Enhancing Safety, Efficiency, and Resilience in Advanced Air Mobility Through Geofencing, Contingency Landing Management, and Optimized Network Strategies

by

Joseph Kim

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2024

Doctoral Committee:

Professor Ella M. Atkins, Co-Chair
Assistant Professor Max Z. Li, Co-Chair
Associate Professor Dimitra Panagou
Professor Nadine B. Sarter

Taehyung Kim
jthkim@umich.edu
ORCID iD: 0000-0001-9292-5139

DEDICATION

To my beloved family,

Without your unwavering love, support, and prayers, I would not have embarked on this journey. To my parents, sister, and grandma, your encouragement and sacrifices have been my strength and inspiration. Your belief in me has fueled my determination to pursue my dreams.

To my dear pastor Yongbaek Shin,

Your spiritual guidance, love, support and prayers have been a guiding light in my life. Your prayers and counsel have provided me with the courage and confidence to face challenges and persevere. Your love in me has fueled my passion to pursue the visions.

And to my Lord,

I am eternally grateful for your blessings, wisdom, and guidance throughout the life journey. You have shaped me, strengthened me, and chastised me, shaping me into the person I am today. You have been my Shepherd, refreshing my soul and guiding me along the right paths. Your protection, love, and grace have been my constant companions, and I am humbled by your presence in my life. Soli Deo Gloria.

ACKNOWLEDGEMENTS

I would like to begin by expressing my heartfelt gratitude to my advisor, Professor Ella Atkins. From the moment I chose to study at University of Michigan and nervously stepped into your office, I truly knew that studying under your guidance would be an extraordinary journey. Your mentorship, unwavering support, and encouragement have been invaluable to me throughout my time here. Whenever I faced challenges, you were there to support me and to provide invaluable insights into tackling research obstacles. Every fond memory I have of my time at Michigan is from the opportunities you provided me to pursue my dreams. Thank you, Professor Atkins, for believing in me and for accepting me as your Ph.D. student. I recall promising you years ago that I would one day make flying cars, now known as Advanced Air Mobility (AAM), a reality and give you a ride. I am making one more step now to make this vision a reality and fulfil my promise.

I am also deeply grateful to my co-advisor, Professor Max Li, for the privilege of being your Ph.D. student. Working alongside you has been nothing short of inspiring, and I am immensely appreciative of your guidance and support. It is also an honor to be the first Ph.D. graduate of the LATTICE lab. Thank you for your guidance and support.

To my esteemed committee members, Professor Nadine Sarter and Professor Dimitra Panagou, I extend my sincere thanks for your invaluable feedback and guidance throughout my Ph.D. journey. Your insights have been instrumental in shaping my research endeavors.

Also, I want to express my heartfelt gratitude to the members of the A2SYS lab, whose camaraderie and support made this Ph.D. journey all the more meaningful. I am grateful to each of you, Akshay Marthur, Mark Nail, Brandon Apodaca, Paul Flanigen, Prashin Sharma, Jeremy Castagno, Prince Kuevor, Matt Romano, Mia Stevens, Cosme Ochoa and Chris Barkey, for your friendship and collaboration. I extend my gratitude to the members of the LATTICE lab as well, Haochen Wu, Hejun Huang, Billy Mazotti, Minghao Chen, Yuxuan Fang, Sihang Wei, Sinan Abdulhak, Armaan Kamat and more. It was great interacting with you, and I am thrilled to hear about the exciting projects and great achievements you will make.

I am indebted to my collaborators, particularly those at Collins Aerospace: Alex Postnikov, Nick Liberko, Stefano Riverso, Giovanni Franzini, Khushboo Wadhwani, Dave

Watkins and NASA: Natasha Neogi, Hanbong Lee, Terry Morris, for their unwavering support, encouragement and guidance. Your collaboration and mentorship have been instrumental in my progress both in research and career.

Lastly, I wish to express my profound gratitude to my friends, brothers, and sisters at Korean New Life Church and Redeemer Church for their prayers and unwavering support. Your encouragement and prayers have been a source of strength. I eagerly anticipate the day when I can share with you the realization of connecting the world through the skies.

PREFACE

Before my thesis begins, I want to shed light on my personal journey and the purpose driving my pursuit of a Ph.D. in Advanced Air Mobility (AAM). Growing up in a small Nigerian town in Africa, I witnessed firsthand the transformative power of engineering in improving lives and communities. In a place where access to clean water and efficient transportation were daily challenges, I saw the profound impact that innovative technologies could have on addressing these fundamental needs; it was a groundbreaking portable water filtration system that helped solve the struggles faced by family and friends. My early experiences instilled in me a strong belief in the potential of engineering that can positively change the villages, cities and the world.

From a young age, I have been captivated by the sky and the remarkable technology about flying vehicles. Then, I found myself pondering the idea: "What if we could travel in ways that enable engineers and innovators to reach those in need more easily?" This fascination ignited my dreams to pioneer the development of flying cars capable of connecting communities efficiently and seamlessly, thereby transforming the way people interact, travel and help others. My dream is to connect the world through sky, from isolated and underdeveloped regions to developed cities and countries, so that essential engineering technologies, education, and healthcare systems can be distributed around the world by people. Advanced Air Mobility will connect the world far beyond what conventional airliners can do to make this happen.

This vision has been the guiding force behind my academic pursuits in aerospace engineering, as well as my professional endeavors, including my service as a second lieutenant in the Republic of Korean Air Force. It is the purpose and these dreams that have led me to pursue a Ph.D. at the University of Michigan. This thesis represents the culmination of years of dedication, passion, and perseverance in pursuit of this ambitious goal. I believe that through innovative research and collaborative efforts, we can harness the power of engineering to shape a better future for our generation and for the generations to come.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

ABSTRACT

Advanced Air Mobility (AAM) is a rapidly emerging sector of the aerospace industry with the potential to significantly improve the transportation system. Its benefits include reduced congestion and travel time, environmentally friendly low-emission operations, and the stimulation of economic growth through new infrastructure. This dissertation offers novel solutions for individual vehicle management, contingency planning, and optimal AAM traffic network management.

The first contribution is a method to generate flight plans with geofencing for low-altitude urban airspace. We create conflict-free geofence volumes that wrap flight routes using computational geometry and polygon decimation. Visibility graph with constant altitude versus terrain following paths are compared by a weighted cost summing distance and energy. By generating conflict-free flight paths for a single vehicle and strategically deconflicting multiple vehicles, the proposed algorithms offer route flexibility relative to fixed corridor solutions.

The second contribution is statistically-guided airspace geofence volume sizing. The optimal geofence sizing is critical for the scalability of AAM operations. If too large, airspace will be wasted; if too small, we risk AAM violating geofences due to uncertainties. We propose a method to determine geofence buffer sizes based on vehicle dynamics, statistical sensor errors, and urban wind models specified by computational fluid dynamics. This method generates statistically-guided geofence sizes with a three-sigma safety level with a fixed-wing Uncrewed Aircraft System (UAS) scaled up to an AAM vehicle. The findings provide insights into optimizing geofence volumes for different environmental conditions and vehicle types.

The third contribution is an extension of two-dimensional polygon geofences to polyhedron geometries for climb, descent, and UAS swarms. We first define and analyze a parallelepiped geofence with top and bottom surface slopes matching the flight path angle. Next, we define a convex hull swarm containment geofencing algorithm and showed how much airspace volume is saved compared to single vehicle geofencing solutions.

The fourth contribution is the development of assured contingency landing management (ACLM) for AAM. The ACLM architecture enables distressed AAM flights to quickly determine safe landing options, thereby enhancing overall safety in AAM operations. ACLM

employs mathematically-provable controllability and reachability logic, and integrates pre-analyzed prepared and unprepared landing sites and plan databases to maximize response efficiency. A multicopter case study uses ACLM to safely land when motor and/or battery failures occur. Multicopter landing options are analyzed as a function of vehicle weight, and parallel threading enables ACLM to execute in milliseconds.

The fifth contribution is AAM traffic management optimization. By strategically sectorizing urban airspace and modeling AAM routes, we formulate centralized management with vehicle, infrastructure, and operation constraints. Then, distributed network management is formulated as bi-level optimization using cooperative game theory and mixed integer programming. The research offers valuable insights into scalable AAM network management strategies by comparing centralized and distributed AAM network managements.

In summary, this thesis contributes to the safety, efficiency, and resilience of AAM. It promotes safety with solutions for low-altitude flight planning through geofencing and assured contingency landing management for distressed vehicles. It improves efficiency by determining optimal geofence geometries and sizes for AAM flights. It enhances AAM operational resilience with strategically optimal traffic management solutions that adapt to time-dependent infrastructural and operational constraints, as well as shifting demands in regional airspace congestion. Research insights pave the way for scalable and efficient AAM and urban airspace management.

CHAPTER 1

Introduction

1.1 Motivation

This dissertation contributes to ensuring safe, efficient, and resilient operation of autonomous aerial systems in contemporary urban environments. This imperative is driven by the proliferation of Uncrewed Aircraft Systems (UAS) for diverse applications [11, 12] and the rapid growth of the Urban Air Mobility (UAM) sector [13], collectively referenced as Advanced Air Mobility (AAM). AAM will offer more efficient urban and regional transportation by bypassing ground congestion. As of this writing, approximately 200 companies are developing electric Vertical Takeoff and Landing (eVTOL) AAM aircraft [14]. Large-scale AAM deployment will boost economic vitality and job opportunities [15]. Figure 1.1 shows prospective AAM operations in urban airspace.

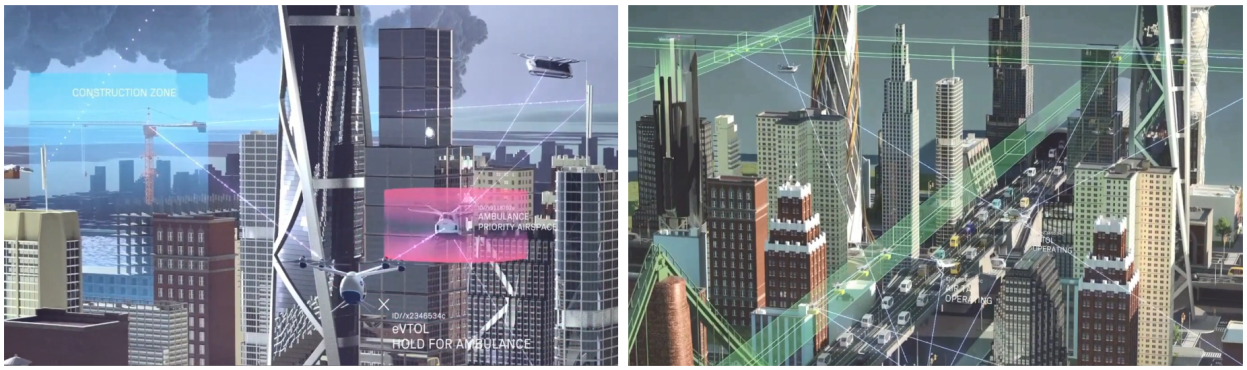


Figure 1.1: Prospective AAM operations in urban airspace. AAM vehicles are integrated into urban airspace, where UAM, air ambulance, and package-delivering UAS are also depicted. Figures are adapted from [1].

Efficient and safe transportation is vital for national well-being. For this reason, the National Aeronautics and Space Administration (NASA) has identified three key areas for

advancing AAM: air vehicle technologies, airspace system design and operations management, and community integration and acceptance [16]. To achieve this vision, five pillars were further established: 1) vehicle development, 2) individual vehicle management and operations, 3) airspace system design and implementation, 4) airspace/fleet management, and 5) community integration [15]. Moreover, NASA and Federal Aviation Administration (FAA) created an AAM Maturity Level scale to assess AAM ecosystems, each distinguished by air traffic density, operational complexity, and reliance on automation [17, 16].

To address the challenges posed by the dynamic future of airspace operations, we have conducted research in collaboration with Collins Aerospace (a subsidiary of RTX Corporation) and NASA. This dissertation contributes to safe and efficient AAM operation by offering novel solutions for airspace management, contingency planning, and AAM traffic network optimization.

1.2 Problem Statement

This dissertation addresses the following questions:

- How can we generate flight planning solutions in low-altitude urban airspace using static and durational airspace geofence volumes [18] to ensure safety and efficiency?
- How should a geofence buffer be sized so that it safely and compactly maintains separation among AAM vehicles operating within a limited airspace volume?
- How can we design spatially efficient climb/descent geofences and containment geofences for multi-agent UAS teams or swarms?
- How do we assure distressed AAM flights can rapidly select and proceed to a safe landing site with minimal decision-making delays?
- How can we optimize AAM traffic flow in both centralized and distributed settings, considering airspace constraints, diverse vehicle types, and service priorities, while also ensuring equity among AAM vehicles?

1.3 Research Approach and Dissertation Outline

This dissertation explores methodologies and system architectures to address each problem stated above. Particularly, this research employs a range of algorithms, including computational geometry, path planning, Guidance, Navigation and Control (GNC), computational

fluid dynamics (CFD) for urban wind analysis, graph theory, and network optimization to address the stated problems. Figure 1.2 outlines the overall research approach offering both vehicle-centric and network-centric perspectives.

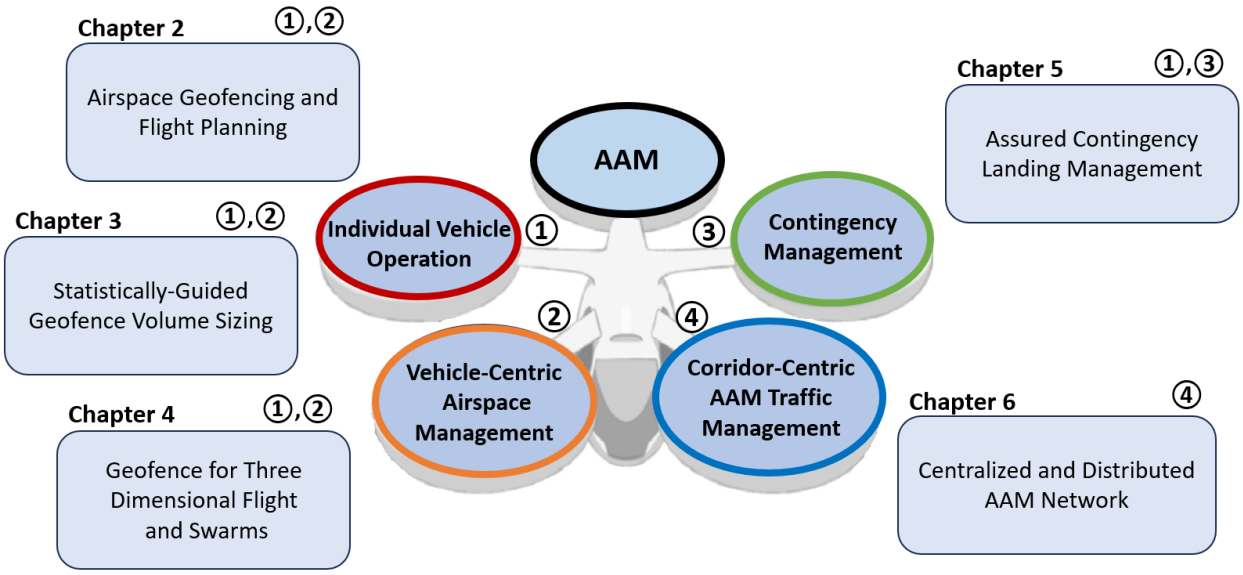


Figure 1.2: Outline of the research approach.

A summary of each dissertation chapter is presented below:

- **[Chapter 2] Airspace Geofencing and Flight Planning for Low-Altitude, Urban, and Small UAS:** This chapter introduces three-dimensional flight volumization algorithms to support airspace geofence management for AAM. Geofencing plays a crucial role in managing low-altitude airspace by separating flyable and no-fly zones [19, 20]. Our work considers low-altitude mapped obstacles and geofence volume requirements in 3D multicopter sUAS flight planning. Airspace deconfliction occurs with a FIFO (first-in, first-out) strategy. A static polygon geofence with limited duration encloses each 3-D flight path that avoids pre-existing keep-out geofences surrounding terrain, infrastructure, and other geofenced flight routes. Urban map data processing algorithms are presented for computationally efficient map construction. Monte Carlo simulations validate our algorithms. Simulation results illustrate geofencing pipeline support for the single-vehicle solution and multiple-vehicle deconfliction scenarios in an urban New York City environment.

- **[Chapter 3] Statistically-Guided Geofence Volume Sizing with an Advanced Air Mobility Vehicle Performance Model:** The scalability of flight operations poses a critical constraint for AAM [21]. As of the end of 2020, the US had around 1.7

million drones, a fleet size seven times larger than both airlines and general aviation combined, according to FAA estimates [22]. Prior research suggested fixed separation distances, but it remains uncertain whether these distances are optimal or excessive in densely populated low-altitude airspace [23, 24]. This chapter introduces a methodology to design geofence buffer sizing for each vehicle trajectory in restricted low-altitude airspace. This method takes into account vehicle dynamics, sensor characteristics, and wind conditions to generate statistically-guided geofence sizes with a three-sigma safety level. Case studies, including a small UAS and a full-size AAM aircraft model, are used to compare different geofence volume sizings that are statistically optimal for the specific AAM scale.

- **[Chapter 4] Space Efficient Airspace Geofence Volume Sizing:** This chapter builds upon the previous chapters by researching more spatially efficient algorithms for climbing and descending geofence volumes. As point-to-point UAS missions increase, the efficiency of geofencing for these flight segments improves. AAM missions involve climb, cruise, and descent flight segments. Efficient airspace allocation for climbing and descending are crucial to support various AAM operations in low-altitude airspace. Additionally, this chapter investigates the spatial and computational efficiency of wrapping a single geofence around multi-agent UAS teams or swarms. Such a geofencing design can improve airspace management efficiency for cooperatively controlled UAS teams. We introduce a parallelepiped geofence geometry for climb/descent and containment geofences for multi-agent UAS teams. We present simulation results analyzing geofence volume savings versus computation time.

- **[Chapter 5] Assured Contingency Landing Management (ACLM):** Contingency management has traditionally relied on human pilots. This is because human pilots can be innovative, and implementing fleet-wide certified software upgrades would be prohibitively expensive. However, the growth of AAM is likely to surpass the availability of experienced pilots, and inexperienced pilots may not respond safely in emergencies, particularly given the tight real-time response requirements associated with regional flights at low altitude. Thus, autonomy in contingency management is crucial for ensuring the safety of AAM operations. This chapter proposes an Assured Contingency Landing Management (ACLM) architecture. ACLM maximizes urgent or emergency landing success by minimizing the need for real-time perception and planning. The ACLM architecture generates and caches contingency landing plans before flight, monitors aircraft controllability and reachability during flight, and reacts with urgency when needed. When no cached plan is suitable for the encountered situation, ACLM

generates an online plan potentially for an unprepared site. This chapter demonstrates ACLM application with hexacopter UAS models, considering battery degradation and propulsion system failures. Monte Carlo simulation results show ACLM's effectiveness with success rates and runtime statistics.

- **[Chapter 6] Centralized and Distributed Optimization of Advanced Air Mobility Strategic Traffic Management:** The growth of AAM as an urban/ regional transportation mode raises the need for efficient air traffic management [25, 26, 27]. Various concepts of operations (ConOps) and architectures have been proposed for UAS/AAM Traffic Management [28, 29, 17]. However, to date, there has been no research investigating systematic and efficient management of low-altitude urban airspace, considering local traffic constraints in flight corridors and vertiport limits, as well as vehicle types, service priorities, and equity. This research aims to address this gap by strategically optimizing AAM traffic management in both centralized and distributed Providers of Services for UAM (PSU) settings. We propose a method to effectively sectorize low-altitude urban airspace and model centrally planned AAM routes considering limited corridor and vertiport capacities. Distributed AAM traffic management is then formulated as bi-level optimization using cooperative game theory and Mixed Integer Programming (MIP). Monte Carlo simulations evaluate AAM flight operations across different vehicle configurations and service priority types. Our research provides insights into the performance of centralized versus distributed AAM network management strategies, highlighting the potential of scalable distributed approaches in addressing urban airspace demands. By offering a comprehensive technical framework, our research informs decision-making in AAM traffic management strategy development and implementation.

1.4 Contributions and Innovations

1.4.1 Contributions

This thesis makes several contributions to AAM technologies. First, it introduces a 3D path planning module integrated with airspace geofencing. Through case studies involving both single UAS and multi-vehicle spatial deconfliction, this research offers insights into the effectiveness of airspace geofencing for low altitude UAS operations compared to traditional corridor methods.

Secondly, this thesis defines a method for geofence volume sizing using a scaled-up AAM vehicle dynamics model and representative inertial state estimate noise. By employing associated guidance and control strategies, trajectory tracking error is statistically analyzed across various scenarios, including those with no wind and low-altitude wind profiles.

The integration of flight planning with mathematically-provable controllability and reachability logic into the Assured Contingency Landing Management (ACLM) architecture is another contribution. This integration enhances the overall efficiency and reliability of the ACLM, facilitating a robust decision-making process that minimizes real-time computations with a contingency plan database and comprehensive suite of landing site options that need not be sensed in real-time.

Case studies illustrate how ACLM incorporates a prepared and categorized landing site database and conducts comprehensive simulations within a realistic environment model. These simulations are essential for assessing system performance under various conditions and refining algorithms for optimal outcomes.

Last, this thesis presents an approach to AAM network management that integrates airspace sectorization, corridor-based route planning, and centralized and distributed AAM network management. This integration aims to optimize airspace utilization and enhance overall system scalability and efficiency.

1.4.2 Innovations

Novel solutions to AAM challenges are proposed. First, this dissertation specifies three-dimensional durational flight trajectory geofence algorithms for vehicle operation, parallelepiped geofence volume definition for climb/descent, and containment geofencing for swarm flight. Unlike traditional geofences with fixed ceiling and floor altitudes, three-dimensional volumes efficiently wrap climbing and descending flight paths with staircase and parallelepiped geometries.

Additionally, the ability of an AAM vehicle to comply with geofence boundaries is validated by incorporating statistical navigation and tracking errors, as well as wind statistics, into geofence volume sizing.

The ACLM architecture with integrated online/offline flight planning capability is novel due to its use of prepared landing site and flight plan data to maximize response efficacy and minimize online response time.

Last, the development of centralized and distributed AAM strategic traffic management represents a novel approach in optimizing AAM traffic corridor flow. Through bi-level opti-

mization utilizing cooperative game theory and mixed-integer programming, our optimizations consider various constraints and priorities to simulate realistic traffic optimization scenarios, ultimately improving system efficiency and safety.

CHAPTER 2

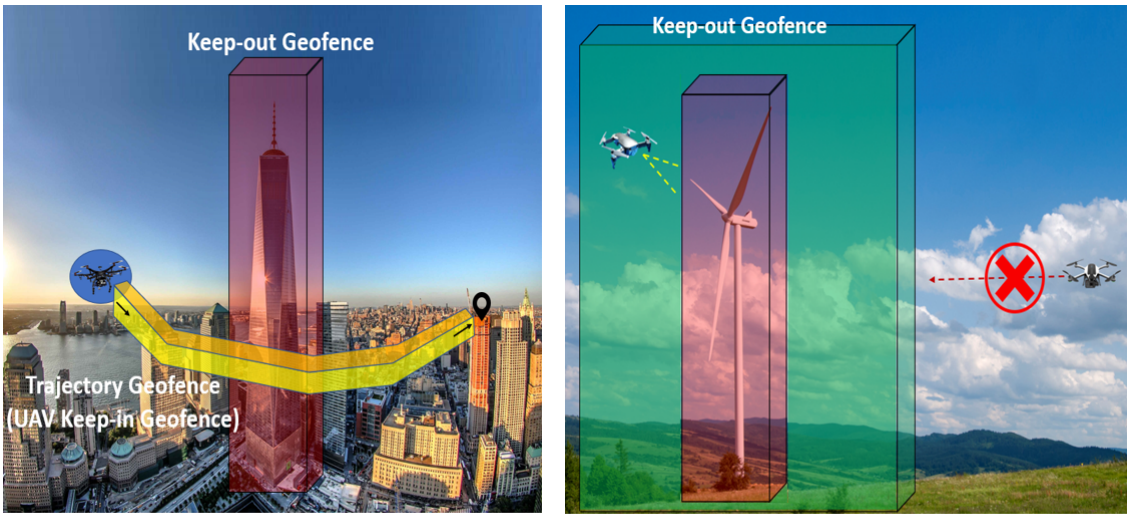
Airspace Geofencing and Flight Planning for Low-Altitude, Urban, and Small UAS

2.1 Introduction

Small Uncrewed Aircraft System (UAS) operations are expected to proliferate [11, 12] for applications such as small package delivery, surveillance, and the visual inspection of assets including wind turbines, construction sites, bridges, and agricultural products. Several challenges must be overcome to enable routine small UAS operations. The aviation community has proposed UAS Traffic Management (UTM) [25, 30, 31] to safely and efficiently manage low-altitude airspace where small UAS are expected to operate. UTM services are expected to be based on web apps and datalinks which facilitate the efficient definition and coordination of UAS flight plans.

Airspace geofencing [19] is one of the key capabilities required for UTM [25]. The envisioned geofencing system will enable safe flight operations by dividing airspace into available fly-zone (keep-in geofence) and no-fly zone (keep-out geofence) volumes with statically and dynamically adjusted virtual barriers or "fences" designed to assure UAS separation from obstacles, sensitive areas, and each other. Geofencing will facilitate safety management (i.e., Situational Awareness (SA) for trajectory monitoring, trajectory deviation alerts/geofence breaches, and contingency plans) and flight management (i.e., route-planning, the selection of take-off/landing sites, and mission priority adjustment) for UTM. Figure 2.1 illustrates airspace geofence examples for UAS flight operations near the One World Trade Center in Manhattan (left) and for wind turbine inspection (right).

Researchers have previously investigated airspace geofencing systems for UTM. Two-dimensional static, durational, and dynamic airspace geofence volumes are defined in [19] with a geofence construction algorithm considering steady wind described in [18]. Real-time geofence violation detection capabilities have been developed using Ray Casting [32] and Triangle Weight Characterization with Adjacency (TWCA) [33] methods. Potential



(a) transiting UAS in urban environment

(b) UAS inspection of wind turbine

Figure 2.1: UAS airspace geofencing examples. The left figure shows a keep-out geofence (red) around One World Trade Center in New York City. A transiting UAS keeps clear of this geofence with a path wrapped by a trajectory or keep-in geofence (yellow). The right figure shows a wind turbine being inspected by a small UAS. During inspection, the usual wind turbine keep-out geofence (red) is expanded as depicted in green to also enclose the inspection UAS. Any other nearby UAS will keep clear of this expanded keep-out geofence (green) during inspection activities. This geofence design assures separation between the two illustrated UAS.

intersections of 2-D geofences can be rapidly detected using a convex hull approach [34]. A constrained control scheme was developed using an Explicit Reference Governor (ERG) in [35]; this approach ensures a UAS does not violate geofence boundaries. This previous research primarily focused on geofence definition, boundary violation detection, and UAS avionics augmentation to support geofencing. Our work's focus on 3D path planning with geofence volumes in a realistically mapped urban environment is complementary.

Cooperative UAS flight tests were also evaluated using "separation by segregation" geofencing features in [36]. To define a local geofence volume for applications such as crop inspection, the maximum flight distance a UAS can travel after flight termination was calculated using vehicle dynamics and position sensors in [37] to define geofence geometry. This research demonstrated that a UAS stays within its prescribed keep-in geofence in both nominal and off-nominal (e.g., flight termination) conditions.

A three-dimensional dynamic geofencing volumization solution was proposed using "operational" and "inverse" volumization functions in [38]. Per [38], *airspace operational volumization* is the process by which a requested flight plan is "wrapped" with a geofence reserved over an approved flight time window. *Inverse volumization* is the opposite process

in which a flight is planned to always remain within a designated airspace geofence volume. This chapter extends our work in [38] in several ways. First, we integrate the individual airspace volumization algorithms into a geofencing pipeline described in Section 2.3. This geofencing pipeline is shown in Figure 2.2. We also construct simplified keep-in/keep-out 3-D geofencing boundaries based on buildings and UAS flight plans, as illustrated in Figure 2.1. This algorithm uses parameters such as vehicle speed, geofence boundary safety buffer size, and polygon simplification parameters to generate a flight plan that does not violate keep-in/keep-out geofences in the surrounding region. We define a trajectory keep-in geofence as the airspace volume surrounding the planned flight path with constant ceiling and floor safety buffers. Pathfinding logic is developed for different start and desired end locations of a vehicle in the flight plan. The algorithms are built on computational geometry, where obstacles, buildings, and flight path keep-in geofences are represented as sets of 3-D polygons. Path planning modules are computed efficiently based on a visibility graph approach and set operations in a 3-D environment.

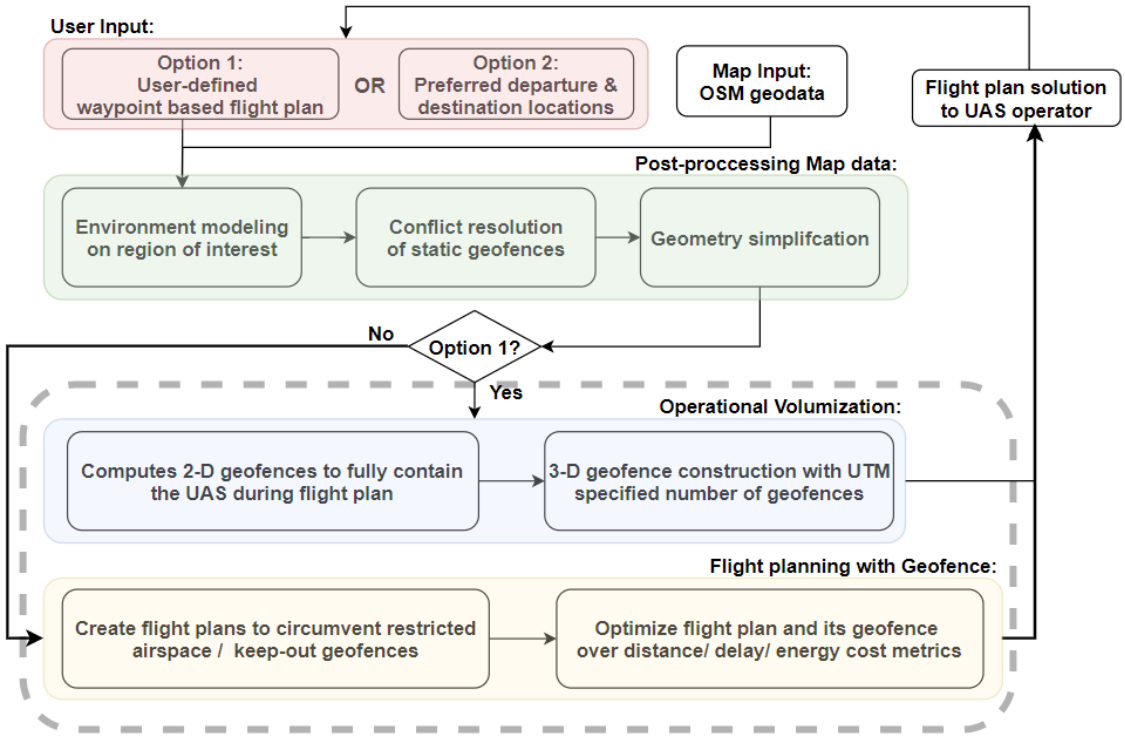


Figure 2.2: Airspace and environment geofencing functionality and data flow.

This work is unique in its joint consideration of low-altitude mapped obstacles and geofence volume requirements in 3D multicopter sUAS flight planning. Urban terrain and building maps necessarily create more complex flight paths and safety constraints [39]. As

an example, consider package delivery UAS in an urban canyon environment. Safe operation requires obstacle-free path planning for all sUAS operating in this shared low-altitude airspace. Planned sUAS paths must therefore treat both physical obstacles (e.g., buildings, power lines, and terrain) and keep-out geofences as impenetrable obstacles that must be circumvented in a safe flight plan. Our work bridges the gap in the existing geofencing literature by focusing on path planning solutions that assure the satisfaction of keep-in/keep-out geofenced airspace volume constraints.

The contributions of this chapter are:

- The specification of formal algorithms to define keep-in/keep-out geofences for obstacles to plan UAS paths with separation assurance;
- The integration of airspace and environmental geofencing processing pipelines with user inputs to construct geofences and geofence-wrapped path plans in a real-world urban environment;
- Map data processing to generate keep-out geofences around buildings and terrain and a process to simplify a detailed map dataset to support a more compact representation and improved path planning efficiency;
- A benchmark comparison of our geofenced path planning solutions with a fixed sUAS airway flight corridor design, and a case study of sUAS route deconfliction in shared airspace.

The remaining structure of this chapter is organized as follows. Section 2.2 summarizes previous work in UAS Traffic Management (UTM), sUAS and robotic path planning, and computational geometry methods used in geofencing algorithms. Section 2.3 defines an airspace geofence, states assumptions made in this work, and introduces sUAS geofencing pipeline algorithms used in the generation of flight trajectory solutions. Section 2.4 describes OpenStreetMap (OSM) data processing steps to minimize computational time in generating solutions. Section 2.5 describes Monte Carlo simulation setups that integrate pipeline algorithms with map data processing. Section 2.6 presents statistics comparing results from our airspace volumization algorithm with a fixed airway flight corridor solution for a region of Manhattan in New York City. Section 2.7 describes a case study for sUAS route deconfliction, while Section 2.8 concludes the chapter.

2.2 Literature Review

This section presents related work in UTM, computational geometry, and path planning, all of which are relevant to our geofencing algorithms.

2.2.1 Unmanned Aircraft System Traffic Management and Geofencing

UTM has been identified as a critical capability for future small UAS operations due to their unique operating profiles at low altitude, near complex infrastructure, and likely in mixed-use airspace [25]. UTM-like concepts have been investigated by industry, government, and academia across the globe. As an example, Single European Sky ATM Research (SESAR) recommended UTM to the European Union (EU) to safely coordinate UAS [40]. Centralized and distributed UTM with airspace volumes distinguished by altitude layer was investigated to deconflict UAS traffic in Sweden [41]. UTM was modeled using a multiplayer network of nodes and airways at low-altitude airspace in Luxembourg [31]. The National Aeronautics and Space Administration (NASA) perhaps first coined the term UTM as a system architecture necessary to accommodate UAS in a low-altitude National Airspace System (NAS) layer not frequently occupied by legacy manned aircraft [25]. Representatives from industry have worked to establish adequate security protocols for managing UTM datalinks [26]. NASA, in cooperation with industry, has pursued a series of flight test events to evaluate cooperative UAS operations in beyond visual line of sight (BVLOS) conditions with a "separation by segregation" geofence design [36]. Airspace capacity estimation was analyzed using keep-in and keep-out geofences in [42]. A roadmap for geofence implementation in urban areas with 5G networks and blockchain was introduced in [43].

Dynamic airspace geofencing algorithms are novel to UTM. Two different but equally important perspectives (i.e., local/global) exist in geofencing designs. One perspective is a classical guidance/navigation/control (GNC) approach, where geofence layering is only generated for the individual UAS that has full knowledge of its control system. This vehicle-centered geofence perspective focuses on controlling UAS to ensure that the vehicle does not violate the geofence boundaries (given expected trajectory tracing errors) [35, 44]. In this work, each UAS monitors its real-time state vector relative to geofence boundaries to detect and react to potential breaches given uncertainties due to sensor errors and wind disturbances.

Vehicle-centered geofencing research is important but does not consider properties of the operating area airspace or the ground-based environment. Geofencing has also been researched from an *airspace system* perspective. With this viewpoint, geofences are managed

by UTM to organize airspace structure and improve Situational Awareness (SA). UTM will not model individual UAS capabilities and uncertainties in detail, but it can conservatively monitor UAS travel through an approved geofence to offer impending breach warnings to the UAS and actual boundary violations to all traffic per [45, 33].

SA is a fundamental requirement for all flight operations, autonomous or human-piloted [46, 47]; while legacy air Traffic Management (ATM) will remain distinct from UTM in the near term, advanced air mobility (AAM) supporting increasingly to fully autonomous flight will motivate the integration of ATM and UTM over the long term. UTM calls for the automation of airspace management tasks. Airspace organization and protection through geofencing can improve SA and in turn safety. Our algorithms can be integrated into both GNC (onboard) and airspace system (UTM) geofencing realizations.

AAM operations, including but not limited to Urban Air Mobility (UAM), are envisioned to have higher altitudes than 400 AGL, where current UTM is designed to serve. Researchers at NASA and Uber investigated the applicability of UTM to coordinate UAM routes safely and efficiently [48]. In their case studies, "Transit-Based Operational Volumes (TBOVs)" were used to wrap the UAM flight path, a notion analogous to the trajectory keep-in geofence discussed in this chapter. Inspired by the static "UAM-authorized airspace" active over a fixed duration [48] as an airspace management alternative to geofencing, in our case studies, we designed fixed flight corridors and simulated sUAS flight missions operating in these flight corridors. This alternative solution offers a benchmark with which our dynamic geofence volumization and path planning solutions are compared (2.5).

2.2.2 Computational Geometry

Computational geometry has been used to construct and deconflict airspace geofence volumes and to detect/prevent airspace boundary violations (onboard). Scaling algorithms have been developed for two-dimensional keep-in/keep-out concave polygon geofences with consideration of steady wind in [18] and with warning and override layers in [49]. This chapter uses vehicle performance constraints and steady wind conditions to generate scaled warning and override geofence boundaries. Once a UAS crosses one of these boundaries, onboard guidance, navigation, and control (GNC) can trigger a corrective response [50] or flight termination. In [51, 52, 53], algorithms for polygon set operations (i.e., polygon intersections and unions), polygon clipping, convex hull, and point-in-polygon were developed. We use these algorithms to detect and resolve geofence boundary conflicts and generate new geofence volumes by merging conflicting boundaries. A UAS geofence violation detection method was defined in [32] using Ray Casting [54]. A Triangle Weight Characterization

with Adjacency (TWCA) algorithm was developed as a faster real-time geofence violation detection method in [33]. TWCA divides geofence into a finite number of triangles and then finds UAS location in a pre-generated adjacency graph. In [34], a 3-D dynamic geofence ("moving geofence") was constructed using the maximum cruise time, speed, and range of the UAS as a pre-departure flight planning algorithm. This chapter also proposes a convex hull approach to find conflicts between current and newly submitted flight plans.

2.2.3 Path Planning

Determining a collision-free geofence-based flight trajectory is central to the design of our geofencing volumization work. A variety of path planning algorithms were considered. Grid-based path planning methods overlay a fixed-resolution grid on top of the configuration space and find discretized line segment paths connecting start state to destination. This search is fast in low-dimensional space but quickly becomes computationally intractable with high-resolution maps and appreciable travel distance. The most notable grid-based path planning algorithms are $\mathcal{A}^*$ [55] and $\mathcal{D}^*$ [56]. A family of roadmap-based path planning algorithms have been developed to offer a more compact search space optimizing a specific cost metric. For example, a visibility graph [57] minimizes travel distance, while a Voronoi diagram maximizes obstacle clearance distance [58]. The application of cell decomposition [59] offers a compact map for discrete search path planning in an obstacle field. Other path planning methods include potential-field algorithms [60] that efficiently build plans with gradient descents but are subject to local minima issues. Sampling-based path planning algorithms [61] have also been developed and are particularly well suited to planning in uncertain environments. Our work utilizes a visibility graph approach to path planning. This approach allows us to directly translate geofence volumes generated with computational geometry into visibility graph roadmaps. As is discussed below, we scale keep-out geofences to assure safe separation is maintained. Note that a visibility graph does not require a rasterized map, enabling geofences to be represented without distortion or approximation.

2.3 Definitions and Algorithms

The term airspace geofence was formally defined in [19] to support a common framework for airspace volume reservation in UTM. Our work follows this definition:

Definition 1 *A Geofence* $g = \{n, v[\ ], z_f, z_c, m, h[\ ]\}$ *is a volume defined by a list of* n *vertices in the horizontal plane* $v = [(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)]$*, where* $n \geq 3$*, and an altitude floor* z_f *and ceiling* z_c*. The volume is defined relative to a set of home locations,* $h_i =$

(x_i, y_i, z_i, t_i), *where* $h[\,]$ *is a list of length* $m \geq 2$. *Lateral home positions can be represented as latitude/longitude pairs* (ϕ_i, λ_i) *or locally referenced Cartesian coordinates* (x_i, y_i). z_i *is the altitude of the home location above Mean Sea Level (MSL).* t_i *is the activation time for home location* i *where* $1 \leq i \leq m$. t_m *is the deactivation time for geofence* g. *For consistency, Cartesian coordinates and altitudes are defined in meters and activation/deactivation times are in seconds.*

This data structure supports geofence types: static, durational, and dynamic. A *static geofence* has a permanent fixed home location $h[\,]$ and typically surrounds physical obstacles such as buildings or sensitive areas (i.e., no-fly zones). A *durational geofence* is active over a finite time interval with a fixed home location $h[\,]$. A *dynamic geofence* is active over a specific time interval; its *home* location can move over time.

The following simplifying assumption is made in this chapter to facilitate path planning and eliminate the need for traffic deconfliction.

Assumption 2 *One aircraft (e.g., UAS) is allocated to each local geofence volume. No other UAS is permitted to cross into this volume. UTM efficiency therefore relies on minimizing each reserved geofence volume and its duration.*

Dynamic airspace volumization for geofencing will enhance safety by wrapping a UAS in an airspace volume that assures separation from other traffic. The below subsections describe our geofencing algorithm pipeline for UTM, where flight plans are designed with keep-in/keep-out geofencing volumes on a low-altitude airspace map. Three-dimensional trajectory keep-in geofence volumes safely wrapping UAS flight paths are described in Section 2.3.1, keep-out geofence construction for a low-altitude urban map is described in Section 2.3.2, and geofence-based path planning solutions are illustrated in Section 2.3.3.

2.3.1 Airspace Operational Volumization

Operational volumization constructs a trajectory keep-in geofence overlaid on a user-defined 3-D flight trajectory. Climb and descent segments are first generated with vehicle dynamics inputs such as velocity and desired time to climb/descend. Then, three-dimensional cruise operational volumes are created between the climb and descent geofence pair. This assures a geofence volume always encloses the flight trajectory with the prescribed safety buffer $\delta_{vehicle}$. This algorithm integrates 2-D flight trajectory operational volumization with the Multiple Staircase Geofence (MSG) algorithm per [38]. Three-dimensional trajectory volumization is shown in Algorithm 1. Figure 2.3 shows an example of a 3-D trajectory with its corresponding three-dimensional geofence volume. A sequence of geofence volumes is constructed by connecting climb, cruise, and descent geofences with user-specified safety buffers.

Algorithm 1: 3D Flight Trajectory Operational Volumization (***3dOperVol***)

Inputs: 2-D Trajectory waypoints $\mathcal{W}$, Velocity $\mathcal{V}$, Time to Climb t_{climb}, Time to Descent t_{desc}, Number of Geofence $\mathcal{N}_{geo}$, UAS Safety Buffer $\delta_{vehicle}$, Cruise Altitude h_{cruise}

Outputs: 3-D Flight Trajectory $\mathcal{P}_{traj}$, 3-D Geofence for 3-D Flight Trajectory $\mathcal{G}$

Algorithm:

1: $[\mathcal{P}_{climb}, \mathcal{G}_{climb}] \leftarrow MSG(\mathcal{W}[1:2], \mathcal{V}, t_{climb}, \mathcal{N}_{geo}, \delta_{vehicle})$ ◁ generate climb geofence
2: $[\mathcal{P}_{desc}, \mathcal{G}_{desc}] \leftarrow MSG(\mathcal{W}[end-1:end], \mathcal{V}, t_{desc}, \mathcal{N}_{geo}, \delta_{vehicle})$ ◁ descent geofence
3:
4: $\mathcal{P}_{cruise} \leftarrow [\mathcal{P}_{climb}[end-1:end], \mathcal{W}[3:end-2], \mathcal{P}_{desc}[1:2]]$ ◁ 3-D Cruise flight
5: $[\mathcal{G}_{cruise}] \leftarrow MSG(2dOperVol(\mathcal{P}_{cruise}, \delta_{vehicle}), h_{cruise})$ ◁ Generate cruise geofence
6: $\mathcal{P}_{traj} \leftarrow [\mathcal{P}_{climb}; \mathcal{P}_{cruise}; \mathcal{P}_{desc}]$
7: $\mathcal{G} \leftarrow [\mathcal{G}_{climb}; \mathcal{G}_{cruise}; \mathcal{G}_{desc}]$
8: **return** $[\mathcal{P}_{traj}, \mathcal{G}]$

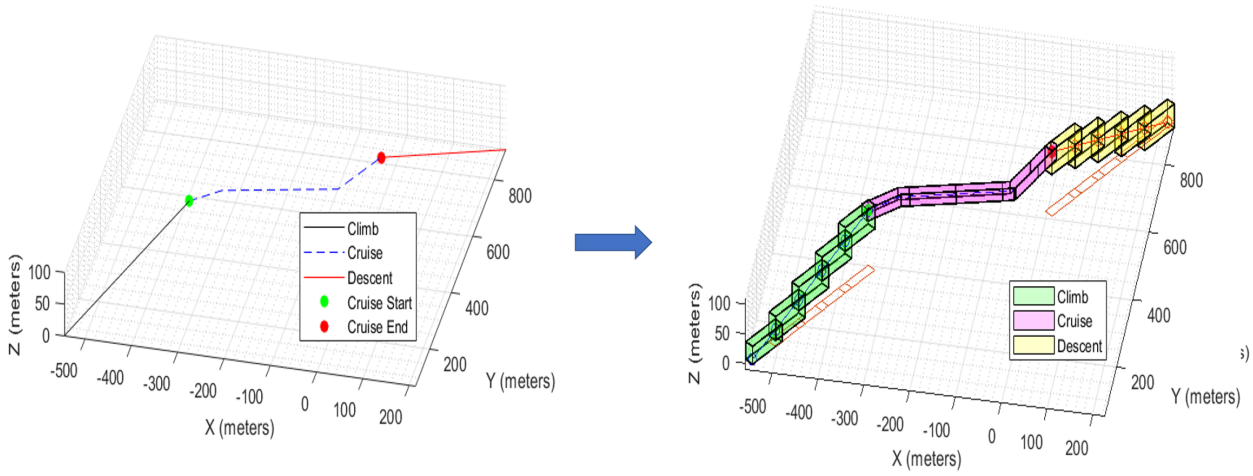


Figure 2.3: Example application of Algorithm 1. A sample 3-D flight path is shown on the left. A corresponding flight trajectory keep-in geofence is shown on the right.

To minimize airspace volume reservation duration, we utilize the shrinking durational geofence (SDG) and multi-stage durational geofence (MDG) algorithms in [38] for the cruise segment. A shrinking durational geofence (SDG) removes a previously occupied geofence volume at each time update in UTM. A multi-stage durational geofence (MDG) has multiple volumes generated over the flight trajectory with temporal or spatial overlap. For transitions between MDG regions, either temporal or spatial overlap is used to guarantee the UAS is always enclosed by at least one MDG. Overlap offers a buffer in case the UAS flies faster or slower than expected. Note that climb and descent segments utilize multiple staircase geofences so that previously occupied staircase geofences can be removed sequentially.

2.3.2 Constructing a Geofence Volume from an Urban Map

Keep-out geofences are constructed around obstacles (i.e., buildings) to assure separation between UAS and obstacles or no-fly airspace zones. The construction of keep-out geofence volumes from a building and terrain map must be efficiently carried out to constrain the computation time needed to generate geofence-based path planning solutions. For this work, we utilize a visibility graph approach to path planning, as illustrated in Section 2.3.3. The time complexity of visibility graph generation is $O(n^2 log(n))$, where n is the total number of vertices in all polygons. In a real-world environment, the number of keep-out geofences in the urban environment can be significant (i.e., 14,000 building cluster geofence polygons in the southern Manhattan map). We utilize two algorithms to achieve map simplification. First, we downsample geofence vertices in the map as shown in Algorithm 2 per [62] with user-defined parameters $n_{maxVert}$ and $p_{dwnSmple}$. This updated set of keep-out geofences is then used to construct a region of interest (ROI) visibility graph. The ROI in the map is first constructed as a rectangular box surrounding departure and destination points. Then, polygon intersection, point-in-polygon, and convex hull operations are used to define the actual region of interest for which geofence-based path planning solutions are generated. Generation of the flight planning visibility graph ROI is shown in Algorithm 3. Figure 2.4 shows an example of polygon vertex set downsampling. Figure 2.5 illustrates an initial rectangular ROI P_{recROI} example.

