thesis, Princeton University, 1996.

[123] Rao, A. V. and Mease, K. D., "Dichotomic basis approach to solving hypersensitive optimal control problems," *Automatica*, vol. 35, no. 4, pp. 633 – 642, 1999.

[124] Rao, A. V., "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497 – 528, 2009.

[125] Rapidlasso, "http://www.rapidlasso.com."

[126] Rasmussen, C. and Williams, C. K. I., *Gaussian Processes for Machine Learning*. MIT press, 2006. ISBN:026218253X.

[127] Raymer, D. P., *Aircraft Design: A conceptual Approach*. AIAA, 2006. ISBN-10: 1600869114.

[128] RTCA, "Operational services and environmental definition (OSED) for unmanned aircraft systems," tech. rep., FAA, June 2010. SC-203.

[129] Rugh, W. J. and Shamma, J. S., "Research on gain scheduling," *Automatica*, vol. 36, no. 10, pp. 1401 – 1425, 2000.

[130] Rugh, W., "Analytical framework for gain scheduling," *Control Systems*, vol. 11, pp. 79 – 84, Jan 1991. ISSN:1066-033X.

[131] Salmah, Sutrisno, Joelianto, E., Budiyono, A., Wijayanti, I. E., and Megawati, N. Y., "Model predictive control for obstacle avoidance as hybrid systems of small scale helicopter," in *International Conference on Instrumentation Control and Automation*, (Ungasan), IEEE, 2013.

[132] Schrage, D. and Mavris, D., "Technology for affordability - how to define, measure, evaluate, and implement it?," in *50th National Forum of the American Helicopter Society*, (Washington, D.C.), 1994.

[133] Shanmugavel, M., Tsourdos, A., and White, B. A., "Collision avoidance and path planning of multiple uavs using flyable paths in 3d," in *In Methods and Models in Automation and Robotics (MMAR)*, pp. 218 – 222, 2015.

[134] Shin, H.-H., Lee, S.-H., Kim, Y., Kim, E.-T., and Sung, K.-J., "Design of a flight envelope protection system using a dynamic trim algorithm," *International Journal of Aeronautical and Space Science*, vol. 12, no. 3, pp. 241 – 251, 2011.

[135] Stastny, T. J., Garcia, G., and Keshmiri, S., "Robust three-dimensional collision avoidance for fixed-wing unmanned aerial systems," in *AIAA Guidance, Navigation, and Control Conference*, 2015.

[136] Stastny, T. J., Garcia, G. A., and Keshmiri., S. S., "Collision and obstacle avoidance in unmanned aerial systems using morphing potential field navigation and nonlinear model predictive control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 1, 2015.

[137] Stastny, T. J., Garcia, G. A., and Keshmiri, S. S., "Collision and obstacle avoidance in unmanned aerial systems using morphing potential field navigation and nonlinear model predictive control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 1, 2015.

[138] Stevens, B. L. and Lewis, F. L., *Aircraft control and simulation*. Wiley, second ed., 2003.

[139] Stoer, J. and Bulirsch, R., *Introduction to Numerical Analysis*. Springer, 2002. ISBN-13: 978-1441930064.

[140] Sundar, S. and Shiller, Z., "Time-optimal obstacle avoidance," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, (Nagoya, Japan), 1995.

[141] Sutrisno, S., Joelianto, E., Budiyono, A., Wijayanti, I. E., and Megawati, N. Y., "Model predictive control for obstacle avoidance as hybrid

systems of small scale helicopter," in *Instrumentation Control and Automation (ICA)*, (Ungasan), IEEE, 2013.

[142] Suzuki, J. and Valasek., J., "Fuzzy logic based forebody vortex flow control.," *AIAA paper*, pp. 1155 – 1165, 1997.

[143] Tom, S., Moor, B. D., Feron, E., and How, J., "Mixed integer programming for multi-vehicle path planning," in *European control conference*, vol. 1, pp. 2603 – 2608, 2001.

[144] Tsukayama, H., "Google buys drone maker titan aerospace," *The Washington Post*, 2014.

[145] Vanderplaats, G. N., *Multidiscipline Design Optimization*. Vanderplaats Research & Development, Inc., 2007.

[146] Verma, V., Rakesh, K., and Hsu, S., "3D building detection and modeling from aerial lidar data," in *2006 IEEE Computer Society Conference*, vol. 2, pp. 2213 – 2220, 2006.

[147] Vlassenbroeck, J., "A chebyshev polynomial method for optimal control with state constraints," *Automatica*, vol. 24, no. 4, pp. 499 – 506, 1988.

[148] Vlassenbroeck, J. and Doreen, R. V., "A chebyshev technique for solving nonlinear optimal control problems," *IEEE Transactions on Automatic Control*, vol. 33, no. 4, pp. 333 – 340, 1988.

[149] von Stryk, O. and Bulirsch, R., "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, vol. 37, no. 1, pp. 357 – 373, 1992.

[150] Watanabe, Y., Calise, A. J., and Johnson, E. N., "Vision-based obstacle

avoidance for UAVs," in *AIAA Guidance, Navigation and Control Conference*, (Hilton Head, South Carolina), 2007.

[151] Weibel, R. E., Edwards, M. W., and Fernandes, C., "Establishing a risk-based separation standard for unmanned aircraft self separation," in *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, (Virginia Beach, VA), AIAA, 2011. AIAA 2011-6921.

[152] Weinstock, R., *Calculus of Variations with Applications to Physics and Engineering*. McGraw-Hill, 1974. ISBN-10: 0486630692.

[153] Wen, N., Zhao, L., Su, X., and Ma, P., "UAV online path planning algorithm in a low altitude dangerous environment," *Automatica Sinica*, vol. 2, no. 2, 2015.

[154] Wieland, F., DeLaurentis, D., and Kubat, G., "Modeling and simulation for UAS in the NAS," tech. rep., NASA, 2012.

[155] Wise, K. A. and Roy., D. J. B., "Agile missile dynamics and control," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 3, pp. 441 – 449, 1998.

[156] Xu, N., Cai, G., Kang, W., and Chen, B. M., "Minimum-time trajectory planning for helicopter uavs using computational dynamic optimization," in *In Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference*, pp. 2732 – 2737, IEEE, 2012.

[157] Xu, N., Cai, G., Kang, W., and Chen, B. M., "Minimum-time trajectory planning for helicopter uavs using computational dynamic optimization," in *In Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pp. 2732 – 2737, IEEE, 2012.

[158] You, S., Hu, J., Neumann, U., and Fox, P., "Urban site modeling from lidar," in *ICCSA'03 Proceedings of the 2003 international conference on Computational science and its applications*, pp. 579 – 588, 2003.

[159] Zhou, Q.-Y. and Neumann, U., "Fast and extensible building modeling from airborne lidar data," in *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, 2008.

# VITA

Youngjun Choi is from Seoul, South Korea. He received Bachelor (2004) and Master degree (2006) in Mechanical Engineering at Gachon University. He was a visiting researcher at the Centre for Power Transmission and Motion Control at the University of Bath in the United Kingdom, sponsored by the British Council and the National Research Foundation of the South Korea. As a researcher, he joined the development programs of two satellite systems and an airborne-radar system for an unmanned aircraft application at the Agency for Defense Development (ADD) and researched vibration suppression techniques using smart materials for a space application. He joined the Aerospace Systems Design Laboratory at the School of Aerospace Engineering at the Georgia Institute of Technology to pursue his doctoral studies. He earned his Master degree in 2013, and his Doctor of Philosophy degree in August 2016.