Algorithm 2: Reduce Map Geofence Vertex Set

Inputs: Set of Keep-out Geofences $\mathcal{S}_{geo}$, Downsample Threshold $n_{maxVert}$, Downsample Tolerance In Percentage $p_{dwnSmple}$
Outputs: Set of Downsampled Keep-out Geofences $\mathcal{S}_{ds}$
Algorithm:

$\mathcal{S}_{ds} \leftarrow [\quad]$ ◁ initialize the output set
for $\mathcal{S} \in \mathcal{S}_{geo}$ **do**
  if $len(\mathcal{S})/2 > n_{maxVert}$ **then**
    $\mathcal{S}_{out} \leftarrow DecimatePoly(\mathcal{S}, p_{dwnSmple})$ ◁ downsample polygon vertices
    $k \leftarrow 1$
    for $j = 1 : len(\mathcal{S}_{out})$ **do**
      $\mathcal{G}[k : k+1] \leftarrow \mathcal{S}_{out}[j, 1:2]$ ◁ obtain geofence data structure
      $k = k + 2$
    end for
  end if
  $\mathcal{S}_{ds} \leftarrow \mathcal{S}_{ds}.add(\mathcal{G})$
end for
return $\mathcal{S}_{ds}$

Algorithm 3: Compute Visibility Graph ROI

Inputs: Departure Point $\mathcal{P}_{start}$, Destination Point $\mathcal{P}_{end}$, ROI Inital Buffer δ_{ROI}, Keep-out Geofence Set $\mathcal{S}_{geo}$

Outputs: Keep-out Geofences in ROI $\mathcal{S}_{ROI}$

Algorithm:

$P_{recROI} \leftarrow getRecROI(\mathcal{P}_{start}, \mathcal{P}_{end}, \delta_{ROI})$ ◁ get Rectangular ROI vertices

//get convexhull ROI where geofencing solutions are generated

$S_{intersect} \leftarrow [\quad]$ ◁ initialize the intersecting geofence set

for $\mathcal{S} \in \mathcal{S}_{geo}$ **do**

  if $searchIntersect(S, P_{recROI}) \neq \emptyset$ **then**

    $\mathcal{S}_{intersect} \leftarrow \mathcal{S}_{intsct}.add(S)$ ◁ Append intersecting geofence

  end if

end for

//Search keep-out geofences inside the convex hull P_{ROI}

$S_{ROI} \leftarrow [\quad]$ ◁ initialize S_{ROI}

$P_{ROI} \leftarrow convexHull(S_{intersect})$ ◁ ROI where geofencing solutions are generated

for $\mathcal{S} \in \mathcal{S}_{geo}$ **do**

  if $searchIntersect(S, P_{ROI}) \neq \emptyset$ **then**

    $\mathcal{S}_{ROI} \leftarrow \mathcal{S}_{intsct}.add(S)$ ◁ Append intersecting geofence

  end if

end for

return S_{ROI}

2.3.3 UAS Flight Planning in a Geofenced UTM Airspace

Flight plans are typically optimized over distance, energy usage, and flight time (delay) cost metrics. A UAS configuration space is first obtained from user-defined safety buffers $\delta_{vehicle}$, $\delta_{building}$ around the vehicle and obstacles, respectively. The UAS can then be treated as a point mass in configuration space with obstacle boundaries expanded for safety by:

$$\delta_{sb} = \delta_{vehicle} + \delta_{building}. \tag{2.1}$$

This safety buffer ensures the vehicle maintains sufficient clearance from any obstacles. $\delta_{vehicle}$ and $\delta_{building}$ are user-specified parameters in this work.

Our proposed geofencing pipeline applies three inverse volumization options per [38] based on user-specified departure and destination locations. The first option is a "turn" solution that calculates climb, cruise, and descent flight trajectories that turn away from nearby obstacles, maintaining a minimum-distance path from start to end. For this module, a low-dimensional visibility graph search with Dijkstra's algorithm [63] plans paths around

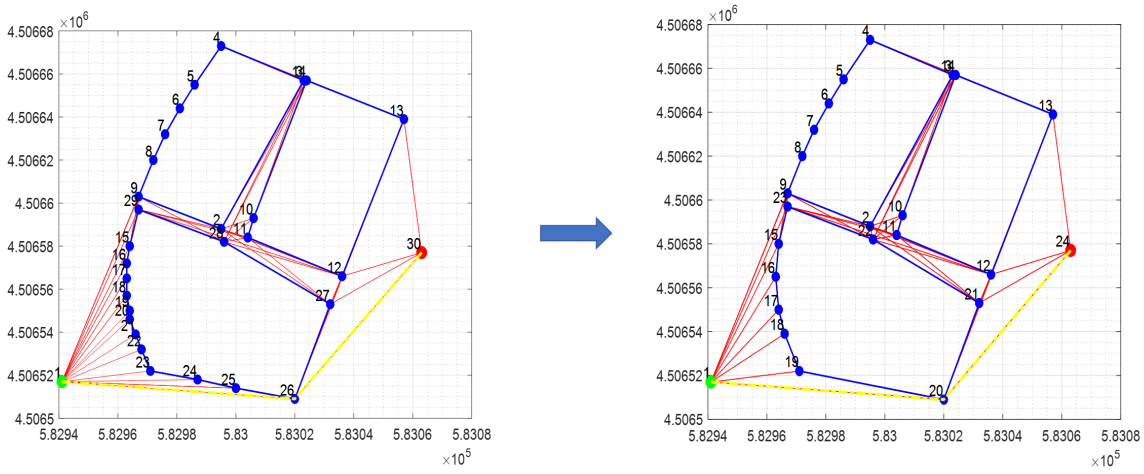


Figure 2.4: Example of reducing the number of vertices to simplify the associated visibility graph. The left illustration shows three original polygons. The right illustration shows the polygons after applying the vertex downsampling algorithm. $n_{maxVert}$ and $p_{dwnSmple}$ are 15 and 60%, respectively. The time complexity of visibility graph generation is $O(n^2 log(n))$, where n is the total number of vertices in all polygons. The number of vertices in the lower polygon illustrated here is reduced from 15 to 9.

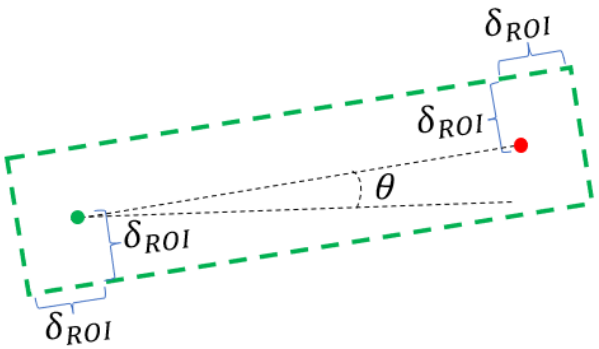


Figure 2.5: Illustration of rectangular ROI generation. Start point, destination point, and ROI initial buffer size δ_{ROI} are used to initialize the rectangular ROI per Algorithm 3.

obstacles (i.e., polygons) defined in a local Cartesian frame. We modeled keep-out geofences on obstacles as open set 3-D polygons extruded from 2-D obstacles with fixed heights. Per Section 2.2.3, an obstacle-free visibility graph or roadmap space can be constructed from geofence and obstacle polygons without rasterization [64, 57].

The second path planning option is a "constant cruise altitude climb" module for which the UAS climbs over no-fly and obstacle volumes until a direct-heading route to the destination is obstacle-free. For this option, a vehicle first climbs to a pre-determined cruise altitude greater than the highest building en route to the destination. Then, the vehicle

flies directly to the destination at cruise altitude. As the vehicle approaches the end of its cruise segment, it descends to the destination free of obstacles along the path. The third path planning option is a "vertical terrain follower" module, where a UAS follows the terrain altitude profile en route to the destination, flying as low as possible. This solution minimizes the time a UAS will spend at a high altitude potentially in conflict with other transiting traffic, but it adds complexity to the altitude profile. Figure 2.6 shows examples of turn, constant cruise altitude, and terrain follower climb solutions per [38].

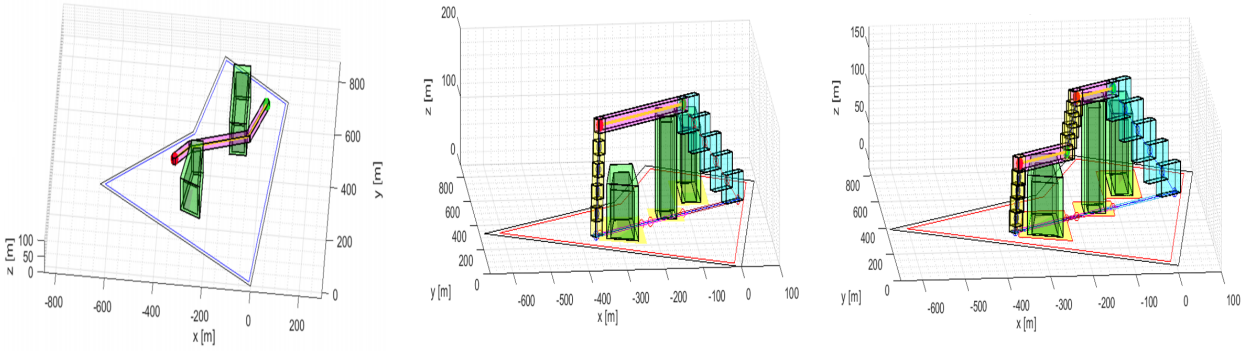


Figure 2.6: Three candidate flight planning solutions respecting keep-out airspace geofence and obstacle "no-fly" volumes. A turn solution uses a visibility graph to define a constant-altitude path around no-fly zones (**left**). A cruise altitude solution climbs to an altitude greater than the highest building enroute to the destination (**center**). The terrain follower defines an altitude profile maintaining minimum safe clearance or greater from no-fly zones (**right**).

To determine which of the three solutions is best, a weighted cost function over time, distance, and energy is defined:

$$\mathcal{C} = \alpha * d_{travel} + \beta * P_{travel} + \gamma * t_{wait}. \tag{2.2}$$

where d_{travel}, P_{travel}, and t_{wait} are distance traveled, power consumption, and time delay until durational geofences disappear, respectively. Weighting factors $\alpha, \beta, and \gamma$ are user-defined. The path planning solution with minimum cost is then suggested to an operator and/or automation. The flight planning process with geofencing is shown in Algorithm 4. In this algorithm, the departure point, destination point, cruise altitude, and keep-out geofence boundary coordinates are input along with cruise velocity and climb/descent times. For the turn solution, a Rotational Plane Sweep (RPS) algorithm is used to find all straight-line segments connecting line-of-sight vertices to form a visibility graph map. Then, Dijkstra's algorithm finds the minimum distance path from departure to destination point. For con-

stant altitude climb and terrain follower solutions, points of intersection between a straight line solution path and obstacles are found using a polygon-line intersection operator. Then, obstacle height at the intersection points are extracted from keep-out geofence data. Three-dimensional flight trajectory "turn", "constant cruise altitude", and "terrain follower" solutions are wrapped with geofences using Algorithm 1. The best solution is the minimum cost module based on Equation (2.2). Note that geofence segment duration is not explicitly considered in this chapter. Instead, it is assumed the flight trajectory keep-in geofence generated using Algorithm 4 remains active from UAS launch to landing.

2.4 Environment Modeling

2.4.1 Map Data Processing

To evaluate the proposed geofencing capability in a complex low-altitude environment, we processed OpenStreetMap (OSM) data for the Manhattan Borough of New York City (USA). OSM is a collaborative global mapping project that creates geographical data and information [65]. OSM is frequently updated and provides map entities including airways, roads, buildings, and more. To minimize map processing overhead for this work, we used pre-processed georeferenced OSM Manhattan building data as described in Ref. [66]. This raw data contain building coordinates represented as polygon vertices, building heights, and street level in WGS 84/UTM zone 18N [67], where units are in meters with East, North, Up (ENU) axes. We applied a combination of set and convex hull [57] operations to simplify geofence geometry for flight planning. Figure 2.7 shows the flowchart for map data post-processing. After post-processing, the dataset was partitioned into four categories: buildings with heights greater than 20 m, 60 m, 122 m, and 400 m. Depending on sUAS start and end altitude (i.e., roof of building, ground), flight planning utilizes one of these four datasets to generate plans and associated geofence volumes.

Figure 2.8 shows a map of southern Manhattan with closely spaced building clusters each enclosed by a single keep-out geofence to simplify the Manhattan urban canyon map. Figure 2.9 shows an example of post-processed georeferenced data and its 3-D keep-out geofence.

A southern Manhattan, New York City map was defined by 14,000 building cluster geofence polygons using the above procedure. To further simplify the map, we downsampled geofence vertices and construct an updated set of keep-out geofences from the ROI visibility graph per Algorithms 2 and 3 in Section 2.3.2. Figure 2.10 shows an example of the rectangular ROI, ROI obstacle polygon, and visibility graph generation pipeline. The "turn" flight planning visibility graph was constructed from keep-out geofences inside the ROI along with

Algorithm 4: Flight Planning With Geofencing

Inputs: Departure Point $\mathcal{R}_{start}$, Destination Point $\mathcal{R}_{end}$, Cruise Altitude h_{cruise}, Keep-out Geofence Boundaries $\mathcal{S}_{geo}$, Aircraft Velocity $\mathcal{V}$, Time to Climb t_{climb}, Time to Descend t_{desc}, Number of Geofences $\mathcal{N}_{geo}$, UAS Safety Buffer $\delta_{vehicle}$

Outputs: Planned Flight Trajectory $\mathcal{P}_{traj}$, Trajectory-wrapping 3-D Geofence Volumes $\mathcal{G}$

Algorithm:

1: //turn solution module
2: $\mathcal{R}_{VG} \leftarrow [\mathcal{R}_{start}; \mathcal{S}_{geo}; \mathcal{R}_{end}] \triangleleft$ Vertices of Visibility Graph
3: $[edges, vert\_ID] \leftarrow RPS(\mathcal{R}_{VG}) \triangleleft$ get visibility graph edges on the map using RPS
4: $[R_{turn}] \leftarrow dijkstraPath(\mathcal{R}_{start}, \mathcal{R}_{end}, edges, vert\_ID) \triangleleft$ get min. distance path
5: $[\mathcal{P}_{turn}, \mathcal{G}_{turn}] \leftarrow 3dOperVol(\mathcal{R}_{turn}, \mathcal{V}, [t_{climb}, t_{desc}], \mathcal{N}_{geo}, \delta_{vehicle}, h_{cruise})$
6: $\mathcal{D}_{turn} \leftarrow getDist(\mathcal{P}_{turn}) \triangleleft$ get turn module flight distance
7:
8: //climb solution modules
9: $\mathcal{R}_{intersect} \leftarrow searchIntersect(\mathcal{R}_{VG}) \triangleleft$ get intersections from $[\mathcal{R}_{start}; \mathcal{R}_{end}]$ to $\mathcal{S}_{geo}$
10: **if** $\mathcal{R}_{intersect} \neq \emptyset$ **then**
11: $h_{intersect} \leftarrow extractHeight(\mathcal{R}_{intersect}, \mathcal{S}_{geo}) \triangleleft$ get heights at intersections
12: $h_{max} \leftarrow max(h_{intersect})$
13:
14: //constant cruise altitude
15: $[\mathcal{P}_{const}, \mathcal{G}_{const}] \leftarrow 3dOperVol(\mathcal{R}_{intersect}, \mathcal{V}, [t_{climb}, t_{desc}], \mathcal{N}_{geo}, \delta_{vehicle}, h_{max})$
16: $\mathcal{D}_{const} \leftarrow getDist(\mathcal{P}_{const}) \triangleleft$ get constant altitude cruise flight distance
17:
18: //terrain follower
19: $[\mathcal{P}_{terr}, \mathcal{G}_{terr}] \leftarrow 3dOperVol(\mathcal{R}_{intersect}, \mathcal{V}, [t_{climb}, t_{desc}], \mathcal{N}_{geo}, \delta_{vehicle}, h_{intersect})$
20: $\mathcal{D}_{terrain} \leftarrow getDist(\mathcal{P}_{terr}) \triangleleft$ get terrain follower flight distance
21: **end if**
22:
23: //cost comparison
24: $[\mathcal{C}_{min}, opt] \leftarrow costCompare(\mathcal{D}_{turn}, \mathcal{D}_{const}, \mathcal{D}_{terrain})$
25: **if** opt == 1 **then**
26: $[\mathcal{P}_{traj}, \mathcal{G}_{traj}] \leftarrow [p_{turn}, \mathcal{G}_{turn}] \triangleleft$ best sol: turn module
27: **else if** opt == 2 **then**
28: $[\mathcal{P}_{traj}, \mathcal{G}_{traj}] \leftarrow [p_{const}, \mathcal{G}_{const}] \triangleleft$ best sol: constant cruise altitude module
29: **else**
30: $[\mathcal{P}_{traj}, \mathcal{G}_{traj}] \leftarrow [p_{terr}, \mathcal{G}_{terr}] \triangleleft$ best sol: terrain follower module
31: **end if**
32: **return** $[\mathcal{P}_{traj}, \mathcal{G}_{traj}]$

departure and destination locations.

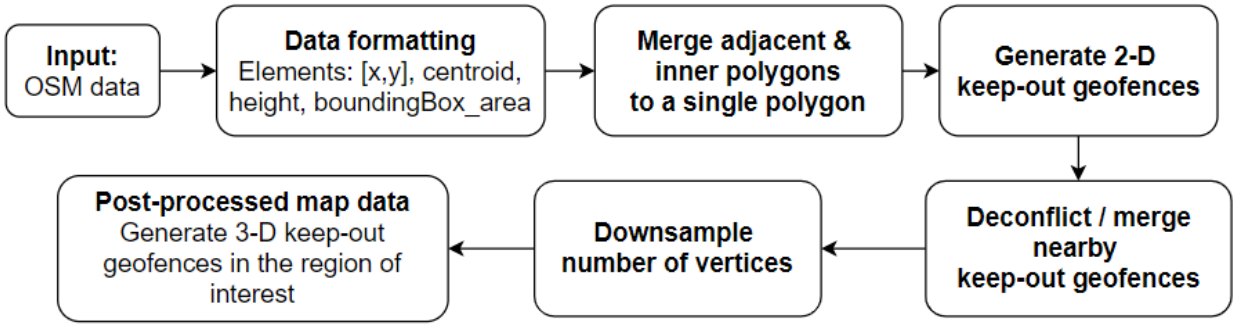


Figure 2.7: Flowchart of post-processing map data. OSM data were converted to a MATLAB format, then processed using polygon set convex hull operators to reduce the number of keep-out geofences in the region of interest (ROI), the area between departure and destination points. If the number of vertices in a geofence is greater than threshold $n_{maxVert}$, it is downsampled to $p_{dwnSmple}$. $n_{maxVert}$ and $p_{dwnSmple}$ are user-defined parameters set to 15 and 60%, respectively, in this work. Algorithms 2 and 3 are used in finding ROI and reducing number of map vertices. Three-dimensional keep-out geofences around buildings are generated with safety buffer $\delta_{building}$.

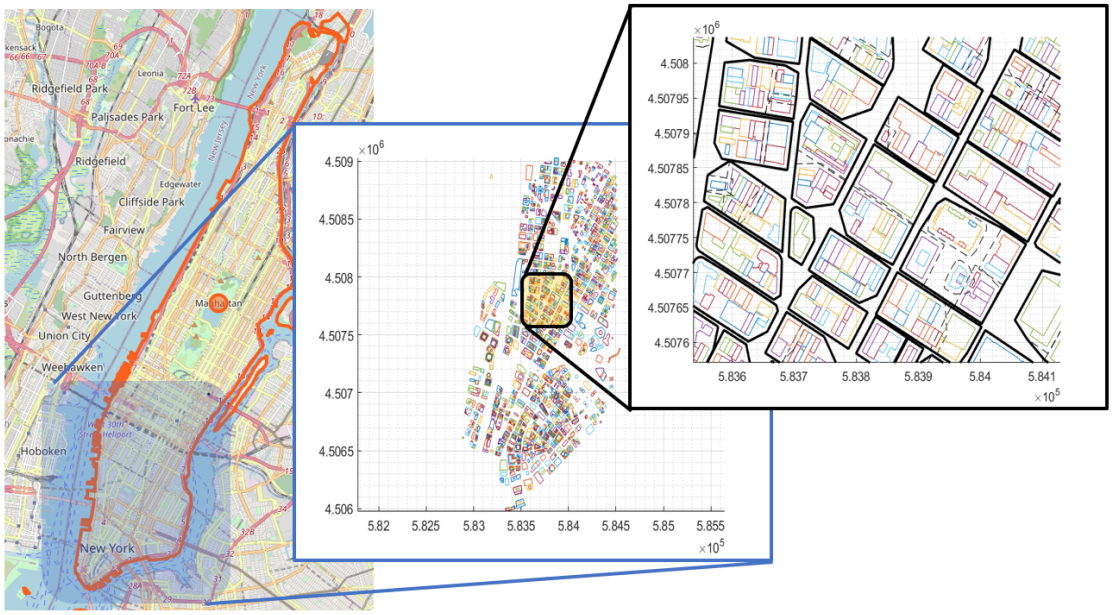


Figure 2.8: Post-processing map data for southern Manhattan. Buildings with heights greater than 20 m are shown. The rightmost plot shows keep-out geofences enclosing building clusters (black solid lines), individual building keep-out geofences (black dashed lines), and building outlines (colored lines). Geofence maps for 60 m, 122 m, and 400 m altitude cross-sections are constructed in the same manner.

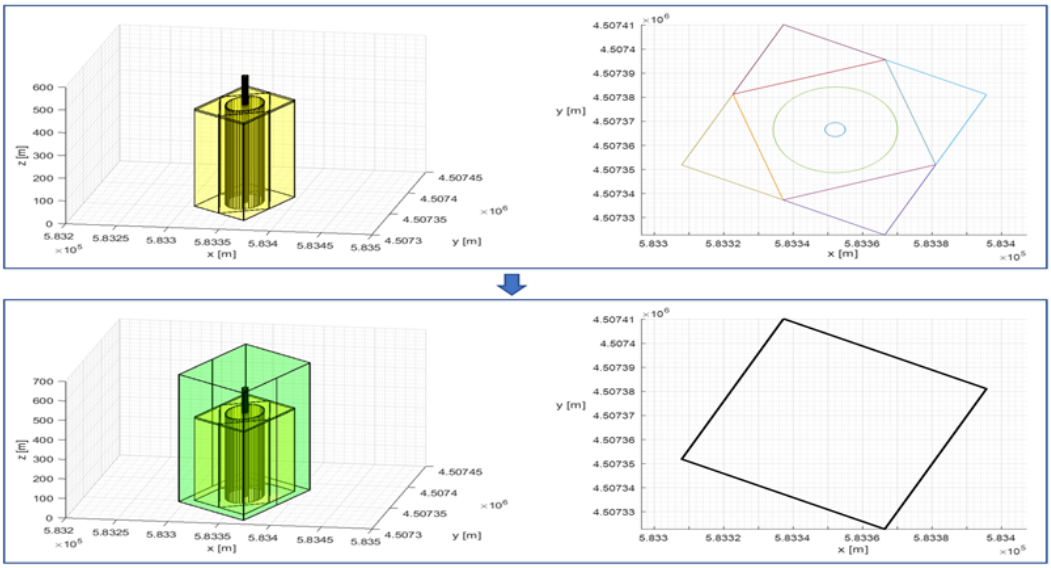


Figure 2.9: Post-processed georeferenced data for the One World Trade Center building in Manhattan. The top left and right show raw OSM data side and top views, respectively. The bottom left and right show post-processed keep-out geofence data (shaded in green) side and top views, respectively.

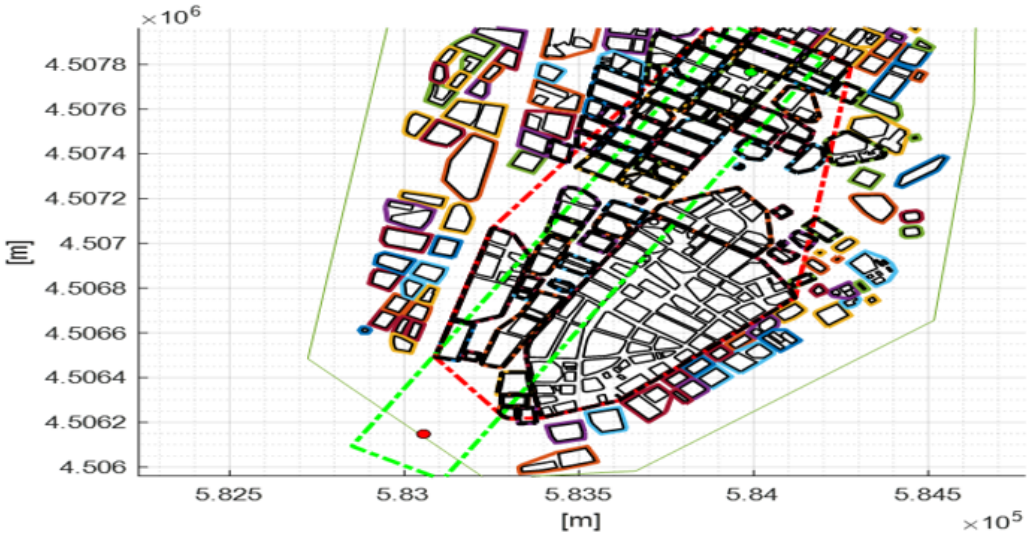


Figure 2.10: Keep-out geofence polygon extraction for UAS flight planning. The initial ROI (green dashed line) is a rectangular box per Figure 2.5. Keep-out geofences (solid black lines) inside or intersecting the rectangular ROI box are found using polygon intersection and point-in-polygon operations. The final ROI (red dashed line) is the convex hull around these keep-out geofences. For our simulation, $\delta_{ROI} = 150$ m.

2.5 Simulation Setup

Monte Carlo simulations were used to evaluate proposed airspace volumization strategies on the Manhattan map. Figure 2.11 shows the flowchart of pathfinding logic in our simulation setup. Pathfinding logic comprises four solution modules for the airspace geofencing algorithm. Once the start and end locations were defined, the keep-out geofence ROI polygons (Figure 2.10) were extracted from post-processed map data. Constant cruise altitude and terrain follower modules were generated by searching the intersection points between the buildings' keep-out geofences and the line that connects UAS start and end waypoints. A pure turn solution was generated if both start and end locations were on the ground. If either start or end location was on the roof of the building (i.e., inside of the keep-out geofence), a constant cruise altitude algorithm was first used to find the flight path from the start/end point to the outside of the keep-out geofence, and the turn module solution was used to calculate the remaining flight path, creating a combined solution.

Control parameters are shown in Table 2.1. To offer an experimentally grounded dataset, a prototype quadplane's power consumption model [10] was used per Table 2.2 to compute P_{travel} in climb, cruise, and descent segments. A quadplane is a hybrid quadrotor/fixed-wing UAS designed to vertically takeoff and land in an urban environment. For our simulations, the quadrotor motors were active in all phases of flight; cruise power would otherwise be lower. Cost function weighting factors $\alpha = 0.6,\ \beta = 0.2,\ \gamma = 0.0$ were chosen to prioritize minimum-distance solutions. Note that γ was set to zero because building obstacles have static or permanent geofences.

Table 2.1: Control parameters for geofenced flight planning case studies.

$\boldsymbol{\mathcal{V}_{vehicle}}$	$\boldsymbol{\delta_{vehicle}}$	$\boldsymbol{\delta_{building}}$	$\boldsymbol{N_{geofence}}$	$\boldsymbol{z_{cruise}}$
5 (m/s)	2 (m)	5 (m)	5	50 (m)

Table 2.2: Flight power consumption data from [10].

Climb	**Descent**	**Forward Flight**
312 (J/s)	300 (J/s)	328 (J/s)

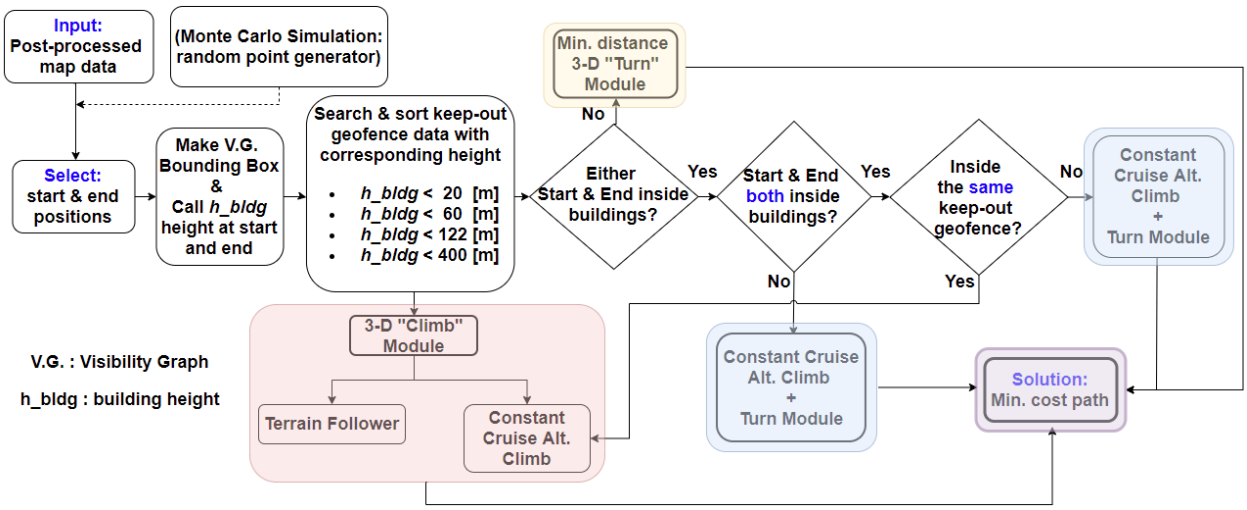


Figure 2.11: Flow chart of pathfinding logic for different start and end locations. In the chart, V.G. abbreviates visibility graph, and h_{bldg} is the height of a geofence around a cluster of buildings. If the departure/destination is not inside the keep-out geofence ROI box, h_{bldg} at start/end point is set to street/terrain altitude.

2.6 Simulation Results

A total of 1010 Monte Carlo simulations were run with our Manhattan maps. For each case, start and destination points were randomly defined. Selected start/end altitudes ranged from 20 m above ground level to the highest building roof. The 20 m value represents an above-ground vertical climb to hover waypoint to ensure the multicopter is well clear of people on the ground when it begins executing its lateral flight plan. If both start and end points had altitudes less than 50 m, the cruise altitude for the turn solution was set at 50 m. Otherwise, cruise altitude was adjusted based on the following condition:

$$z_{cruise} = max\{h_{start}, h_{end}\} \quad \text{if } h_{start} > 50m \mid h_{end} > 50m. \tag{2.3}$$

Our airspace volumization algorithm used this condition to choose one of the fixed-altitude datasets described in Section 2.4. As z_{cruise} becomes larger, fewer obstacles were present, so fewer calculations were needed to generate and plan a flight through the visibility graph. For each case, cost values of the four planning options ("turn", "constant cruise alt.", "terrain follower", "combined (constant cruise altitude + turn)") were calculated using Equation (2.2), and the minimum cost solution was selected as the best solution. Note in the Manhattan data the "wait" solution was never used because buildings are permanent, resulting in static geofence obstacles only.

Monte Carlo results offer an opportunity to compare our airspace volumization solutions

against a manual fixed airway or "flight corridor" airspace design. A conventional fixed-altitude airway is permanently designated on a map to enable traffic "queues" to organize in a way that can be managed by human air traffic controllers. It is unclear whether UTM will benefit from this legacy design practice, motivating our comparison of path costs for our airspace volumization and fixed airway solutions. Unlike our airspace volumization, fixed airway/flight corridor maps only require a local search for the closest airway to join. The UAS then follows fixed airway routes until exiting over a short final segment to the end state. We generated a pair of low-altitude horizontal and vertical airways through our Lower Manhattan map to illustrate the airways concept and support our evaluation.

The designed vertical airway in Lower Manhattan follows Broadway, the north–south main thoroughfare, from its origin at Bowling Green to Houston Street. The horizontal airway follows Chambers Street from River Terrance in the west to Municipal Plaza in the east, and then follows the Brooklyn Bridge until it reaches the East River. We provided two sets of the same cross airways at 150 m and 500 m to offer each UAS an altitude choice since more obstacles are present at 150 m but the climb will be more substantial to 500 m. Figure 2.12 shows our manually defined airway corridors. To offer a practical comparison, only randomly generated start and end points that do not lie in the same quadrants (i.e., 712 out of 1010 simulation examples) were considered. If randomly-generated start and end points were located in the same quadrant, the airways were unused, thus offering no benefit to efficiency or airspace organization.

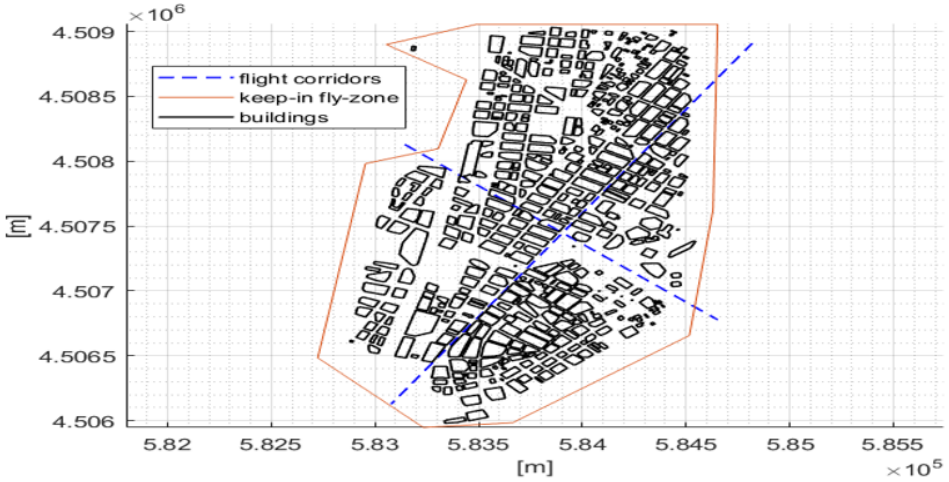


Figure 2.12: Example horizontal and vertical airway corridors in Manhattan.

There are many considerations when designing UAS/UAM airways in urban canyons. Due to the operations in low-altitude airspace by its nature (i.e., package delivery, air taxi services,

etc.), the UAS/UAM airways must be designed to avoid the complex airspace integration problem near flight restriction areas such as international airports. Such airways also need to consider daytime/nighttime population density in the affected regions. Furthermore, the security and human factor aspects must also be considered. Once the airway is designed to serve the UAS operations, discordance in community acceptance can severely impinge on the practical operations of such airways. As an example, conflicts of interest between uninvolved bystanders (i.e., homeowners) and business sectors/regulators include concerns of privacy violation, noise, visual disturbance, and daily safety vs. economic benefits. Furthermore, systematic studies must be performed on traffic flow and queuing to estimate the demand and capacity of such airways with varying mission types. Without the holistic analysis of air traffic simulation, testing, and survey, the designed airways will cause severe safety problems in urban canyons and economic loss. In this sense, our flight corridors in Figure 2.12 do not reflect any kind of thorough studies on such factors. Instead, the purpose of our flight corridor analysis is to compare the path costs between potential airways and our airspace volumization, providing a foundation for establishing future dynamic airspace geofencing, either in terms of our airspace volumization or fixed-altitude flight corridors.

Figure 2.13 shows a top-down route view comparing our airspace volumization and flight corridor solutions. Cost weights $\alpha = 0.6$, $\beta = 0.2$, $\gamma = 0.0$ were again used, so d_{travel} was prioritized in minimizing overall cost. For the illustrated case, the "turn" solution is best. Flight corridor solution cost was in fact typically higher than any of our airspace volumization solutions. For the same example, altitude vs. time plots for each solution are shown in Figure 2.14. Examples of geofencing solutions are shown in Figures 2.15–2.17, where three alternative trajectory solutions are generated, ensuring the avoidance of no-fly zones. Building keep-out geofences are shown in green.

For each Monte Carlo simulation, the minimum-cost $\mathcal{C}$ solution was compared to flight corridor solution costs at 150 m and 500 m per Table 2.3. Since the flight corridor at 150 m was almost always better than the flight corridor at 500 m, benchmark data compare the best solution obtained using dynamic airspace volumization with the flight corridor at 150 m. The results indicate our airspace geofencing volumization solutions generally have lower cost than flight corridors at 150 m or 500 m do.

The average distance and power consumption of the two-dimensional straight-line path between each start and destination location are shown in Table 2.4.

The mean and standard deviation for d_{travel}, p_{travel} for the minimum cost airspace volumization solution are summarized in Table 2.5. The percent frequency distributions of the four solution options are shown in Figure 2.18.

A similar analysis was performed to compute travel distance and power consumption

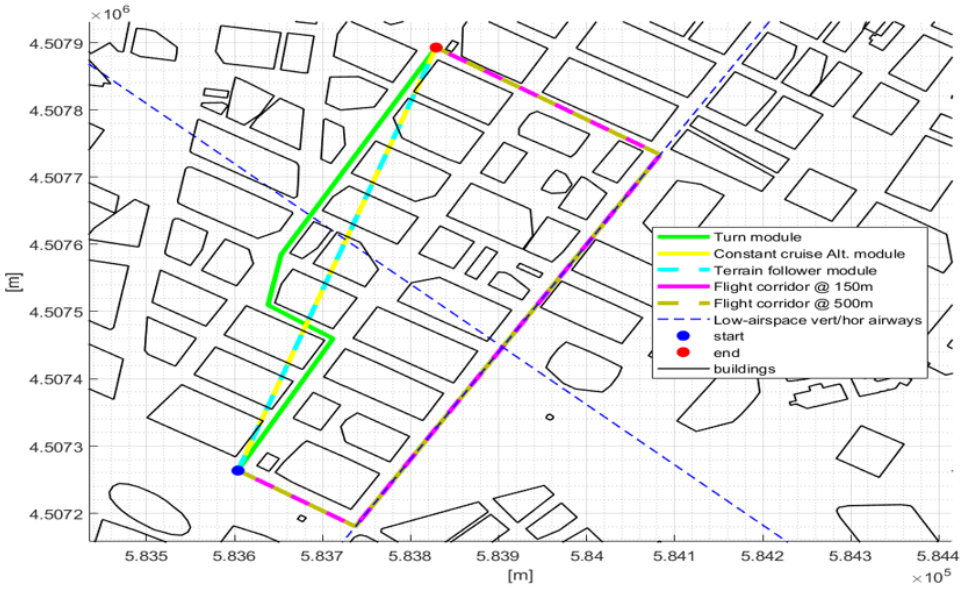


Figure 2.13: Top-down view of example flight paths for airspace volumization and fixed flight corridor solutions. Distances traveled are 770 m (turn), 1051 m (constant cruise), 1139 m (terrain follower), 1528 (150 m flight corridor), and 1977m (500 m flight corridor).

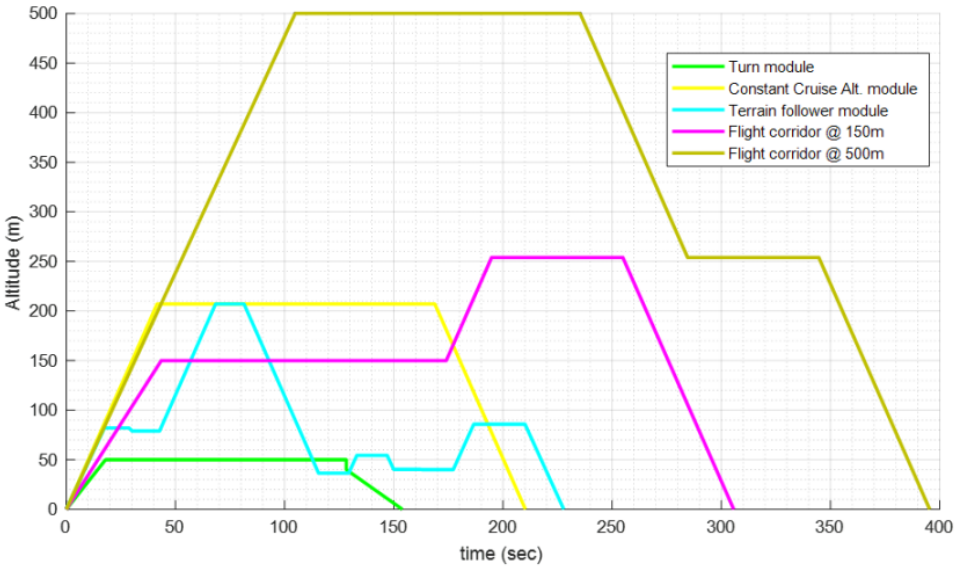


Figure 2.14: Flight altitude time histories for airspace volumization and flight corridor solutions for Figure 2.13 example.

statistics for the flight corridor solutions at 150 m and 500 m, as shown in Tables 2.6 and 2.7.

Dynamic airspace volumization and flight corridor solutions at 150 m are normalized by the two-dimensional straight-line path parameters, indicating the percent increase in average

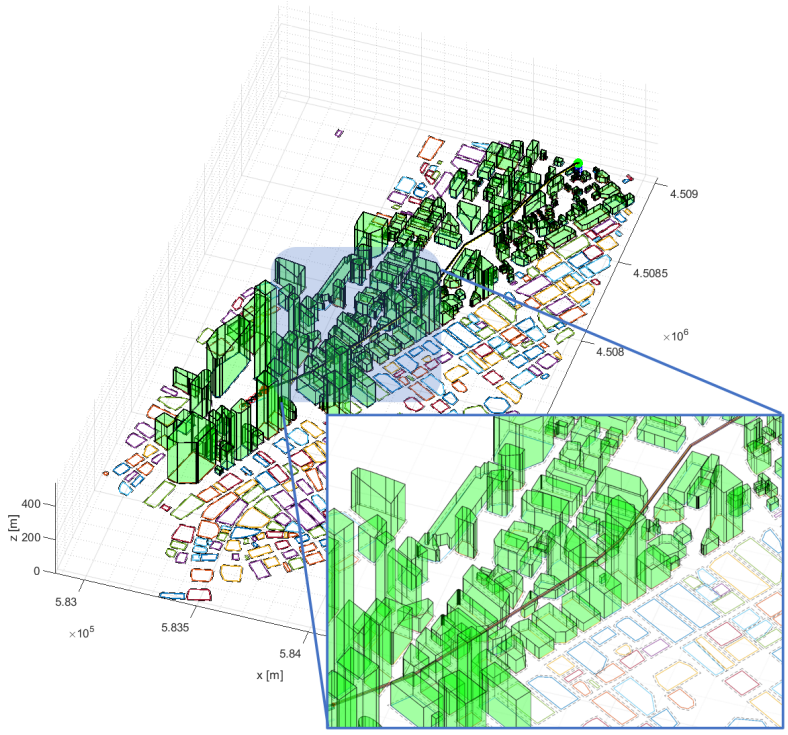


Figure 2.15: Example of a 3-D geofence wrapping a "turn" flight plan solution. Polyhedra in green denote keep-out geofences around buildings near the trajectory's keep-in geofence. The remaining 2-D polygons denote keep-out geofences around buildings that are more distant from the sUAS flight path.

Table 2.3: Number of cases where airspace volumization *vol* has minimum cost (left) and number of cases where the flight corridor at 150 m has lower cost than the corridor at 500 m.

$\#\ \{\mathcal{C}_{vol.method} < \mathcal{C}_{150m}\}$	$\#\ \{\mathcal{C}_{150m} < \mathcal{C}_{500m}\}$
698 out of 712 cases	702 out of 712 cases

Table 2.4: Average distance (d), power consumption (P), and minimum and maximum distances of 2D straight-line paths between start and destination states for the Monte Carlo simulations.

$\mu_{d_{2D\ path}}$	$\mu_{P_{2D\ path}}$	$min\{d_{2D\ path}\}$	$max\{d_{2D\ path}\}$
1391 (m)	91259 (J)	189 (m)	3003 (m)

travel distance and power consumption. A normalized benchmark comparison is shown in Table 2.8. On average, our 3-D airspace geofencing solution increased travel distance by

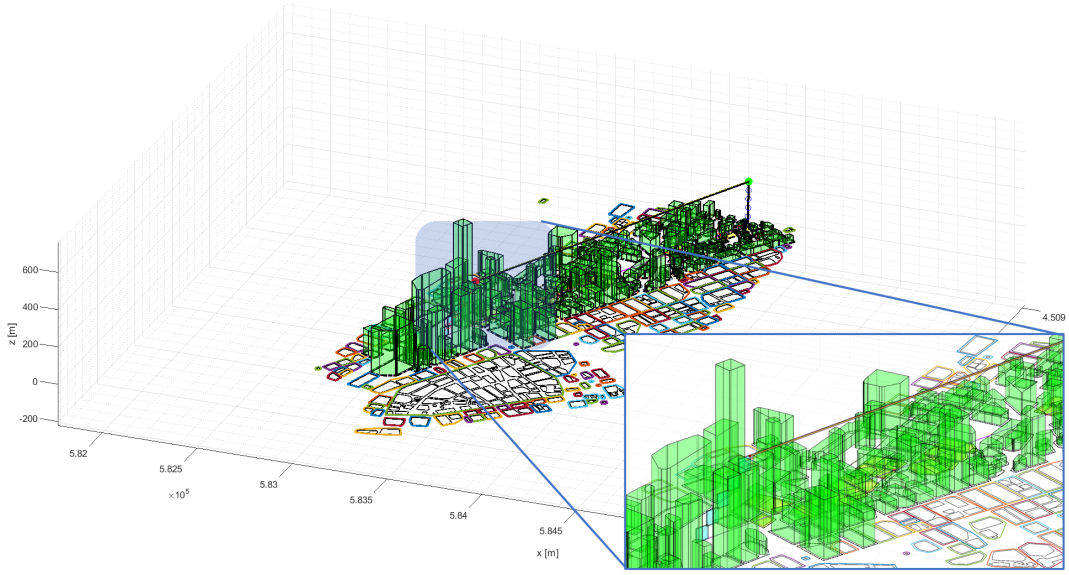


Figure 2.16: Example 3-D geofencing solution for a "constant cruise altitude" flight plan solution. Polyhedra in green denotes keep-out geofences around buildings near the trajectory's keep-in geofence. The remaining 2-D polygons denote keep-out geofences around buildings that are more distant from the sUAS flight path.

Table 2.5: Mean μ and standard deviation σ of the minimum-cost airspace volumization solution.

$\mu_{d_{travel}}$	$\sigma_{d_{travel}}$	$\mu_{P_{travel}}$	$\sigma_{P_{travel}}$	$min\{d_{travel}\}$	$max\{d_{travel}\}$
1595 (m)	606 (m)	94,338 (J)	39,609 (J)	254 (m)	3349 (m)

Table 2.6: Mean μ and standard deviation σ of 150 m flight corridor solutions.

$\mu_{d_{150m}}$	$\sigma_{d_{150m}}$	$\mu_{P_{150m}}$	$\sigma_{P_{150m}}$	$min\{d_{150m}\}$	$max\{d_{150m}\}$
2303 (m)	820 (m)	149,084 (J)	53,449 (J)	479 (m)	4464 (m)

Table 2.7: Mean μ and standard deviation σ of 500 m flight corridor solutions.

$\mu_{d_{500m}}$	$\sigma_{d_{500m}}$	$\mu_{P_{500m}}$	$\sigma_{P_{500m}}$	$min\{d_{500m}\}$	$max\{d_{500m}\}$
2796 (m)	788 (m)	179,363 (J)	51,502 (J)	1142 (m)	4836 (m)

15% and power consumption by 3% compared to 2-D straight-line paths from start states to destination states. On the other hand, the travel distance increased by 66 % and power increases by 63% when comparing minimum-cost 3-D geofencing solutions with 150 m flight corridor solutions. This analysis indicates our airspace geofencing algorithm generates routes

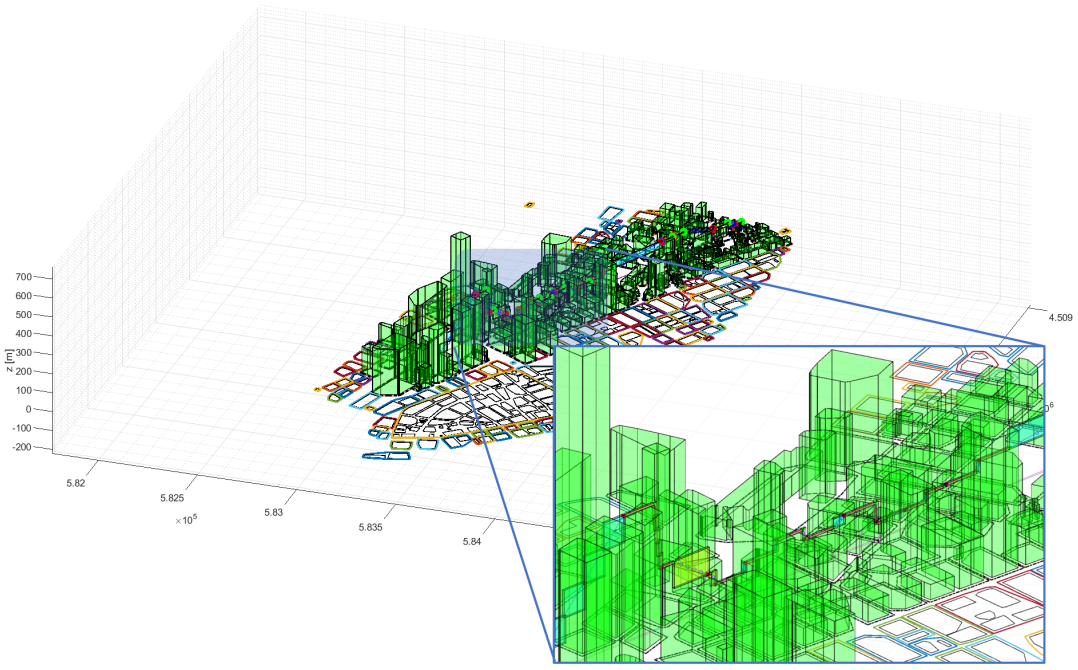


Figure 2.17: Example 3-D geofencing solution for a "terrain follower" flight plan solution. Polyhedra in green denotes keep-out geofences around buildings near the trajectory's keep-in geofence. The remaining 2-D polygons denote keep-out geofences around buildings that are more distant from the sUAS flight path.

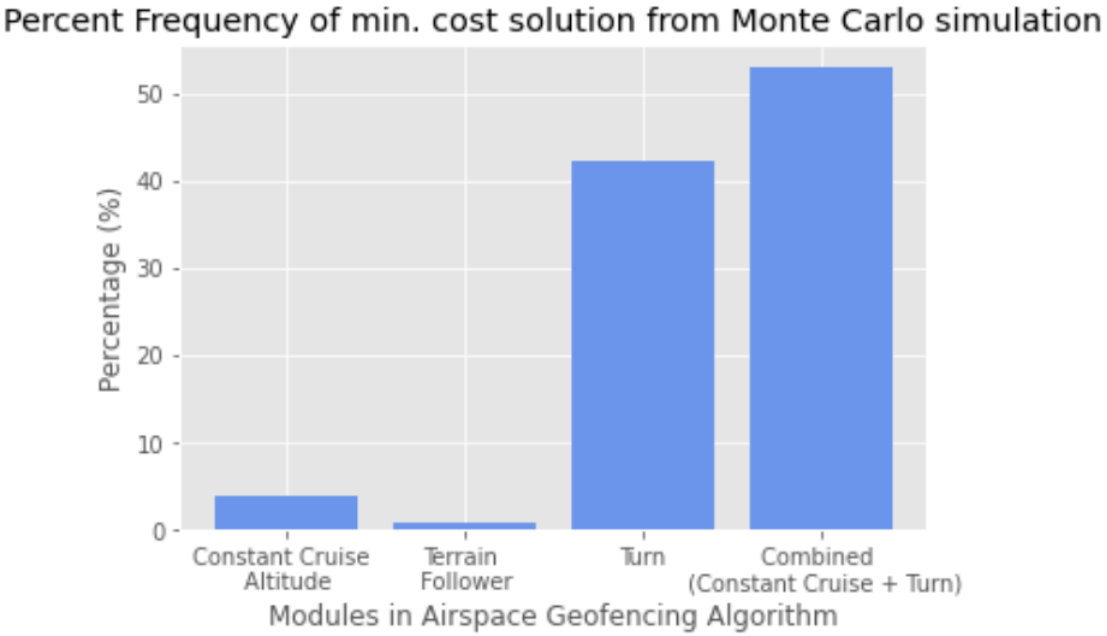


Figure 2.18: Percent frequency distribution of minimum-cost solutions over Monte Carlo simulations.

that offer nontrivial distance (time) and power (energy) reductions relative to flight corridor paths, at least for Manhattan.

Table 2.8: Normalized travel distance comparison between airspace geofencing and 150 m flight corridor solutions.

$\mu_{d_{travel}}/\mu_{d_{2D\ path}}$	$\mu_{P_{travel}}/\mu_{P_{2D\ path}}$	$\mu_{d_{150m}}/\mu_{d_{2D\ path}}$	$\mu_{P_{150m}}/\mu_{P_{2D\ path}}$
115 (%)	103 (%)	166 (%)	163 (%)

All simulations were executed on a standard laptop PC using uncompiled MATLAB code. The mean runtime and standard deviation over all 1010 Monte Carlo simulations were computed. The average runtime was 10.98 s with $\sigma = 12.68$. The minimum runtime was 0.13 s, and the maximum runtime was 90.66 s. As the number of obstacles inside the fly-zone increase, runtime also increased, as might be expected. A more computationally efficient visibility graph algorithm could be implemented in future work [68], particularly with a large obstacle set. Migration from uncompiled MATLAB to a compiled code (e.g., in C++) will also improve performance.

A Monte Carlo simulation generated a suite of random launch (start) and landing (end) points for a single sUAS flying in Lower Manhattan. Start and end points were either located on the ground or a flat building roof to simulate the diverse sUAS flight cases that might be encountered in a densely populated urban environment. Keep-out geofences were generated at each building or around blocks of clustered buildings, representing no-fly zones for the sUAS. Our airspace geofencing pipeline successfully generated flight plans and enclosing geofence volumes for four flight trajectory solution options for all 1010 Monte Carlo simulations. The minimum distance and energy cost was chosen as the best solution. Our geofence-based path planning solutions outperformed a more traditional fixed flight corridor routing option.

Our Monte Carlo simulations did not limit the maximum altitude for UAS flight, so the trajectories for some solutions had cruising altitudes greater than 400 ft AGL, beyond the UTM and sUAS ceiling. Our Monte Carlo results showed the "combined" solution option (i.e., constant cruise and turn) was preferred most often. A maximum altitude constraint would eliminate all solutions that climbed above UTM-managed airspace, likely resulting in the more frequent use of visibility graph "turning" solutions. The results in Table 2.8 showed that our algorithm generates solutions that are 51% and 60% more efficient than flight corridor solutions at 150 m altitude in terms of normalized average flight distance and power consumption, respectively. It is likely that for AAM airspace corridors accessible to sUAS, above 400 ft AGL will be designated. For longer-distance flights, a flight plan might

use an efficient dynamically geofenced route to/from a high-altitude transit tube, potentially requiring a hybrid combination of dynamic flight planning and geofencing at UTM-managed altitudes and fixed corridor transit at altitudes managed by legacy ATM.

2.7 Case Study with sUAS Route Deconfliction

The above results describe single geofenced sUAS routes through a complex urban landscape. In general, UTM will manage multiple sUAS in shared airspace. This section presents a case study illustrating how the proposed geofencing pipeline supports multiple-sUAS deconfliction. For this study, we assume airspace is allocated first-come-first-served. Suppose $sUAS_1$ and $sUAS_2$ request flight plans each defined by departure and destination coordinates (WGS 84/UTM zone 18N), cruise speed, and targeted cruise altitude as defined in Table 2.9. Further, suppose $sUAS_1$ receives approval for its flight plan and associated geofence volume before $sUAS_2$ contacts UTM. $sUAS_2$ will then need to plan a flight that avoids the Manhattan terrain and building geofences as well as the flight trajectory geofence wrapping the $sUAS_1$ route. Figure 2.19 shows the resulting flight plans for $sUAS_2$ as a top-down route view comparing our airspace volumization and flight corridor solutions. For this example, altitude vs. time plots for the $sUAS_2$ solutions are shown in Figure 2.20.

Table 2.9: Flight plan parameters for $sUAS_1$ and $sUAS_2$.

	$\boldsymbol{P_{Departure}}$ **(m)**	$\boldsymbol{P_{Destination}}$ **(m)**	$\boldsymbol{V_{UAS}}$ **(m/s)**	$\boldsymbol{h_{targetCruise}}$ **(m)**
$sUAS_1$	[584,085; 4,508,093; 0]	[584,248; 4,506,598; 0]	30	50
$sUAS_2$	[583,600; 4,507,000; 0]	[584,460; 4,507,660; 0]	20	50

Since $sUAS_1$ and $sUAS_2$ have the same target cruise altitude, a maneuver was required for $sUAS_2$ to deconflict its "turn" route from the $sUAS_1$ flight trajectory, making this the longest distance solution option. On the other hand, the "constant cruise altitude" and "terrain follower" solutions were not influenced by the $sUAS_1$ trajectory because the minimum building height along the straight line path from departure to destination for $sUAS_2$ was greater than $sUAS_1$'s target cruise altitude. If building height placed $sUAS_2$ at $sUAS_1$'s cruise altitude, $sUAS_2$ would also need to climb over the $sUAS_1$ geofence. Note that if $sUAS_1$'s airspace volume reservation duration was minimized using SDG or MDG, $sUAS_2$'s path had a lower probability of being impacted. Example 3-D $sUAS_2$ in "turn", "constant cruise altitude", "terrain follower" solutions are shown in Figures 2.21–2.23.

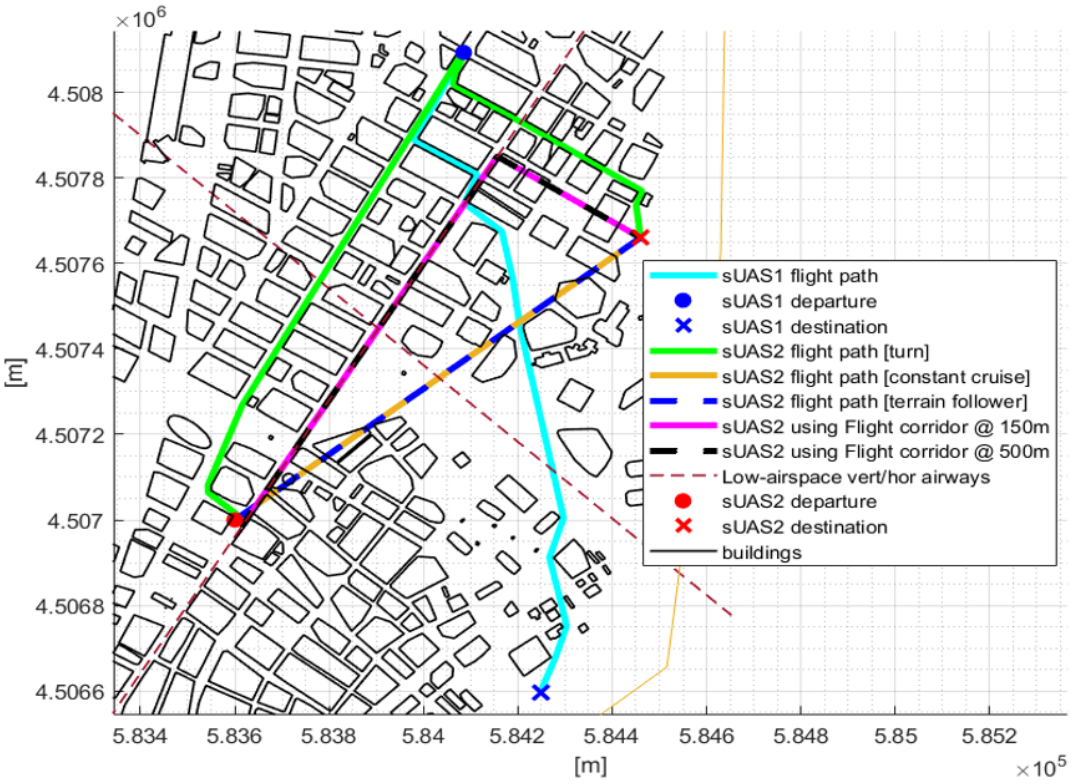


Figure 2.19: Top-down view of $sUAS_2$ sample solutions. Five flight trajectory solutions are generated for $sUAS_2$. Each solution provides route deconfliction from Manhattan terrain and building geofences and from the $sUAS_1$ flight trajectory geofence. Distances traveled are 2008 m (turn), 1585 m (constant cruise), 1634 (terrain follower), 1983 (150 m flight corridor), and 2395 (500 m flight corridor). The minimum-cost solution for $sUAS_2$ is the constant cruise altitude option.

2.8 Conclusions and Future Work

This chapter applied airspace geofencing volumization and path planning to support UTM management of low-altitude airspace. Layered durational geofences wrapping flight trajectories ensure the UAS will fly without conflict in designated or reserved airspace volumes. Our airspace volumization algorithms generated four conflict-free paths for any keep-in/keep-out geofence volume set based on turn, constant cruise, terrain follower and combination turn/cruise options. The algorithm ranked these paths using a weighted distance, energy, and time cost function, then selected the minimum-cost solution. A city map data of Lower Manhattan was used to construct keep-out geofences around buildings. Monte Carlo simulation studies validate our geofence algorithms and support the statistical characterization of performance including run time. A benchmark comparison of our dynamically geofenced flight plans and conventional flight corridor solutions is provided, showing that our solutions

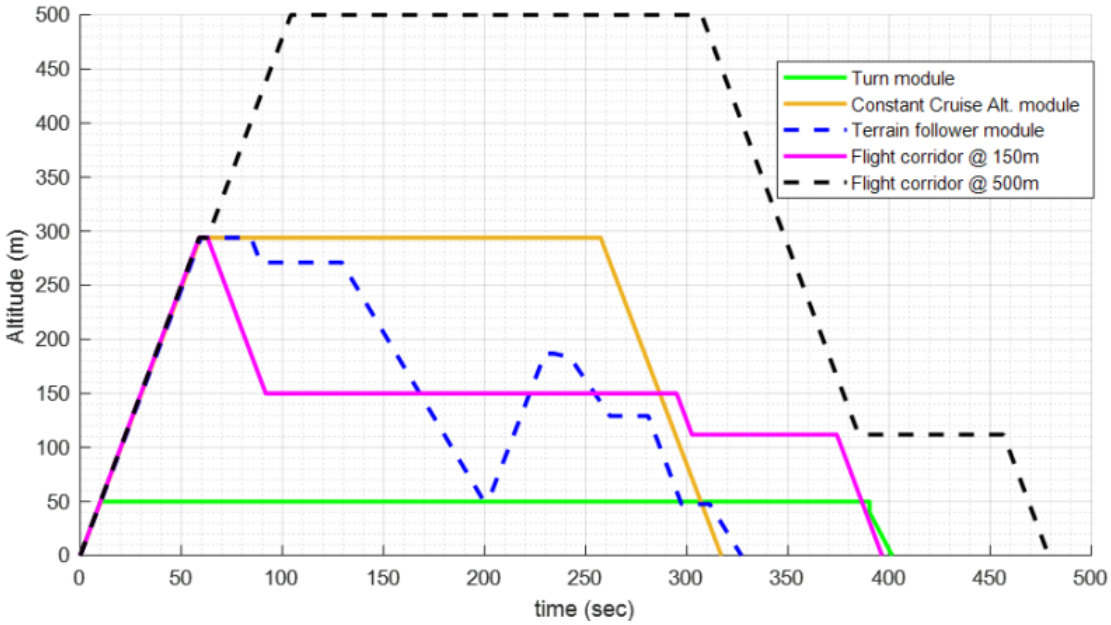


Figure 2.20: Flight altitude time histories for airspace volumization and flight corridor solutions for $sUAS_2$ in Figure 2.19 example.

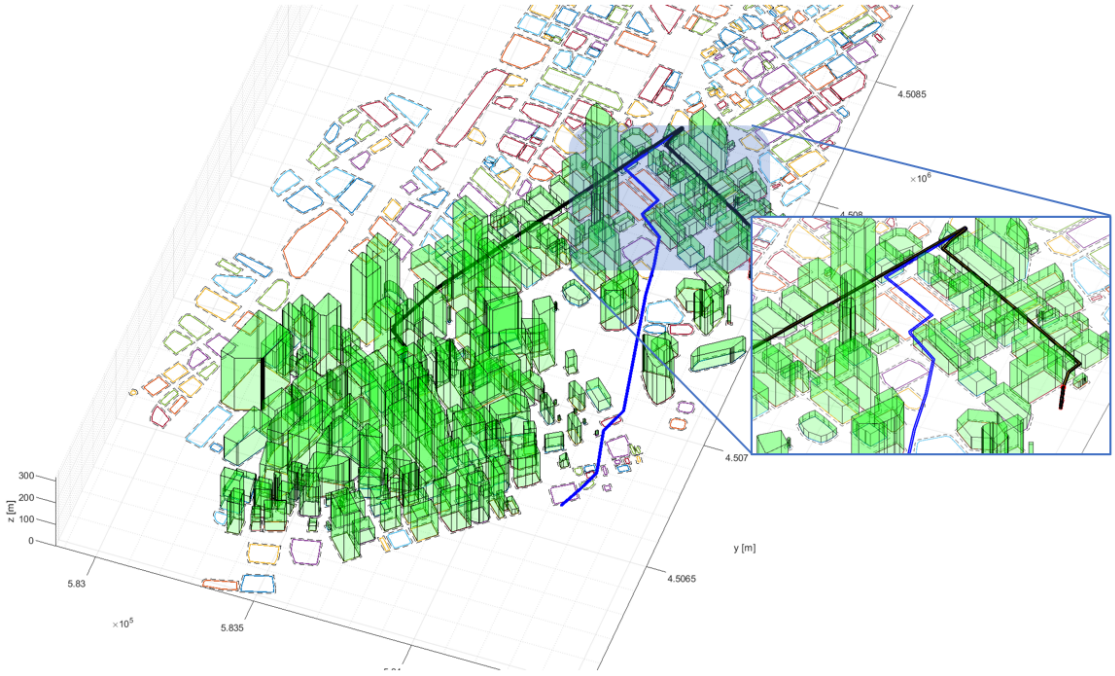


Figure 2.21: Example of a 3-D geofence wrapping a "turn" flight plan for $sUAS_2$. The $sUAS_2$ trajectory is shown in black, and the $sUAS_1$ trajectory is shown in blue. Polyhedra (green) denotes keep-out geofences around buildings. The remaining 2-D polygons denote keep-out geofences around buildings that are outside the combined ROI.

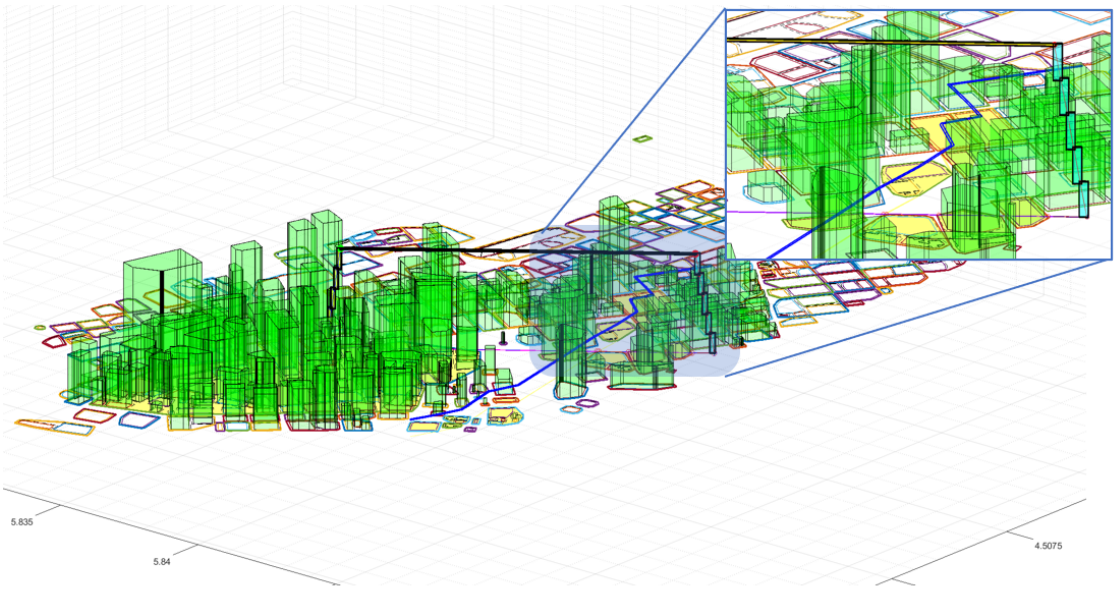


Figure 2.22: Example of a 3-D geofence wrapping a "constant cruise altitude" flight plan for $sUAS_2$. The $sUAS_2$ trajectory is shown in black, and the $sUAS_1$ trajectory is shown in blue. Polyhedra (green) denotes keep-out geofences around buildings. The remaining 2-D polygons denote keep-out geofences around buildings that are outside the combined ROI.

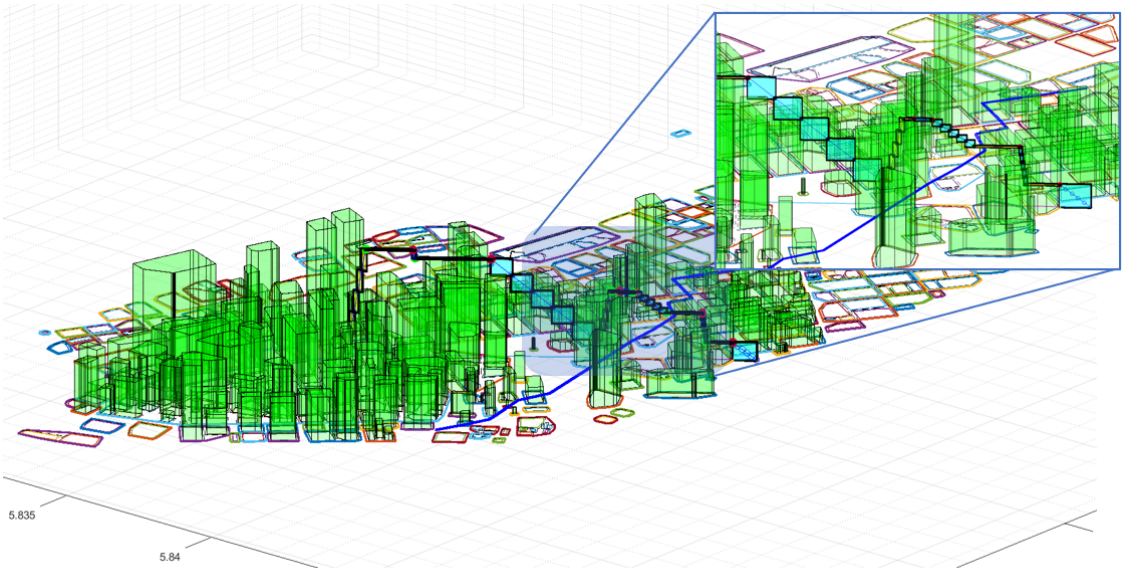


Figure 2.23: Example of a 3-D geofence wrapping a "terrain follower" flight plan for $sUAS_2$. The $sUAS_2$ trajectory is shown in black, and the $sUAS_1$ trajectory is shown in blue. Polyhedra (green) denote keep-out geofences around buildings. The remaining 2-D polygons denote keep-out geofences around buildings that are outside the combined ROI.

reduce flight distance and power compared to fixed corridor solutions. A case study of two sUAS flight planning demonstrated how the proposed geofencing pipeline supports multiple sUAS deconfliction. Algorithms and definitions from this chapter can contribute to future UTM dynamic airspace geofencing operational standards.

This work simplifies flight planning to geometric paths. Future work will incorporate aircraft dynamics into flight plans and geofence layer sizing, and extend airspace volumization to enclose cooperative groups of sUAS. Additionally, the altitude constraint and other factors such as day/night local population density, GPS dependency, air traffic volume, and vehicle-specific parameters should be incorporated in the geofenced path planning algorithm to generate solutions that are more realistic for UTM-specific applications. We hope to apply machine learning to large-scale flight track data and urban maps to generalize and optimize geofencing volume designs based on area topology, day/night occupancy, infrastructure, and existing air traffic patterns. We will also explore auto-code generation and Python/C++ implementations to improve path planning computational performance.

CHAPTER 3

Statistically-Guided Geofence Volume Sizing with AAM Vehicle Performance Model

3.1 Introduction

The main goal of AAM is to revolutionize air transportation for passengers and cargo in traditionally underserved areas. By the end of 2020, there were approximately 1.7 million drones in the US, a fleet size seven times larger than that of both airlines and general aviation combined, as estimated by the FAA [22]. Scalability in air traffic control is a critical constraint for AAM [21]. Previous research proposed fixed separation distances for AAM, but it's unclear whether these distances are optimal or excessive [23, 24].

This chapter outlines the methodology for establishing statically guided geofence volume sizing to ensure safety among AAM vehicles operating within limited airspace.

As explained in the previous chapter, geofencing is a key enabler for versatile and inclusive UTM [25, 19]. Geofencing divides airspace into available fly (keep-in) and no-fly (keep-out) zones with virtual boundaries to assure UAS separation assurance and obstacle/terrain avoidance. We developed methodologies to construct three-dimensional geofence-based flight path planning solutions for randomly generated maps in [38] based on the airspace geofencing algorithm suite defined in [49, 33, 18]. Ref. [38] minimized airspace volume extent with time-based diminishing geofence volumes wrapping planned flight trajectories.

This chapter extends the previous chapter by addressing how to size safety volume geofences wrapping individual aircraft. Our methodology models aircraft dynamics and guidance, navigation and control (GNC) systems based on realistic sensor and wind uncertainties. A traditional Proportional, Integral and Derivative (PID) controller is integrated with an extended Kalman Filter (EKF) state estimator using sensor and model noise covariance estimates from [2, 69]. A fixed-wing flight kinematics model [2] is implemented for lateral and longitudinal guidance. Visibility graph [57, 64] and Dubins path [70, 71] solvers are used to

generate reference aircraft flight paths. Map data are constructed using Open Street Map (OSM) data, and map-based geofence algorithms in [3].

Wind is an important parameter for geofence buffer sizing, especially for aircraft operating at low-altitude where wind can be impacted by nearby terrain and buildings. A computational fluid dynamics (CFD) model is used to generate a wind vector field around planned flight trajectories in the simulated OSM environment. Figure 3.2 shows a schematic of the simulation environment applied to UAS and larger-scale Advanced Air Mobility (AAM) aircraft case studies. p denotes flight plan, $r(t)$ is the reference state vector, $\hat{x}(t)$ is the estimated state vector, and $e(t)$ is the difference or error between reference and estimated state vectors. $u(t)$ denotes control input vector, $w(t)$, $v(t)$ are process and sensor noise, respectively, and $T(t)$ is the wind vector at time t. This simulation is used to statistically characterize navigation and trajectory tracking error for UAS and AAM platforms as a function of sensor noise, wind, and vehicle performance. The goal is to balance safety and efficiency in low-altitude settings, especially in densely populated urban areas. These errors are translated to UAS geofencing safety buffer sizes that statistically guarantee a UAS will stay inside a flight trajectory keep-in geofence with minimal airspace reservation. We aim to determine the minimum geofence buffer size needed to statistically ensure safety at 3-sigma level, aligning with FAA priorities for safety in urban air mobility. Figure 3.1 shows the urban airspace with diverse AAM vehicle types and their associated trajectory geofence volumes.

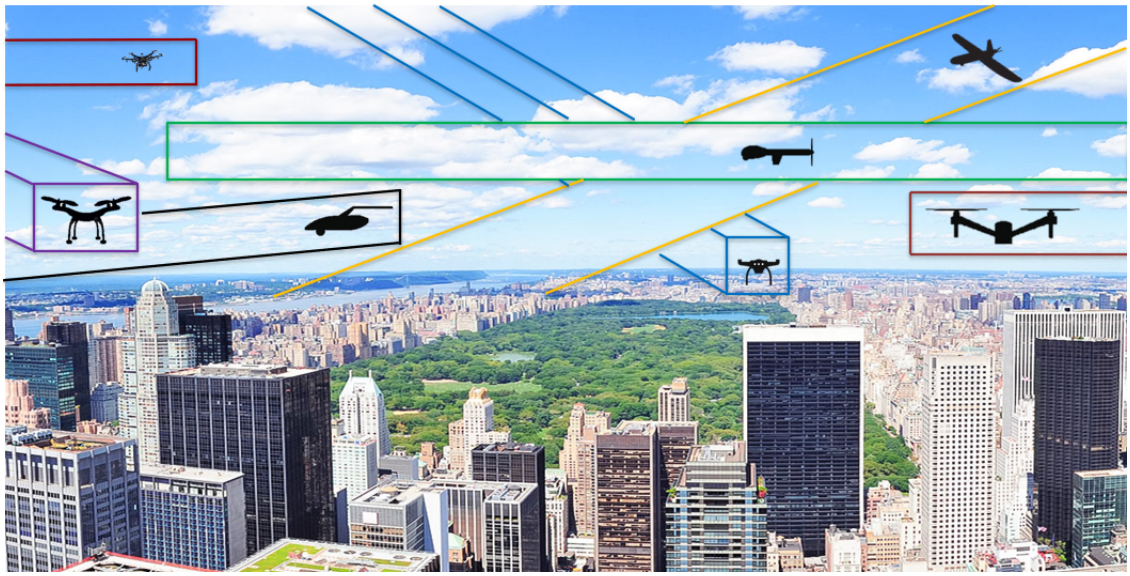


Figure 3.1: The upcoming urban airspace with diverse AAM vehicle types and their associated trajectory geofence volumes.

Figure 3.3 shows a visualization of the geofence safety buffer zone dimensions [δ_{sb_x}, δ_{sb_y},

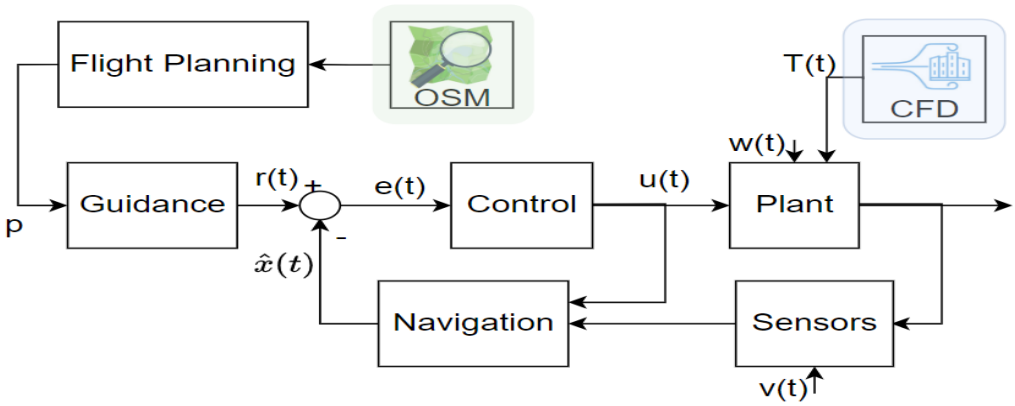


Figure 3.2: Aircraft simulation block diagram used to assess navigation and trajectory tracking error in UAS and AAM case studies under urban wind field simulated with computational fluid dynamics (CFD).

δ_{sb_z}] computed in this chapter. The EKF-based state estimator for navigation requires sensor and aircraft models with covariance (noise) estimates. We present two case studies for geofencing buffer sizing, one for a small fixed-wing UAS and another for an upscaled Advanced Air Mobility (AAM) design of a similar configuration. UAS buffer sizing was determined using a model of the Aerosonde, manufactured by Textron Systems. A model of the Aerosonde is adapted from [2]. Since there is not yet an openly published AAM reference model, we scaled the Aerosonde to 322 kg (710 lbs) to estimate AAM buffer sizing dimensions as described further below.

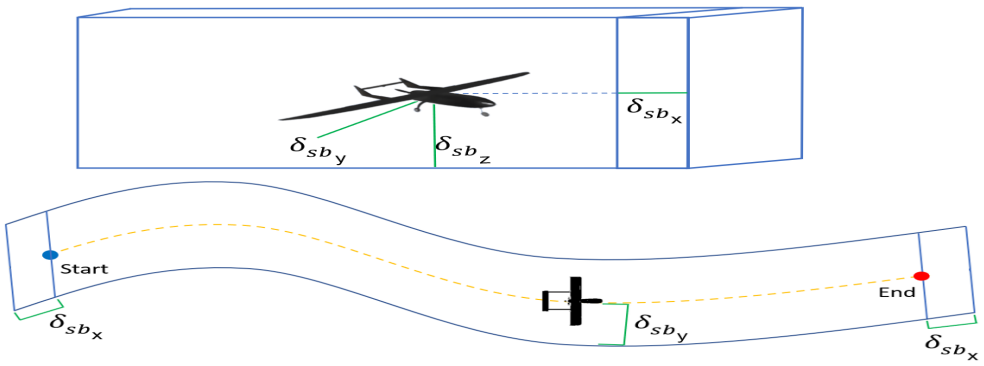


Figure 3.3: Geofencing buffer sizing parameters δ_{sb_x}, δ_{sb_y}, δ_{sb_z} in side and top views, respectively. The boundary lines shown in blue represent the flight trajectory geofence, the yellow dashed line is the flight trajectory, and green lines indicate geofence buffer dimensions in the aircraft body frame. Vehicle start and and end waypoints are shown in blue and red circles, respectively.

The chapter is organized as follows. Section 3.2 presents UAS kinematics and dynamics,

PID control design, and EKF to estimate the mean and covariance of each vehicle state along with guidance models for trajectory following. Section 3.3 describes an AAM design obtained by upscaling the Aerosonde UAS model. Section 3.4 illustrates simulation setup in Manhattan City, flight management, and CFD wind estimation using ANSYS Fluent. Section 3.5 presents case studies with different vehicle and sensor noise and wind uncertainties. Section 3.6 concludes the chapter.

3.2 Aircraft Dynamics, Guidance, Navigation, and Control Models

3.2.1 Kinematics and Dynamics

This section presents fixed-wing aircraft kinematics and dynamics with reference to the specific Aerosonde UAS dynamics model found in [2]. The kinematics equations describe aircraft's three-dimensional positions and velocities, while dynamics equations compute aircraft accelerations from applied forces and moments assuming a rigid body. External forces and moments arise from longitudinal and lateral aerodynamics including control surface deflections, propulsion modules, and Earth's gravity. Each force/moment can be expressed in an aircraft body or ground (inertial) coordinate frame with a rotation matrix or equivalent converting results between the two reference frames. By convention, linear velocity is typically expressed in a body reference frame, and linear position is commonly expressed in an inertial reference frame. Since a fixed-wing aircraft does not typically undergo large-scale attitude changes, Euler angles as well as rotation matrices are utilized in the simulation code. The aircraft equations of motion expressed in first-order ordinary differential equation (ODE) form are shown below in Eqs. 3.1-3.6. Here, $[p_n, p_e, p_d]$ are inertial North, East, Down Earth coordinates of UAS position, $[u, v, w]$ is the ground velocity vector with respect to the aircraft body frame, and $[\phi, \theta, \psi]$ are roll, pitch, and yaw angles in the vehicle frame, respectively. Vector $[p, q, r]$ describes body frame angular rates, and $[f_x, f_y, f_z]$ is the body frame applied force vector due to gravity, aerodynamics, and propulsive thrust. The vector $[l, m, n]$ defines body frame applied moments and J is the aircraft inertia matrix.

$$\begin{bmatrix} \dot{p_n} \\ \dot{p_e} \\ \dot{p_d} \end{bmatrix} = R_b^v \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.1a}$$

$$R_b^v = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \tag{3.1b}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3.2}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.3}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{bmatrix} \begin{bmatrix} \Gamma_3 l + \Gamma_4 n \\ \frac{1}{J_y} m \\ \Gamma_4 l + \Gamma_8 n \end{bmatrix} \tag{3.5}$$

$$\Gamma_1 = \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma}, \quad \Gamma_2 = \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma} \tag{3.6a}$$

$$\Gamma_3 = \frac{J_z}{\Gamma}, \quad \Gamma_4 = \frac{J_{xz}}{\Gamma}, \quad \Gamma_5 = \frac{J_z - J_x}{J_y} \tag{3.6b}$$

$$\Gamma_6 = \frac{J_{xz}}{J_y}, \quad \Gamma_7 = \frac{(J_x - J_y)J_x + J_{xz}^2}{\Gamma} \tag{3.6c}$$

$$\Gamma_8 = \frac{J_x}{\Gamma}, \quad \Gamma = J_{xz} - J_{xz}^2 \tag{3.6d}$$

Applied forces and moments require nonlinear aerodynamic models. In particular, wing stall characteristics when computing lift are modeled as in [2]. Our Simulink model uses the nonlinear lift and drag equations in [72] to accurately capture aerodynamic forces over a wide range of wing angles of attack. Since we could not utilize a wind tunnel to determine the nonlinear relationship between angle of attack and moments, a linear model was used for longitudinal and lateral moments. Figure 3.4 shows an example of modeled nonlinear lift coefficient values as a function of angle of attack.

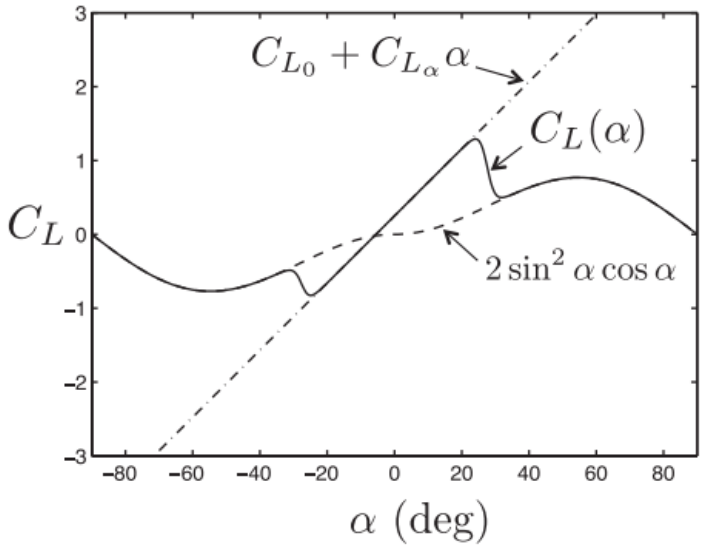


Figure 3.4: Nonlinear lift coefficient as a function of angle of attack [2]. The lift coefficient function is approximated by blending a linear function with the lift coefficient for a flat plate. Stall performance reduction is captured.

3.2.2 Control

A traditional proportional-integral-derivative (PID) feedback controller was implemented to minimize tracking error for attitude, airspeed, altitude, and course angle or heading. Servo saturation limits were imposed. Decoupled longitudinal and lateral PID flight controllers were designed and tuned for Aerosonde and AAM models distinctly. Particularly, we implemented longitudinal and lateral controller based on the derivations in [2], where each control gain was tuned from the desired response of closed-loop dynamics. Gains were first tuned from inner loop (i.e., attitude angles), and then outer loop (i.e., altitude, course angle, airspeed, etc.). Figure 3.5 shows block diagrams of the implemented lateral controller in our Simulink model.

3.2.3 Navigation and Guidance

The navigation function computes current aircraft state vector mean and covariance values. The guidance model outputs a reference trajectory at each time step. These values are differenced to compute trajectory tracking error $e(t)$ at each time t. We use an Extended Kalman Filter (EKF) as an optimal linear estimator for a nonlinear system to calculate state and associated covariances as described in [73]. Then, straight line and orbit reference guidance paths [74, 75] are used to compute kinematic error bounds for longitudinal and lateral path segments, respectively. By obtaining position standard deviations σ and maximum guidance

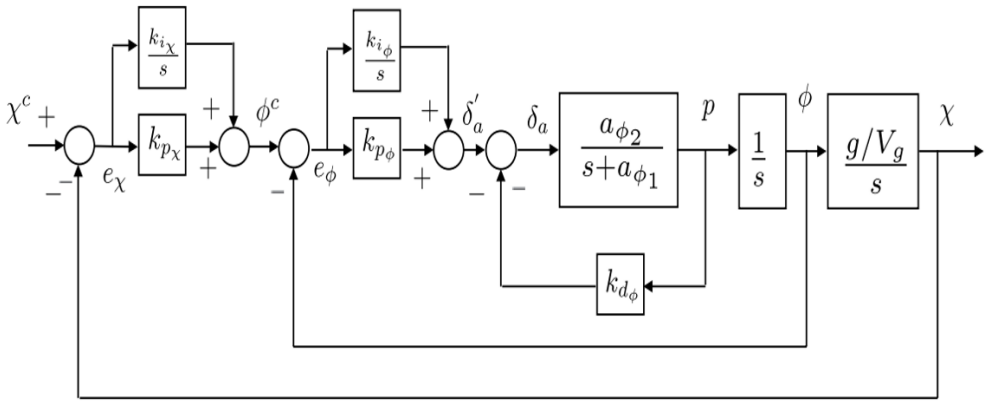


Figure 3.5: Lateral controller block diagrams [2] implemented in Simulink. K_p, K_d, K_i denote proportional, derivative and integral gains, respectively. χ and ϕ stand for course angle and roll angle, where the superscript c denotes commanded input. δ_a stands for aileron input, and V_g is the ground speed.

errors for each simulated flight, navigational and guidance uncertainties are captured and translated to geofencing safety buffer sizing requirements.

In our simulation, static pressure, differential pressure, angular rate (gyro), and Global Positioning System (GPS) sensors are used to find navigational uncertainty. Particularly, covariances of angular rate (gyro), static pressure, and differential pressure were taken from [2], while GPS covariances (i.e., $\sigma_n, \sigma_e, \sigma_d$) were taken from [69] to capture realistic GPS accuracy with respect to street level position and relative altitude in an urban environment. Figure 3.6 below shows the discrete-time EKF used in our Simulink model.

3.2.4 Flight Trajectory Geofence Buffer Dimensions

Aircraft-specific geofence volume sizes assure airspace is appropriately reserved for safety and without waste. Realistic simulations plus real flight data when available offer statistical confidence in computed geofence sizing as a function of wind conditions. Position coordinate uncertainties are a function of navigation error, control disturbances, and ambient wind conditions. We use the EKF to find position covariances, and then square those values to find the positional uncertainties as standard deviations. Vehicle performance parameters and guidance strategy also impact tracking errors. We define the following equation for flight trajectory keep-in geofence safety buffer sizing:

$$\delta_{sb_x} = 3\sigma_x + \frac{1}{2} f_l \tag{3.7a}$$

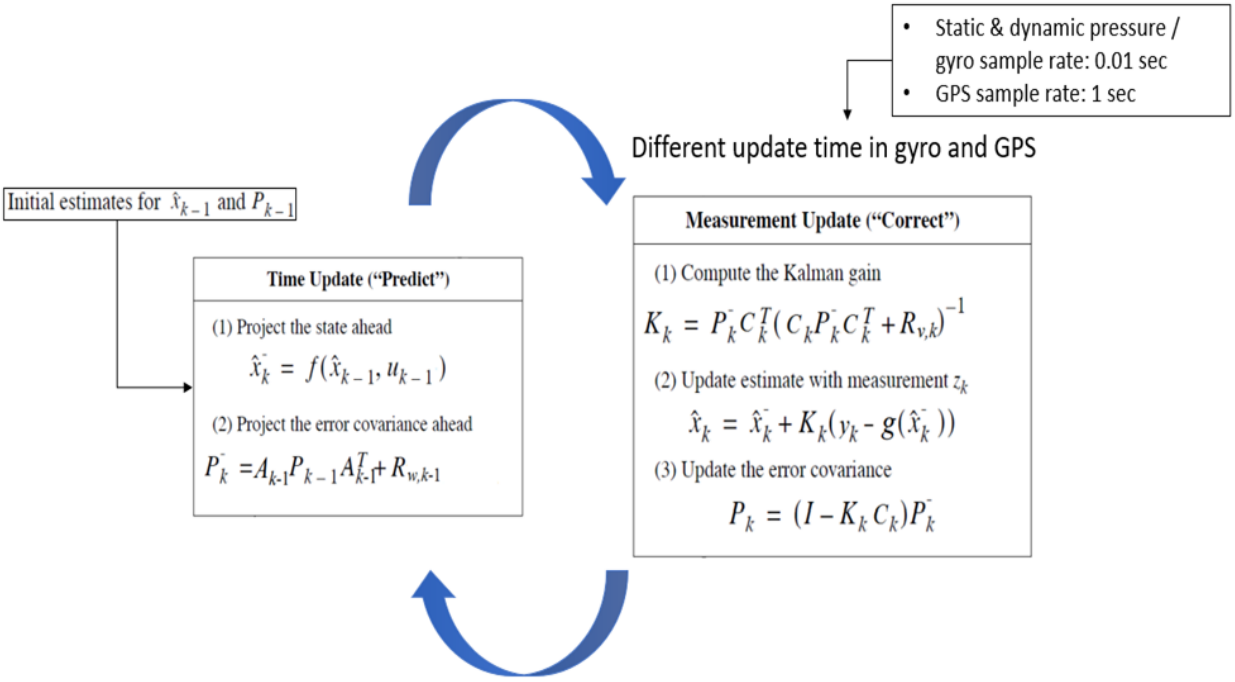


Figure 3.6: Discrete-time extended Kalman filter used in case study simulations. An initial state vector estimate is provided, states are propagated over the system dynamics model, and estimates are corrected using a Kalman gain matrix with sensor measurements. Figures are adopted and modified from [2].

$$\delta_{sb_y} = 3\sigma_z + \epsilon_{guidance_y} + \frac{1}{2}b \tag{3.7b}$$

$$\delta_{sb_z} = 3\sigma_z + \epsilon_{guidance_z} + \frac{1}{2}h \tag{3.7c}$$

$$\delta_{sb} = max[\delta_{sb_y}, \delta_{sb_z}] \tag{3.7d}$$

Here, $[\sigma_x, \sigma_y, \sigma_z]$ represent the vehicle's maximum positional standard deviation in vehicle body frame $[x, y, z]$ throughout the flight path. The navigational position confidence was then calculated by multiplying those σ values by 3, achieving 99.7% confidence in estimated position at each body frame axis, given the vehicle and sensor properties as well as the wind condition. $\epsilon_{guidance_y}$ and $\epsilon_{guidance_z}$ are the maximum guidance error (i.e., trajectory error between estimated vehicle position and nominal flight path) in lateral direction and altitude, respectively. The cross-sectional geofencing safety buffer size, δ_{sb}, was then calculated as the maximum of $[\delta_{sb_y}, \delta_{sb_z}]$, denoting the maximum deviations in combined navigation and guidance errors as well as the vehicle dimensions (i.e., wingspan and height) in the y

and z axis. This process models the flight trajectory keep-in geofence cross-section to be square in the simulation. δ_{sb_x} adds an additional longitudinal buffer. Equation 3.7 calculates optimal safety buffer sizes given maximum positional uncertainties, maximum guidance error throughout the flight as well as vehicle dimensions. Figure 3.7 shows a visualization of optimal buffer sizing in the lateral direction.

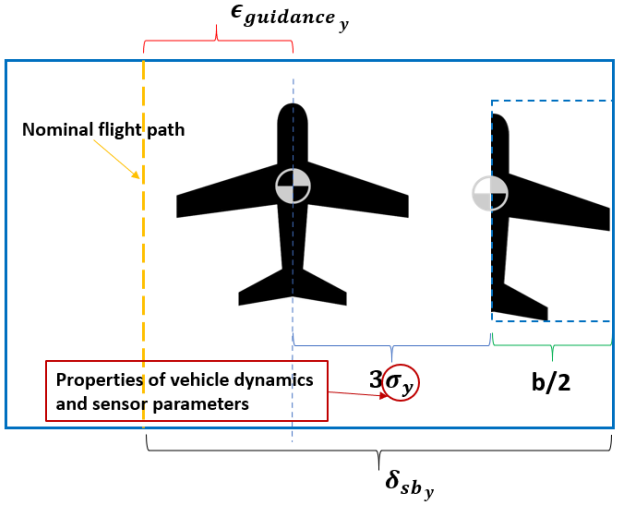


Figure 3.7: Optimal geofence buffer size calculation per vehicle. σ_y is the vehicle's lateral positional standard deviation. $3\sigma_y$ statistically guarantees that the vehicle's position is in 99.7% confidence. b denotes aircraft wing span, and $\epsilon_{guidance_y}$ denotes the guidance error in lateral direction. The optimal lateral geofence buffer dimension is calculated using lateral navigational and guidance error as well as the vehicle's wingspan.

3.3 AAM Design

The AAM model was designed by upscaling the Aerosonde UAS with known and openly published aerodynamic characteristics. The benefit of upscaling is that the non-dimensional aerodynamic properties stay the same regardless of the size. Specifically, the AAM reference vehicle wing span and chord were scaled to be three times the Aerosonde's wing dimensions to have a gross weight greater than 700 lbs. The AAM fuselage was upscaled approximately three times to Aerosonde's such that one passenger can fit inside the cockpit. To calculate the inertia tensor, the geometry of the AAM reference vehicle was simply designed with fuselage and tail booms represented as cylinders, while wing and tail were represented as flat plates. An engine with a 70 kg weight was chosen for the AAM. Figure 3.8 shows the schematics of Aerosonde UAS and our AAM model that weighs 333 kg (i.e., 734 lbs), where aircraft structure was assumed to be mostly carbon fiber with density $\rho = 250kg/m^3$. Tables 3.1 and

3.2 show Aerosonde's and AAM's component dimensions and inertia tensors, respectively. Note that there are missing dimensions in the Aerosonde UAS model because we do not know the Aerosonde's fuselage and tail geometries and placements. The AAM was designed with simple geometric shapes and scaled such that its gross weight is greater than 700 lbs.

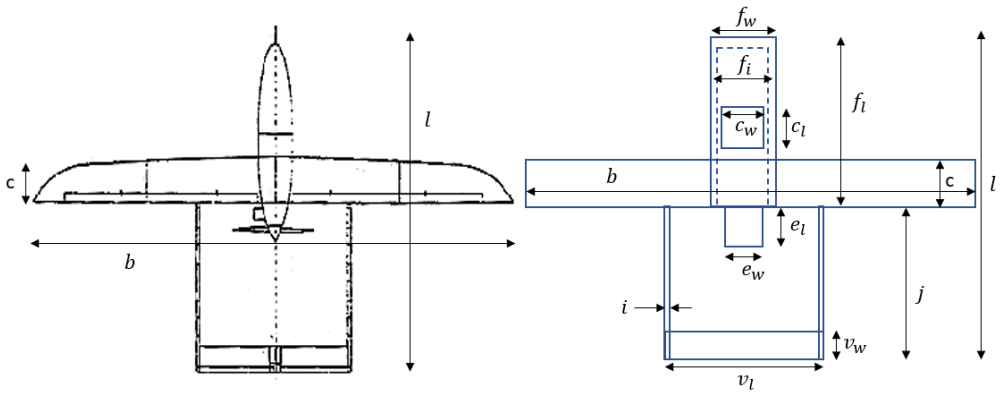


Figure 3.8: Aerosonde UAV (left) and AAM model top view (right). The dimensions of the AAM vehicle are shown in Table 3.1.

Table 3.1: Aerosonde and AAM component dimensions [m]

Label	Aerosonde	AAM
l	1.72	5.718
f_w	-	0.9
f_i	-	0.8
f_l	0.72	2.538
c_w	-	0.8
c_l	-	0.8
c	0.189	0.57
b	2.896	8.7
j	-	3.18
e_1	-	0.399
e_w	-	0.162
i	-	0.054
v_l	-	2.64
v_w	-	0.36

Table 3.2: Aerosonde and AAM inertia tensors $[kg/m^2]$

	Aerosonde	AAM
I_{xx}	0.824	12.047
I_{yy}	1.135	258.48
I_{zz}	1.759	246.44
I_{xz}	0.120	24.277

3.4 Simulation Environment

The simulation was configured for UAS and AAM platforms to fly at 300m mean sea level (MSL) altitude in Manhattan, New York City, New York at a constant airspeed of 35 m/s. The subsections below describe how the simulation modules were coded and integrated.

3.4.1 Environment Map and Wind Model

Manhattan geofenced map data were constructed using algorithms in [3] with OpenStreetMap (OSM). Here, only buildings greater than 300m MSL altitude are extracted since the cruise altitude of both Aerosonde UAS and AAM models is set to 300m in the simulation. Keep-out geofences are constructed on all mapped buildings with a building safety buffer size of 10m. Figure 3.9 shows the Manhattan map with mapped buildings and their geofences.

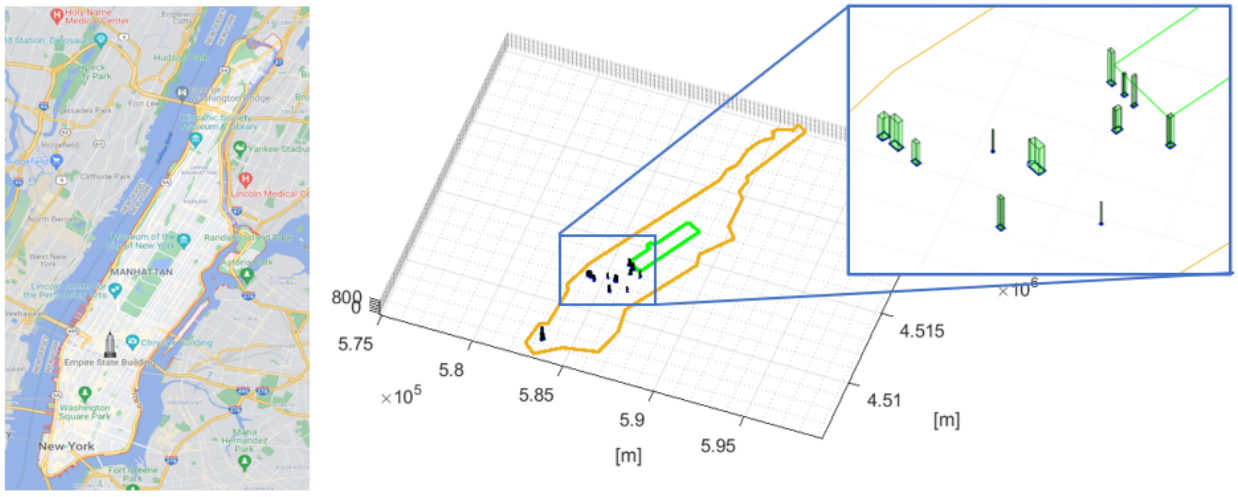


Figure 3.9: Geofenced Manhattan map constructed using algorithms in [3]. The map shows buildings with heights greater than 300m only. The Aerosonde UAS and AAM vehicle cruise altitudes are set to 300m in the simulation.

In the simulation, a constant west wind of 3 m/s was injected in the environment. Due to the urban terrain with high-rise buildings in Manhattan, the actual wind direction and its magnitude are highly variable over the terrain as the wind disperses/collides in the city [76]. To capture the complex interaction between airflow and buildings and to simulate the atmospheric wind flow effect on geofence buffer sizing, a computational fluid dynamics (CFD) analysis was performed on the region for which each flight plan is generated. For CFD analysis, the turbulent kinetic energy (k-epsilon) method was chosen with 5% turbulence intensity to capture the effect of the turbulence after wind impacts the buildings. The semi implicit pressure linked equations (SIMPLE) method was used to solve the CFD with no-slip conditions on buildings. Figure 3.10 shows wind velocity contour simulation results for the region of interest. Figure 3.11-3.13 shows the component-wise wind vector in the region of interest. These CFD results were imported into Simulink and interpolated to compute the wind vector at each flight state as the vehicle travels along its path.

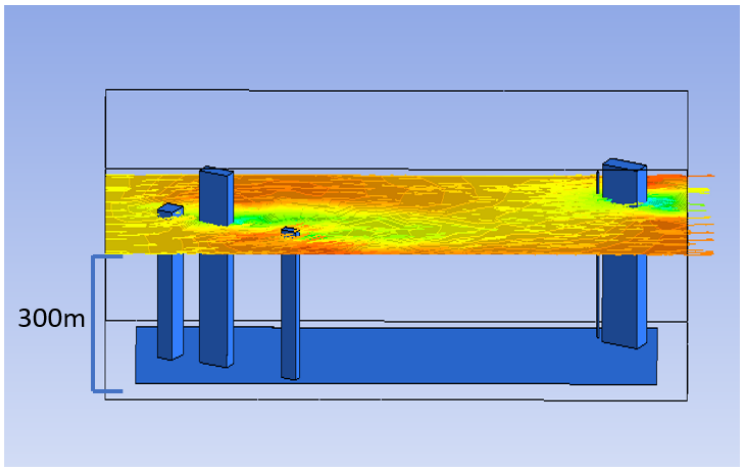


Figure 3.10: CFD wind velocity contour plot for 300m MSL.

3.4.2 Flight Planning

Each flight path was defined at a constant 300m MSL altitude. A visibility graph was used to find a minimum travel distance waypoint sequence from an initial state to a destination. A Dubins path was then defined to introduce smooth turning flight path segments to connect visibility graph waypoints assuming a constant altitude and constant velocity per [71].

Figure 3.14 shows an example visualization of a flight path with minimum geofence buffer sizing around arbitrary obstacles. Minimum geofence safety buffer size was calculated using Equation 3.7, defined to assure the aircraft will stay inside the yellow shaded keep-in geofence volume with three sigma (3σ) or 99.7% confidence.

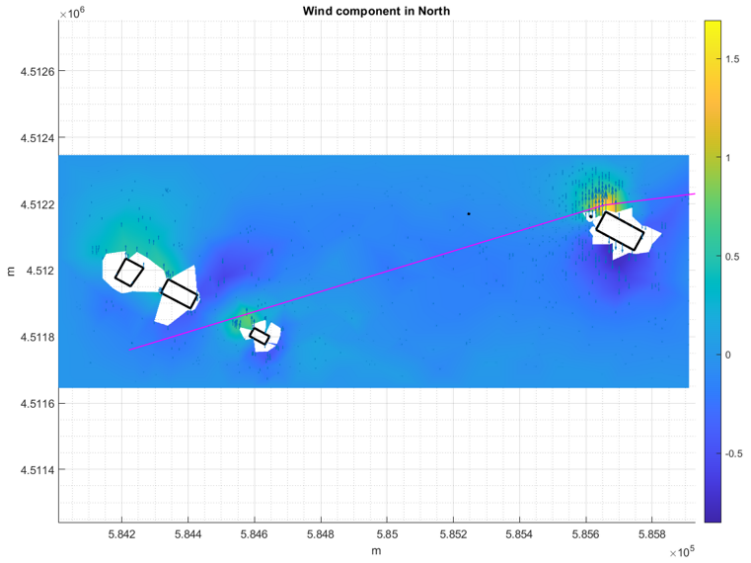


Figure 3.11: Wind vector field North component at 300m MSL.

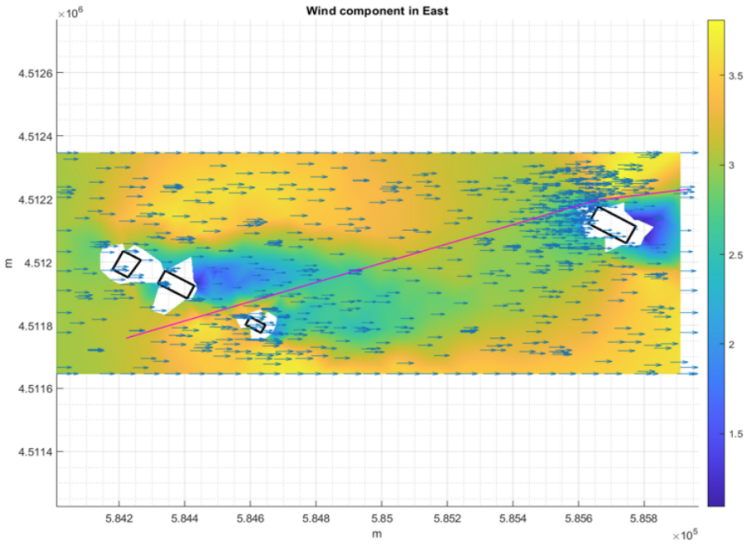


Figure 3.12: Wind vector field East component at 300m MSL.

3.5 Case Studies

The below case studies present geofence safety buffer sizing analyses for the Aerosonde UAS and our reference AAM model. Sensor parameters for each model are shown in Table 3.3. The sensor parameters of the Aerosonde UAS were taken from [2, 69], while the GPS and pressure sensor parameters of AAM were adjusted to have slightly less covariance values with an assumption that AAM has more accurate/costly sensor units. We assume sensor

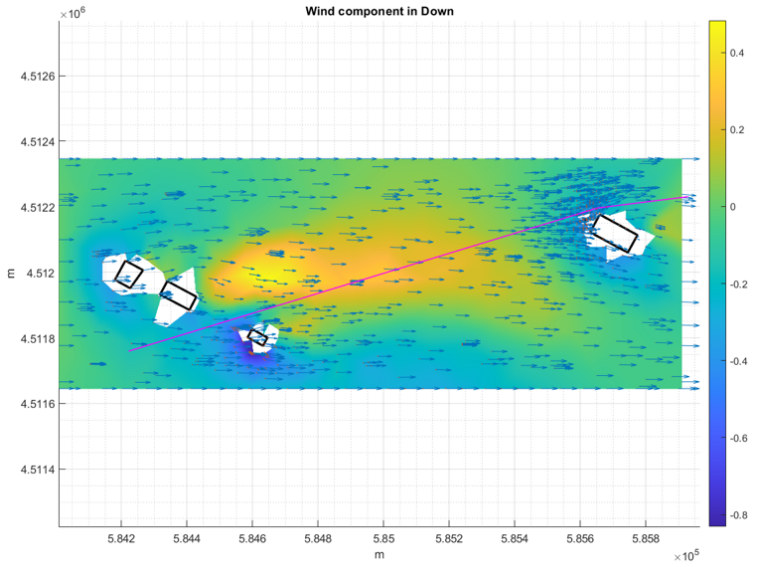


Figure 3.13: Wind vector field Down component at 300m MSL.

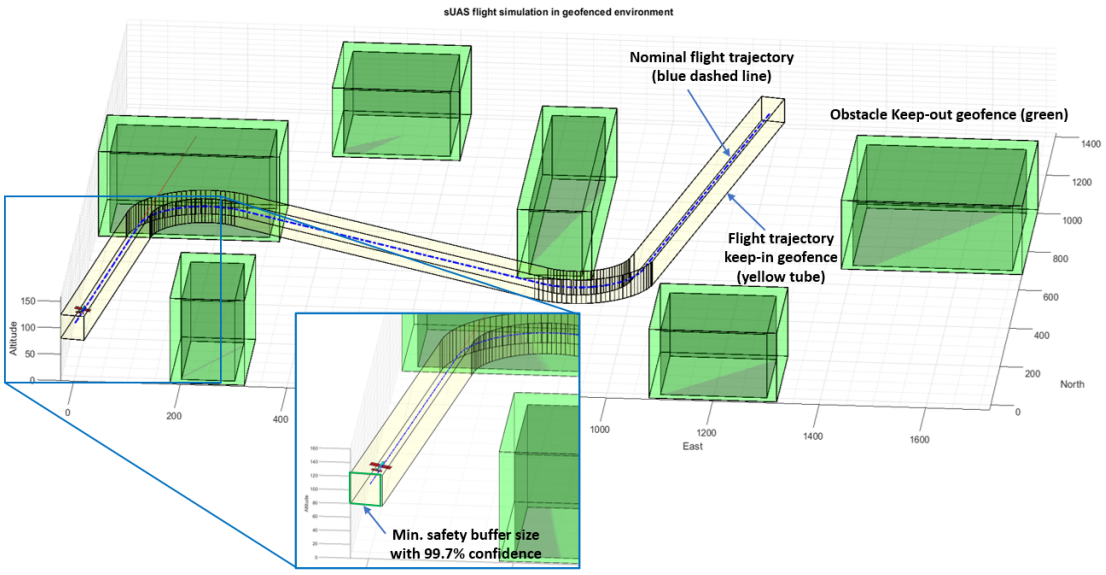


Figure 3.14: Visualization of a UAS flight plan with minimum geofence buffer sizing. Flight path segments with keep-in geofence wrappers are shown in yellow, and building/obstacle keep-out geofences are shown in green.

bias terms are removed through calibration. In our simulations, the cruising altitude for both Aerosonde UAS and AAM models was set to 300m MSL.

Figure 3.15 shows a visualization of UAS flight in Manhattan under a West wind of 3m/s to match our CFD analysis. The reference flight trajectory was defined near high-rise buildings so that the wind magnitude and direction change along the flight trajectory. The wind

Table 3.3: UAS sensor parameters

	Aerosonde	AAM
$\sigma_{accel}\ [m/s^2]$	0.024	0.024
$\sigma_{gyro}\ [rad/s]$	0.002	0.002
$\sigma_{staticpress}\ [Pa]$	10	7
$\sigma_{diffpress}\ [Pa]$	2	1.5
$\sigma_{GPS_n}\ [m]$	7.2	6.5
$\sigma_{GPS_e}\ [m]$	7.2	6.5
$\sigma_{GPS_h}\ [m]$	3.67	3
$\sigma_{GPS_{Vg}}\ [m/s]$	0.2	0.2
$\sigma_{GPS_\chi}\ [rad]$	0.05	0.05

component in North changed the most near buildings per the CFD vector field diagrams showing turbulence. Optimal geofence buffer sizings for urban terrain were therefore calculated for this particular 3m/s West wind condition. Vehicle parameters, state estimation uncertainties, and tracking errors for Aerosonde and AAM are shown in Table 3.4. The fuselage length and height of the Aerosonde UAV were estimated from the schematics in Figure 3.8 as the true dimensions are not publicly available. Note that positional standard deviation $\sigma_x, \sigma_y, \sigma_z$ were calculated by taking square of variances $[cov(x,x),\ cov(y,y),\ cov(z,z)]$ from EKF. Using Eq. 3.7, geofencing safety buffer sizes for Aerosonde UAS and AAM models are calculated and listed in Table 3.5. The result shows that the AAM geofence buffers are 4.47m and 0.76m larger than that of Aerosonde model in cross-sectional and longitudinal buffer, respectively, even though GPS and pressure sensors of the AAM are slightly better than for the Aerosonde due to the larger vehicle dimensions of the AAM model. Recall from Figure 3.7 the width of the flight trajectory geofence is $2\delta_{sb}$. Therefore, the width of the geofence for the for Aerosonde UAS is approximately $8m$ smaller than that for the AAM aircraft.

Table 3.4: Geofence buffer sizing parameters for each model [Unit: m].

	Aerosonde	AAM
$\epsilon_{guidance_y}$	16.90	18.11
$\epsilon_{guidance_z}$	5.45	4.86
σ	[1.08, 0.66, 0.28]	[1.03, 0.78, 0.23]
b	2.89	8.7
f_l	0.72	2.538
h	0.3	0.9

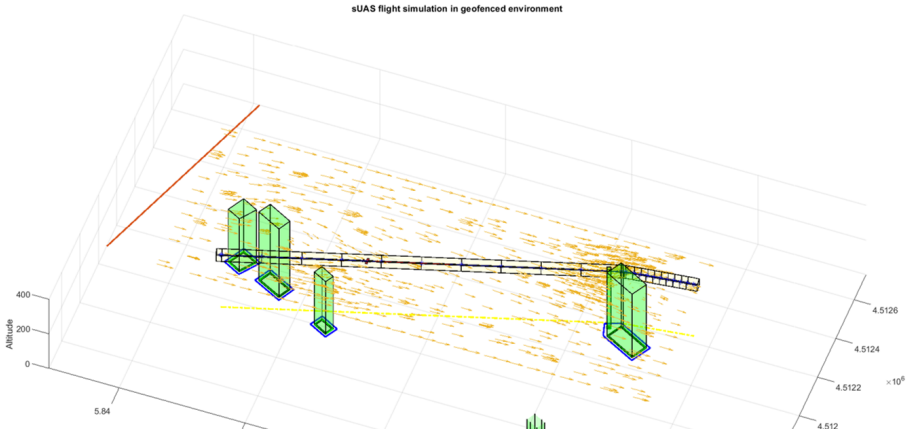


Figure 3.15: Visualization of UAS flight in Manhattan with a 3m/s West wind. Flight path wrapping keep-in geofences are shown in yellow, and building keep-out geofences are shown in green. The wind vector field is shown with orange arrows.

Table 3.5: Geofence buffer size for each model.

Model	δ_{sb} [m]	δ_{sb_x} [m]
Aerosonde UAS	20.33	3.6
AAM	24.8	4.36

Wind is an important source of tracking error for both the UAS and AAM platforms. The authors injected a much larger wind (10m/s magnitude) into the same map and found that both vehicles have more difficulty following the flight path leading to significantly larger trajectory tracking errors. This result confirms that optimal geofence buffer sizing particularly with irregular urban or mountain terrain must take wind properties into account to find geofence sizings that match a combination of vehicle, topographic environment, and wind conditions. Note that in the limit, a flight vehicle will be grounded when wind conditions are forecast to exceed safe operating constraints.

Figure 3.16 shows state estimates over time for Aerosonde UAS and AAM platforms. Specifically, position vector $[\hat{p_n}, \hat{p_e}, \hat{h}]$, airspeed $\hat{V_a}$, and attitude $[\hat{\phi}, \hat{\theta}, \hat{\psi}]$ are shown. Figure 3.17 shows a comparison of true vehicle states over time for the Aerosonde UAS and AAM platforms, including angle of attack α and actuation commands aileron δ_a, elevator δ_e, and normalized throttle δ_t]. The state estimation comparison shows position estimates for the AAM platform follow the commands (i.e., desired altitude at 300 meters, and desired airspeed

at 35 m/s) better as the GPS sensor covariances are lower in the AAM. The true state comparison shows that more thrust was required to fly the Aerosonde UAS under the same wind condition, while north and east position, airspeed, roll, and pitch angles displayed similar values in both the Aerosonde UAS and AAM platforms. At a time of 40 seconds, the vehicle approaches close to the building on the right, and the magnitude of wind flow in the North increased due to the turbulence created near the building. Wind influenced the roll and pitch of the Aerosonde UAS more than the AAM platform since the UAS has a comparatively low weight.

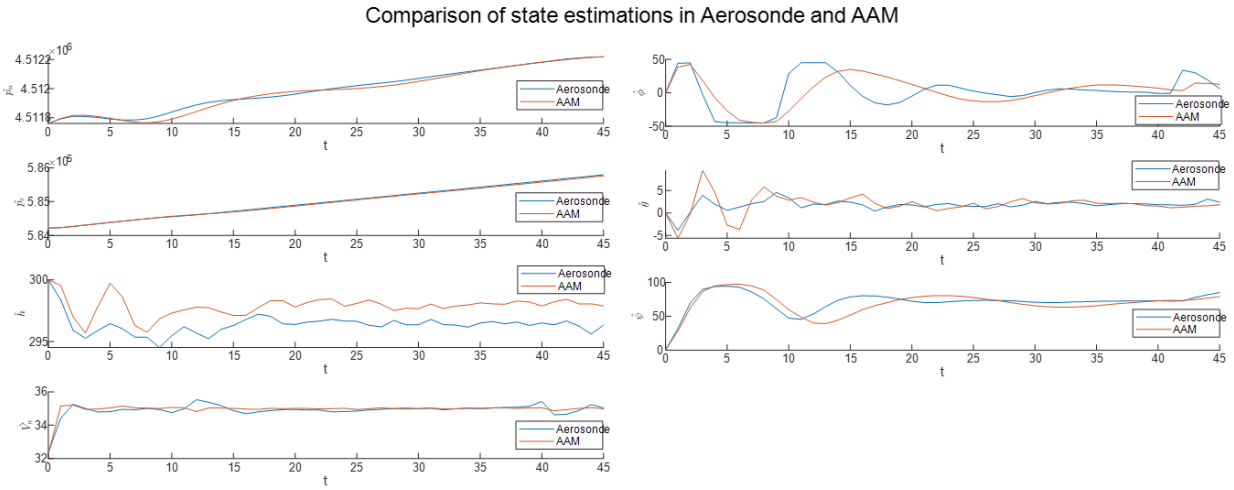


Figure 3.16: State estimate time series for UAS and AAM simulations.

3.6 Conclusion

This chapter has proposed a methodology to calculate flight trajectory geofence safety buffer sizings using vehicle performance, GNC, and expected wind models. Uncertainties in state estimates and trajectory tracking error were statistically modeled and translated along with vehicle dimensions to geofence buffer size. A passenger-carrying AAM aircraft model was defined by upscaling an Aerosonde UAS model. A CFD model was generated around a group of buildings in Manhattan taller than 300m MSL to simulate the effect of wind flow in an urban environment. Case studies show geofence buffer sizes computed for both the Aerosonde and AAM models. In future work additional environments, wind speeds, and vehicle models need to be analyzed to better understand how geofence buffer dimensions vary as a function of each parameter. We hypothesize that small fixed-wing UAS geofence buffers will have similar sizings to the Aerosonde in similar wind conditions but further

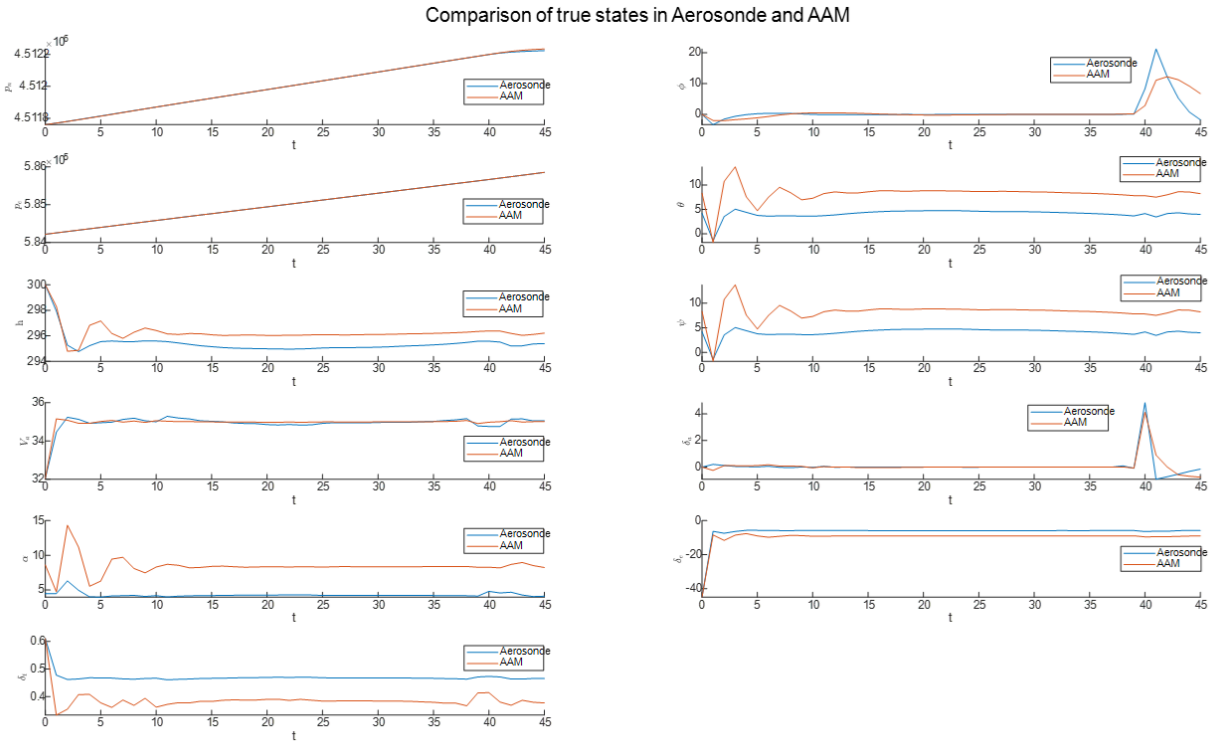


Figure 3.17: True state time series for UAS and AAM simulations.

analysis is required to confirm this hypothesis. For future analysis, we plan to generate a database for varying wind conditions in different city models (i.e., buildings and terrains) and investigate the effect of different wind conditions and vehicle types on geofence buffer sizes.

CHAPTER 4

Geofencing for Three Dimensional Flight and Swarms

4.1 Introduction

As the number of Uncrewed Aircraft Systems (UAS) increases, low-altitude airspace aircraft density will also increase, especially in the urban environment [20]. The Unmanned Aircraft System Traffic Management (UTM) system was proposed to efficiently organize the airspace of a large number of UAS [77]. These UTM systems will need to utilize this airspace in the most efficient way possible while still prioritizing the safety of buildings, people, and the UAS themselves. With the deployment of UAS for delivery, inspection, and first response, these UTM systems will also need to deconflict numerous flights with diverse mission profiles.

This chapter extends our previous work [38] by choosing a more space-efficient design for climbing and descending geofence volumes. [1] The efficiency of geofencing for these parts of the flight path increases as point-to-point UAS missions increase. Package delivery and air taxi cast as urban air mobility (UAM) or more generally advanced air mobility (AAM) rely on being able to ascend and descend through what will ultimately become a busy low-altitude airspace. Airspace must be efficiently allocated and utilized to support all missions. Our previous work utilized a constant ceiling and floor multiple-staircase geofence (MSG) in order to maintain a constant safety buffer around a UAS in flight. The method outlined in this chapter is to instead wrap a climbing or descending UAS with a parallelepiped geofence (PG) volume, extending the definition in [19] to utilize full polyhedra rather than lateral polygons in geofence definitions. The polyhedra in PGs require more data and processing overhead to manage. This chapter compares the difference in MSG and PG computational complexity as their relative efficiency with respect to reserved airspace volumes.

This chapter also explores the spatial and computational efficiency of two different meth-

[1] This chapter was also published as [78].

ods to create geofences for multi-agent UAS teams or swarms. The spatial and traffic management efficiency of such geofences becomes important as cooperatively controlled UAS teams enter the airspace. These UAS groups must be able to use the same UTM systems as individual UAS in order to ensure a safe environment without requiring individual geofences for each UAS, a technique that is not scalable. As such, there must be geofence volumes to handle these groups of UAS in an efficient way. The first swarm geofencing method we consider builds on the geofence definition of [19] with a constant floor and ceiling. This geofence volume is a bounding box that surrounds the UAS group. The second method uses a 3D convex hull algorithm [79] to create a more geometrically complex but spatially efficient geofence based on the group's geometry. This chapter compares differences in the computational complexity of these two methods as well as airspace volume utilization.

The chapter is organized as follows. A brief literature review is followed by an updated geofence definition and specific definitions of climb and descent MSG and PG geometries. Simulation results for MSG and PG benchmarking are followed by the definition of geofence containment volumes for UAS swarm geofencing. Containment geofence simulation results are followed by a brief conclusion.

4.2 Literature Review

UTM has been researched as a way to safely utilize low-altitude airspace. The Single European Sky ATM Research 3 Joint Undertaking has recommended that the EU adopt a UTM-type system for its UAS traffic management [80]. While most UTM research focuses on geofencing and its optimization for path planning, there is research into an Internet of things for uncrewed aerial vehicles [81]. This research proposes a system of interconnected nodes to facilitate UAS traffic. In addition, NASA has conducted research into UTM systems including a five-day flight test to research a UTM system beyond visual line of sight [82]. They concluded that a "UTM system can provide support to enable the proliferation of UAS operations." NASA expanded on this research in a partnership with ride-share company Uber to test their UTM system in use with urban air mobility [83]. They found that their UTM service could be used in this capacity, but there were significant concerns over the timing of the system. An overload of their UTM system could mean that information about deconfliction and flight planning could not be sent in a timely manner. This research into urban air mobility also adds to other work that describes the infrastructure needed to obtain pilotless urban air mobility [20]. In addition to research from NASA, there is research by the telecommunications company ATT to develop a UTM system using 5G networks as a basis for communication to the UAS [84].

Geofencing provides safe separation from buildings, people, and other UAS [19].NASA has has evaluated their geofencing hardware and software solution named SAFEGUARD in [85]. In addition, research has been done to classify and define the urban airspace environment in order to better understand the problems a UTM system will face in the real world[28].

4.3 Parallelepiped Geofence Definition

The parallelepiped geofence (PG) extends our previous fixed ceiling and floor definition with polygon cross-section to a three dimensional polyhedron specification. Additionally, the PG has a time element that describes which geofence in a sequence is active/inactive over time. The space efficiency of the PG definition comes at the cost of additional computational complexity for both generation and deconfliction of the geofences. Below is the formal PG definition accompanied by its illustration in Figure 4.1.

Definition 3 (Parallelepiped Geofence (PG)) *A parallelepiped geofence* $pg = \{n, v[\ ], m, e[\ ], h, t\}$ *is a volume defined by a list of eight three-dimensional Cartesian coordinate vertices* $v = [v_1, v_2, \cdots, v_8] = [(x_1, y_1, z_1), (x_2, y_2, z_2), \cdots, (x_8, y_8, z_8)]$ *and a list of twelve corresponding edges defining a parallelepiped geometry. The* (v, e) *graph is positioned relative to a home location* h *given by latitude, longitude, and altitude* (ϕ, λ, z) *Mean Sea Level (MSL). The volume is active during time interval* $t = [t_s, t_f]$ *where* t_s *is start or activation time and* t_f *is final or deactivation time.*

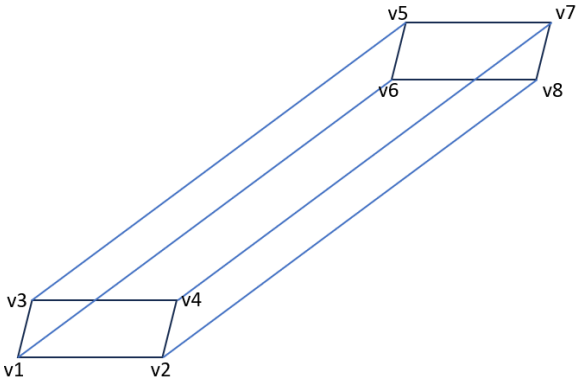


Figure 4.1: Illustration of a Parallelepiped Geofence (PG).

4.4 Space-Efficient Climb/Descent Geofence

Our previous work [38] constructed constant ceiling and floor multiple-staircase geofence (MSG) volumes for UAS climb/descent flight phases. The MSG design creates multiple stairs or "blocks" of geofences that progressively activate and deactivate to assure the UAS will always stay inside at least one MSG volume during its climb/descent. MSG was shown to reduce total airspace volume reserved compared to a single constant floor/ceiling geofence volume enclosing the entire climb/descent. We construct parallelepiped geofence volumes to further reduce geofence airspace volumes reserved during climb/descent trajectories. Although geofence definition and usage complexities are increased with parallelepiped construction per [19], our work shows significant airspace volume savings with the parallelepiped. Figure 4.2 illustrates MSG and parallelepiped volume designs.

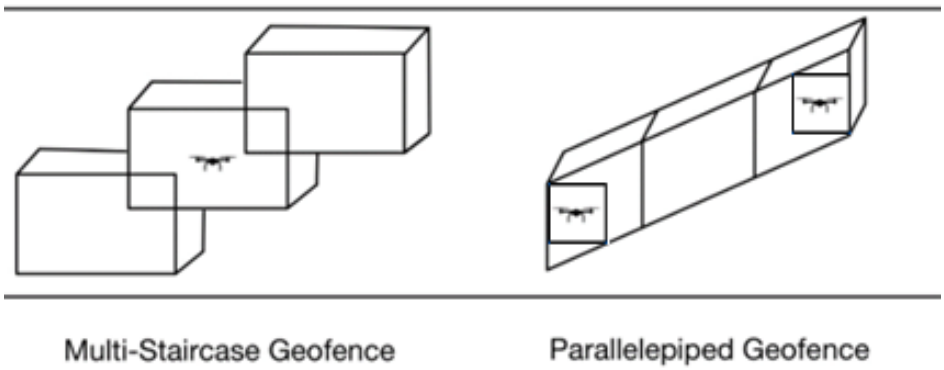


Figure 4.2: Multiple staircase geofence (MSG) (left) and parallelepiped geofence (PG) (right) for a steady climbing flight example.

4.4.1 Methodology and Algorithm

The parallelepiped geofence is constructed to maintain a minimum safety buffer around the UAS at every point of its climb. The safety buffer is treated as a constant distance in all body frame directions. Initially, a box (i.e., cube) representing that safety buffer is drawn around the UAS, and another box is constructed at the end of the UAS' climb/descent. These boxes are then connected by their outside edges to ensure the drone will always be within the safety buffer. Figure 4.1 illustrates the completed parallelepiped geofence. The parallelepiped geofence contains "blocks" much like the MSG in order to reduce the total amount of airspace used at any one time. These blocks activate and deactivate to ensure that the UAS will always be inside the volume with its safety buffer.

Parallelepiped geofence (PG) construction takes as its inputs a UAS' departure point $\mathcal{R}_{start}$, velocity $\mathcal{V}$, climb time $\mathcal{T}$, number of geofence blocks N_{geo}, and safety buffer δ_{sb}. The endpoint of the PG is determined by the departure point $\mathcal{R}_{start}$, velocity $\mathcal{V}$, and climb time $\mathcal{T}$. The time spent in each PG block is determined by dividing the climb time $\mathcal{T}$ by the number of geofences blocks N_{geo}. Each PG block is then created individually by determining that block's start and end points, start and end box vertices, and connecting the start and end boxes. These blocks are then added to the overall data structure.

4.4.2 Simulation Results

Figure 4.3 shows the result of **a geofence volume sizing comparison** between PG and MSG in climb/descent over **a range of safety buffer sizes and the number of geofence blocks**. The number of geofence blocks or partitions, flight path angle, and geofence safety buffer size were collectively used to generate geofence volumes. Percentages of volume saved with PG designs are shown in the figure. Particularly, the safety buffer size varied from 10m to 100m, and the number of blocks ranged from 5 to 50. As the number of blocks and safety distance decrease, the PG becomes more spatially efficient compared to the MSG (shown in yellow). As the number of blocks increases, the MSG approximates a smooth climb boundary resulting in similar volume sizing to the PG. When safety buffer distance increases, however, parallelepiped geofence efficiency is reduced due to the large amount of extra space created to accommodate large safety buffer height and length.

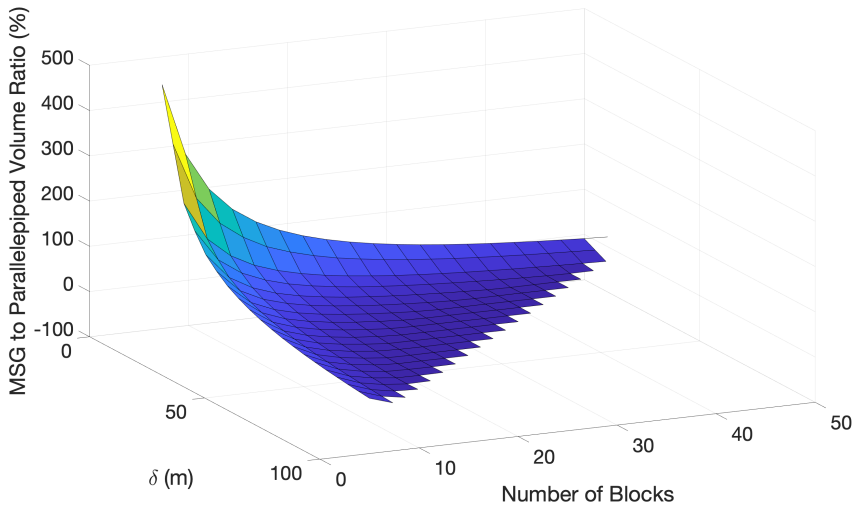


Figure 4.3: Comparison of MSG and parallelepiped geofence volumes as a function of number of geofence blocks and safety buffer distances for a 1000m climb with 45° flight path angle.

Figure 4.4 shows **a volume sizing comparison** of parallelepiped and multiple-staircase geofences over **a range of flight path angles γ and safety buffer sizes** δ. For this comparison, the flight path angle changes from 0° to 90°, and the safety buffer size varies from 10m to 40m. Results show that the parallelepiped geofence is most efficient relative to the MSG at a flight path angle of 45° and at a small safety buffer distance. This occurs due to MSG space inefficiency at 45°. Small safety buffers make these spatial inefficiencies more pronounced. At 0° and 90°, MSG and parallelepiped volumes are identical as the geofences became rectangular prisms to wrap horizontal and vertical flight paths.

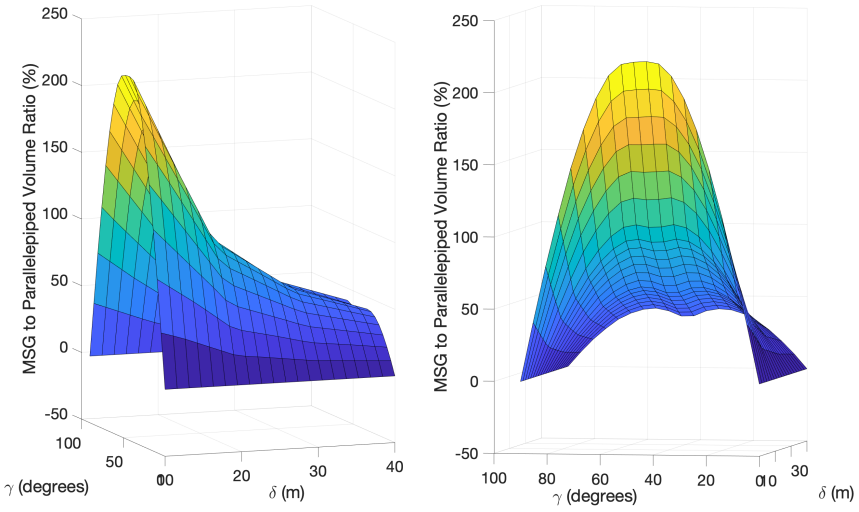


Figure 4.4: Comparison of MSG and parallelepiped geofence volumes as a function of flight path angle and safety buffer size for a 1000m climb distance with 10 geofence blocks.

Figure 4.5-4.6 show **the runtime comparison** of PG and MSG over **a number of blocks and safety buffer distance**. In both cases, the PG resulted in a slower computation time than the MSG. Particularly, with an increasing number of blocks, the PG runtime displayed a more significant delay in computing the volumes. However, the magnitude of the runtime was in the order of 10^{-3} second for a single climb/descent, still reliable for real-time volume generation in the UTM.j

Overall, the PG is more spatially efficient than the MSG in terms of volume sizing, but less efficient in terms of computation speed. Particularly, in its worst case, the PG results in the same amount of volume required for the MSG at γ of 0° and 90°. MSG spatial efficiency is comparable to PG efficiency outside of these extremes only for cases in which UAS safety buffer distances are large relative to climb/descent path length. However, the MSG can utilize existing geofence definitions per [19] and is computationally more efficient in generating the volume.

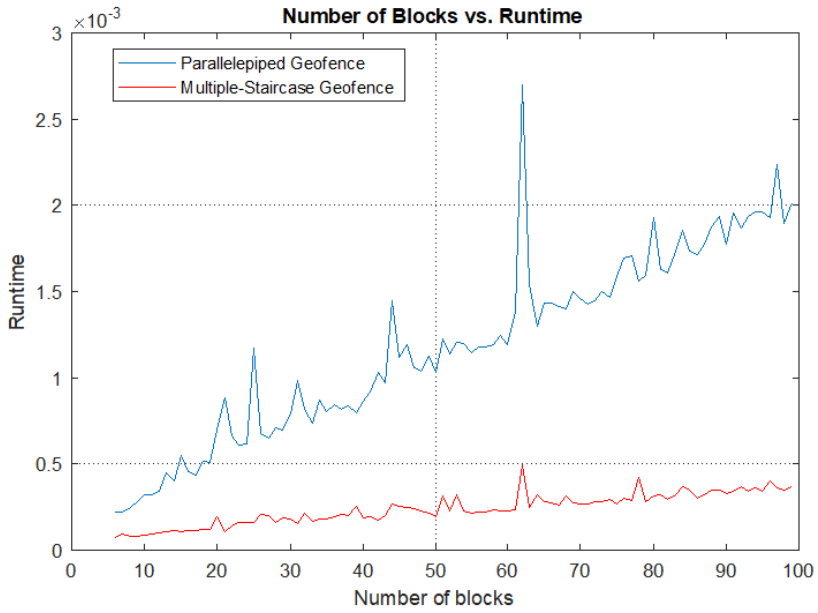


Figure 4.5: Runtime comparison of MSG and parallelepiped geofence as a function of the number of blocks.

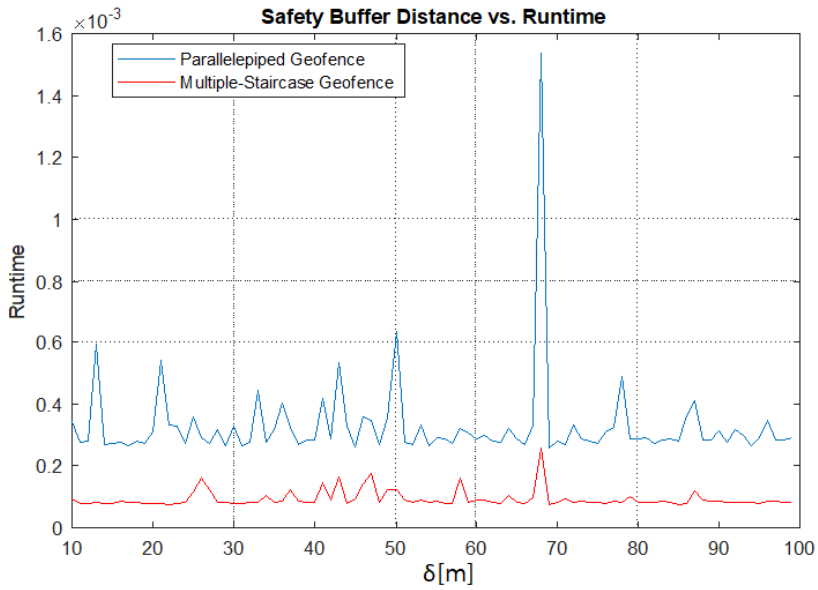


Figure 4.6: Runtime comparison of MSG and parallelepiped geofence as a function of safety buffer size.

4.5 Space-Efficient Containment Geofence for Swarm Formation

4.5.1 Methodology and Algorithms

A containment geofence for a cooperative UAS team can be generated by first creating minimum separation boxes around each UAS in the swarm, and then generating a three-dimensional convex hull containment geofence as shown in Figure 4.7. We used 3-D convex hull algorithms to create these volumes in O(nlogn) complexity [79]. A bounding box geofence is generated to compare the volume saved using a more complex but tight convex hull containment geofence. The bounding box (shown in red) was constructed by taking the largest value in the x, y, and z axes to generate a box. In addition, 3D kinematics can be used to rotate and move volumes in space when needed. These 3D movements can be obtained by using 4x4 rotation and translation matrices[86]. Analogous to the parallelepiped geofence, the convex hull geofence volume has more in-plane geometric complexity and also may not have a constant floor and ceiling. However, the volume saved from the containment geofence can be significant. As seen in Figure 4.7, there is much space within the red bounding box that is not needed to provide a safety buffer for the swarm due to the geometry of the swarm itself. This amount of unused space in the bounding box compared to the convex hull containment geofence changes with the geometry of the swarm as well as the size of the safety buffer. The containment geofence generation algorithm is shown in Algorithm 5.

Algorithm 5: Convex Hull Geofence Construction

Inputs: Drone Positions $\mathcal{D}_{pos}$, Safety Buffer δ_{sb}

Outputs: Vertices $v[\,]$, Edges $e[\ \ ]$

Algorithm:

1: $v_{\text{cubes}} \leftarrow$ empty N x 3 matrix
2: **for** $i \in length(D_{pos})$ **do**
3: $\quad v_{D_{pos}} \leftarrow \text{createCubicGeofence}(D_{pos}(i,:), \delta_{sb})$
4: $\quad v_{\text{cubes}} \leftarrow \text{add}(v_{D_{pos}}) \lhd$ all vertices in cube geofence added to the matrix v_{cubes}
5: **end for**
6: $[v, k] \leftarrow \text{ConvexHull3D}(v_{cubes}) \lhd$ 3D convex hull algorithm
7: $[v, e] \leftarrow RemoveEdges(v, k) \lhd$ removed unnecessary edges from convex hull volume

Once the containment geofence is generated using the Algorithm 5, homogeneous transformation [86] was applied to rotate and translate the geofence along the trajectory that a cooperative UAS team flies.

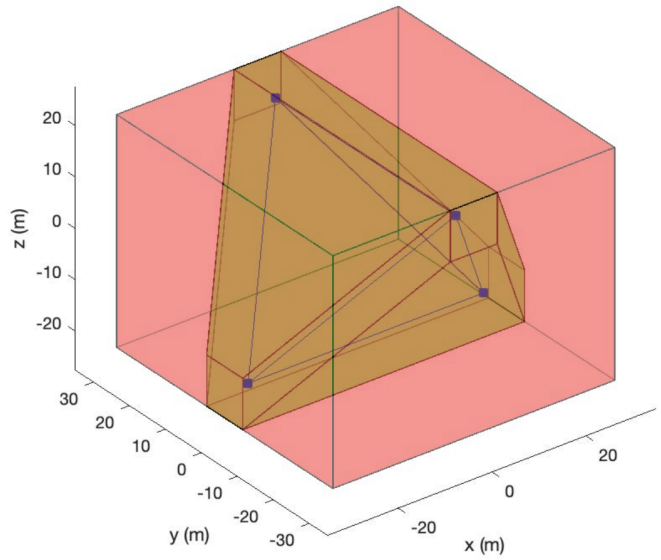


Figure 4.7: Containment geofence Volume case study for a four UAS team. A bounding box geofence volume is shown in red, and a convex hull containment geofence is shown in green. Blue squares illustrate the UAS positions.

4.5.2 Simulation Results

Figure 4.8 shows a volume sizing comparison between containment geofence and bounding box geofence to support a coordinated flight of four UAS. The swarm flight formation in this example case defines a regular tetrahedron shape. Results show that the containment geofence volume is smaller compared to the bounding box given relatively small safety buffer distances and relatively large separations between the four UAS. As the distances between the UAS in a formation increase, the bounding box geofence generates a volume much greater than the necessary volume to contain the coordinated UAS team.

Figure 4.9-4.10 show **the runtime comparison** between containment geofence and bounding box geofence to support a coordinated flight of four UAS. In both cases, the runtime of the containment geofence resulted in slower computation time than the single bounding box geofence. The containment geofence relies on a three-dimensional convex hull algorithm, which penalizes the runtime compared to the single bounding box. Particularly, the result showed that as the distance between individual drones increased, the runtime increased. However, note that the magnitude of the runtime is in the order of 10^{-3} seconds for a single containment geofence, still reliable for real-time volume generation in the UTM.

Complementary swarm configurations and their containment geofence vs. single bounding box volume comparisons are shown in Figures 4.11 - 4.13. Such tested configurations are

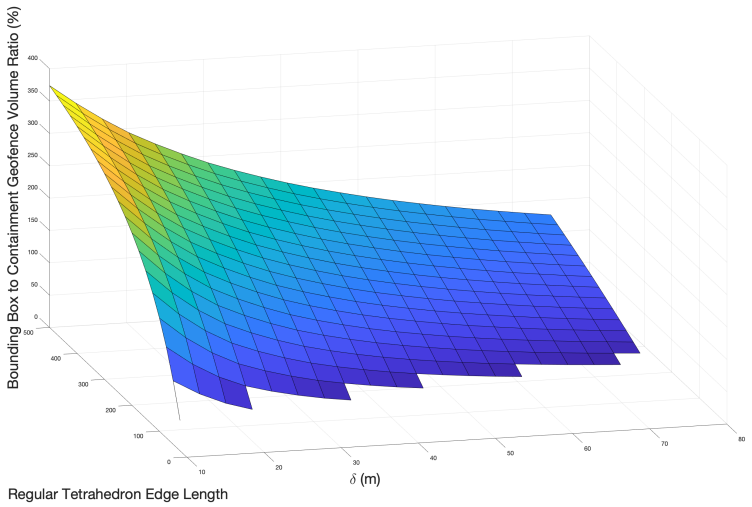


Figure 4.8: Volume Comparison of containment and bounding box cooperative UAS team geofencing designs as a function of distances between UAS and safety buffer sizing.

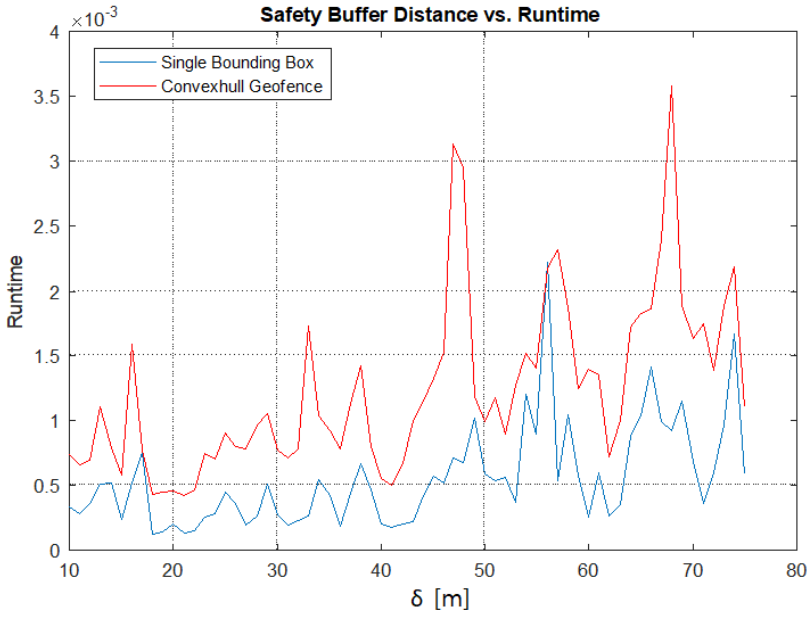


Figure 4.9: Runtime Comparison of containment and bounding box cooperative UAS team geofencing designs as a function of safety buffer distance. The distance between UAS is fixed to 200m in the simulation.

"straight-line", "inverted V" and "3-D Prism" swarm formations. The containment geofence becomes a single bounding box in the case of straight-line swarm formation and therefore showed no volume saving in containment geofence in this particular case. However, for "inverted-v" and "3-D prism" configurations, the simulation showed that as the distance

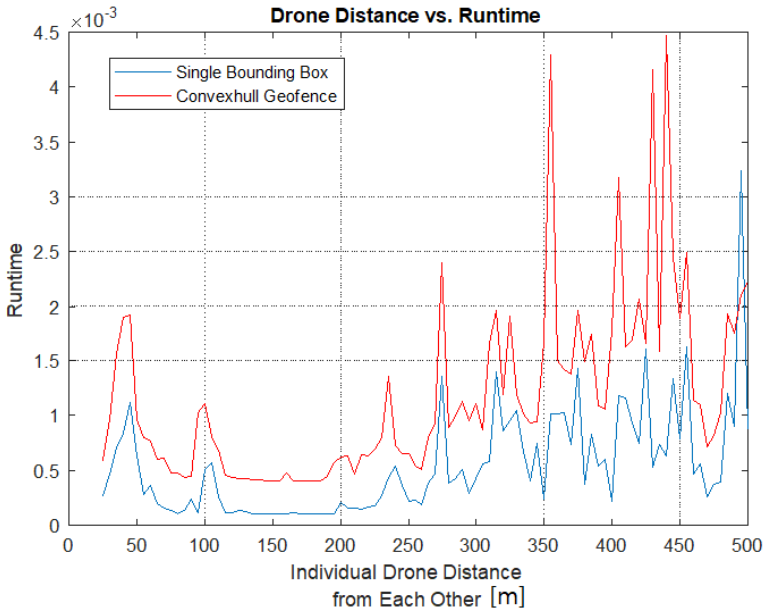


Figure 4.10: Runtime Comparison of containment and bounding box cooperative UAS team geofencing designs as a function of the distance between UAS. The safety buffer is fixed to 10m in the simulation.

between individual drones increases while the safety buffer size is small, relative containment geofence airspace volume reduction also increases. This reveals that containment geofence can reduce the reserved airspace volume in arbitrary swarm formations, but the magnitude of volume saving depends on the distance between individual drones and the safety buffer size.

4.6 Discussion

The simulation results show that the parallelepiped geofence algorithm for climb and descent flight paths, as well as convex hull swarm containment geofencing, provide more space-efficient volumes compared to the MSG and single bounding box geofence. It is important to note that while this is true, the computational cost of such geofences may make them unfeasible in the real UTM system. Further research needs to be conducted to determine the trade-off between spatial efficiency and computational cost in more general cases. Furthermore, geofence volume deconfliction among different PG, MSG, and swarm containment geofences using the above definition is needed.

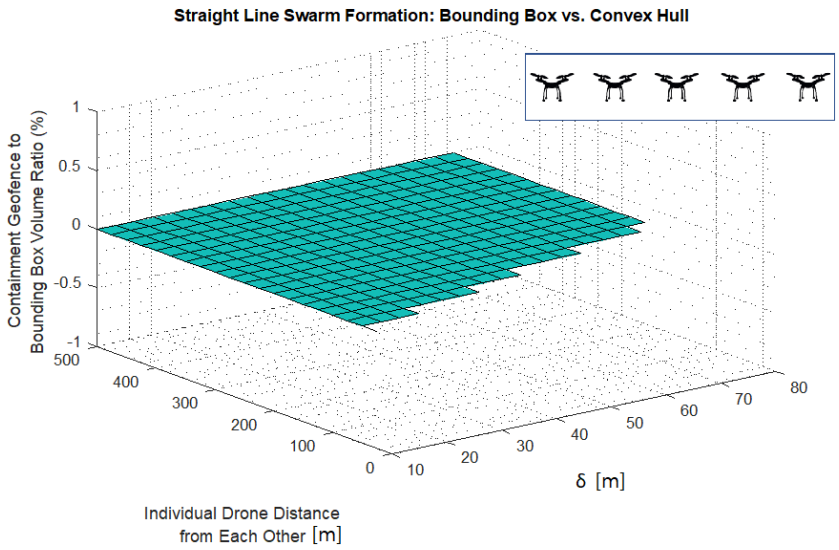


Figure 4.11: Volume Comparison of containment and bounding box cooperative UAS team in "straight-line" swarm configuration.

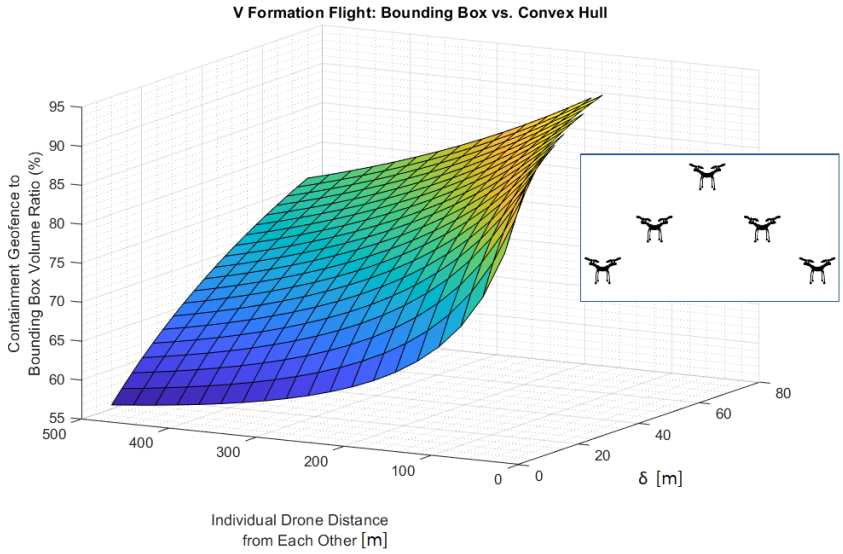


Figure 4.12: Volume Comparison of containment and bounding box cooperative UAS team in "inverted V" swarm configuration.

4.7 Conclusion

Space-efficient UAS geofencing will minimize the airspace volume reserved in densely populated airspace to maximize airspace availability for other UAS missions. This chapter has presented parallelepiped climb/descent and convex hull swarm containment geofencing designs along with a summary of results indicating space efficiency gains as a function of flight

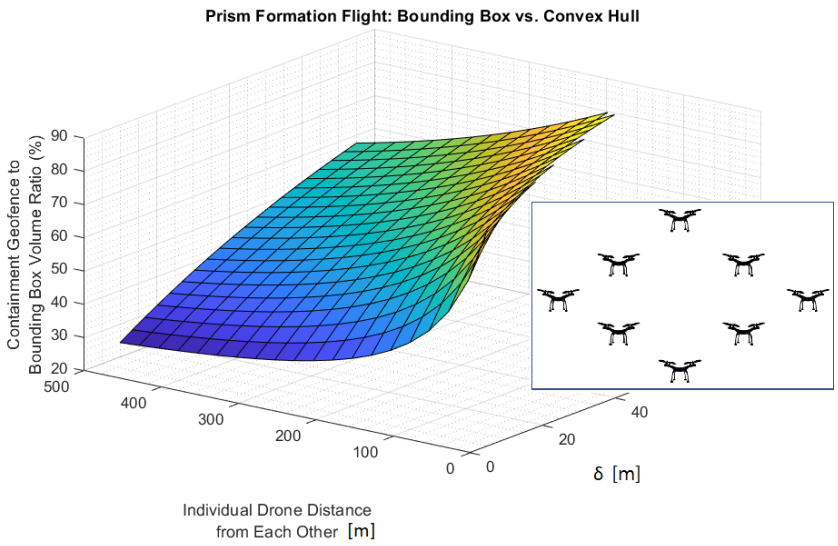


Figure 4.13: Volume Comparison of containment and bounding box cooperative UAS team in "3-D prism" swarm configuration.

path angle, safety buffer, number of geofence blocks, and trajectory length. We also compared the run times of these geofences to geofences created using a polygon cross-section with constant altitude ceiling and floor. Our polyhedra geofence volumes did provide more efficiency in airspace use, but, on average, polyhedra geofence processing time is greater than polygon geofence (MSG) processing time.

Future work will extend this chapter's geofence volume definitions to account for wind and safety buffer requirements that may not be identical for all vehicles in a team or swarm.

CHAPTER 5

Assured Contingency Landing Management for AAM

5.1 Introduction

Advanced Air Mobility (AAM) is expected to provide flexible on-demand passenger and package regional air transportation [87]. To ensure safe and economically viable operations particularly in urban mixed-use airspace, AAM scalability will require increasing levels of autonomy for both crewed and uncrewed aircraft [20]. Certified flight management system (FMS) and other fly-by-wire aircraft automation have improved safety of flight through reliable system management and flight plan following. Contingency management has typically been left to the flight crew, however, because human pilots can be innovative and because fleet-wide certified software upgrades would be costly. AAM growth will likely outstrip experienced pilot availability, and inexperienced pilots are less likely to react safely in case of emergency, especially given the tight real-time response constraints associated with regional flight at low altitude. Contingency management autonomy is therefore critical for safe AAM.

When a safety-critical problem is encountered in flight, the best course of action is typically to safely land as soon as possible. This chapter proposes an assured contingency landing management (ACLM) autonomy framework suitable for both human-occupied aircraft and uncrewed aircraft systems (UAS). By fusing a pre-analyzed landing site database that are categorized into prepared and unprepared, ACLM enables distressed AAM flights to swiftly define a safe landing plan, minimizing reaction time and preventing missed opportunities due to decision delays.

Fig. 5.1 shows the ACLM architecture applied to a multicopter UAS. The vehicle system includes guidance, navigation, and control (GNC) functions as well as prognostics functions that include fault detection and identification (FDI). ACLM receives as input the current state vector $\vec{x}(t)$, time remaining until battery system End of Discharge (EOD) t_{EOD}, rotary propulsion unit health parameters $\eta_r(t)$, and rotor thrust commands $f_r(t)$ attenuated

based on FDI analysis. ACLM returns a contingency flight plan $\mathcal{W}_{ACLM}$ based on six interconnected modules: 1) A controllability and reachability (C&R) watchdog function that rapidly identifies the need for contingency or emergency landing, 2) A pre-flight (offline) flight planner that builds and indexes contingency plans that can be rapidly retrieved, 3) An online flight planner that can build new flight plans in real-time when needed, 4) A landing strategy selector to retrieve a cached plan when available or initiate real-time flight planning to a minimum-risk landing site otherwise, 5) A continue/hold selector to hold (e.g., hover) or continue as well as possible along the original flight plan while contingency flight planning and initiation actions are completed, and 6) A flight termination (e.g., parachute deployment) function available as a last resort. The output $\mathcal{W}_{ACLM}$ is an ordered list of n waypoints defining the contingency landing plan or $\emptyset$ indicating flight termination.

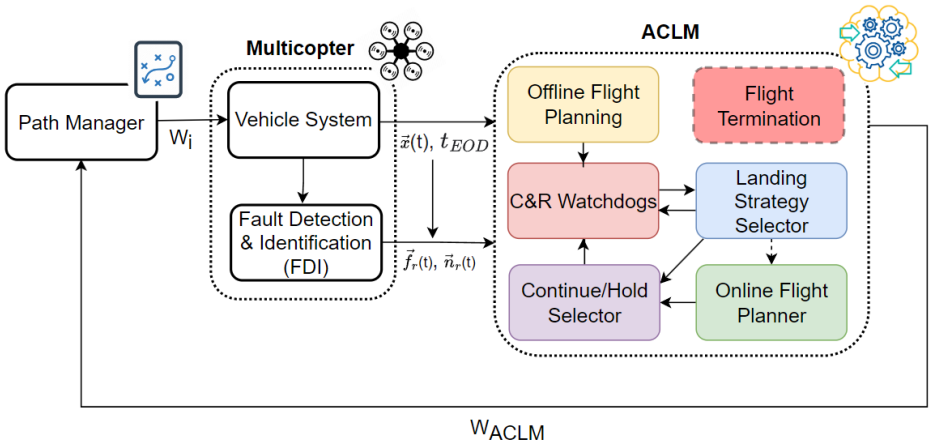


Figure 5.1: Assured contingency landing management (ACLM) with a multicopter application.

This chapter proposes a single-aircraft ACLM solution applied to a multicopter with motor and battery faults as potential failure modes. These two failure modes are prevalent in small UAS [88] and have a substantial impact on aircraft controllability and reachability which in turn trigger and influence contingency response. We present the logic and mathematical derivation of the ACLM functions designed to achieve Assurance Efficacy (AE) in auto-mitigation responses. ACLM ensures adequate control authority and contingency flight planning in failure scenarios. Case study simulations illustrate the influence of motor and battery degradation and failure cases on safe landing decisions. We apply ACLM to a small hexacopter UAS with experimentally validated performance as well as motor and battery degradation models [89, 90], and integrate Guidance, Navigation, and Control (GNC) functions with experimentally validated performance, dual battery pack, and propulsion unit

models. Monte Carlo simulations are conducted using real-world Manhattan map data to create random failure scenarios. Results demonstrate the functionality of ACLM in motor and battery degradation case studies.
Contributions of this work include:

- ***Logic and data flow of the ACLM architecture:*** We integrate mathematically-provable positive controllability and reachability logic to assure a contingency landing is triggered whenever the current plan is no longer feasible. While controllability and reachability principles are well established, their integration into the ACLM watchdog based on prognostic feedback such as battery system t_{EOD} is novel.

- ***Assured offline landing plan integration:*** Contingency landing plans prepared before flight can be optimized and validated by both software and human experts. ACLM therefore stores and indexes pre-flight contingency landing plans for each filed flight plan segment and each anticipated degradation scenario. ACLM can in turn ensure that each cached contingency plan has been validated and will be retrieved/initiated with minimal delay. While alternate landing plans have been prepared by pilots and FMS long-term, ACLM's plan database indexed by flight segment and prognostics status is novel.

- ***Integration of categorized landing site database:*** While landing at a prepared vertiport site is preferred, no such site may be reachable given motor failure and/or battery degradation. ACLM therefore caches a map database of unprepared landing sites to facilitate identifying a nearby risk-minimizing unprepared landing site when real-time planning is required. Most previous work has assumed a combination of real-time site mapping or strictly use of prepared vertiport/runway sites.

- ***Simulations in a realistic environment model:*** Simulation studies apply ACLM to a suite of in-flight failure and degradation scenarios. Our combination of real-world Manhattan building and landing site models plus experimentally validated vehicle, motor, and battery performance and degradation models is novel, improving the realism of our results compared to previous contingency management simulation studies.

The structure of this chapter is as follows. Section 5.2 summarizes related work in emergency flight planning and decision-making. Section 5.3 describes the underlying algorithms in each ACLM function, illustrating how assurance is achieved for each. Section 5.4 outlines key assumptions and our procedure to construct environmental models and landing

site databases. Additionally, it summarizes Monte Carlo simulation studies designed to test the ACLM pipeline in different failure cases. Section 5.5 presents Monte Carlo simulation and ACLM solutions for lighter-weight and heavier-weight package-carrying hexacopters operating in Manhattan, New York City, under different in-flight anomaly scenarios. Section 5.6 discusses strategies for minimizing the need for flight termination in future work, and Section 5.7 concludes the chapter.

5.2 Literature Review

A fault-tolerant control scheme is crucial to ensure the safe completion of missions in AAM flight operations. Such a scheme is proposed for multicopters in [91], using a reduced controllability index to compute a constrained control allocation scheme. This work employs a Nonlinear Dynamic Inversion (NDI) controller to land the faulty multicopter safely. Hamadi et al. [92] present a data fusion architecture to increase tolerance to sensor and software faults without modifying the flight plan or control law. This architecture focuses anomaly detection and recovery services to multi-sensor perception systems.

Even degraded AAM platforms must be assured to safely reach emergency landing sites. Assurance can be formulated with reachability analysis in physical space. The Hamilton-Jacobi reachability method [93], [94] computes a set of states that drive the system to a target set while satisfying time-varying state constraints. Hamilton Jacobi Bellman (HJB) reachability analysis was employed to determine the feasible landing region in [95]. Reachability can be defined in terms of feasible post-failure trim states and transitions between them, utilizing a database of feasible trim states and transitions to construct landing plans [96]. Ref. [97] proposes computation of an approximate footprint to rapidly define reachable landing sites given fixed-wing aircraft experiencing loss of thrust. Ref. [98] extends this work by maximizing range as a function gliding turn parameters.

A real-time contingency planning system is presented in [99] with metrics that assess mission safety and feasibility. Flight states are categorized as Nominal, Off-Nominal, Alternate Land, and Land Now using a finite state machine with state transitions that can trigger replanning. In [100], the Safe2Ditch architecture integrates real-time landing site selection with visual situation awareness to identify and assess potential landing sites within onboard sensor field of view. In [101, 102], architectures to plan safe trajectories for autonomous helicopters are proposed that combines deterministic and learning functions. Procedures for managing lost link in remotely piloted aircraft are proposed in [103], leveraging historical traffic volume data to minimize encounter risk. These procedures are evaluated through human-in-the-loop simulations to assess operator workload. A Markov Decision Process (MDP) formulation was

used for contingency landing management based on motor and battery prognostics feedback in [104, 105].

The contingency management solutions cited above require mission and flight planning. Search-based planners typically cannot guarantee real-time execution given limited embedded computing resources. Further, requiring each aircraft to identify landing sites with onboard sensors can also be time-consuming and limits results to the field of view. Our ACLM framework, initially presented in [106], integrates controllability and reachability analysis to assure safety and utilizes a map including a landing site database to generate and cache contingency landing plans along the planned route prior to flight. Simulations show that ACLM retrieves or computes contingency responses in milliseconds when triggered by ACLM's controllability and reachability (C&R) watchdog.

5.3 Assured Contingency Landing Management

As described above in Fig. 5.1, ACLM integrates six functional modules: offline flight planning, C&R watchdog, landing strategy selector (LSS), continue/hold selector (CHS), online flight planner, and flight termination. The logic and algorithm for each is discussed below. The output of ACLM is a waypoint-based contingency flight plan $\mathcal{W}_{ACLM} = \{w_{i,1}, \cdots, w_{i,j}, \cdots, w_{i,n}\}$, where each waypoint $w_{i,j}$ is given by $\{\sigma_j, \lambda_j, h_j\}$. Here σ_j denotes waypoint latitude, λ_j denotes waypoint longitude, and h_j is waypoint flight altitude. i indicates a database index for the selected landing flight plan, and j indicates the waypoint index in that plan.

5.3.1 Offline Flight Planning

A nominal flight plan $\mathcal{W}_{nom}$ is created from start and destination locations, an aircraft performance model, company preferred route information, and airspace/traffic constraints. As shown in Fig. 5.2, ACLM's offline flight planning module receives $\mathcal{W}_{nom}$ as input from which it calculates and stores a database of contingency landing plans that begin at the midpoint and endpoint of every nominal flight plan segment.

Flight plan $\mathcal{W}_{nom} = \{w_1^{nom}, w_2^{nom}, \cdots, w_n^{nom}\}$ is defined by a sequence of waypoints $w_i^{nom} = \{\sigma_i, \lambda_i, h_i\}$. A set of approximate reachable footprints denoted $\mathcal{F} = \{F_1^{mid}, F_1^{end}, \cdots, F_n^{mid}, F_n^{end}\}$, defines midpoint and endpoint landing footprint geometries for each flight segment. Each midpoint footprint $F_i^{mid} = \{R_i^{mid}, \sigma_i^{mid}, \lambda_i^{mid}\}$ is defined by a circle with radius R_i^{mid} and center $(\sigma_i^{mid}, \lambda_i^{mid})$ corresponding to the midpoint of waypoint segment $[w_i, w_{i+1}]$. Endpoint footprint F_i^{end} is defined analogously.

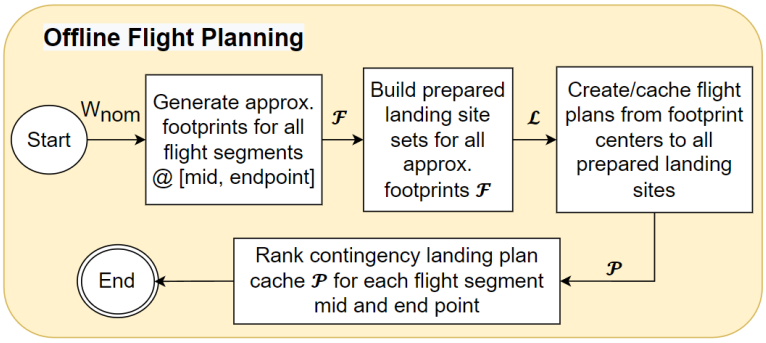


Figure 5.2: Offline flight planning logic flow.

For each flight segment midpoint footprint F_i^{mid}, the set of prepared landing sites is defined as $\mathcal{L}_i^{mid} = \{L_{i,1}^{mid}, \cdots, L_{i,k}^{mid}, \cdots, L_{i,m}^{mid}\}$, where each landing site $L_{i,k}^{mid}$ is characterized by latitude ($\sigma_{i,k}^{mid}$), longitude ($\lambda_{i,k}^{mid}$), altitude ($h_{i,k}^{mid}$) coordinates and landing site risk ($\mathcal{R}_{i,k}^{mid}$). The set of prepared landing sites for the endpoint footprint F_i^{end} is defined analogously. The landing site risk model is adopted from [107], incorporating cost terms including landing site area, terrain/landing site type. For this work, we define *prepared* landing sites as protected vertiports and runways intended for aircraft use, and *unprepared* landing sites as terrain, building rooftops, roads, et al that are not protected and not typically utilized by aircraft. Section 5.4.1 provides more details. The flight plan database generated offline contains landing solutions to prepared landing sites, all of which are considered low-risk. Additional landing sites for small UAS, e.g., unoccupied flat building rooftops approved for emergency landing, are also considered "prepared" thus low-risk in this work. The remaining *unprepared* landing sites that are mapped but higher risk serve as alternative options for the online flight planner when no low-risk or prepared landing sites are reachable.

ACLM builds and caches a contingency landing plan database $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ during preflight. Each nominal flight segment i has contingency landing plan set $P_i = \{P_i^{mid}, P_i^{end}\}$ where each $P_i^* = \{P_{i,1}^*, P_{i,2}^*, \cdots, P_{i,m}^*\}$ for all reachable landing sites in $\mathcal{L}_i^*$ where $* \in \{mid, end\}$. The cost of a contingency flight plan $P_{i,j}^*$ is calculated using Equation 5.1.

$$\mathcal{C}_{i,j}^* = \beta \cdot d_{travel} + \gamma \cdot \mathcal{R} \tag{5.1}$$

where β and γ are cost function weights, d_{travel} is landing path distance traveled, and $\mathcal{R}$ is the risk of landing site $j \in \mathcal{L}_i^*$. The landing site flight plans within each footprint are then ordered by cost and indexed by $\{i, j, *\}$ in a contingency landing plan lookup table. Computing and storing contingency landing plans during preflight minimizes ACLM execution time since real-time flight planning is the most computationally expensive ACLM function. Fig. 5.3

provides a visual depiction of the flight plan database.

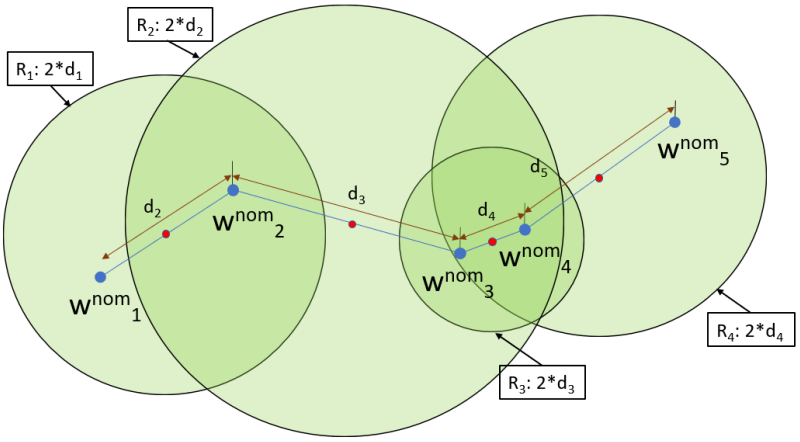


Figure 5.3: Footprints utilized to define a contingency landing plan database during preflight planning at each flight segment midpoint, represented graphically. Analogous footprints are constructed for each segment endpoint. The blue dots represent waypoints in $\mathcal{W}nom$, while the red dots indicate the midpoints of the flight segments. The green circles represent approximate footprints with radii $R_i = 2 * d_i$ at each midpoint.

5.3.2 Controllability and Reachability (C&R) Watchdog

The C&R watchdog checks the controllability and reachability of the current landing site, performed during each ACLM sampling rate. The logic flow of the C&R watchdog is illustrated in Fig. 5.4. The first step is to test for positive controllability $C(t)$. Condition $C(t) < 0$ indicates that the controllability matrix is not full rank or the system lacks sufficient control authority to follow the prescribed flight plan. In this case, the flight termination thread is initiated. Reachability $R(t)$ in this work is computed by projecting the executing flight plan from the current state to the landing waypoint with consideration of an updated estimate of the battery's End of Discharge (EOD) time and the current vehicle performance model. If $R(t) == 1$, no warning is issued, and the current flight plan continues as planned. In the event that $R(t) == 0$, indicating that the current landing site is not reachable, the C&R watchdog triggers the Landing Strategy Selector (LSS) parallel thread to find contingency plans, and the "Planning In Progress (PIP)" flag is set to true until a reachable contingency plan is found. Meanwhile, the C&R watchdog counts the number of times the watchdog executes with an unreachable status. If this count exceeds a preset threshold, the watchdog triggers flight termination within the Landing Strategy Selection (LSS) process, indicating an LSS failure.

Execution of flight termination in this version of ACLM is appropriate for a parachute-equipped small UAS. A passenger-carrying AAM platform would be designed with more redundancy and resilience to avoid flight termination to within [TBD] certification standards.

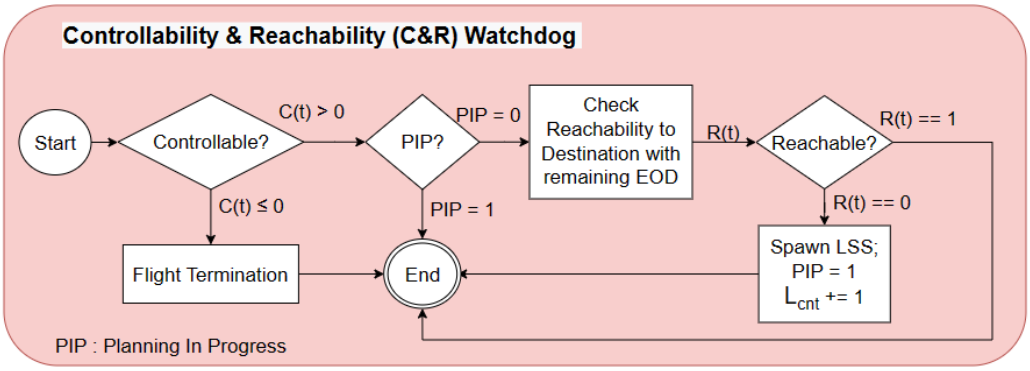


Figure 5.4: Controllability and Reachability (C&R) watchdog logic.

5.3.2.1 Controllability of a Small Multicopter UAS

A multicopter is a nonlinear dynamic system that relies on positive control inputs, specifically the unidirectional thrust generated by its rotors. Classical controllability theory for linear systems, as discussed in [108, 109], demonstrates that the Kalman rank test alone is insufficient to assess the controllability of a multicopter with bounded control inputs. To address this, positive controllability theory provides necessary and sufficient conditions for analyzing the controllability of systems with constrained inputs [110]. Du et al. [111] introduced the Available Control Authority Index (ACAI), and Saied et al. [112] proposed Small-time Local Controllability (STLC) with a positive controllability theorem. A method derived from [111] is utilized in our multicopter system and ACLM C&R thread, allowing real-time assessment of the controllability conditions in rotor failure cases. This is done by employing a proposed linearized state-space method.

The thread takes as input the fault/exception flag vector $\vec{F}(t)$, rotor health parameters $\vec{\eta}_r(t)$, and rotor thrust commands $\vec{f}_r(t)$ to calculate controllability $C(t)$ of the vehicle at any time t. The rotor health parameters $\vec{\eta}_r(t)$ form a vector of length equal to the number of motors. Each element of this vector ranges from 0 to 1, representing the health status of the corresponding rotor. In cases where the multicopter remains controllable despite a failure, the landing strategy selection (LSS) process is typically activated to define a safe contingency landing solution because reachability may be compromised due to the excess power required for the remaining functional motors to follow the original flight path to completion. In cases where the multicopter becomes uncontrollable, flight termination is

activated with parachute assist. Fig. 5.5 shows example octocopter configurations and their positive controllability results with up to two propulsion unit failures.

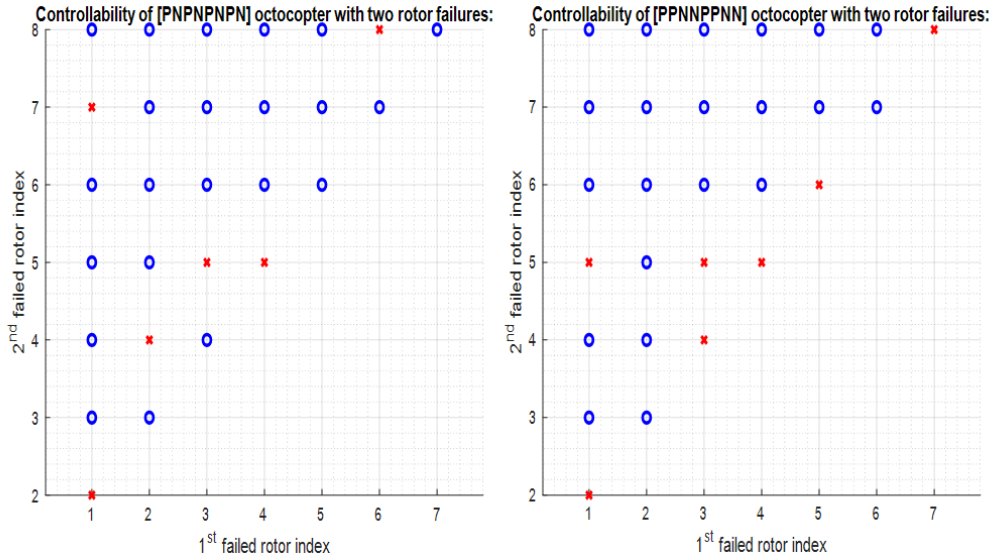


Figure 5.5: Controllability of configuration type "PNPNPNPN" (left) and "PPNNPPNN" (right) octocopters with two propulsion unit failures. A blue circle indicates a controllable system even though the indicated propulsion units have failed. A red cross indicates an uncontrollable system due to the two failed propulsion units.

5.3.2.2 Reachability of a Small Multicopter UAS

Landing site reachability $R(t)$ is assessed using an estimation of the Battery End of Discharge (EOD) time, the current state vector $\vec{x}(t)$, and the remaining distance to the destination. Algorithm 6 outlines the procedure for computing the reachability of landing sites within the ACLM solution. Battery EOD time is determined by considering the average current draw during the mission profile at each ACLM sampling rate. To ensure a conservative estimate, the average current draw is multiplied by a safety factor. Furthermore, the flight plan duration is calculated by subtracting a safety margin from the EOD time, providing an additional buffer period.

5.3.3 Landing Strategy Selector (LSS)

The landing strategy selector (LSS) thread is initiated when landing site reachability becomes false ($R(t) = 0$) and the aircraft remains under control ($C(t) = 1$). Fig. 5.6 shows the logic flow of LSS. The selection process takes as input the current state vector $\vec{x}(t)$, time

Algorithm 6: EOD-based Reachability

Input: EOD,t_{sf}, $\mathcal{V}$, $\mathcal{L}_{rem}$
Output: R_{EOD} Flag
// EOD = `Battery End of Discharge (s)`
// t_{sf} = `Safety Margin (s)`
// $\mathcal{V}$ = `vehicle speed (m/s)`
// $\mathcal{L}_{rem}$ = `Remaining flight time (s)`
$t_{rem} = \mathcal{L}_{rem}/\mathcal{V}$
if $(EOD \geq t_{rem} + t_{sf})$ **then**
  $R_{EOD} = 1$
else
  $R_{EOD} = 0$
end

until battery system End of Discharge (EOD) t_{EOD}, and segment number i. Utilizing the flight plan database described in Section 5.3.1, LSS first searches for an existing contingency flight plan in set P_i^* that leads to a suitable prepared landing site. Note that the value of $* \in \{mid, end\}$ is determined by whether $\vec{x}(t)$ is before or after the segment i midpoint, respectively.

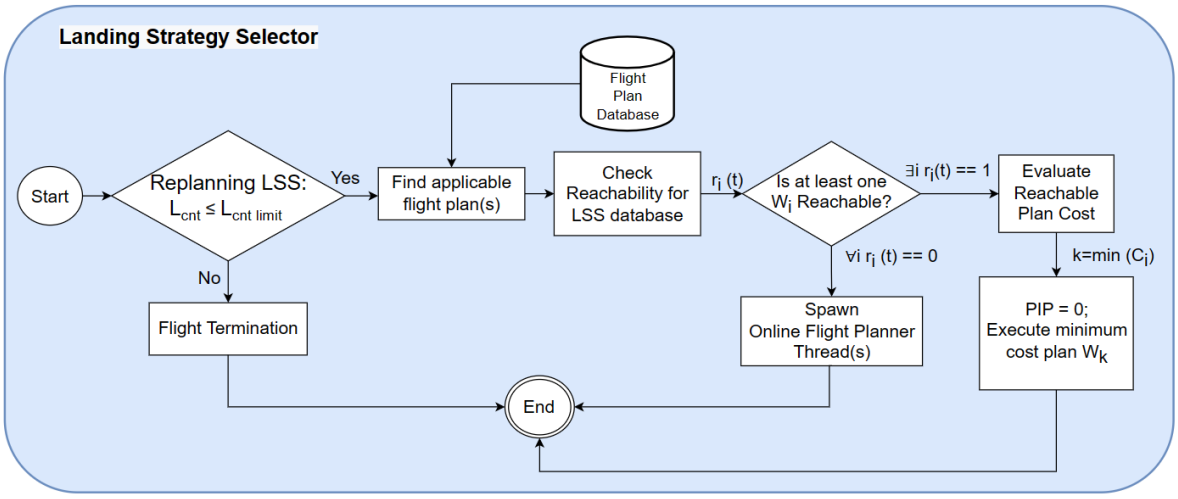


Figure 5.6: Landing strategy selector logic flow.

Each candidate landing plan in P_i^* is evaluated for reachability under degraded performance conditions, considering factors such as reduced battery EOD and reduced thrust due to propulsion unit failures and a potentially degraded battery condition. If there is at least one reachable flight plan serving as an emergency landing option, that plan can be executed. In cases where multiple flight plans satisfy the reachability criteria, the landing plan with

minimum cost $C^*_{i,j}$ is selected as defined in Equation 5.1.

Fig. 5.7 provides an example that demonstrates the process of generating contingency flight plans to prepared landing sites.

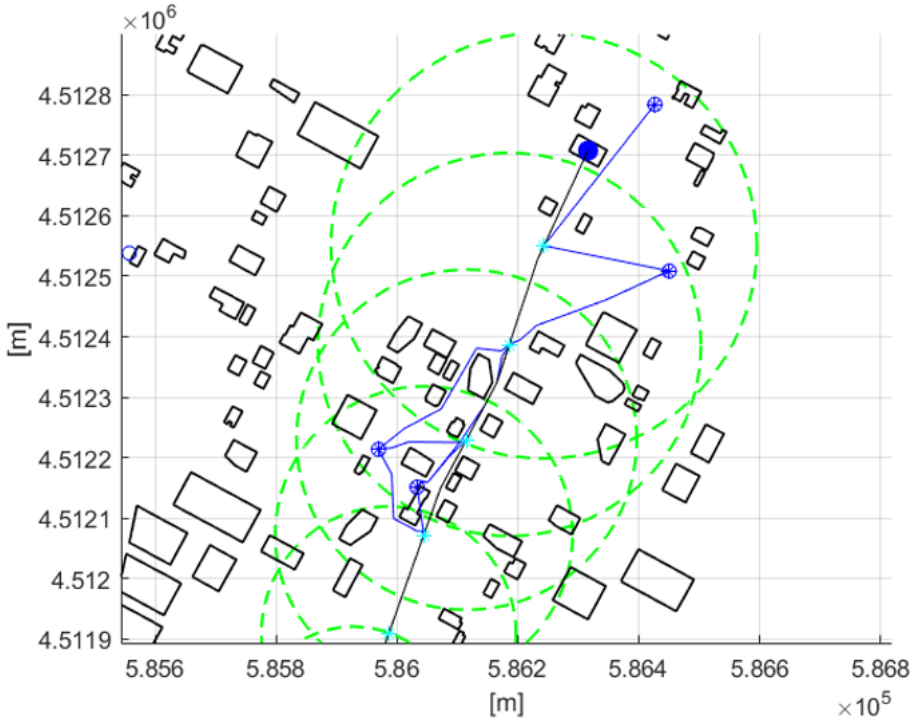


Figure 5.7: Visualization of LSS generating contingency flight plans to prepared landing sites. The approximate footprint from the offline flight planning database is represented by a green dotted circle. The reachable prepared landing sites are depicted as blue asterisk circles. The midpoint and endpoint of each flight trajectory are marked with cyan dots. The solid blue lines represent the generated contingency flight paths.

Per Section 5.3.2, when the number of replanning instances in the Landing Strategy Selection (LSS) process exceeds the threshold, flight termination is triggered. This occurs when the vehicle is unable to reach consecutive landing sites.

5.3.4 Continue/Hold Selector

Typically, the ACLM operates within milliseconds, allowing for quick retrieval and execution of pre-flight computed contingency landing plans. However, in the worst-case scenario where no prepared landing site is available within the aircraft's current footprint, real-time planning becomes necessary. Although the execution time of ACLM in such cases can still be fast, it may be nontrivial. To address this, a continue/hold selector thread runs in parallel to determine whether it is safer to hold/loiter or continue following the current flight plan while

generating a safe landing solution. This thread is activated within the online flight planner when the LSS thread triggers it. It calculates the anticipated future location where the online flight planner constructs a real-time contingency plan for moderate-risk landing sites. Typically, if the current overflight area has low population density and a minimal possibility of traffic conflicts, holding/loitering may be preferred. Fig. 5.8 shows Continue/Hold selector logic.

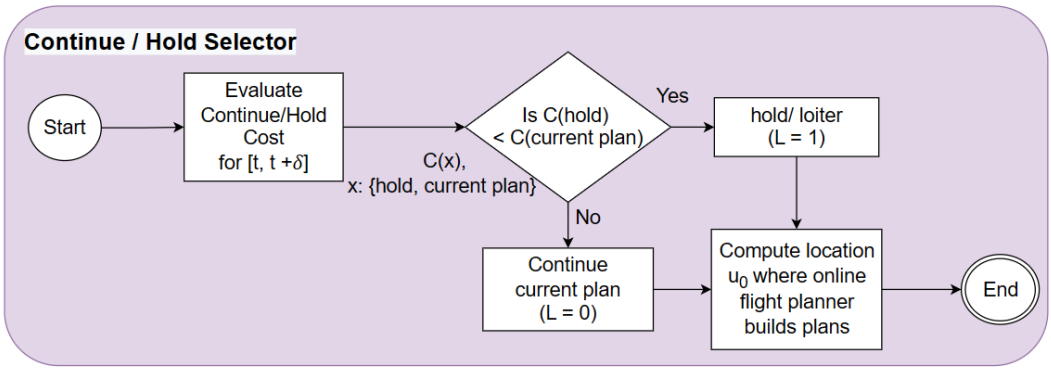


Figure 5.8: Continue/Hold selector logic flow.

In cases where the computation time for LSS exceeds the threshold, the costs of loitering versus continuing the current flight path are evaluated within a near-term time horizon of $[t, t+\delta]$. The decision of whether to continue or hold is based on the cost function defined as follows:

$$\begin{aligned} \mathcal{C}^i(\cdot) &= \sum \{\alpha_k * \mathcal{C}^i_k\} \\ &= \alpha_1 * \mathcal{C}^i_P(\cdot) + \alpha_2 * \mathcal{C}^i_{air}(\cdot) + \alpha_3 * \ \mathcal{C}^i_{\mathcal{L}^*_j}(\cdot), \quad \text{where} \ \ \mathcal{C}^i_{\mathcal{L}^*_j}(\cdot) = \frac{1}{|\mathcal{L}^*_j|} \end{aligned} \tag{5.2}$$

The superscript i in the above equation represents the options of holding/loitering ($i = 1$) or continuing ($i = 2$). The weights α_i are the coefficients of the cost function, and they are typically chosen such that their sum equals 1. The variables $\mathcal{C}^i_P$ and $\mathcal{C}^i_{air}$ represent the population density and air traffic risk (i.e., airspace density) respectively, at either hold/loiter area or the current flight path within the near-term horizon. It is assumed that values for $\mathcal{C}^i_P$ and $\mathcal{C}^i_{air}$ are preloaded for the nominal flight region before the flight begins. The variable $\mathcal{C}_{\mathcal{L}^*_j}$ represents the proximity to prepared landing sites, which is defined as the inverse of the total number of prepared landing sites $|\mathcal{L}^*_j|$ located on nominal flight plan segment j near or further along the future flight plan.

5.3.5 Online Flight Planner

If the LSS does not contain prepared landing sites that are reachable based on current T_{EOD} and for which preflight plans are cached for retrieval, it is necessary to generate a landing plan to a higher-risk unprepared landing site in real time. Despite the higher costs associated with these higher-risk landing sites in terms of terrain and property, landing under control by following a feasible collision-free path to a designated landing site is typically safer than flight termination. The online flight planner is initiated with a reachable footprint defined by the updated values of $[t_{EOD}, \vec{x}(t)]$. Fig. 5.9 shows the logic flow of the online flight planner. The process of rapidly computing a reachable approximate footprint is described below.

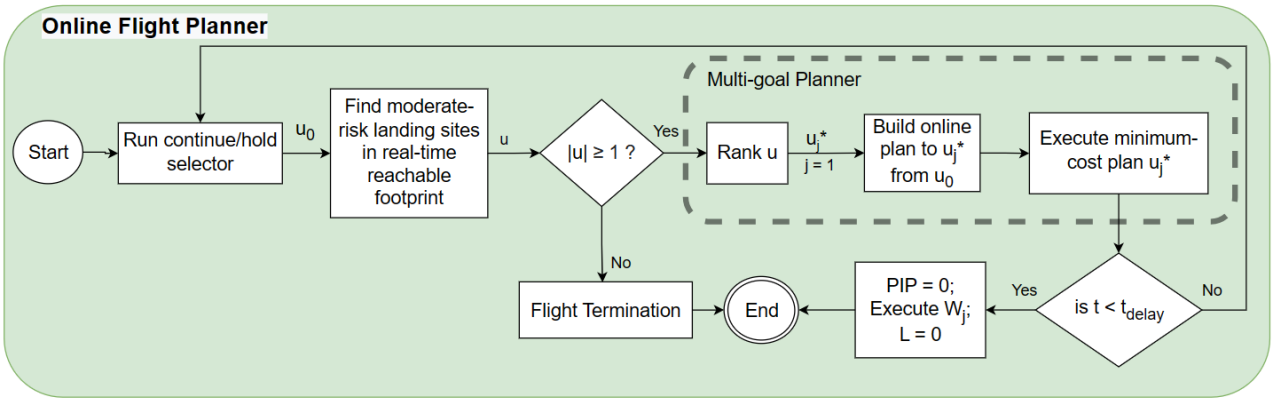


Figure 5.9: Online flight planner logic flow.

5.3.5.1 Reachable Footprint Computation for an Energy-constrained Multicopter

A real-time analytical approach for estimating the reachable footprint of a multicopter is calculated based on battery t_{EOD} at constant cruise speed. To achieve this, we adopted a conservative approximation by considering longitudinal level flight for the multicopter as depicted in Fig. 5.10.

In this approximation, the equations of motion are defined as follows, where $u = -T\sin\theta$ with T representing thrust (N) and θ denoting the pitch angle in radians:

$$\begin{aligned} m\ddot{x} &= u - d_x\dot{x} \\ mg &= T\cos\theta \end{aligned} \tag{5.3}$$

The state space representation of equation (5.3), incorporating the forward position ($x_1 =$

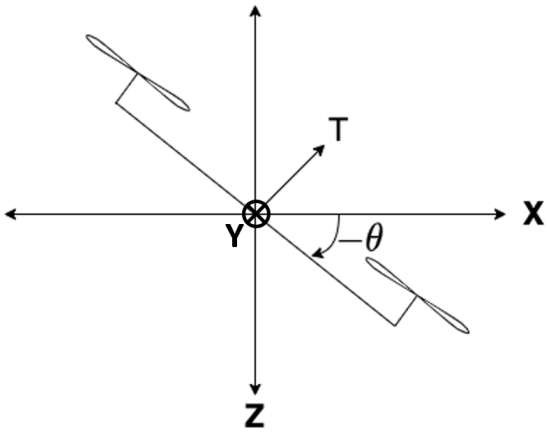


Figure 5.10: Multicopter in 1-D motion

x) and velocity ($x_2 = \dot{x}_1$), is provided below. Additionally, the cost functional determining the minimum control effort for projecting the footprint over a specified time interval ($t_0 = 0$, $t_f = T_{EOD}$), is expressed as follows:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}u - \frac{d_x}{m}x_2 \end{aligned} \tag{5.4}$$

$$J = \frac{1}{2}\int_{t_0}^{t_f} u^2 \, dt \tag{5.5}$$

Pontryagin's Minimum Principle (PMP) [113] determines the optimal control input u^*. By considering simplified longitudinal, level multicopter flight dynamics, an analytical solution can be obtained using the PMP, with computations typically taking milliseconds and requiring only modest computing resources. The Hamiltonian associated with this problem is defined as follows:

$$H = \frac{1}{2}u^2 + [\lambda_1 \ \lambda_2] \begin{bmatrix} x_2 \\ \frac{1}{m}u - \frac{d_x}{m}x_2 \end{bmatrix} \tag{5.6}$$

The obtained solution for the optimal states (x_1^*, x_2^*), co-states (λ_1^*, λ_2^*), and controls (u^*) can be expressed by the following set of equations:

$$x_1^*(t) = -\frac{C_1}{d_x^2}t + \frac{C_2}{2d_x^2}\exp\left(\frac{d_x}{m}t\right) - \frac{C_3}{d_x}m\exp\left(\frac{-d_x}{m}t\right) + C_4 \tag{5.7}$$

$$x_2^*(t) = -\frac{C_1}{d_x^2} + \frac{C_2}{2d_x m}\exp\left(\frac{d_x}{m}t\right) - C_3\exp\left(\frac{-d_x}{m}t\right) \tag{5.8}$$

$$u^*(t) = -\frac{C_1}{d_x} + \frac{C_2}{m}\exp\left(\frac{d_x}{m}t\right) \tag{5.9}$$

$$\lambda_1^*(t) = C_1 \tag{5.10}$$

$$\lambda_2^*(t) = \frac{mC_1}{d_x} - C_2\exp\left(\frac{d_x}{m}t\right) \tag{5.11}$$

To determine the maximum distance covered by the hexacopter in both the forward and backward directions, a problem is formulated with a free final state and a fixed final time to determine constants C_1, C_2, C_3, C_4. Then, the following boundary conditions are established to define the maximum distance traveled in the forward direction, and the maximum distance traveled backward given an initial forward velocity:

$$x_1(t_0) = x_0,\ x_2(t_0) = v_{init},\ x_2(t_f) = v_{init},\ \lambda_2(t_f) = 0 \tag{5.12}$$

$$x_1(t_0) = x_0,\ x_2(t_0) = v_{init},\ x_2(t_f) = -v_{init},\ \lambda_2(t_f) = 0 \tag{5.13}$$

By maintaining equal initial and final velocity magnitudes, a constant velocity profile is represented in the forward direction, while a constant steady-state velocity is depicted in the backward direction. This leads to a nearly constant current draw from the battery, causing the battery t_{EOD} to decrease consistently until it reaches the final position for level flight. Solving the system of linear equations with the given boundary conditions allows us to determine the values of C_i. In the implementation, the exponential function is replaced by an nth-order MacLauren Series expansion, where for the current approach, we utilize an order of $n = 5$. Once the optimal control input $u^*(t)$ is calculated for steady-level flight, the values of T and θ are derived using following equations:

$$\theta_{opt}(t) = atan^{-1}\left(\frac{-u(t)}{mg}\right) \tag{5.14}$$

$$T_{opt}(t) = \frac{mg}{cos(\theta_{opt}(t))} \tag{5.15}$$

An estimation of the online reachable footprint is then computed based on the updated

real-time prediction of t_{EOD} as shown in Algorithm 7. The real-time reachable moderate-risk landing sites are found using point-in-polygon operation [54] under the footprint boundary. Fig. 5.11-5.13 illustrate the online reachable footprint concept with example footprints given a multicopter cruise speed of $7m/s$ at different t_{EOD}. The figure illustrates that as the EOD values increase, the multicopter possesses enough duration to decelerate and move in the backward direction. Conversely, lower t_{EOD} values limit the distance that can be traveled backward.

Algorithm 7: Online Footprint Approximation

Input: v_{init},EOD,T_{Max}
Output: Footprint Limits
// v_{init} = `Current Velocity (x,y)`
// EOD = `Battery End of Discharge (s)`
// T_{Max} = `Maximum Total Thrust (N)`
1 Assign the boundary conditions as per Eq. (5.12)-(5.13)
2 Calculate C_i values for forward and backward motion
3 Determine u^*_{fwd},u^*_{bkwd} using eq. (5.9)
4 Determine [$\theta_{opt-fwd}$, $T_{opt-fwd}$] and [$\theta_{opt-bkwd}$, $T_{opt-bkwd}$] using eq. (5.15)-(5.14)
5 Bound the [θ_{opt} , T_{opt}] with $\theta_{min/max}$ obtained from available T_{Max} for level flight
6 Determine u_{bound} using eq. (5.3) and bounded values of θ_{bound},T_{bound}
7 Simulate system given by eq. (5.4) until t_{EOD} using u_{bound}
8 Determine footprint limits based on the position at end of simulation

5.3.5.2 Real-time Multi-Goal Flight Planning

The online flight planner in ACLM incorporates a **multi-goal planning algorithm** [107] to calculate plans to reachable unprepared (higher-risk) landing sites in real-time when no prepared landing sites are accessible through LSS. This algorithm utilizes a three-dimensional rasterized map, created from airborne point cloud data known as Digital Surface Maps. Each voxel cell in the map contains occupancy and risk information, allowing for the construction of a comprehensive 3D environment representation. This occupancy map is then utilized to determine an optimal collision-free trajectory towards the minimum-cost moderate-risk landing site within the online reachable footprint. The A* path planning algorithm was employed to generating the online contingency flight planning solution. The cost of a flight plan to a reachable unprepared landing site i is defined using Equation 5.16 with terms analogous to those used in preflight planning (Eq. 5.1):

$$\mathcal{C}_i = \beta \cdot d_{travel} + \gamma \cdot \mathcal{R} \tag{5.16}$$

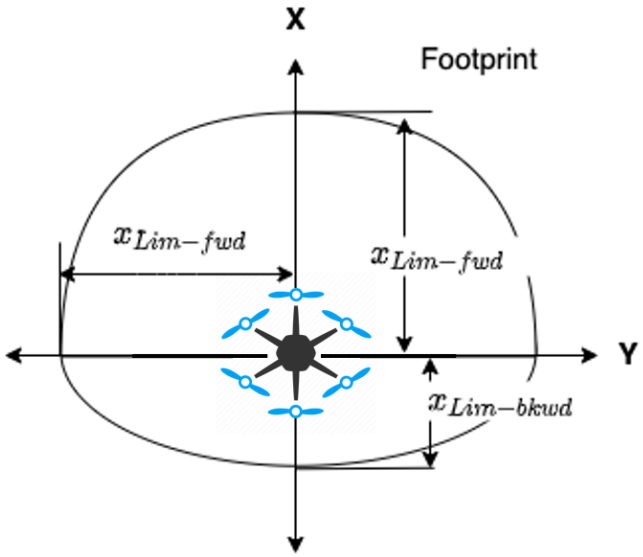


Figure 5.11: Reachable footprint of a multicopter based on level-flight forward/backward range at constant cruise speed.

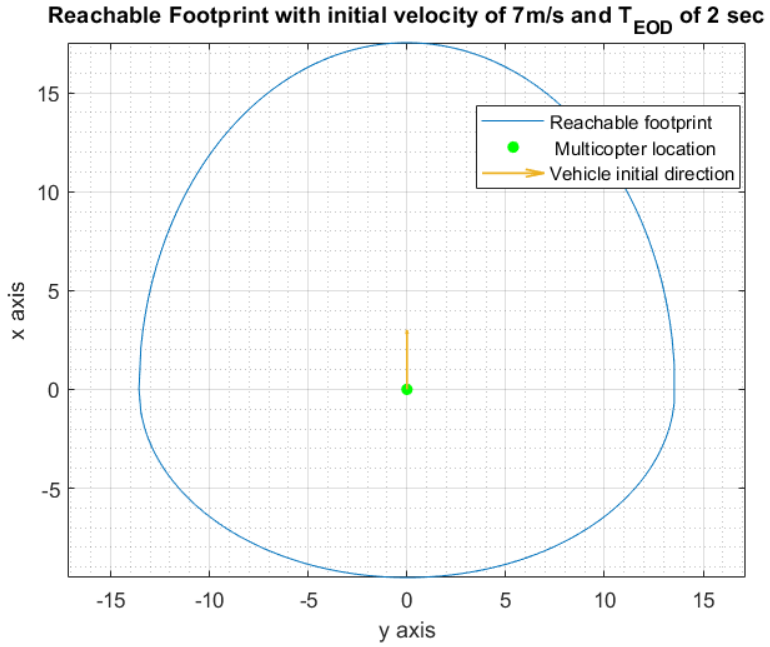


Figure 5.12: Reachable footprint for a multicopter, initially moving in the x-axis direction with a velocity of 7m/s and a time until battery End of Discharge (T_{EOD}) of 2 seconds.

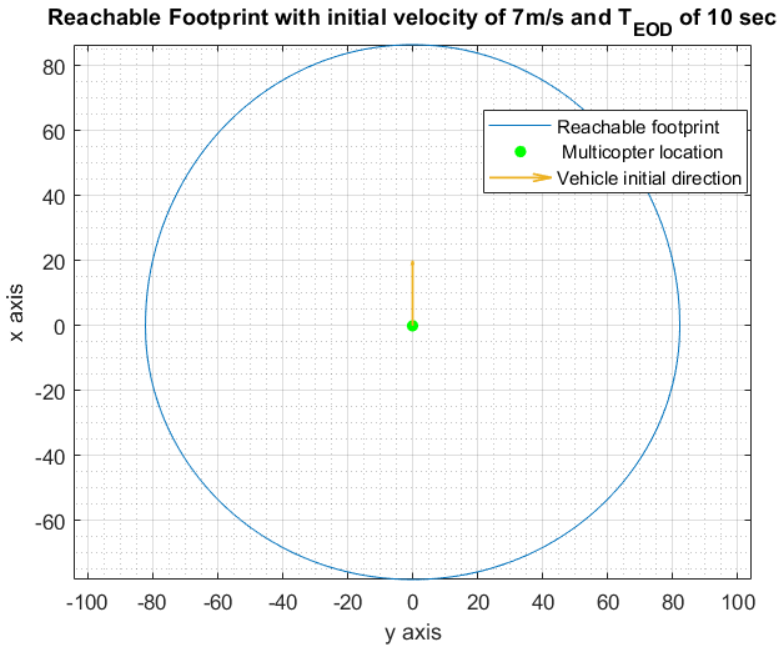


Figure 5.13: Reachable footprint for a multicopter, initially moving in the x-axis direction with a velocity of 7m/s and a time until battery End of Discharge (T_{EOD}) of 10 seconds.

5.3.6 Flight Termination

The activation of the flight termination thread can occur within various components such as the C&R watchdog, LSS, or online flight planner, as illustrated in each respective thread. Fig. 5.14 shows the flow chart of the flight termination thread. Within this thread, the attitude angles (pitch and roll) and their angular rates (pitch rate and roll rate) are continuously provided to the ACLM at its designated sampling rate. The thread's purpose is to identify instances where the vehicle system is not tumbling or when the tumbling rate is sufficiently slow to initiate parachute ejection and system shutdown as shown in the Algorithm 8.

5.3.7 Assurance for individual ACLM sub-components

We ensure the reliability and safety of the entire ACLM system by establishing assurance for each of its sub-components. For **offline flight planning**, assurance is established through offline validation of the prepared landing site database and by validation of each contingency landing plan with respect to constraints on t_{EOD} and flight performance/controllability constraints. Because preflight contingency plans need not be generated or checked quickly, they can be validated mathematically and in simulation.

For the **C&R watchdog**, assurance is derived from control theory. We apply positive controllability theory (i.e., for multicopter unidirectional thrust configurations). Reachability

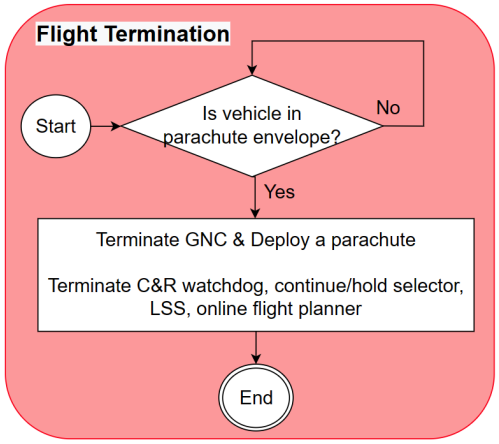


Figure 5.14: Flight termination logic flow.

is assessed based on energy constraints, and optimal control solutions are obtained through the application of Pontryagin's minimum principle. For the **landing strategy selector**, assurance is made through the retrieval and execution of landing site plans computed from validated map and landing site data and with plans whose flight performance and obstacle clearance constraints can be validated in simulation before flight. In the **continue/hold selector**, the decision of whether to proceed with the current flight plan or initiate a hold/loiter pattern while an online ACLM solution is established is determined by evaluating the solution generation time against a predefined threshold. This is an "informed judgement call" that neither guarantees nor compromises safety of flight. Finally, for the **online flight planner**, assurance is achieved as follows First, the online flight planner uses an unprepared landing site database that is validated but higher risk in that the site is not protected from occupancy by people, objects, etc. Second, because contingency landing plans are short in distance and time, search-based planning time can be bounded in depth thus time. These layers of assurance collectively form a foundation upon which ACLM operates, ensuring the safety and reliability of its contingency management processes.

While it is impossible to provide an absolute guarantee of flight termination (FT) avoidance, ACLM has been designed to make FT the final resort. This is achieved by systematically minimizing the likelihood of executing FT through a series of ACLM execution stages:

- **ACLM Required**: For most flights, ACLM will not be needed. In this case, thenominal flight will proceed to the original destination.
- **Assured Offline Solution**: If contingency response is needed, the first step is to employ an assured offline solution. This solution isgenerated in advance, ensuring its reliability and feasibility. Further, collective use of the C&R watchdog and plan

Algorithm 8: Flight Termination Condition

Input: $\phi, \theta, \dot{\phi}, \dot{\theta}$
Output: R_{FT}
// $\phi =$ `roll angle (rad)`
// $\theta =$ `pitch angle (rad)`
// $\dot{\phi} =$ `roll rate (rad/s)`
// $\dot{\theta} =$ `pitch rate (rad/s)`
// $R_{FT} =$ `Flight Termination Flag`
if *(*$\phi \in [-0.7854, 0.7854]$ & $\theta \in [-0.7854, 0.7854]$*)* **then**
  if *(*$\dot{\phi} \in [-0.2, 0.2]$ & $\dot{\theta} \in [-0.2, 0.2]$*)* **then**
1 $R_{FT} = 1$
  end
else
2 $R_{FT} = 0$
end

database minimizes the delay between the triggering event and contingency landing plan execution.

- **Assured Online Solution**: In cases where no viable offline solution is available, ACLM generates an online solution that is assured to meet current controllability and reachability constraints. While landing at a mapped but unprepared site carries higher risk than landing at a prepared site, this procedure carries lower risk than trying to discover a landing site in real-time or continuing to a prepared site outside the reachable footprint, which would exceed t_{EOD} thus result in FT due to motor shutdown.

- **Flight Termination (FT) as a Last Resort**: Only when all previous steps fail to provide a feasible contingency plan and no other safe alternatives exist, flight termination (FT) is executed as a last resort. Because the C&R Watchdog initiates FT as soon as controllability is lost, the UAS is unlikely to have tumbled into an adverse attitude state, so probability of safe parachute deployment is maximized.

5.4 Simulation Setup

Package delivery will require UAS to operate in a low-altitude urban airspace as shown in Fig. 5.15.

In this, work, ACLM was implemented into the hexacopter modeled in [89], and missions were simulated in the urban environment of Manhattan, New York City. Each simulated package delivery flight involved rotor failure, battery degradation, or a combination of both,

Figure 5.15: Multicopter use case for urban package delivery. The top left image shows Matternet delivering a medical item. The top right image shows Amazon Prime Air, and the bottom image depicts Geopost (formerly DPDgroup) for parcel delivery.

requiring the ACLM algorithm to ensure the safety of the distressed vehicle using its decision-making algorithms. To test more realistic scenarios, simulations were conducted using both lighter-weight and heavier-weight package-delivering hexacopters, where the limited maximum thrust of the vehicles resulted in different outcomes for the ACLM decision-making process.

During the simulation of ACLM, several **key assumptions** are assumed: First, it is assumed that contingency management plans need not be deconflicted with other air traffic. This condition is reasonable as emergency landings receive priority handling. Second, we assume an accurate and precise map marking prepared and unprepared landing sites is available for use. This map forms the basis for all navigation-related calculations and decision-making. Finally, the simulation assumes the presence of accurate and effective Fault Detection and Identification (FDI) mechanisms. These play a critical role in the activation and execution of ACLM solutions, effectively identifying and responding to vehicle system anomalies.

The below subsections describe environment modeling and setup for our Monte Carlo simulations.

5.4.1 Environment Modeling

A 3-D map of Manhattan City, New York City was created using OpenStreetMap (OSM) data. OSM is a collaborative global mapping project that creates geographical data [65].

It provides regularly updated map entities such as roads, buildings, airways, and more, along with specific details like centroid and area data. The raw OSM data includes building coordinates represented as polygon vertices, and height using the WGS 84 / UTM zone 18N coordinate system [67]. To convert the unit of measurement to meters in the local coordinate system, we applied a projected coordinate system (EPSG: 26918). Subsequently, geofence map processing algorithms in [3] were applied within the region of interest (ROI) to generate static keep-out geofence layers around each building. This ensures a collision-free path for the UAS during both nominal flight planning and contingency flight planning.

Because OSM does not map UAS landing sites, we generated a database of potential landing site from multiple data sources. We utilized a feature-rich database from [4] to create two distinct categories of landing sites: **prepared landing sites** for offline flight planning and **moderate-risk or unprepared landing sites** for online flight planning. The landing site database was constructed by combining data from multiple sources, including Open Street Map (OSM), LIDAR data, and terrain height data. This comprehensive approach ensures that the landing site database incorporates diverse and relevant data for accurate and effective contingency flight planning. Fig. 5.16 shows the data processing of landing sites for offline and online flight planning.

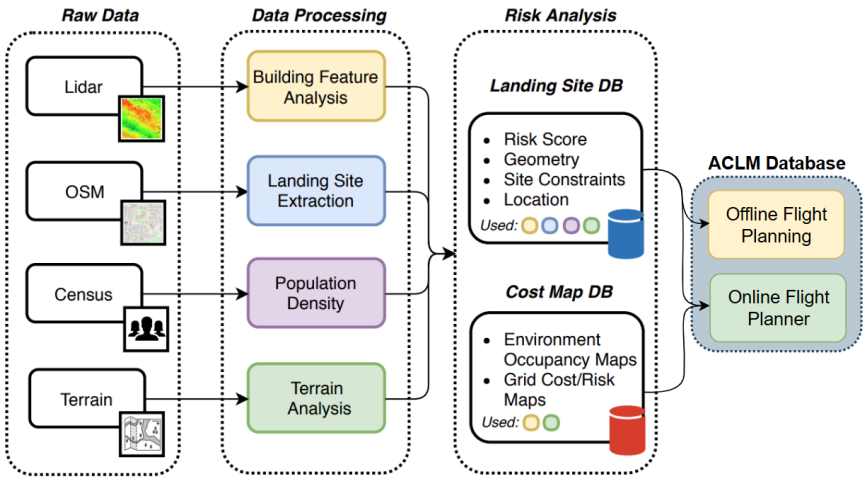


Figure 5.16: Pre-processing of data [4] to generate prepared/ unprepared landing site offline and online flight planning database.

The risk level of each landing site is assigned a normalized value between 0 and 1 to categorize the associated risk. The equation used to calculate the landing site risk is presented below, where w_t, w_p, w_s are weight factors, and $\mathcal{C}_t$, $\mathcal{C}_p$ and $\mathcal{C}_s$ are terrain cost, population cost, and property cost, respectively, as defined in [4]:

$$\mathcal{R} = w_t \cdot \mathcal{C}_t + w_p \cdot \mathcal{C}_p + w_s \cdot \mathcal{C}_s \tag{5.17}$$

For our work, the **prepared (i.e., low-risk) landing sites** ($\mathcal{L}_{low}$) and **unprepared (i.e., moderate-risk) landing sites** ($\mathcal{L}_{moderate}$) are further categorized as follows, where $h_{\mathcal{L}_i}$ is the height of the landing site:

$$\mathcal{L}_{low} = \{\mathcal{L}_i \mid \mathcal{R} \geq 0.6 \ \& \ h_{ls} < 50m\} \tag{5.18}$$

$$\mathcal{L}_{moderate} = \{\mathcal{L}_i \mid \mathcal{R} < 0.6 \ \& \ h_{ls} < 50m\} \tag{5.19}$$

Fig. 5.17 shows the ROI where the Monte Carlo simulation was conducted. The map shows the prepared landing sites and unprepared moderate-risk landing sites in the ROI. The total number of available landing sites in the ROI is shown in Table 5.1.

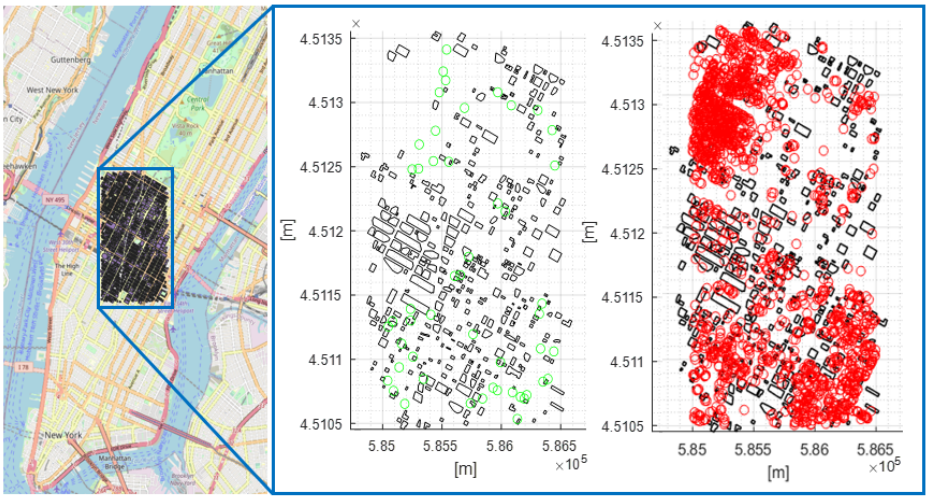


Figure 5.17: Locations of prepared landing sites and moderate-risk landing sites visualized in a region of interest (ROI) in Manhattan, New York City. On the left side, prepared landing sites are indicated by green circles, while on the right side, moderate-risk landing sites are represented by red circles. To ensure clarity, only buildings with a height greater than 60m are displayed as solid black polygons.

Table 5.1: Total number of available landing sites

Prepared landing sites	Unprepared landing sites
50	1581

After categorizing the landing sites based on their normalized risk, the offline flight planning database for each prepared landing site was generated. This was achieved by applying the Visibility Graph algorithm [57] to determine the minimal travel distance from the mid and endpoint of each flight segment to each landing site on the vectorized map. This approach ensures a collision-free path to the destination without any distortion or approximation. Subsequently, a prepared landing site database was created and ranked for the mid and endpoint of each flight segment using Equation 5.1 in Section 5.3.1. In a similar manner, a moderate-risk unprepared landing site database was created and ranked in real-time using Equation 5.16. Table 5.2 below summarizes the weight factor parameters used in the Monte Carlo simulation.

Table 5.2: Weight factor parameters for landing sites

w_t	w_p	w_s	β_l	γ_l	β_m	γ_m
0.5	0.25	0.25	0.6	0.4	0.5	0.5

5.4.2 Vehicle and ACLM System Modeling

The hexacopter model for vehicle simulation was constructed using the physical parameters provided in [90]. Fig. 5.18 shows the hexacopter configuration, and Table 5.3 provides a summary of the key parameters used in designing the simulation model. The parameter L represents the arm length of the hexacopter, and C_q and C_t correspond to the moment coefficient and thrust coefficient of the motor, respectively. The inertia matrix elements are denoted as I_{xx}, I_{yy}, and I_{zz}. Additionally, Table 5.4 presents the mass of the hexacopter, which carries lighter-weight and heavier-weight packages.

Table 5.3: Parameters for hexacopter models [SI unit]

L	C_q	C_t	T_{max}	I_{xx}	I_{yy}	I_{zz}
0.225	$2.08e^{-10}$	$1.57e^{-8}$	27	0.0169	0.0132	0.0327

Table 5.4: Mass of package-carrying hexacopter models [kg]

lighter-weight hexacopter	heavier-weight hexacopter
1.2	1.837

The feedback controller employed a Linear Quadratic Regulator (LQR) design tuned for both no-rotor failure and single-rotor failure cases. In the event of a single-rotor failure, the system

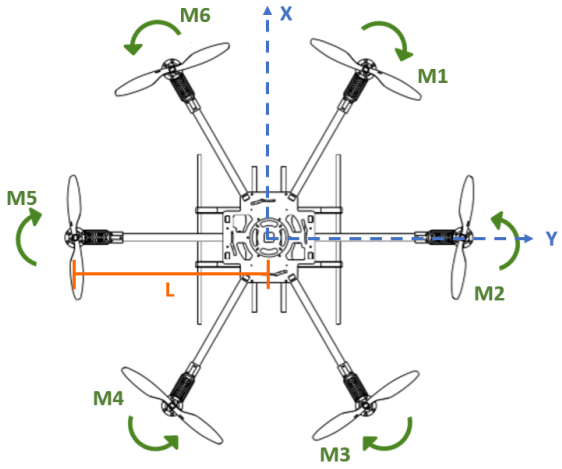


Figure 5.18: Hexacopter model configuration used in the simulation.

was reconfigured into a quadcopter configuration by deactivating the opposite rotor, relying on the remaining four rotors to control the vehicle. A quadcopter degraded configuration made the heavier-weight hexacopterdescend at a constant rate because it cannot generate sufficient thrust to balance weight. Thisled to a significantly contracted reachable footprint. Vehicle controller execution rate was faster than ACLM watchdog execution rate to assure good controller performance as shown in Table 5.5.

Table 5.5: Sampling rate of controller and ACLM [Hz]

LQR controller	ACLM
200	1

To determine the reachable footprint, t_{EOD} was calculated using the corresponding motor angular speeds, average current draws, and initial battery capacity (amp-hours). Our simulations used the experimentally determined curve-fitted equation from [90] that relates the current draw and varying motor speed (rad/s) for this specific hexacopter model to calculate the current draw. Then, the average current draws of both the lighter-weight and heavier-weight hexacopter models were determined by considering the constant vehicle speed across multiple simulations. Subsequently, the time until the battery end of discharge was then calculated using the following equation based on [114]:

$$I_{avg} = mean(\sum_{i=1}^{6} 3.0282e^{-9} \cdot \omega_i^3) \; per \; flight \tag{5.20}$$

$$T_{EOD} = \mathcal{C}_{battery}/I_{avg} * 3600 \tag{5.21}$$

where ω is the individual motor speed, and $\mathcal{C}_{battery}$ is the battery capacity.

Table 5.6 provides a summary of the average current draw under nominal operating conditions, with a safety factor of 10% applied, as well as the average current draw under rotor failure conditions, with a safety factor of 20% applied. Throughout the simulation, hexacopter models carried two battery packs that have just enough combined battery capacity to complete each nominal flight trajectory. This design ensures that in the event of rotor failure or battery degradation, it is likely that the vehicle will be unable to reach its original destination, consequently resulting in ACLM's C&R watchdog triggering LSS for a contingency response.

Table 5.6: The average current consumption and altitude deviation of the hexacopter models under normal operating conditions and rotor failure scenarios.

	Hexacopter	
	lighter-weight	heavier-weight
Avg. current draw [A] (normal operating)	32	60.9
Avg. current draw [A] (with rotor-failure)	36	81.4
Altitude lost [m/s] (after rotor-failure)	-	0.5

The hexacopter model and ACLM were integrated with each other seamlessly using Simulink and Stateflow. Fig. 5.19 shows the Simulink and Stateflow models with inputs and outputs to each other. Particularly, designing ACLM using Stateflow allowed for the implementation of parallel threads enabling synchronization between states. This synchronization ensured that events in one thread triggered actions in another thread as illustrated in the ACLM logic flows in Section 5.3.

5.4.3 Monte Carlo Parameter Setup

Monte Carlo simulation parameters were configured to analyze the ACLM outputs with a package-carrying hexacopter designed in Section 5.4.2. The parameters used in the simulation are shown in Table 5.7.

In the simulation, the hexacopter initiates the mission by performing vertical takeoff and landing maneuvers. Furthermore, it is assumed that there is no wind and there are no state estimation errors or uncertainties.

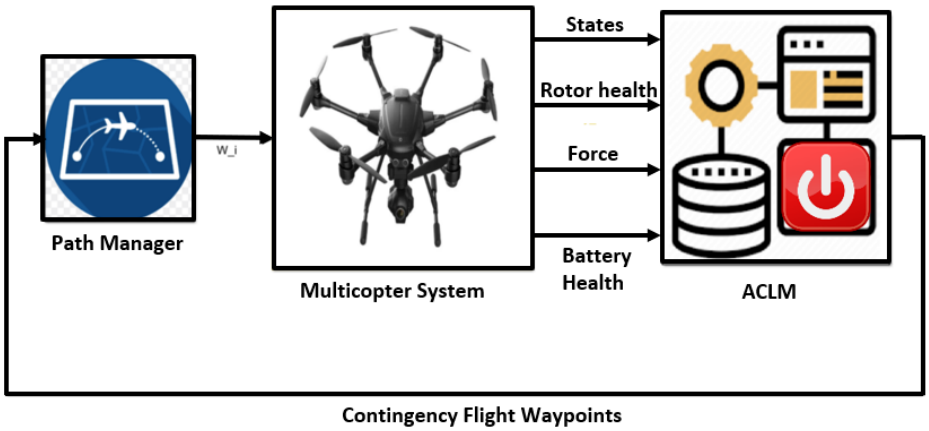


Figure 5.19: Hexacopter model and ACLM integration using Simulink and Stateflow. Icon images are adapted and modified from [5].

Table 5.7: Monte Carlo Simulation Parameters

Parameter	**Description/Constraints**
Start and Destination Points	Random; Flight distance: 300 - 1200 m AGL
Rotor Failure Time	Random
Battery Degradation Time	Random
Single Battery Degradation Percentage	No degradation, 10%, 15%, 50%
Failed Rotor Index	Random (1-6)

5.5 Simulation Results

The Monte Carlo simulations were conducted using MATLAB 2022b on a computer equipped with an AMD Ryzen 9 5900HX mobile processor (8-core) and an NVIDIA GeForce RTX 3060 graphics card, running 64-bit Windows 10. Simulink codes were executed in parallel, utilizing all 8 cores, and the runtime of each thread was recorded for each simulation. A total of 10,398 Monte Carlo simulations were run for lighter-weight and heavier-weight package-delivering hexacopter, respectively. During the simulation, a random selection of four different battery degradation states, as described in Section 5.4.3, was incorporated as part of the Monte Carlo parameters. The frequency of occurrence for each degradation percentage is presented in Fig. 5.20.

Three scenarios were considered for both hexacopter models: 1) a single-rotor failure, 2) a single-battery degradation, and 3) simultaneous single-rotor failure and battery degradation. These scenarios occurred randomly at different starting points, destinations, and times throughout the Monte Carlo simulation. A total of 3,466 simulations were conducted for each scenario case, resulting in a cumulative total of 10,398 simulations for each hexacopter

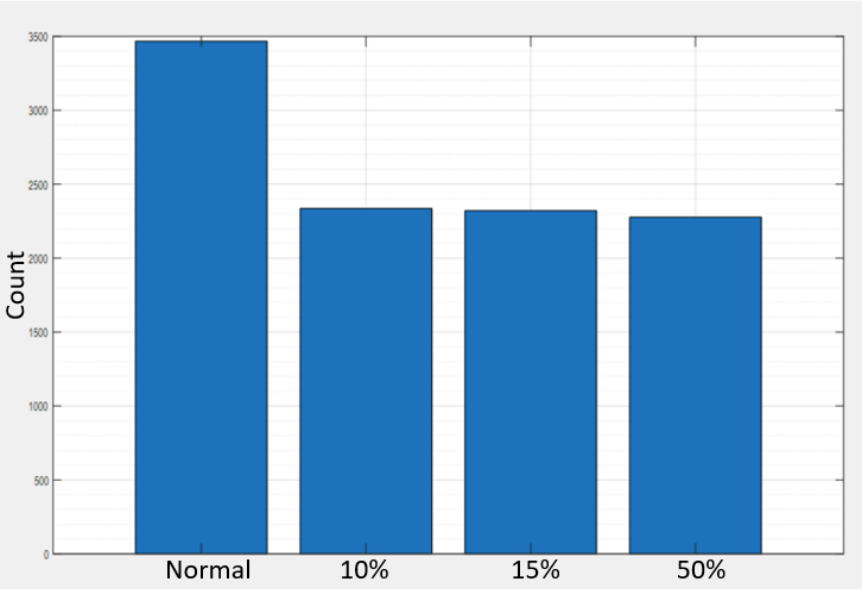


Figure 5.20: Monte Carlo simulation of in-flight battery anomalies for various degradation cases. During each flight, one of the following scenarios was simulated: no battery degradation, battery capacity degradation of 10%, 15%, or 50% in one of the two battery packs.

model. The visualization of the package-delivering AAM flying in Manhattan, New York City during the simulation is shown in Fig. 5.21. Additionally, Fig. 5.22-5.23 display the visualization of the offline contingency flight path generated using LSS and the online flight planner solution, respectively.

The output results of the simulation were analyzed to derive statistics on key features for both the lighter-weight and heavier-weight hexacopter models, as well as to monitor the time-varying system states in the ACLM. An example of ACLM outputs in a specific scenario is presented in Fig. 5.24. In this scenario, the lighter-weight package-delivering hexacopter experiences a 15% battery degradation in one of its batteries at 142 seconds. As a result, the original destination cannot be reached due to the reduced end of battery discharge time. ACLM then searches for and identifies a prepared landing site with the minimum cost to reach and modifies the flight path at the nearest flight segment. Subsequently, at 413 seconds, the vehicle encounters two adjacent rotor failures, leading to a loss of control. This ultimately triggers the flight termination of the vehicle system and the ACLM, with the deployment of a parachute.

Flight termination can also occur when there are no reachable prepared landing sites or unprepared landing sites identified through the LSS and online flight planner after the vehicle experiences an in-flight anomaly. Table 5.8 presents the flight termination rates for both the lighter-weight and heavier-weight hexacopter models when they are unable to find reachable

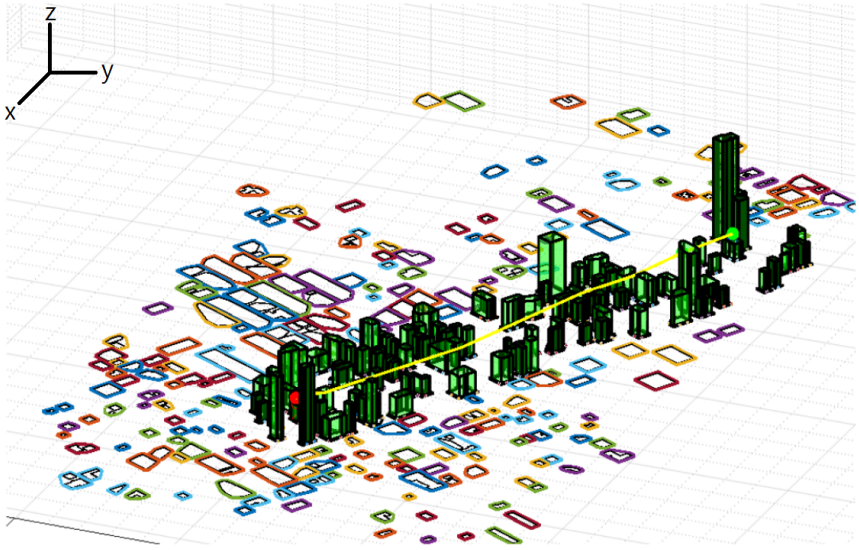


Figure 5.21: Visualization of the nominal flight plan and simulated flight trajectory. The yellow line indicates the nominal flight path, while geofenced buildings are constructed around the nominal flight trajectory (shown in green).

landing sites in the database. In either case of flight termination, when such termination becomes necessary, ACLM ensures the safety of the vehicle by verifying its position and triggering the proper deployment of the parachute as shown in Fig. 5.14. Despite the relatively low occurrence of flight termination events, successful parachute deployment for a small UAS minimizes kinetic energy on impact.

Table 5.8: Flight Termination (FT) rate of hexacopter models

Hexacopter	lighter-weight	heavier-weight
FT rate (%)	2.95	4.56

5.5.1 Lightweight Package-carrying Hexacopter

A lighter-weight package-delivering hexacopter is equipped with a maximum thrust capability of 27 N, as indicated in Table 5.3. This level of thrust is more than sufficient to effectively guide, control, and navigate the hexacopter according to the mission plan, even in the event of a single rotor failure that results in a quadcopter configuration. Specifically, for the lighter-weight hexacopter with a mass of 1.2 kg, the four remaining rotors can generate 18 N of force, providing an additional available thrust of approximately 6 N beyond what is required for maintaining a hover position. Consequently, when encountering rotor failure

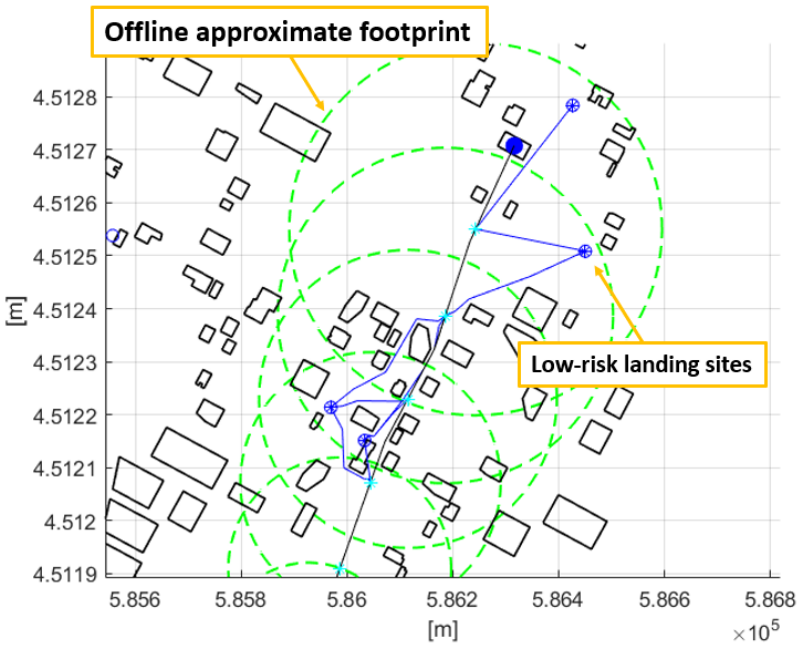


Figure 5.22: Visualization of the offline contingency flight path generated using Visibility Graph with polygon decimation. The green dashed circles represent the approximate footprint, while the cyan asterisk indicates the mid and endpoint of each flight segment. The blue lines depict the stored flight path to prepared landing sites prior to the flight. For better visualization, only buildings with a height exceeding 60m are displayed as solid black polygons.

or a combination of rotor failure and battery degradation, the primary influencing factor for identifying reachable landing sites through LSS or the online flight planner is the T_{EOD}.

In each scenario case, a total of 3,466 simulations were conducted, and the frequency of different ACLM states/outputs was collected throughout the simulation. Fig. 5.25 illustrates the frequency of ACLM states/outputs specifically for **scenario case 1**. Here, "$R_t == 0$" indicates cases where the vehicle cannot reach the final destination, necessitating the use of ACLM to search for offline or online flight plans. "$R_{LSS} == 1$" represents cases where ACLM finds a reachable offline flight plan. "$R_{online\ FP} == 1$" represents cases where ACLM finds a reachable online flight plan when LSS cannot find solutions. "$FT == 1$" indicates cases where ACLM determines that there are no reachable offline/ online flight plans for the distressed AAM to safely land, resulting in the deployment of a parachute.

The execution runtime statistics for **Scenario case 1** is in Fig. 5.26, where the runtime of main threads are recorded: 1) C&R watchdog, 2) LSS, 3) Online flight planner. Each statistic shows the information of runtime medians, quartiles, and outliers from the simulation. Note that the multi-goal planning algorithm in the online flight planner may need to search for multiple landing sites to determine real-time reachability to the moderate-risk unprepared

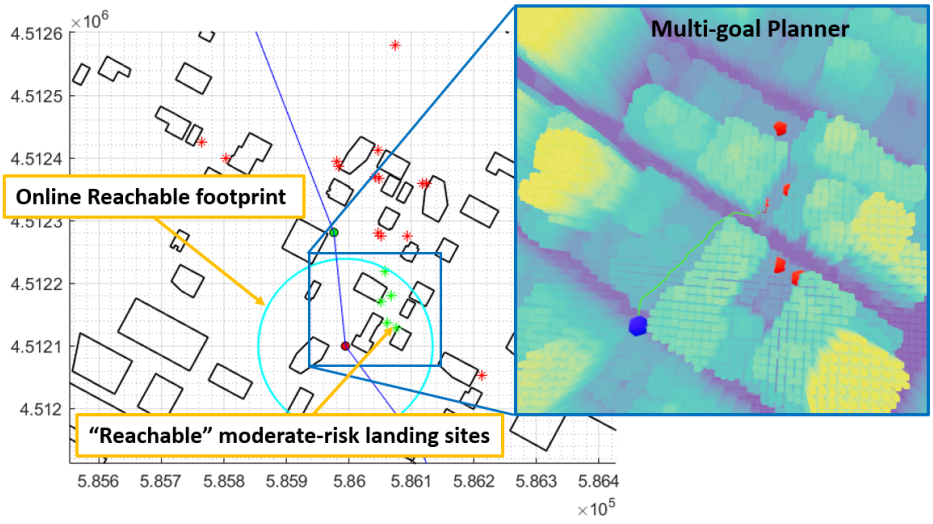


Figure 5.23: Visualization of the online flight planner solution. The reachable moderate-risk landing sites are indicated by green asterisks within the reachable footprint shown in cyan circle. The right figure illustrates the visualization of the multi-goal planner, where the online contingency flight path is generated using voxel A^* algorithm.

landing site. This can lead to a higher number of outliers with extent influenced by the topology of the available unprepared landing sites within the reachable footprint.

Similarly, the Monte Carlo simulation was used to collect the frequency of various ACLM states/outputs for **Scenario case 2**. The results are displayed in Fig. 5.27. As the battery degradation reduces the t_{EOD} to a greater extent than a single rotor failure does, the result indicates that the ACLM tends to rely slightly more on the online flight planner to identify reachable landing sites for **scenario case 2**. The runtime statistics for **scenario case 2** are presented in Fig. 5.28, using the same labels as shown in Fig. 5.26.

Finally, the Monte Carlo simulation was used to collect the frequency of various ACLM states/outputs for **Scenario 3**. The results are displayed in Fig. 5.29. In this scenario, the hexacopter experiences simultaneous rotor failure and battery degradation, resulting huge decrease in T_{EOD}. This resulted in ACLM finding fewer reachable prepared landing sites, and therefore relying more on online flight planner to find paths to unprepared landing sites in real-time. Note that this trend is reflected in the number of available prepared landing sites and unprepared landing sites in Table 5.1. The runtime statistics for **scenario case 3** are presented in Fig. 5.30, using the same labels as shown in Fig. 5.26.

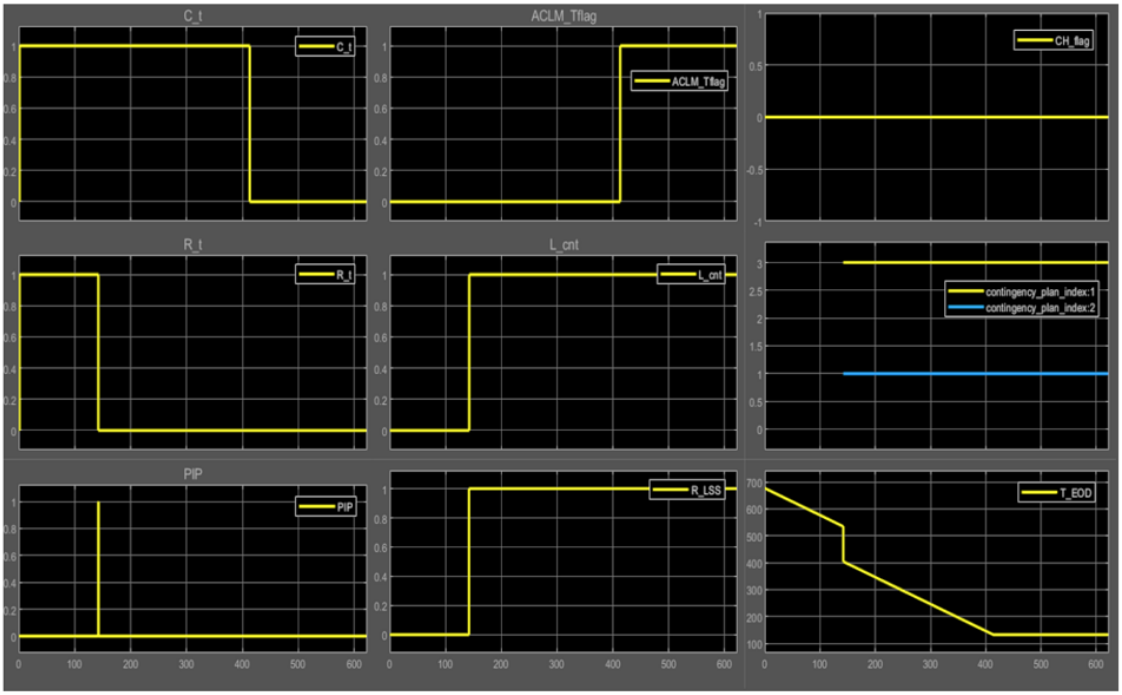


Figure 5.24: Example simulation ACLM outputs . C_t represents controllability and R_t represents reachability in C&R watchdog. PIP indicates the ongoing plan being executed by the LSS algorithm, which searches for a safe landing site from an offline database. ACLM_Tflag is a system termination flag that activates when the hexacopter loses controllability. L_cnt indicates the number of times ACLM executed the LSS algorithm, and R_LSS is a flag indicating the availability of a reachable prepared landing site. CH_flag is the continue/hold flag, contingency_plan_index refers to the map data index for the prepared landing site, and T_EOD represents the end of discharge time of the battery.

5.5.2 Heavyweight Package-carrying Hexacopter

A heavier-weight package-delivering hexacopter is equipped with the same maximum thrust capability of 27 N as in the lighter-weight hexacopter model. This level of thrust is sufficient to effectively guide, control, and navigate the hexacopter in nominal conditions. However, in the event of a single rotor failure that results in a quadcopter configuration, the heavier-weight hexacopter with a mass of 1.837 kg cannot generate sufficient thrust to simultaneously guide, control, navigate, and maintain its desired cruise altitude. Consequently, when encountering rotor failure or a combination of rotor failure and battery degradation, the heavier-weight package-delivering hexacopter consistently experiences a loss of altitude while attempting to reach contingency landing sites. A visualization of a heavier-weight hexacopter model performing trajectory after experiencing a single rotor failure is shown in Fig. 5.31.

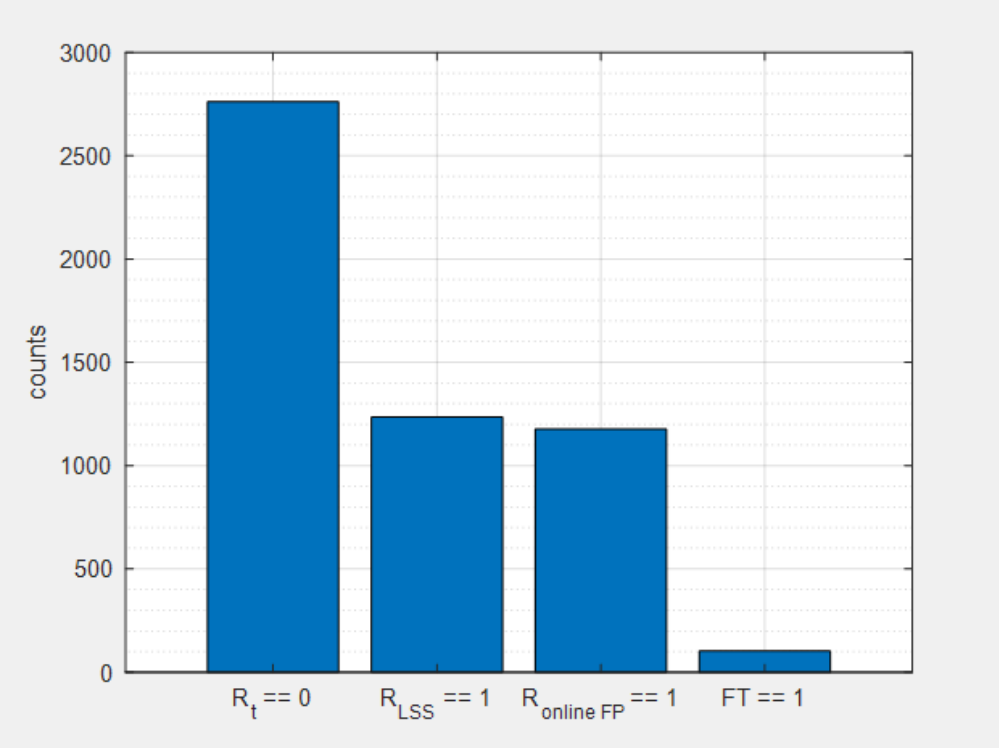


Figure 5.25: Monte Carlo simulation of **lighter-weight** hexacopter with **case 1** simulation: a rotor failure during flight. A total of 3,466 flight simulations were performed, and the frequency of occurrence for different ACLM statuses is presented.

As in thelighter-weight hexacopter model, a total of 3,466 simulations were performed for each scenario. The frequency of different ACLM states/outputs for **Scenario 1** is shown in Fig. 5.32. With the heavier-weight hexacopter, the average current draw following a rotor failure is considerably higher compared to the lighter-weight hexacopter, as indicated in Table 5.6. Furthermore, following a rotor failure, the heavier-weight hexacopter model experiences a descent in altitude, leading to a decrease in the number of reachable prepared landing sites. This occurs because some of the prepared landing sites in the original footprint become inaccessible due to the loss of altitude over time. As a result, the ACLM placed greater reliance on online flight planners to determine accessible landing sites.

The execution runtime statistics for **Scenario 1** is presented in Fig. 5.33, where the runtime of main threads are recorded: 1) C&R watchdog, 2) LSS, 3) Online flight planner. Each statistic shows the information of runtime medians, quartiles, and outliers from the simulation.

Similarly, the Monte Carlo simulation was used to collect the frequency of ACLM states/outputs for **scenario case 2**. The results are displayed in Fig. 5.34. Just as the lighter-weight package-delivering hexacopter, as the battery degradation reduces the T_{EOD}

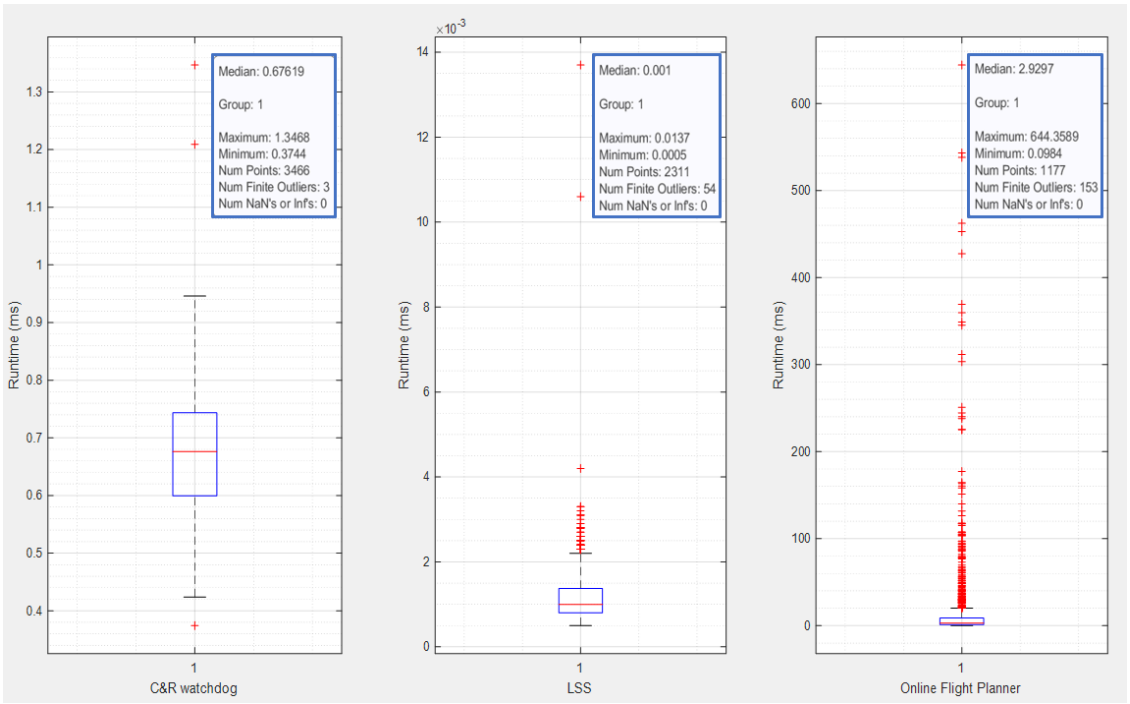


Figure 5.26: Execution time of main threads in ACLM for **case 1** with **lighter-weight** hexacopter.. Median, quartiles, as well as outliers, are presented for C&R watchdog, LSS, and online flight planner.

to a greater extent than a single rotor failure does, the result indicates that the ACLM tends to rely slightly more on the online flight planner to identify reachable landing sites for **Scenario 2**.

The runtime statistics for **Scenario 2** are presented in Fig. 5.35, using the same labels as shown in Fig. 5.26.

Finally, the Monte Carlo simulation was used to collect the frequency of various ACLM states/outputs for **scenario case 3**. The results are displayed in Fig. 5.36. In this scenario, the hexacopter experiences simultaneous rotor failure and battery degradation, resulting huge decrease in T_{EOD} as well as a decrease in the number of available prepared landing sites. This resulted in ACLM identifying fewer reachable prepared landing sites, and therefore relying more on online flight planning with unprepared landing sites.

The runtime statistics for **Scenario 3** are presented in Fig. 5.37, using the same labels as shown in Fig. 5.26.

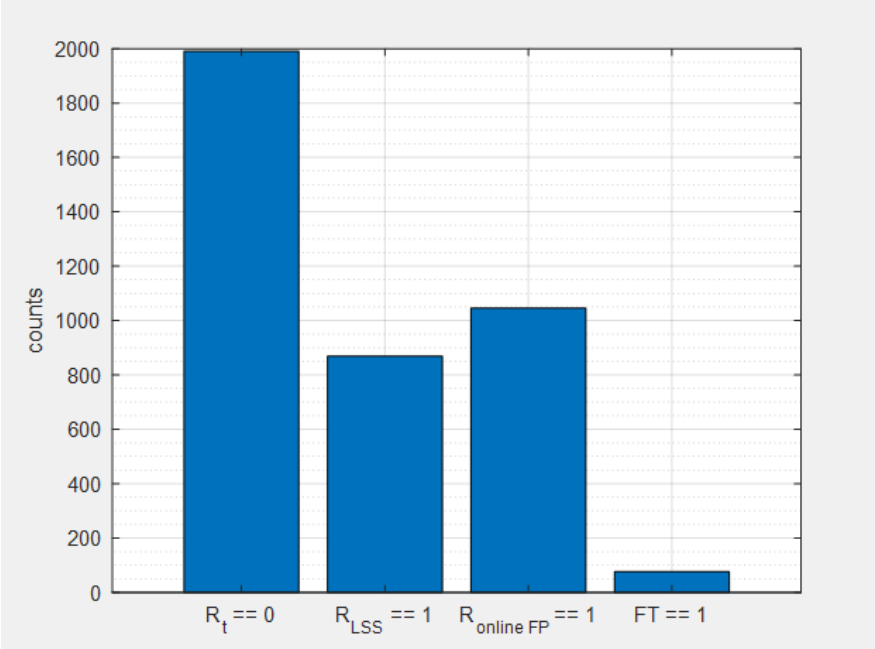


Figure 5.27: Monte Carlo simulation of **lighter-weight** hexacopter with **case 2** simulation: a single-time battery degradation during flight. A total of 3,466 flight simulations were performed, and the frequency of occurrence for different ACLM status values is presented. Each label corresponds to the labels in Fig. 5.25.

5.6 Discussion

While the Monte Carlo simulation results indicated a relatively low occurrence of Flight Termination (FT), there remain strategies to further reduce the probability of FT in emergency scenarios. One approach is generating flight plans that avoid potential dead zones (i.e., overflown areas with no reachable prepared landing sites) to offer ACLM more options in the event of a failure. By planning routes that are in close proximity to prepared landing sites, ACLM can reduce the chances of FT in emergency scenarios.

ACLM assurance can be only achieved when diagnostics and prognostics systems are present and function properly. These systems play a critical role in providing real-time information about component health (i.e., rotor) and predicting future performance (i.e., estimating the remaining useful life of battery) as in [105]. Such systems are crucial in determining the controllability and reachability of landing sites.

ACLM plans developed in preflight and real-time will need to consider urban wind and visibility conditions to calculate footprint and efficacy of each potential landing site. Although computationally expensive and sensitive to prevailing winds, CFD can be used to accurately predict wind near landing sites as in [115]. Also, incorporating navigational un-

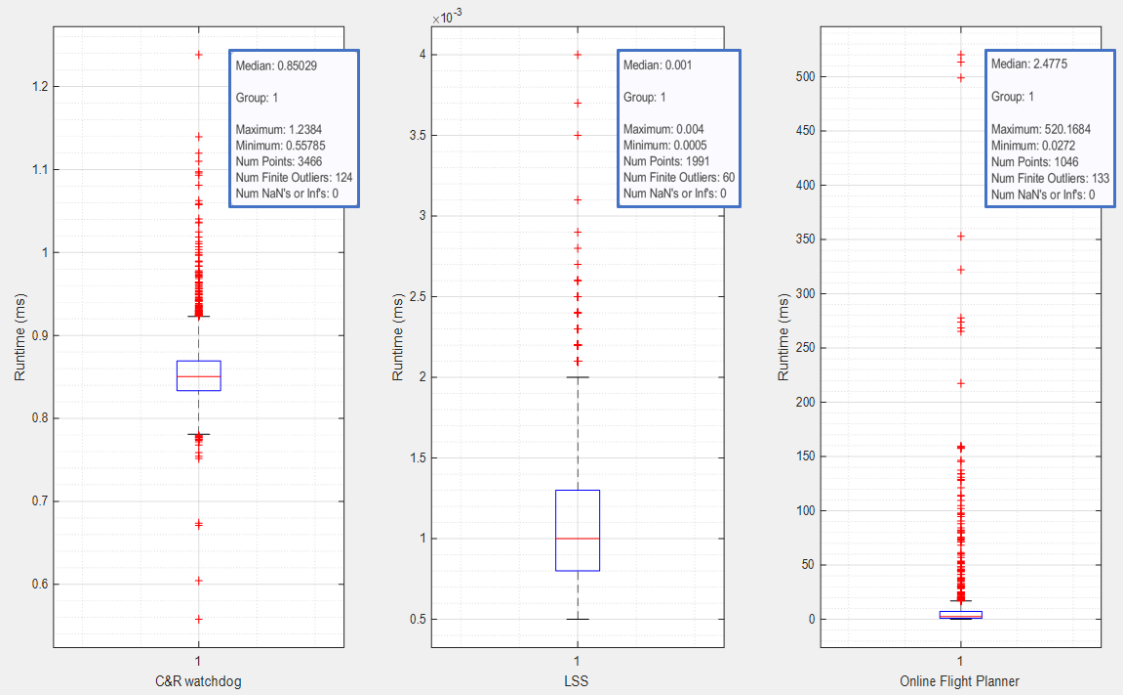


Figure 5.28: Execution time of the main threads in ACLM for **case 2** with **lighter-weight** hexacopter.. Median, quartiles, as well as outliers, are presented for C&R watchdog, LSS, and online flight planner.

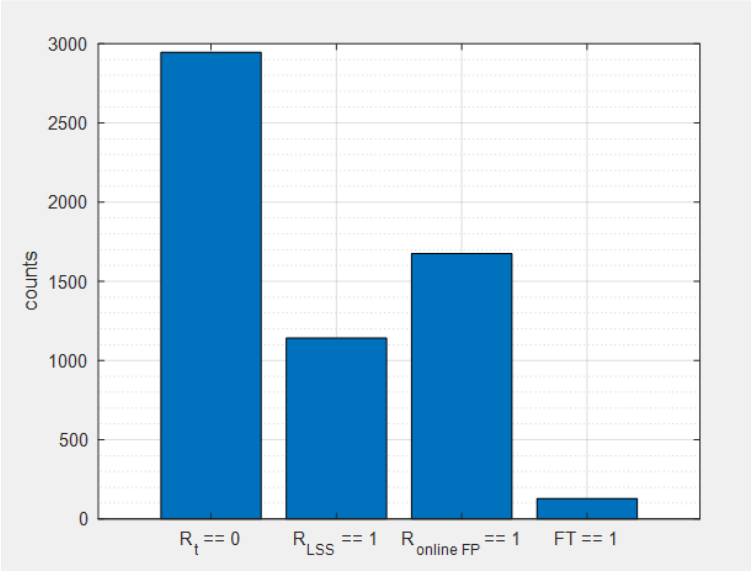


Figure 5.29: Monte Carlo simulation of **lighter-weight** hexacopter with **case 3** simulation: the simultaneous occurrence of rotor failure and single-time battery degradation during flight. A total of 3,466 flight simulations were performed, and the frequency of occurrence for different ACLM statuses is presented. Each label corresponds to the labels in Fig. 5.25.

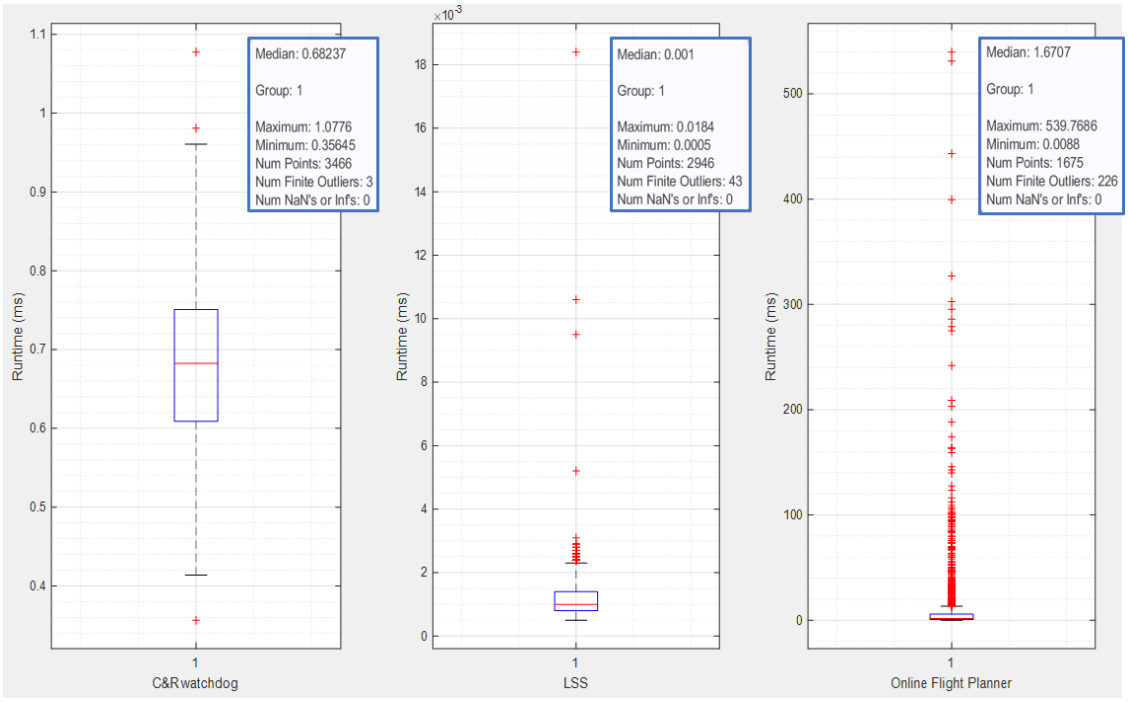


Figure 5.30: Execution time of main threads in ACLM for **case 3** with **lighter-weight** hexacopter. Median, quartiles, as well as outliers, are presented for C&R watchdog, LSS, and online flight planner.

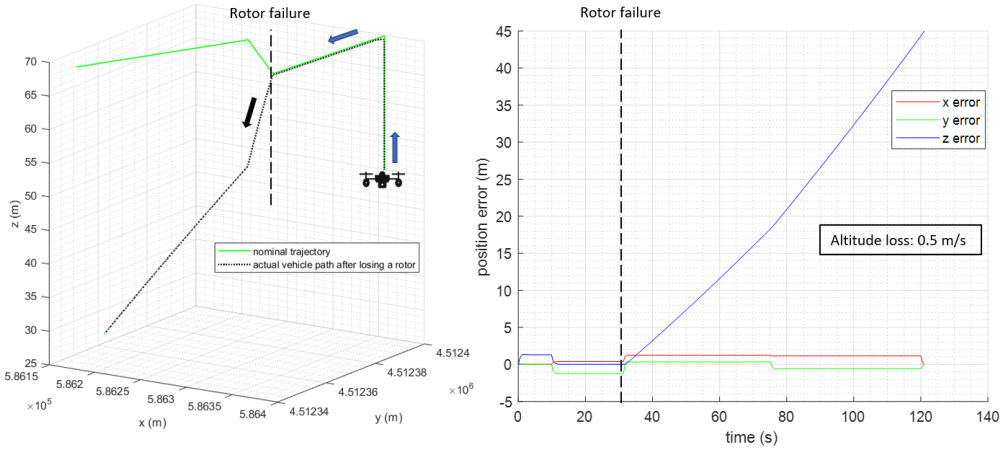


Figure 5.31: Visualization of a **heavier-weight** hexacopter experiencing a loss of altitude following a rotor failure. At the 30-second mark, one rotor loses thrust. On the left side, the 3D representation displays the nominal flight trajectory (depicted in green), and the actual vehicle path (shown as a black dashed line). The figure on the right illustrates the positional error in altitude after the rotor failure. The reachable footprint diminishes, resulting in a reduced number of reachable prepared landing sites and moderate-risk landing sites.

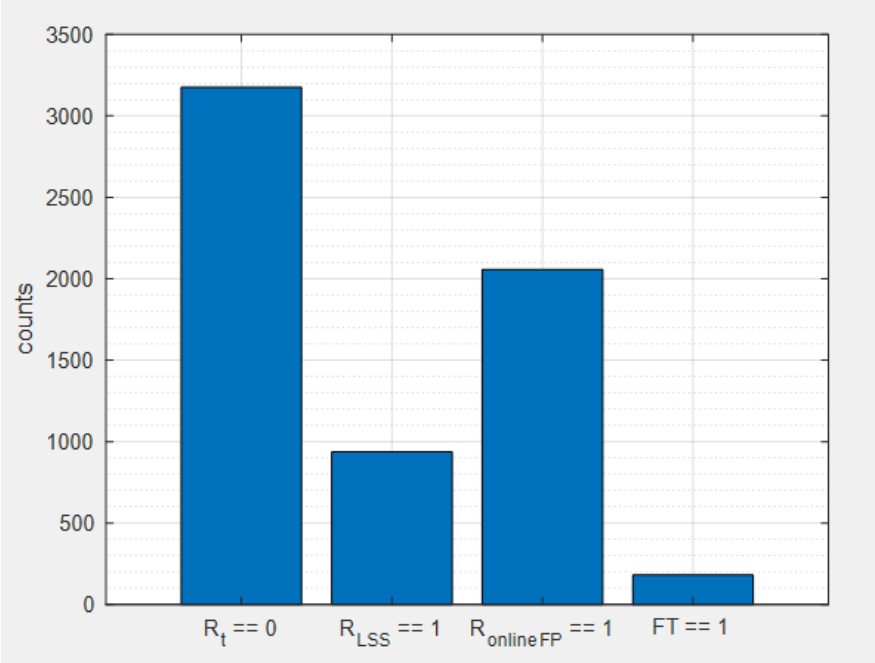


Figure 5.32: Monte Carlo simulation of **heavier-weight** hexacopter with **case 1**: a rotor failure during flight. A total of 3,466 flight simulations were performed, and the frequency of occurrence for different ACLM statuses is presented. Each label corresponds to the labels in Fig. 5.25.

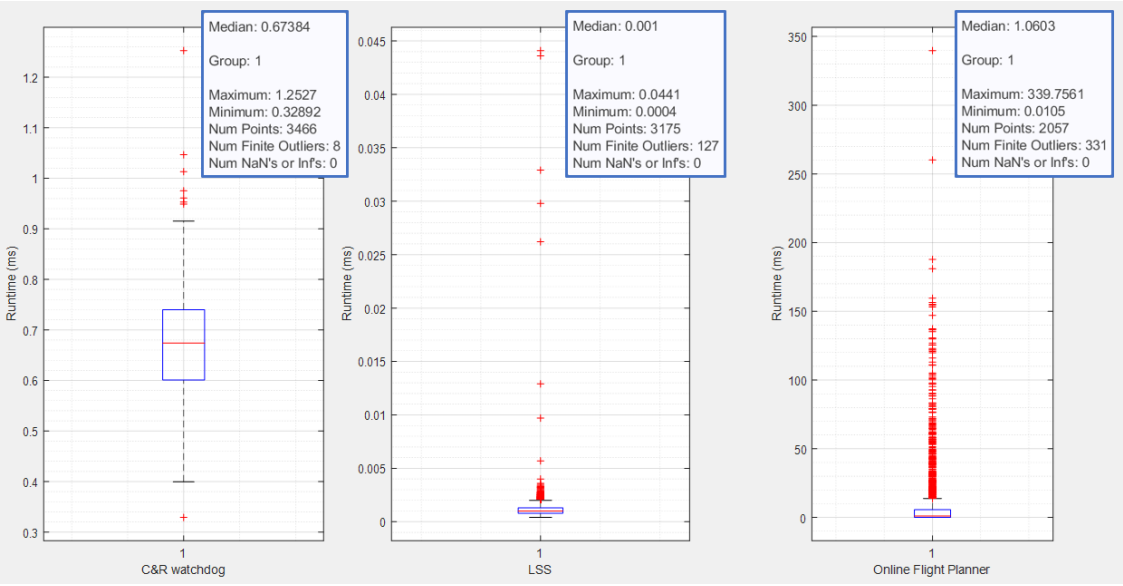


Figure 5.33: Execution time of main threads in ACLM for **case 1** with **heavier-weight** hexacopter. Median, quartiles, as well as outliers, are presented for C&R watchdog, LSS, and online flight planner.

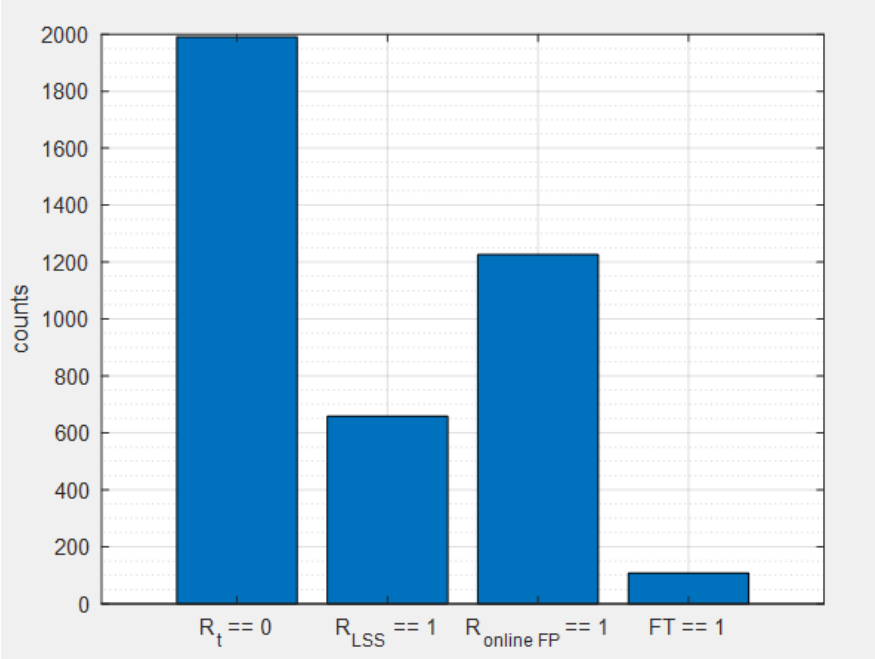


Figure 5.34: Monte Carlo simulation of **heavier-weight** hexacopter with **case 2**: a single-time battery degradation during flight. A total of 3,466 flight simulations were performed, and the frequency of occurrence for different ACLM statuses is presented.

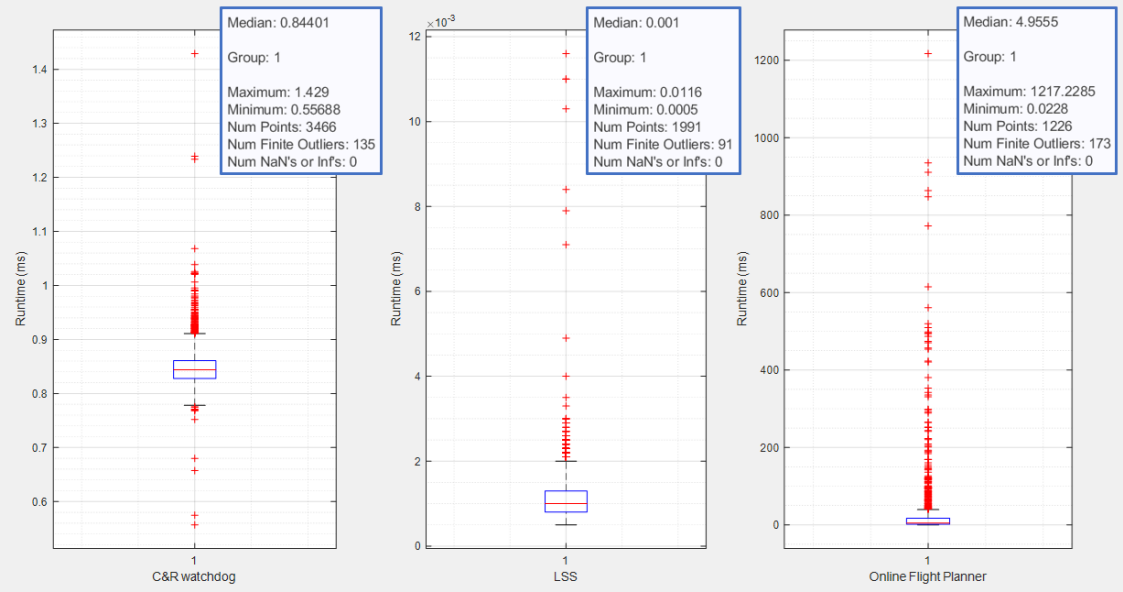


Figure 5.35: Execution time of main threads in ACLM for **case 2** with **heavier-weight** hexacopter. Median, quartiles, as well as outliers are presented for C&R watchdog, LSS and online flight planner.

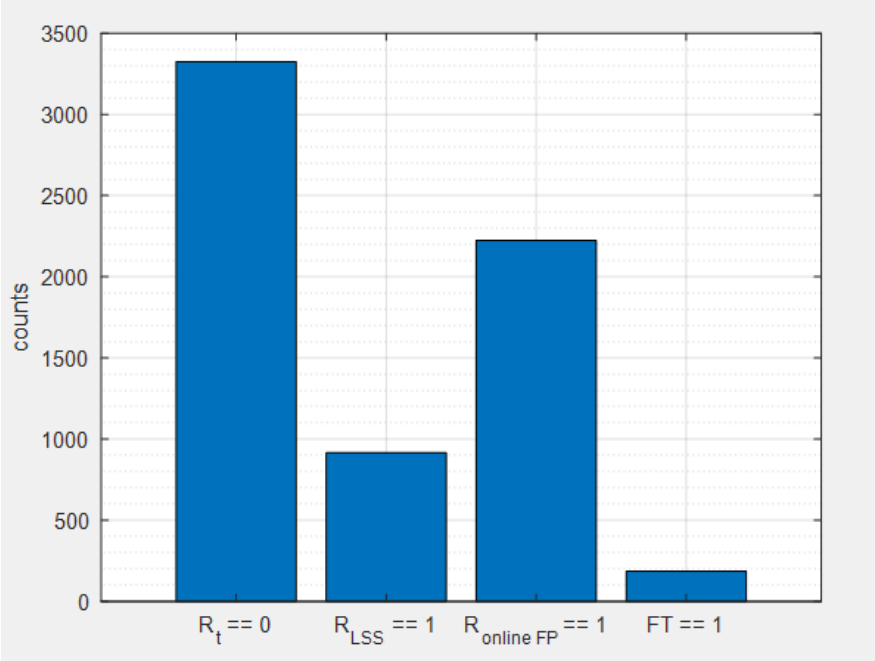


Figure 5.36: Monte Carlo simulation of **heavier-weight** hexacopter with **case 3**: the simultaneous occurrence of rotor failure and single-time battery degradation during flight. A total of 3,466 flight simulations were performed, and the frequency of occurrence for different ACLM statuses is presented.

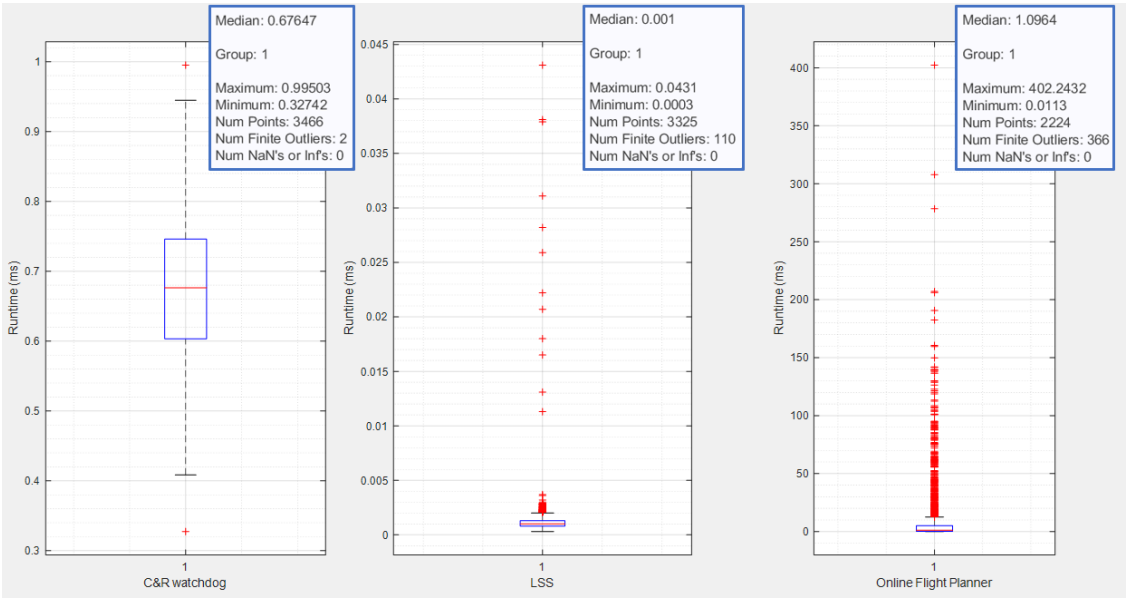


Figure 5.37: Execution time of main threads in ACLM for **case 3** with **heavier-weight** hexacopter. Median, quartiles, as well as outliers, are presented for C&R watchdog, LSS, and online flight planner.

certainty from GPS and sensor errors in urban terrain [116] will add an additional buffer layer to the reachable footprint calculation.

This work has assumed prepared and unprepared landing site data is complete and correct. While prepared landing sites are have a high probability of being clear and ready to support an emergency landing, unprepared landing site conditions are uncertain. In reality, a UAS would need to survey the targeted unprepared landing site once it entered onboard sensor field of view and divert or execute FT as necessary. Incorporation of onboard sensor data and real-time observations from other data sources, e.g., air or ground-based, can improve estimates of risk and availability in future work.

5.7 Conclusion

This chapter introduces an Assured Contingency Landing Management (ACLM) solution for Advanced Air Mobility (AAM) operations. By utilizing a combination of prepared and unprepared landing site data, ACLM enables distressed AAM flights to quickly determine the safest landing option, minimizing reaction time and improving overall safety.

Our work includes the development of the algorithms based on mathematical theory and database. The proposed ACLM solution is specifically designed for multicopter systems with more than four rotors. However, ACLM can be seamlessly adopted in other vehicle configurations, including fixed-wing and hybrid vehicles. Degradation models for battery modules and rotor failures are incorporated into the Monte Carlo simulations to evaluate the impact of different failure modes on safe landing decisions.

The statistical performances showed that ACLM can successfully find contingency landing solutions in the presence of various in-flight anomaly scenarios. By addressing the challenges posed by AAM operations in low-altitude airspace, our research contributes to the ongoing efforts to achieve safe and efficient future autonomous transportation solutions. Further advancements in ACLM will play a crucial role in the full realization of AAM and its safe operation.

In future work, we plan to enhance our ACLM system by integrating urban wind conditions, which will enable a more accurate calculation of the reachable footprint. Additionally, we aim to incorporate navigational uncertainty resulting from GPS and sensor errors, specifically considering the influence of urban terrain. This inclusion will add an additional buffer layer to the reachable footprint calculation.

CHAPTER 6

Centralized and Distributed Optimization of AAM Strategic Traffic Management

6.1 Introduction

The growth of AAM as an urban/ regional transportation mode raises the need for efficient air traffic management [25, 26, 27, 117]. Therefore, various concept of operations (ConOps) and architectures have been proposed for UAS/ AAM Traffic Management [28, 117, 29, 118, 17]. Figure 6.1 illustrates FAA's envisioned AAM transportation system [6] and the roles of each stakeholder. However, to our knowledge, no research has investigated systematic and efficient low-altitude urban airspace management, considering local traffic constraints in flight corridor and vertiport limits, as well as vehicle types, their service priorities and equity. This research aims to fill this gap by strategically optimizing AAM traffic management in both centralized and distributed Providers of Services for UAM (PSU) settings. Our research addresses three key questions:

1. **Airspace Sectorization**: How should urban airspace be divided so that local traffic managers (PSUs) can handle AAM effectively?

2. **AAM Route Planning**: How can we efficiently plan AAM routes when both airspace and vertiports have limited capacities in each PSU? This involves strategies such as modifying departure times and adjusting AAM flight speeds while traveling through corridors.

3. **Distributed Management**: When multiple PSUs oversee neighboring airspace sectors, how can we coordinate airspace management for smoother AAM operations while ensuring each PSU's traffic flow capacity is met? We employ a cooperative game theoretic approach to coordinate AAM traffic management among PSUs.

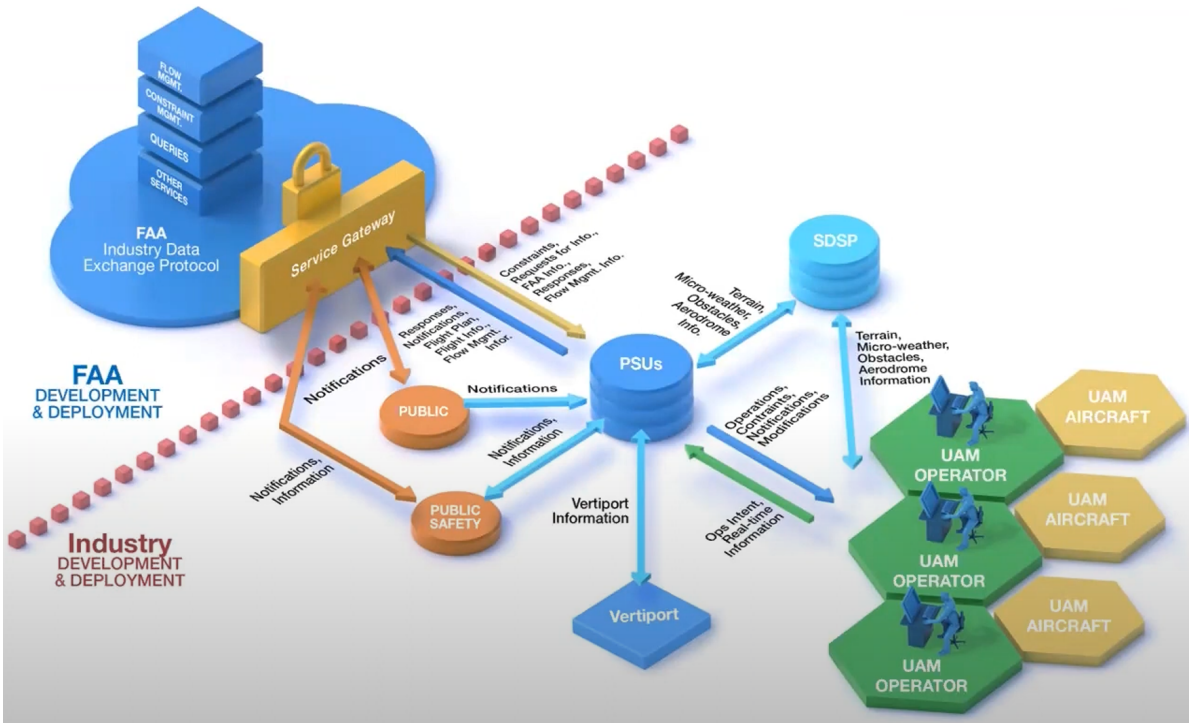


Figure 6.1: Envisioned AAM architecture with a PSU network [6].

This research takes into account the roles of various stakeholders, including traffic management (PSUs), AAM operators, and communities. The proposed traffic management systems for both centralized and distributed PSU environments are evaluated by their runtimes, objective costs, and average departure and airborne delays. This aims to compare different models and design safe, efficient, and scalable AAM traffic management for the growing AAM industry. In our work, we adopt flight corridors for AAM route planning. This aligns with the ConOps proposed by government entities, research laboratories, and AAM companies [28, 119, 120, 121, 122, 123, 124, 125]. The summarized ConOps are shown in Figure 6.2.

In this problem set, we consider the PSU's role as providing critical data (i.e., terrain, obstacle, and weather services), segmenting airspace, setting geofences, and strategically planning AAM flight routes [28] with the following criteria/ goals:

1. Each PSU manages multiple vertiports and computes distinct air traffic flow management (ATFM) capacity limits based on factors like vertiport size and public concerns.

2. PSU must ensure conflict-free operations, spatially or temporally, for all AAM flights.

3. To ensure safe AAM operations, especially during the early stages of low-autonomy AAM deployment [126, 127, 128], PSUs aim to allocate optimized flight corridor routes,

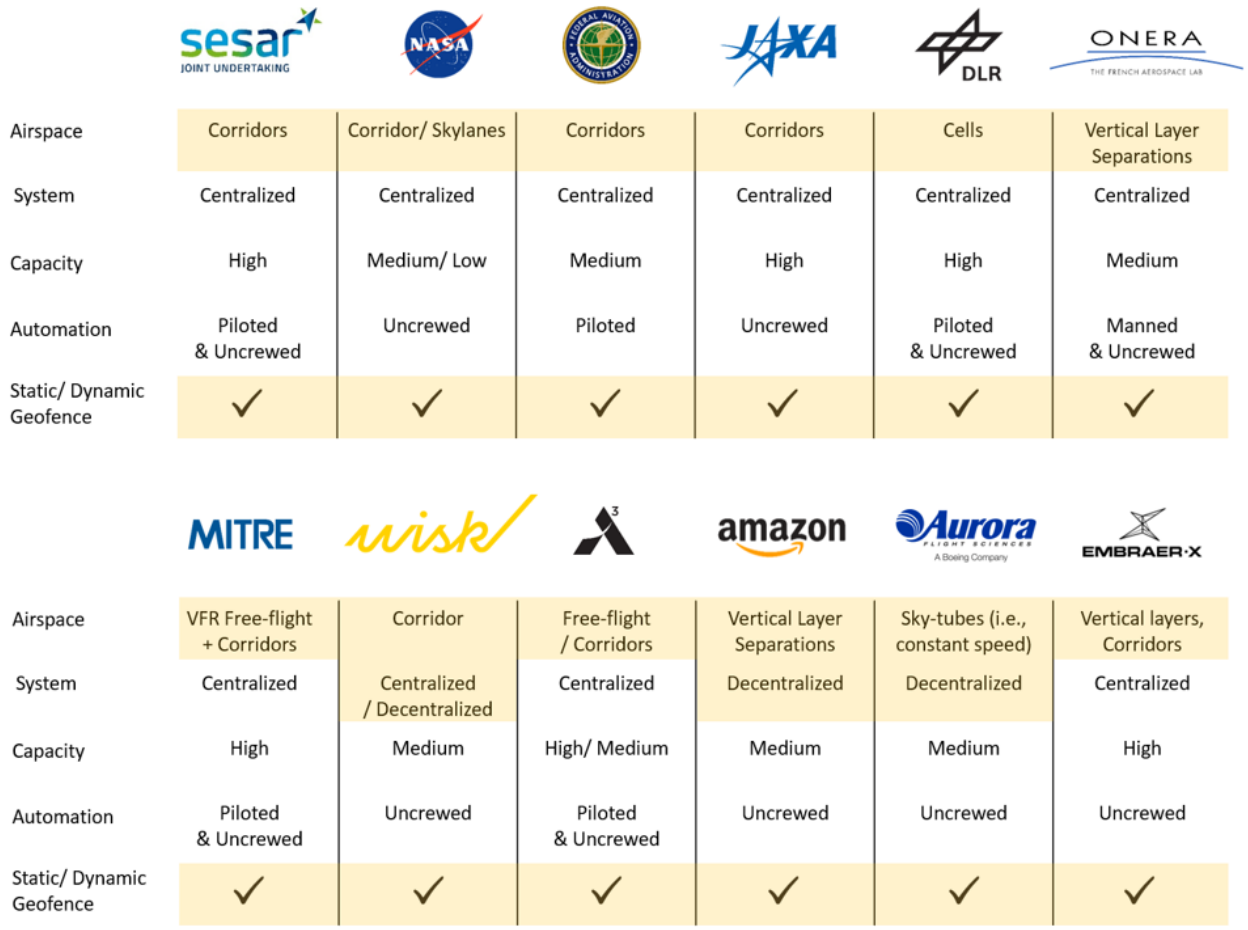


Figure 6.2: AAM airspace design ConOps.

minimizing travel time while considering airspace, vertiport capacities as well as vehicle service priorities.

4. PSUs aim to ensure equitable allocation of departure time for each AAM vehicle, considering factors such as operational service priorities and vehicle characteristics (i.e., min & cruise speeds and range).

In the simulation, we categorize AAM vehicles into three configurations: multicopter, vectored thrust, and lift + cruise, representing the most common designs [129, 130, 131]. Then, each vehicle's design-specific flight speeds and range are considered when solving AAM traffic solutions. Additionally, service priorities are classified into regular, express, and medical categories to accommodate various customer needs. For instance, medical centers and biotech companies aim to expedite organ transfers using AAM vehicles [132]. Simulations cover both centralized and distributed PSU systems in an artificially created map, testing scenarios with 150 and 300 vehicles in a Monte Carlo simulation. Objective costs, flight delays,

and scalability (i.e., runtime) between centralized and distributed systems are compared.

The structure of the chapter is as follows. Section 6.2 summarizes related work in AAM traffic management. Section 6.3 describes the methodologies and algorithms for airspace sectorization, corridor route planning and AAM traffic optimization. Section 6.4 outlines key assumptions and a procedure for the simulation setup. Section 6.5 presents the simulation results. Section 6.6 concludes the chapter.

6.2 Literature Review

The technical maturity of aviation technology is creating a new era of transportation. AAM services include various operations such as passenger transportation, package/ medical delivery, humanitarian / rescue missions, surveillance, and ground data gathering, all within low-altitude airspace. However, ensuring the safe and efficient operation of AAM requires collaborative efforts among governments, industries and academia to overcome many challenges [20, 27, 28]. In this literature review, our focus centers on the challenges of AAM traffic management, addressing the complexities of high-tempo, low-altitude, high-density AAM operations within the national airspace system (NAS) to ensure safe and efficient operations.

To start off, NASA has been conducting collaborative research to integrate AAM operations into the NAS. This involves investigating procedures and algorithmic tools to address the integration challenges of AAM air traffic operations. NASA's high-level initial airspace integration concept, as well as its envisioned strategic and tactical management components, are outlined in [27]. The main challenges in AAM traffic management include AAM congestion management, separation management and vertiport take-off/landing capacity sizing [27, 28, 133].

Governments, research laboratories, and AAM companies released their ConOps, envisioning AAM operation through corridors [28, 119, 120, 121, 122, 123, 124, 125]. The FAA formally defines a corridor as "an airspace volume defining a three-dimensional route, potentially divided into multiple segments, with associated performance requirements" [117]. FAA and NASA envision that AAM vehicles are cooperatively managed within these corridors, governed by set rules, to support increasing operational tempo and new service demands [117]. Furthermore, AAM maturity levels are categorized from 0 to 5 based on density and operation scheduling within a federated service network and PSUs [17].

AAM traffic management comprises strategic and tactical components aimed at reducing congestion and ensuring collision-free routing, similar to traditional ATFM [134, 135, 136]. Recent research explores various algorithmic approaches. For example, in [137], dynamic

geofencing (i.e., trajectile geofence that moves along the trajectory) and linear programming were used to develop a pre-departure first-come, first-serve (FCFS) sequential conflict-free trajectory planning for AAM vehicles. Similarly, a FCFS vertiport scheduling algorithm was developed in [138], considering the throughput and capacity of different vertiport configurations. In [139], a graph reinforcement learning method for online schedule planning was introduced, addressing dynamic demand and uncertainties such as take-off delays and weather-induced route closures. In [140], a message-based decentralized computational guidance algorithm is developed for providing tactical guidance commands, ensuring safe arrivals of AAM vehicles, and avoiding line-of-sight events. The paper formulated a multi-agent Markov decision process for cooperative AAM vehicles in free-flight scenarios and solved using Monte Carlo tree search.

In our work, we adopt the corridor architecture for AAM flight operations and develop safe, efficient and scalable AAM traffic strategic traffic management. Particularly, we formulate vehicle-type-aware and service-priority-aware traffic management systems that achieve optimal solutions while considering vertiport take-off/ landing capacities, corridor throughput capacities, and equity. We explore both centralized and distributed PSU architectures (i.e., bi-level optimization incorporating cooperative game theory) to evaluate their scalability. Additionally, we compare distance-based route planning with weighted/optimized route planning in both centralized and distributed PSU settings.

6.3 Methodologies and Algorithmic Approaches

6.3.1 Airspace Sectorization for (distributed) PSU

The Airspace Sectorization Problem (ASP) aims to divide airspace into manageable sectors, originally for controller workload balance and coordination reduction [141]. In the context of AAM traffic management, PSU takes over these tasks, yet sectorization remains vital. Effective airspace sectorization prevents overburdening individual PSUs, ensuring efficient traffic handling within each sector [142]. Furthermore, it reduces the coordination efforts needed between neighboring sectors and PSUs. Also, areas with higher population density will require more detailed sectorization due to increased traffic and potential conflicts. Lastly, the vertiport topology may vary, having different numbers of touchdown and liftoff (TLOF) pads, gates, and parking spaces [133, 143]. Airspace sectorization should reflect these parameters to prevent congestion in individual PSU.

It is important to note that some AAM companies may operate and manage their own vehicles, but the overall air traffic management in the region is overseen by the PSU. For

instance, vertiports may be shared among different AAM companies, each with its own capacity limits. Integrating the number of vehicles that can take off and land at each vertiport seamlessly into the overall air traffic management is governed by the PSU. By grouping the region based on common features such as vertiport capacity and population density (i.e., airspace sectorization), PSU can effectively manage the regional airspace and provide optimal AAM traffic management solutions.

While traditional airspace design may not directly apply to PSU airspace sectorization due to differing operational complexities and infrastructure, studying airspace sectorization methods enhances our understanding of designing sectorization for PSUs. The below Table 6.1 summarizes general airspace sectorization methods for conventional Air Traffic Management (ATM) [142, 144, 145, 146] and their advantages and drawbacks.

Table 6.1: Airspace Sectorization Methodologies for conventional ATM

Method/Algorithm	**Pros**	**Cons**
Grid-Based	Divide into regular grid layouts	Can't adapt to dynamic traffic pattern
Voronoi Diagram	Geometric method for partitioning. No optimization involved	Computationally intensive for large-scale
Graph Partitioning	Useful for identifying natural traffic flow	Complex computations
Machine Learning	Adapts to real-world evolving traffic patterns (i.e., dynamic airspace sectorization)	Requires substantial training data. May result in interpretability problems
Genetic Algorithm	Good for complex, multi-objective problems	Requires fine-tuning parameters. Does not guarantee solution stability

Our PSU airspace sectorization is achieved by assessing similarities between pairs of vertiport areas. The airspace is then sectorized by identifying communities with shared conditions among vertiport pairs. The steps for achieving PSU airspace sectorization are as follows:

1. Create initial connections between vertiport pairs, forming flight corridors that adhere to a minimum and maximum flight range threshold of corridor.

2. Create an undirected, weighted graph. The weight is determined based on the normalized distance, connectivity (number of associated flight corridors per vertiport), likeness in population density, and likeness in vertiport capacity.

3. Utilize the Louvain method of community detection algorithm [147] to identify non-overlapping communities (i.e., groups of nodes) within the spatially conflicted network

4. Implement Voronoi diagrams [148] to generate the union of Voronoi cells associated with individual vertiports within the same community, constructing airspace sectors for each PSU

The Louvain method for community detection is a heuristic algorithm that optimizes modularity score to find non-overlapping communities from large networks [147]. Modularity measures how well a network is divided into communities, with a higher score indicating a better division. A score between 0.3 and 0.7 is generally considered optimal [149]. A Voronoi diagram [148] is a geometric method that divides a space into regions based on the distance to a given set of points. Each region consists of all the points closer to a particular input point than to any other point in the set.

The weight equation for flight corridor network is shown in Eq. 6.1-6.3. The weight factor of each parameter is represented as α_1, α_2, α_3, α_4 respectively, where the sum of the weight factors equal 1. $\mathcal{G}_i$ is the normalized distance between vertiports u and v (i.e., edge i). $\max(d_{\text{corridor}})$ is the maximum corridor distance, and $d_{u,v}$ is the distance between vertiport u and v. $\mathcal{N}_i$ denotes the average connectivity between vertiport u and v. $\max(m_{\text{vertiport}})$ denotes the maximum connectivity that occurs among the vertiports. $\mathcal{H}_i$ stands for the population similarity score in edge i, and $\mathcal{Q}_i$ represents the vertiport capacity similarity score in edge i. p_u and p_v are population densities of the cities, where vertiport u and vertiport v are located. Similarly, c_u and c_v are vertiport u's and v's capacities. Figure 6.3 visualizes PSU airspace sectorization.

$$w_i = \alpha_1 \cdot \mathcal{G}_i + \alpha_2 \cdot \mathcal{N}_i + \alpha_3 \cdot \mathcal{H}_i + \alpha_4 \cdot \mathcal{Q}_i \tag{6.1}$$

$$\mathcal{G}_i = \frac{\max(d_{\text{corridor}}) - d_{u,v}}{max(d_{\text{corridor}})} \tag{6.2a}$$

$$\mathcal{N}_i = \frac{m_u + m_v}{2\max(m_{\text{vertiport}})} \tag{6.2b}$$

$$\mathcal{H}_i = \exp\left(-\frac{|p_u - p_v|}{\max(p_u, p_v)}\right) \tag{6.2c}$$

$$\mathcal{Q}_i = \exp\left(-\frac{|c_u - c_v|}{\max(c_u, c_v)}\right) \tag{6.2d}$$

$$\sum_{i=1}^{4} \alpha_i = 1, \quad 0 < \alpha_i < 1 \tag{6.3}$$

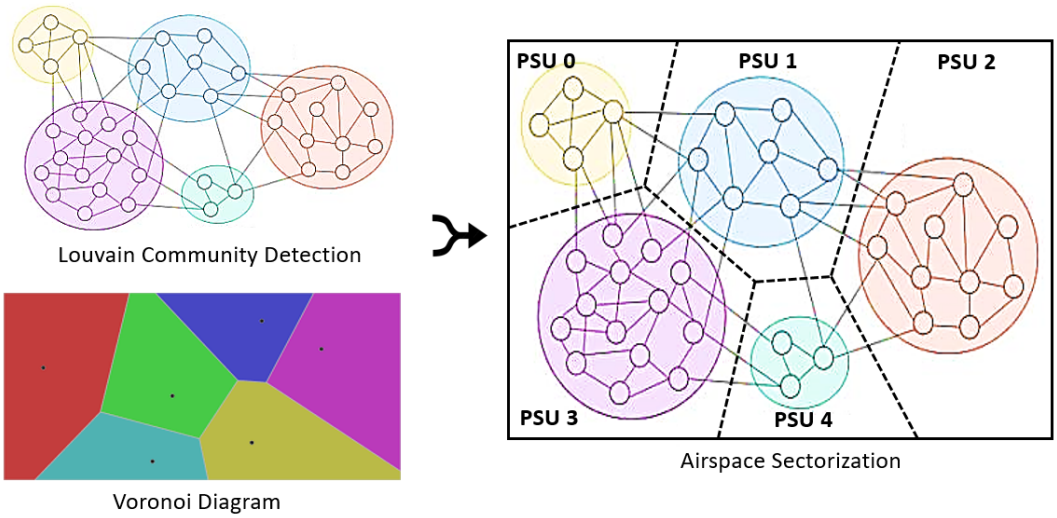


Figure 6.3: Illustration of PSU airspace sectorization. Each small circle represents a vertiport. Solid grey edges represent the corridors, connecting pairs of vertiports. Black dashed lines indicate the PSU boundaries.

6.3.2 Corridor-based Route Planning

6.3.2.1 Distance-based vs. Weighted/Optimized Path Construction

Through the designed PSU, AAM traverses flight corridors connecting departure and destination vertiports, possibly passing through intermediate vertiports on its way. The best corridor routes are determined using Dijkstra's algorithm [150], which finds the shortest corridor routes between departure and destination vertiports. In our work, we explore two different approaches to corridor-based route planning. The first method employs the typical distance-based Dijkstra algorithm to find the shortest route between each pair of departure and destination vertiports. The second approach utilizes corridor weights in Eq. 6.1-6.3 to find the path that minimizes the total weights. The weighted path allows PSU to consider multiple factors to determine the optimal route while avoiding potentially congested regions in low-altitude urban airspace. By exploring an alternative route planning approach, we investigate whether distributed AAM traffic management can be further optimized, reducing overall flight delays in congested airspace. In Section 6.5, the Monte Carlo simulation results are analyzed from the randomly generated AAM flight operations (i.e., 150 vehicles, 300 vehicles) in sectorized PSU regions. The drawbacks and advantages of using weighted/optimized path approach are also discussed.

Figure 6.4 illustrates the two corridor route planning methods. Blue circles represent vertiports, each labeled with a vertiport ID. Gray lines depict corridors connecting pairs of vertiports. Gray boxes highlight potential congestion regions where vertiport connec-

tivity (i.e., number of corridors connected at the vertiport), population density, and vertiport take-off/landing capacities are high. The distance-based (shortest distance) path and the weighted/optimized path are shown as red and black dashed lines, respectively. The weighted/optimized path finds the minimum-cost solution, avoiding potentially congested regions (i.e., high-weighted routes).

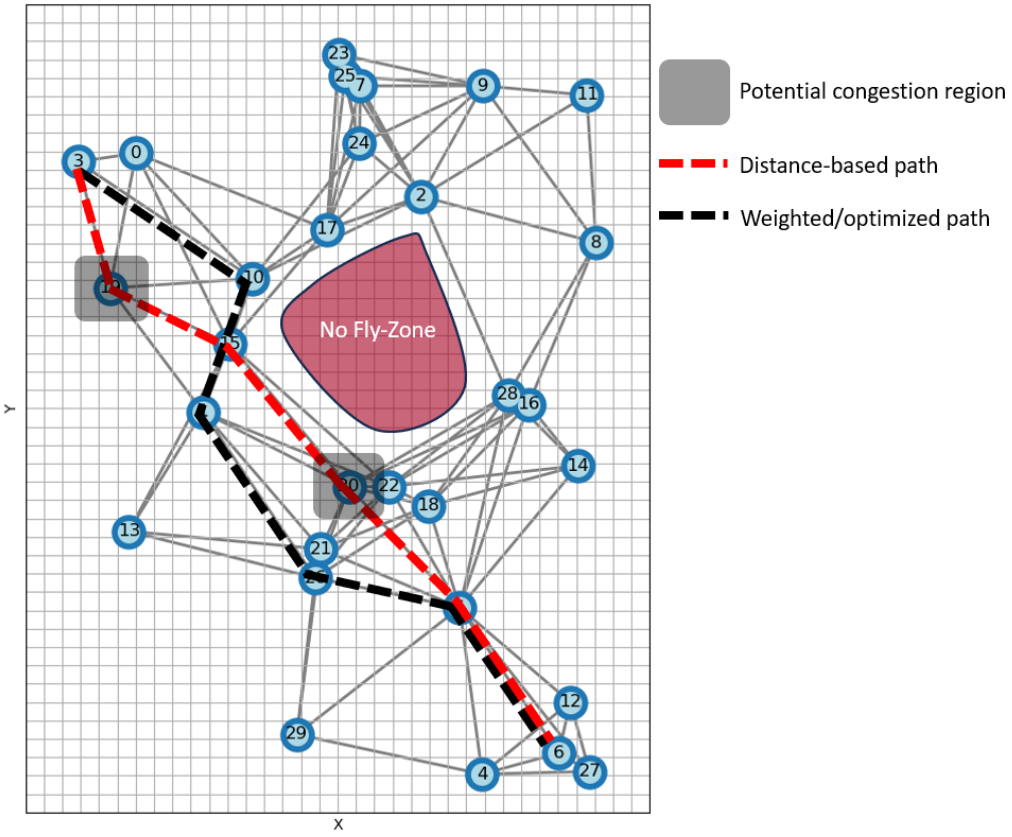


Figure 6.4: Illustration of corridor-based route planning using distance-based and weighted/optimized methods.

6.3.2.2 Corridor Design and Spatial Conflict Detection & Resolution Strategy

AAM vehicles traverse through corridors that connect vertiports. Two types of spatial conflicts can occur while AAM vehicles fly through corridors. The first type occurs within each corridor itself, which we model as a three-dimensional geofence with finite dimensions at a fixed altitude. Because corridors have finite volumes, the maximum number of vehicles that can traverse the corridor simultaneously is limited, leading to spatial conflicts if the capacity is exceeded. Our optimization model in Section 6.3.3 considers each corridor's maximum throughput capacity as a constraint and adjusts vehicle speeds to maintain safe separation

distances between AAM vehicles.

The maximum throughput capacity $\overrightarrow{k_i}$ of the directional corridor i is defined in Eq. 6.4. d_s is the minimum inter-vehicle separation distance (MIVSD) inside the corridor. *dist(i)* is the length of the corridor i. h_i is the number of vertical lanes that construct the directional corridor. For example, a corridor of length 50 km with MIVSD of 2 km and 3 vertical lanes has a maximum throughput capacity of 75 vehicles. In our work, we construct a three-vertical lane bi-directional corridor to connect each pair of vertiports, and MIVSD is set to 2 km. Figure 6.5 illustrates the constructed corridor shape. This design offers the benefit of segregating AAM vehicles by their speeds, particularly advantageous as it facilitates a smooth transition of speed and altitude for AAM flights approaching to land at a destination vertiport or departing from a departure vertiport. However, it's important to note that our optimization method for solving AAM traffic management is not tied to a specific corridor structure. For instance, whether it's a three-vertical lane bi-directional corridor or a three-horizontal lane bi-directional corridor, the results remain the same.

$$\overrightarrow{k_i} = \frac{dist(i)}{d_s} \cdot h_i \tag{6.4}$$

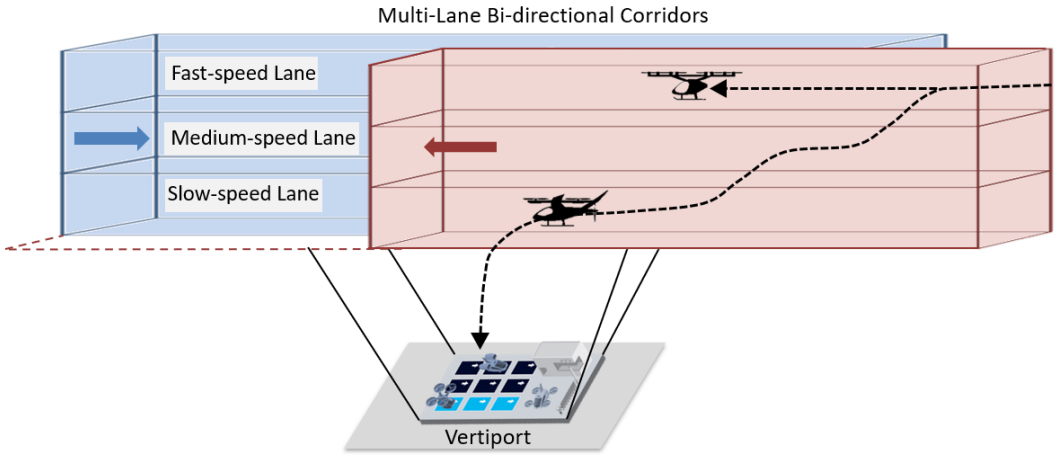


Figure 6.5: Visualization of multi-lane bi-directional corridor

The second type of conflict arises when directional corridors spatially overlap. Such conflicts can occur in multiple scenarios. For example, when multiple corridors converge at a common vertiport or when corridors connecting different pairs of vertiports intersect, spatial conflict occurs. Note that we define a corridor to connect a pair of vertiports. If an AAM flight traverses through multiple vertiports, the flight path consists of multiple corridors in sequence. Figure 6.6 illustrates seven types of spatial conflicts that can occur, with circled numbers indicating the locations of such conflicts in directional corridors. ① and ② cases

denote conflicts when an AAM vehicle is taking off/landing (i.e., yellow flight path) while another AAM vehicle is passing through the corridor (i.e., green flight path) shared through a common vertiport. Case ③ occurs when AAM flights with different corridor paths cross each other at a shared vertiport (i.e., purple and green flight paths). Case ④ occurs when AAM flights with different corridor paths cross each other in mid-air (i.e., purple and yellow flight paths). ⑤ and ⑥ cases denote conflicts where AAM flights with different corridor paths share departure/destination vertiports and the connecting corridors (i.e., purple and yellow flight paths in the center figure). Lastly, case ⑦ is when an AAM corridor path is spatially conflicted with another AAM flight's corridor path by sharing more than one vertiport inside (i.e., purple flight path in the rightmost figure).

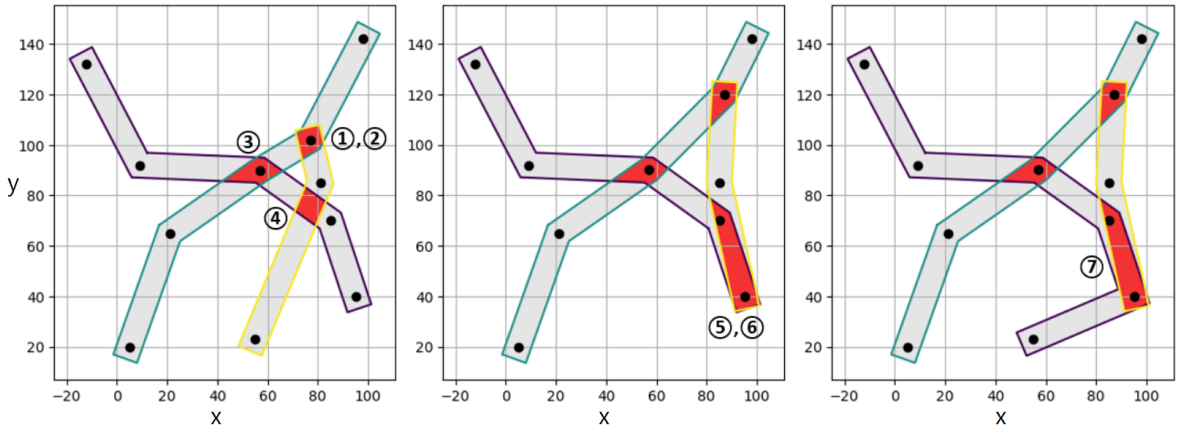


Figure 6.6: Visualization of corridor spatial conflict types.

Each pair of spatially conflicted AAM flight paths can be characterized by a tuple of two conflict types, one for each AAM flight path, from the seven types mentioned above. Once categorized, the next step involves deconflicting the AAM flights temporarily to ensure only one AAM flight passes through the spatially conflicted region at a time, avoiding simultaneous passage. This guarantees that AAM vehicles will not collide in the spatially shared corridor region. Three strategies are available for temporal conflict resolution: 1. delaying the departure time of the AAM flight, 2. adjusting the vehicle speed in the conflicted corridor region, and 3. implementing both options 1 and 2 simultaneously. In strategic AAM traffic management, it is generally preferable to delay the departure time rather than adjust the vehicle speed, considering the energy cost. However, since each vertiport has limited take-off and landing pads, gates, and parking spaces, a system-wide AAM traffic management approach may require implementing all three options to resolve spatially conflicted flight pairs while maintaining scheduled departure and arrival times.

Next step towards temporal conflict resolution involves calculating the time at which

each conflicted AAM flight enters and exits the conflicted region in corridors. This involves finding the boundary coordinates of the shared region and identifying the closest boundary coordinates to each AAM flight path. Orthogonal projection is then used to determine the point of entry and exit, and time at which each flight path enters/ exits the shared region. This method is similar to the spatial conflict detection method employed in [151]. Our method extends it by incorporating various spatial conflict types and comprehensive temporal conflict resolution strategies that adjust vehicle-specific speeds (and even based on the service priority of AAM vehicles).

Figure 6.7 visualizes how spatially conflicted corridor region is detected and temporally deconflicted in each conflicted flight pair. On the left side of the figure, three flight paths with their trajectory geofences (i.e., purple, green, and yellow) are depicted. Spatially conflicted corridor regions are highlighted in red polygons. On the right side, a closer view of spatially conflicted corridors i and j for two conflicted flight paths m and n is shown. The purple plus sign indicates the location before a flight enters a conflicted region (i.e., $B_{m,i}$ and $B_{n,j}$), The purple X sign indicates the location where a flight exits a conflicted region (i.e., $E_{m,i}$ and $E_{n,j}$). Conflicted AAM vehicles are temporally deconflicted if one vehicle enters the conflicted region after the other vehicle exits, as follows (i.e., eq. 6.5):

$$B_{m,i} > E_{n,j} + t_s \quad or \quad B_{n,j} > E_{m,i} + t_s \tag{6.5}$$

Here, t_s is an additional safety separation time buffer. Section 6.3.3.1 outlines the formulation of the temporal conflict resolution as optimization constraints and explains the objective cost design for solving AAM traffic management.

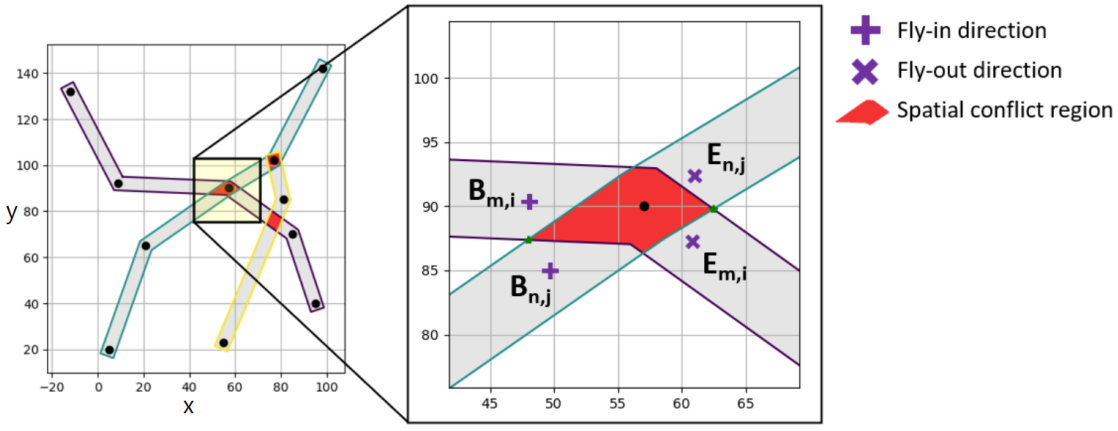


Figure 6.7: Visualizaton of spatial conflict regions between two flight paths (m, n) within corridors (i, j). B and E denote locations before and after a flight enters/ exits a spatial conflict region.

6.3.3 AAM Traffic Flow Management

AAM traffic flow management aims to minimize the total delay time of all flights within centralized/ distributed PSU airspaces. Mixed-integer programming (MIP) [152] is used to solve the optimal solution, adjusting individual vehicle speed along its flight corridors and assigning departure times (which may differ from scheduled departure times). The solution also resolves temporal conflicts in spatially conflicted flight paths. The key parameters and constraints for optimizing AAM traffic are outlined in Table 6.2 and Figure 6.8.

Objective Function	**Parameters**
Minimize Departure & Airborne Delay	Min & Cruise Speed per Vehicle Type
	Departure, Arrival Vertiport Capacities
Consideration	Each Corridor's Max Throughput Capacity
Equity of Assigning Departure Time	Scheduled Departure & Arrival Time per Vehicle
	Cost of Departure Delay & Airborne Delay per Vehicle Type

Table 6.2: AAM traffic optimization objective cost and parameters.

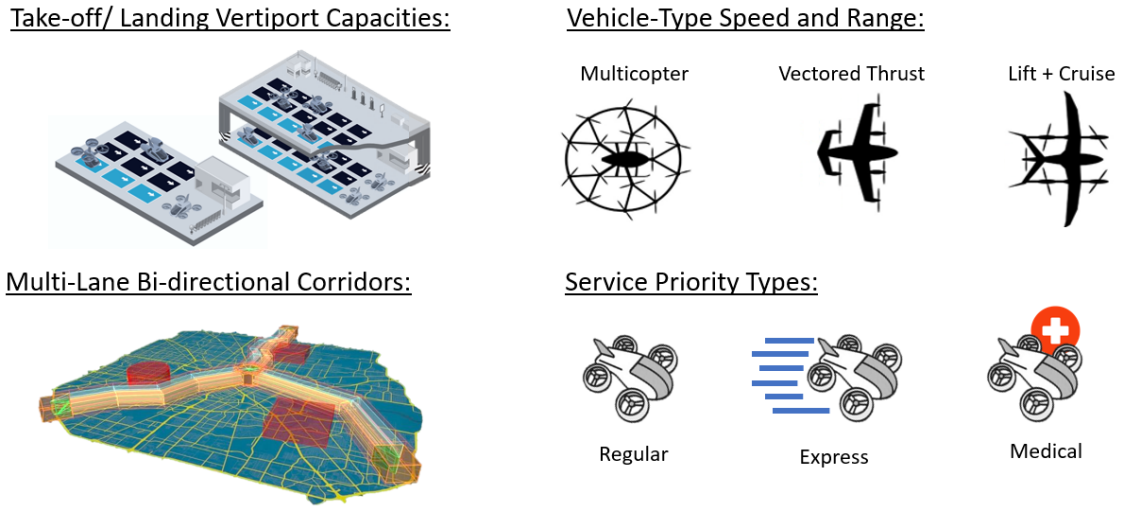


Figure 6.8: AAM traffic optimization constraints. Icons are adapted and modified from [7, 8, 9].

6.3.3.1 Centralized AAM Traffic Flow Management

MIP equations and constraints are formulated to solve a large number of AAM flight operations within a centralized system. The constraints include take-off/landing vertiport capacities and AAM maximum throughput capacities in multilane bi-directional corridors. Additionally, our model accommodates AAM vehicle types (i.e., multicopter, vectored thrust, lift+cruise configurations) with their speeds and ranges. Each vehicle's speed is bounded between minimum cruise to ideal cruise speed, which is specific to the vehicle type. Service priorities are also factored into the formulation, categorized as regular, express, and medical in order of higher priority. Decision variables and parameters for centralized AAM traffic management are given in Table 6.3.

The formulation of the MIP problem for centralized AAM traffic flow management is shown in Eq. 6.6-6.15. The decision variable, $w_{f,t}^{k}$, is based on the well-established ATFM model [153], but modified for solving AAM traffic management in the PSU airspace environment. We define $w_{f,t}^{k}$ equals 1 if flight f arrives at corridor k by time t, and 0 otherwise. The objective function minimizes the total delay cost, which is the sum of airborne delay and departure delay of all AAM vehicles. The objective function is expressed $2 \cdot$ total delay (i.e., airborne delay + departure delay) - departure delay, following the expression in [154]. By expressing this way, we can include the ϵ term (i.e., the delay equity weight) as a super-linearity function in MIP objective cost. This ensures the optimization finds a solution that allows equitable allocation of departure time for each AAM vehicle. For our simulation, we assign ϵ to be 0.05. Lastly, the service priority s_f and the cost ratio (γ) of airborne delay to departure delay of each AAM flight are multiplied to each delay term. The expressed objective function is formulated to fairly assign each vehicle's departure time while considering vehicle-specific parameters and their service priorities.

$$\begin{aligned} \text{Minimize} \quad & \sum_{f \in \mathcal{F}} \Bigg\{ \sum_{t \in \mathcal{O}} \left(\gamma \cdot s_f \cdot (t - a_f)^{1+\epsilon} \cdot (w_{f,t}^{arrival} - w_{f,t-1}^{arrival}) \right) \\ & - \sum_{t \in \mathcal{O}} \left((\gamma - 1) \cdot s_f \cdot (t - d_f)^{1+\epsilon} \cdot (w_{f,t}^{departure} - w_{f,t-1}^{departure}) \right) \Bigg\} \end{aligned} \tag{6.6}$$

subject to

$$\sum_{t \in \mathcal{O}} (w_{f,t}^{departure} - w_{f,t-1}^{departure}) \leq \mathcal{T}_{v,t} \quad \forall f \in \mathcal{F}, \quad \forall v \in \mathcal{V} \tag{6.7a}$$

$$\sum_{t \in \mathcal{O}} (w_{f,t}^{arrival} - w_{f,t-1}^{arrival}) \leq \mathcal{L}_{v,t} \tag{6.7b}$$

Table 6.3: MIP Input Data for Centralized AAM Traffic Management

Set	
$\mathcal{F}$	Set of flights
$\mathcal{V}$	Set of vertiports
$\mathcal{D}$	Set of departure vertiports
$\mathcal{A}$	Set of arrival vertiports
$\mathcal{C}$	Set of corridors
$\mathcal{O}$	Set of flight operation time
$\mathcal{S}$	Set of spatially conflicted flight corridors
Decision Variables	
$w_{f,t}^{departure}$	1: if AAM flight f leaves at departure vertiport by time t. 0: otherwise.
$w_{f,t}^{arrival}$	1: if AAM flight f arrives at destination vertiport by time t. 0: otherwise.
$w_{f,t}^{k}$	1: if AAM flight f arrives at corridor k by time t. 0: otherwise.
$\mathcal{X}_c$	Binary variable, where c = (m,n,i,j) $\in \mathcal{S}$. $\mathcal{X}_c = 1$ if AAM flight m exits conflicted corridor i before AAM flight n enters conflicted corridor j. Otherwise, 0.
x_c	Binary variable, where c = (m,n,i,j) $\in \mathcal{S}$. $x_c = 1$ if AAM flight n exits conflicted corridor j before AAM flight m enters conflicted corridor i. Otherwise, 0.
Parameters	
a_f	scheduled arrival time of AAM flight f
d_f	scheduled departure time of AAM flight f
$l_{f,k}$	minimum time that AAM flight f takes to travel through corridor k
$u_{f,k}$	maximum time that AAM flight f takes to travel through corridor k
a_f	scheduled arrival time of AAM flight f
s_f	service priority of AAM flight f
t_s	safety separation time of spatially conflicted flight pairs
ϵ	delay equity weight
γ	cost ratio of airborne delay to departure delay
$\mathcal{T}_{v,t}$	take-off capacity at vertiport v at time t
$\mathcal{L}_{v,t}$	landing capacity at vertiport v at time t
$\mathcal{M}_k$	throughput capacity at corridor k
$B_{f,i}$	ratio of the conflict region's start point within corridor i relative to its full length, for flight f
$E_{f,i}$	ratio of the conflict region's end point within corridor i relative to its full length, for flight f

$$\sum_{t\in\mathcal{O}}(w_{f,t}^{k} - w_{f,t}^{k+1}) \leq \mathcal{M}_k \quad \forall f \in \mathcal{F}, \quad \forall k \in \mathcal{C} \cup \mathcal{D} \cup \mathcal{A} \tag{6.8}$$

$$\sum_{t\in\mathcal{O}} t\cdot(w^{k'}_{f,t}-w^{k'}_{f,t-1})-\sum_{t\in\mathcal{O}} t\cdot(w^{k}_{f,t}-w^{k}_{f,t-1})\geq l_{f,k} \quad \forall f\in\mathcal{F},\quad \forall k,\ k'\in\mathcal{C}\cup\mathcal{D}\cup\mathcal{A} \tag{6.9a}$$

$$\sum_{t\in\mathcal{O}} t\cdot(w^{k'}_{f,t}-w^{k'}_{f,t-1})-\sum_{t\in\mathcal{O}} t\cdot(w^{k}_{f,t}-w^{k}_{f,t-1})\leq u_{f,k} \tag{6.9b}$$

$$w^{k}_{f,t-1}-w^{k}_{f,t}\leq 0 \quad \forall f\in\mathcal{F},\quad \forall k\in\mathcal{C}\cup\mathcal{D}\cup\mathcal{A},\quad \forall t\in\mathcal{O} \tag{6.10}$$

$$\sum_{t\in\mathcal{O}} w^{k}_{f,t}\geq 1 \quad \forall f\in\mathcal{F},\quad \forall k\in\mathcal{C}\cup\mathcal{D}\cup\mathcal{A} \tag{6.11}$$

$$t\cdot w^{departure}_{f,t}\geq d_f \quad \forall f\in\mathcal{F},\quad departure\in\mathcal{D},\quad \forall t\in\mathcal{O} \tag{6.12}$$

$$\begin{aligned}&\sum_{t\in\mathcal{O}} t\cdot(w^{i}_{m,t}-w^{i}_{m,t-1})+\left(\sum_{t\in\mathcal{O}} t\cdot(w^{i+1}_{m,t}-w^{i+1}_{m,t-1})-\sum_{t\in\mathcal{O}} t\cdot(w^{i}_{m,t}-w^{i}_{m,t-1})\right)\cdot E_{m,i}+t_s\\ &\quad\leq M\cdot(1-\mathcal{X}_c)+\sum_{t\in\mathcal{O}} t\cdot(w^{j}_{n,t}-w^{j}_{n,t-1})\\ &\qquad+\left(\sum_{t\in\mathcal{O}} t\cdot(w^{j+1}_{n,t}-w^{j+1}_{n,t-1})-\sum_{t\in\mathcal{O}} t\cdot(w^{j}_{n,t}-w^{j}_{n,t-1})\right)\cdot B_{n,j}\\ &\qquad\qquad\forall(m,n,i,j)=c_i\in\mathscr{C}\end{aligned} \tag{6.13}$$

$$\begin{aligned}&\sum_{t\in\mathcal{O}} t\cdot(w^{j}_{n,t}-w^{j}_{n,t-1})+\left(\sum_{t\in\mathcal{O}} t\cdot(w^{j+1}_{n,t}-w^{j+1}_{n,t-1})-\sum_{t\in\mathcal{O}} t\cdot(w^{j}_{n,t}-w^{j}_{n,t-1})\right)\cdot E_{n,j}+t_s\\ &\quad\leq M\cdot(1-x_c)+\sum_{t\in\mathcal{O}} t\cdot(w^{i}_{m,t}-w^{i}_{m,t-1})\\ &\qquad+\left(\sum_{t\in\mathcal{O}} t\cdot(w^{i+1}_{m,t}-w^{i+1}_{m,t-1})-\sum_{t\in\mathcal{O}} t\cdot(w^{i}_{m,t}-w^{i}_{m,t-1})\right)\cdot B_{m,i}\\ &\qquad\qquad\forall(m,n,i,j)=c_i\in\mathscr{C}\end{aligned} \tag{6.14}$$

$$\mathcal{X}_c+x_c=1 \tag{6.15}$$

Constraints are as follows. Eq. 6.7a-6.7b are time-dependent take-off and landing capacity constraints at each vertiport v. Eq. 6.8 is the corridor throughput capacity constraint,

ensuring that the total number of flights traversing any flight corridor k does not exceed its capacity. Eq. 6.9a-6.9b define the minimum and maximum traversal times for AAM vehicles through each flight corridor. This essentially adjusts the speed at which each vehicle flies through a specific corridor based on its vehicle type (i.e., multicopter, vectored thrust, lift+cruise). Eq. 6.10-6.11 collectively enforce the constraint that an AAM vehicle must be confined to a single flight corridor at any given time during its flight, ensuring its position remains unique. Eq. 6.12 ensures that the assigned departure time for any flight must be greater than or equal to its scheduled departure time to avoid early departures. Note that this constraint is relaxed in distributed AAM traffic management, and the objective function is adjusted. In distributed management, an AAM vehicle may depart earlier than the scheduled time, if no feasible solution exists to coordinate flights passing through multiple PSUs.

Eq.6.13-6.14 are temporal conflict resolution constraints that delay either one of two flights in spatially shared corridor regions. They are formulated using a big M term (i.e., a large positive constant) to express logical conditions in Eq. 6.5. $B_{m,i}$ is the ratio of the conflict region's start point within corridor i relative to its full length, for flight m. $E_{m,i}$ is the ratio of the conflict region's end point within corridor i relative to its full length, for flight m. $B_{n,j}$ and $E_{n,j}$ follow the same definition, with n and j replacing m and i respectively. Additional safety separation time (t_s) is added to deconflict spatially conflicted flight pairs. Eq. 6.15 ensures that conflicted flights exit the spatially shared corridor region in order. The binary decision variables, $\mathcal{X}_c$ and x_c, follow the definition by Li, et al [151]. $\mathcal{X}_c$ and x_c cannot be 1 simultaneously. Essentially, Eq. 6.13-6.15 enforce the condition expressed in Eq. 6.5.

In a distributed AAM traffic flow management below (i.e., Section 6.3.3.2), each PSU applies slightly modified optimization with additional constraints, and cooperative game theory is used to facilitate fair and coordinated negotiation of AAM traffic among conflicting PSUs. The formulation shows how PSU connectivity and PSU's negotiable bargain power influence overall AAM traffic management.

6.3.3.2 Distributed AAM Traffic Flow Management

As AAM operation grows both in terms of the number of vehicles and the operational regions, the AAM traffic management system requires a distributed model. This is similar to the evolution of conventional air traffic management from centralized to distributed networks. The present aviation traffic management operates within a complex network, incorporating various entities such as the FAA, ICAO, flight operations centers, and traffic management units at en route centers [155]. The traffic management is distributed across manageable units, with redundancies to enhance safety. Whether human operators play an active role

in AAM traffic management or not, distributed PSU traffic management will be crucial for efficiently handling low-altitude, densely populated local air traffic. A significant advantage of distributed PSU traffic management lies in its scalability. It enables the safe and efficient operation of a large number of AAM vehicles by locally optimizing traffic solutions and coordinating/resolving conflicts only for vehicles transitioning through multiple PSUs. The results presented in Section 6.5 shows that the coordinated AAM traffic management solution within individual PSUs can be generated between 1.4 to 30 times faster compared to the time required for centralized management to generate a global solution. With an increase in the number of AAM flight operations, the complexity of the AAM network also rises. Therefore, significantly enhanced scalability of AAM traffic management can be obtained through the distributed system. Refer to Figure 6.1 for the FAA's envisioned AAM architecture involving multiple stakeholders and complexity of the network.

Initially, our centralized AAM traffic management model appears to be scaled for individual PSUs. However, within this framework, each PSU independently manages local air traffic within its designated region. Without coordinated communication and agreements among neighboring PSUs, optimal AAM operation schedules of one PSU may negatively affect adjacent PSUs. This occurs because each PSU independently optimizes its airspace management, potentially leading to one PSU expediting the outflow of many AAM vehicles through other PSU regions to minimize its airspace traffic density. Consequently, adjacent PSUs may experience traffic congestion due to the large inflow of AAM vehicles into their regions. To minimize such congestion and ensure fairness across neighboring PSUs, a coordinated system is crucial with optimized corridor usage and airspace efficiency. Ultimately, PSUs will need to collaborate and coordinate their traffic management solutions to achieve a harmonized and efficient global AAM traffic management system.

To address this challenge, we introduce a collaborative decision-making model among PSUs, leveraging a bi-level optimization approach. The lower-level optimization involves each PSU independently optimizing its AAM traffic management using MIP. Then, the upper-level optimization employs cooperative Nash bargaining game theory [156, 157] to resolve conflicts among the game players (i.e., conflicting PSUs). This cooperative bargaining phase allows negotiation and agreement among conflicted pairs of PSUs to achieve a fair and cooperative agreement on vehicles transitioning between PSU regions. This is done by comparing each PSU's negotiable bargaining power at each operation to reach stability (i.e., Nash bargaining solution). Table 6.4 and Figure 6.9 show the key parameters, objective function, and flow diagram of bi-level optimization for distributed AAM traffic management.

In addition to the variables listed in Table 6.3, we define additional decision variables and parameters for distributed AAM traffic management, outlined in Table 6.5. The PSU

	High-Level Optimization (Game Theory)
Decision Variables	Bargaining power of PSU during operation time
Objective Function	Fair and coordinated negotiation among conflicted PSUs

	Low-Level Optimization (MIP)
Decision Variables	Departure time for each AAM within each PSU airspace
Objective Function	Minimize overall departure and airborne delay within each PSU airspace
Constraints	Speed, capacity and temporal conflict resolution

Table 6.4: Distributed AAM Traffic Management Variables

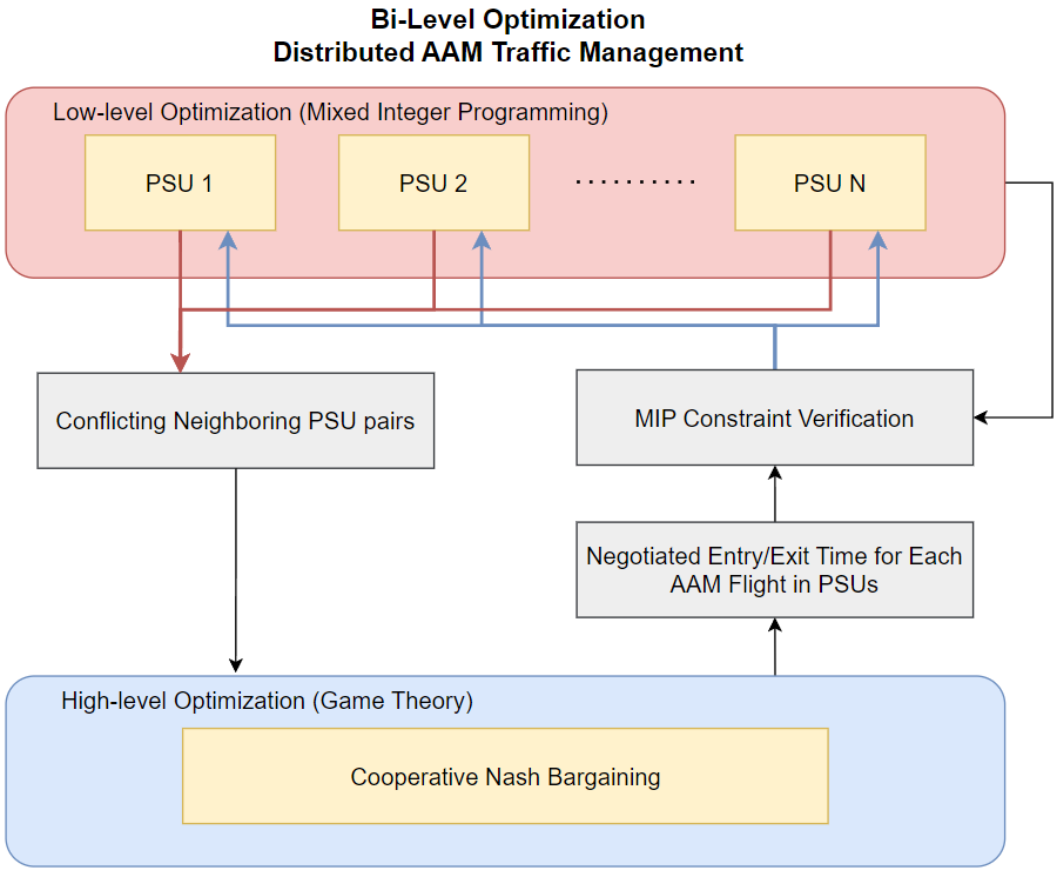


Figure 6.9: Distributed AAM Traffic Management Flowchart

cooperative Nash bargain equations are shown in Eq. 6.16-6.17b below. U_1 and U_2 are transition time equity functions (i.e., utility functions) that show how much each PSU feels about individual AAM's negotiated time of entry/ exit in its airspace region. The decision variable $y_{i,j}^f$ represents the payoff of AAM flight f traveling through conflicted PSUs i and j, with its value ranging between 0 and 1. The decision variable p_i^f is a payoff of AAM flight f entering/ exiting PSU i, and its value ranges between 0 and 1. The equation below

essentially determines the payoff $y_{i,j}^{f}$.

Table 6.5: Additional MIP Input Data for Distributed AAM Traffic Management

Decision Variables	
$b_{f,i}$	bargained entry/ exit transition time of AAM flight f in PSU i
$y_{i,j}^{f}$	payoff $\in [0,1]$ of AAM flight f traveling through conflicted PSUs i and j
p_{i}^{f}	payoff $\in [0,1]$ of AAM flight f entering/ exiting PSU i
Parameters	
n_i	negotiable bargain power of PSU i during flight operation time window $\mathcal{O}$
$t_{f,i}$	optimal entry/ exit time of AAM flight f in PSU i from low-level MIP
$\delta_{i,j}^{f}$	time difference between an AAM flight f's optimal departure from PSU i and its optimal arrival at adjacent PSU j, where PSUs i and j are in conflict
$\mathcal{P}_f$	transition pseudo-vertiport(s) for flight f
$\mathscr{C}_{f,i}$	total number of corridors AAM flight f travels through inside PSU i
$\mathscr{T}_i$	total number of transition corridors in PSU i
$\mathscr{S}_i$	total number of spatially conflicted flight paths inside PSU i
β_{1-3}	bargain parameter weight factors for $\mathscr{C}_{f,i}$, $\mathscr{T}_i$, $\mathscr{S}_i$

$$\begin{aligned} &\text{Maximize} \quad U_i \cdot U_j \\ &\quad \text{subject to} \quad n_j \geq n_i \quad \forall f \in \mathcal{F}, \quad \forall \text{ [conflicted PSU pair (i,j)]} \end{aligned} \tag{6.16}$$

$$U_i(y_{i,j}^{f}) = 1 - y_{i,j}^{f} \tag{6.17a}$$

$$U_j(y_{i,j}^{f}) = \left(y_{i,j}^{f}\right)^{\frac{n_i}{n_j}} \tag{6.17b}$$

Here n_i is the negotiable bargain power of PSU i during flight operation time window $\mathcal{O}$, expressed as Eq. 6.18a-6.18b. The equation comprises three parameters, each multiplied with weight factors (i.e., β_1, β_2, β_3). These weight factors collectively sum up to 1. Each parameter represents AAM density in PSU i, the total number of transition corridors (i.e., corridors connecting to adjacent PSUs), and the total number of spatially conflicted flight paths inside the PSU i.

$$n_i = \beta_1 \cdot \sum_{f=1}^{N} \frac{1}{\mathscr{C}_{f,i}} + \beta_2 \cdot \mathscr{T}_i + \beta_3 \cdot \mathscr{S}_i \tag{6.18a}$$

$$\sum_{i=1}^{3} \beta_i = 1, \quad 0 < \beta_i < 1 \tag{6.18b}$$

It is crucial to recognize that the negotiable bargaining power of PSUs varies with each operational time window $\mathcal{O}$. For instance, during morning hours when AAM is heavily utilized for commuting from residential to commercial PSU areas, there's significant inbound traffic from residential to commercial PSU sectors. Conversely, during the evening rush when people return home from work, there's a surge in outbound traffic from commercial to residential areas. This fluctuation in PSU bargaining power mirrors the demand dynamics within each PSU's operational time window. This establishes a cooperative negotiation framework among PSUs, ensuring that no single PSU maintains a permanent advantage over others. Instead, bargaining power equitably reflects demand, conflict complexity, and the strength of connectivity between adjacent PSUs.

Now, the process of calculating the actual bargained entry/exit transition time $b_{f,i}$ for each AAM flight f in PSU i is shown in Algorithm 9. Note that $\delta^f_{i,j}$ represents the time difference between an AAM flight f's optimal departure from PSU i and its optimal arrival at adjacent PSU j, where PSUs i and j are in conflict.

Algorithm 9: Cooperative Bargained Entry/Exit Time of AAM Flight f Transitioning PSUs i and j

Data: $\delta^f_{i,j}$, $y^f_{i,j}$, $[t_{f,i}, t_{f,j}]$, PSU ID: i, j s.t. $n_j \geq n_i$
Result: $[b_{f,i}, b_{f,j}]$
$p^f_i \leftarrow 1 - y^f_{i,j}$;
$p^f_j \leftarrow y^f_{i,j}$;
if $t_{f,i}$ ¿ $t_{f,j}$ **then**
    $b_{f,i} \leftarrow t_{f,i} - round(\delta^f_{i,j} \cdot p^f_i)$;
    $b_{f,j} \leftarrow t_{f,j} + round(\delta^f_{i,j} \cdot p^f_j)$; `/* round(): rounds floating point number to integer */`
else
    $b_{f,i} \leftarrow t_{f,i} + round(\delta^f_{i,j} \cdot p^f_i)$;
    $b_{f,j} \leftarrow t_{f,j} - round(\delta^f_{i,j} \cdot p^f_j)$;

A new constraint is introduced after cooperative negotiations among conflicting PSU pairs. This constraint pertains to each flight's coordinated entry/exit time in shared corridors between conflicted PSU pairs. Furthermore, to make bi-level optimization feasible, the objective function is modified in distributed AAM traffic management, and the constraint from Eq. 6.12 is relaxed. If a feasible solution cannot be found, the modified optimization and relaxed constraint may choose to depart the AAM flight earlier than the scheduled

departure time. This occurs when the vehicle cannot accelerate beyond its ideal cruise speed to meet the coordinated entry/exit time. The objective function penalizes these early departures more heavily to minimize their occurrence. The modified objective function and new constraint are shown in Eq. 6.19-6.20. Algorithm 10 shows bi-level optimization for distributed AAM traffic management.

$$\begin{aligned} \text{Minimize} \quad & \sum_{f\in\mathcal{F}} \Bigg\{ \sum_{t\in\mathcal{O}} \left(\gamma \cdot s_f \cdot (t-a_f)^{1+\epsilon} \cdot (w_{f,t}^{arrival} - w_{f,t-1}^{arrival})\right) \\ & - \sum_{t\in\mathcal{O}} \Big((\gamma-1) \cdot s_f \cdot (t-d_f) \cdot (w_{f,t}^{departure} - w_{f,t-1}^{departure}) \Big) \Bigg\} \end{aligned} \tag{6.19}$$

subject to additional constraint:

$$w_{f,b_{f,i}}^{\mathcal{P}_f} + w_{f,b_{f,i}-1}^{\mathcal{P}_f} == 1 \tag{6.20}$$

The bi-level optimization yields sub-optimal solutions compared to the centralized MIP approach, as each conflicted PSU pair coordinates its solutions to resolve conflicts. Nonetheless, the objective cost for each PSU will be lower than that of centralized AAM traffic management, as each PSU manages fewer vehicles than the centralized system. Additionally, the computation time for finding solutions will also be reduced in each PSU. Section 6.5 provides a more detailed examination of objective costs and runtime.

Algorithm 10: Bi-level Optimization for Distributed AAM Traffic Management

Data: ∀ AAM Flight Paths, Vertiports and Corridors ∈ Each PSU
Result: Sub-optimal Coordinated Entry/Exit Time of Flights in Each PSU
PSU_airspace_partition(∀ Flight_Paths);
for *each PSU* **do**
  flights_each_PSU ← *solve_MIP_eq.*(6.6);
mismatched_PSUs ← *find_mismatching_entry/exit_times*(∀ flights_each_PSU);
for *mismatched PSU pairs* **do**
  for *flight* **do**
    *solve*_eq.(6.16);
    negotiated_flight_entry/exit_time ← *solve*_alg.(9);
  save(coordinated_flights)
if *any[violate_speedConstraints(coordinated_flights)]* **then**
  adjust_entry/exit_time(constraint_violated_flight) ← *relax_eq.(6.12)*
for *each PSU* **do**
  coordinated_flights_each_PSU ← *solve_MIP_eq.*(6.19);

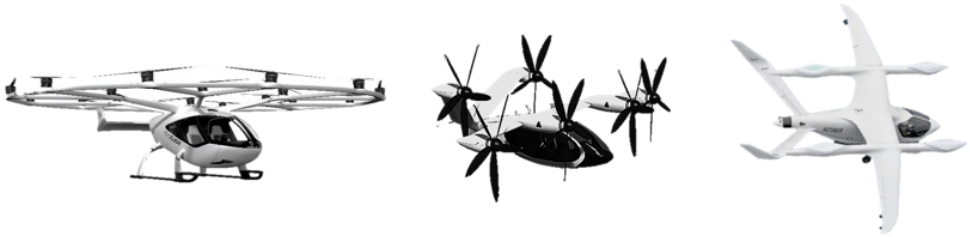
6.4 Simulation Setup

The simulation is tested using an artificially created map. Population density is randomly assigned to each town/ city, and vertiports are constructed to reflect AAM demand [158]. No-fly zone with keep-out geofence is constructed. The maximum distance of a single flight corridor is limited to 60 km, matching the minimum AAM operational range (i.e., multicopter range). The parameters for constructing the artificial map and AAM flight operations are summarized in Table 6.6. Here, vehicle types 1, 2, 3 correspond to Volocity (i.e., multicopter), Joby Aviation (i.e., vectored thrust), and Beta Technologies (i.e., lift + cruise) respectively [159, 160, 161], as shown in Table 6.7. Service priority types 1, 2, 3 refer to regular, express, and medical AAM flight operation, respectively. The geofence width of directional corridor is determined based on [3]. Their study calculates statistically optimal trajectory geofence buffer sizes using guidance, navigation, and control (GNC) and computational fluid dynamics (CFD) wind simulations in an urban environment. The operation time window is set to 4 hours (i.e., from 7 am to 11 am to simulate the morning commute). During the operation time, individual AAM scheduled departure times are randomly generated. The actual simultaneous take-off and landing are constrained by the concurrent take-off and landing (TLOF) capacity at each vertiport. Otherwise, there are 5-minute intervals between scheduled departure times for AAM flights. Finally, we set the number of AAM flights in each vehicle type and corresponding service priority percentage distribution the same. This allows us to analyze how optimization solutions affect each vehicle type and service priority type in both centralized and distributed AAM traffic management settings.

Artificial Map Construction Parameters	
Min/ Max Town Population	26/ 967
Concurrent TLOF Capacity	5 ~ 15
Directional Corridor Max Length	60 km
Directional Corridor Geofence Width	50 m
PSU Sectorization Weights $\alpha_1, \alpha_2, \alpha_3, \alpha_4$	[0.55, 0.25, 0.1, 0.1]

AAM Flight Operation Parameters	
Operation Time Window	4 hr
Scheduled Departure Time Interval	5 min
t_s	30 sec
Vehicle Type Distribution Ratio	Type 1, 2, 3: [1/3, 1/3, 1/3]
Service Priority Percentage per Vehicle Type	Type 1, 2, 3: [50%, 40%, 10%]
PSU Bargain Power Weight $\beta_1, \beta_2, \beta_3$	[0.3, 0.45, 0.25]

Table 6.6: Simulation Parameters.



	Volocity	Joby Aviation	Beta Technologies
Range [km]	35 ~ 65	240	500
Min. Cruise Speed* [km/hr]	30	100	100
Ideal Cruise Speed [km/hr]	90	320	270
Seating Capacity	2	4	4

Table 6.7: AAM Operator Specifications. Minimum cruise speed* is arbitrarily chosen for the simulation.

Figure 6.10 visualizes the simulation map environment with PSU airspace sectorization. In this figure, vertiports are indicated by numbering with blue circles. Flight corridors are shown in gray lines, forming the AAM corridor network. Corridor transition points between PSUs are indicated by numbering with orange circles. Distributed PSU airspace regions are indicated with purple, blue, yellow and green tints. In the simulation, the weight ratio (γ) of airborne delay to departure delay is set to 2 for all vehicle types. This helps analyze how the distributed AAM traffic management algorithm generates the optimal solution incorporating different ranges and speeds of vehicles. However, in reality, the delay weight ratio will vary by vehicle type. Below are the assumptions for the Monte Carlo simulation:

- Hovering is excluded from AAM traffic management due to its energy inefficiency. Only departure delay (ground delay) and airborne delay (adjusting cruise speed) are considered.
- Each AAM flight operates as a one-way trip; multiple trips for each flight are not taken into account.
- Concurrent TLOF capacity at each vertiport and maximum throughput capacity at each corridor remain constant throughout the operation time window.
- The cruise altitude for all AAM vehicle types is set to 1 km.

- AAM flight information is shared knowledge (i.e., scheduled departure/arrival time, departure/destination vertiports), meaning it is accessible to adjacent PSUs for distributed traffic management.

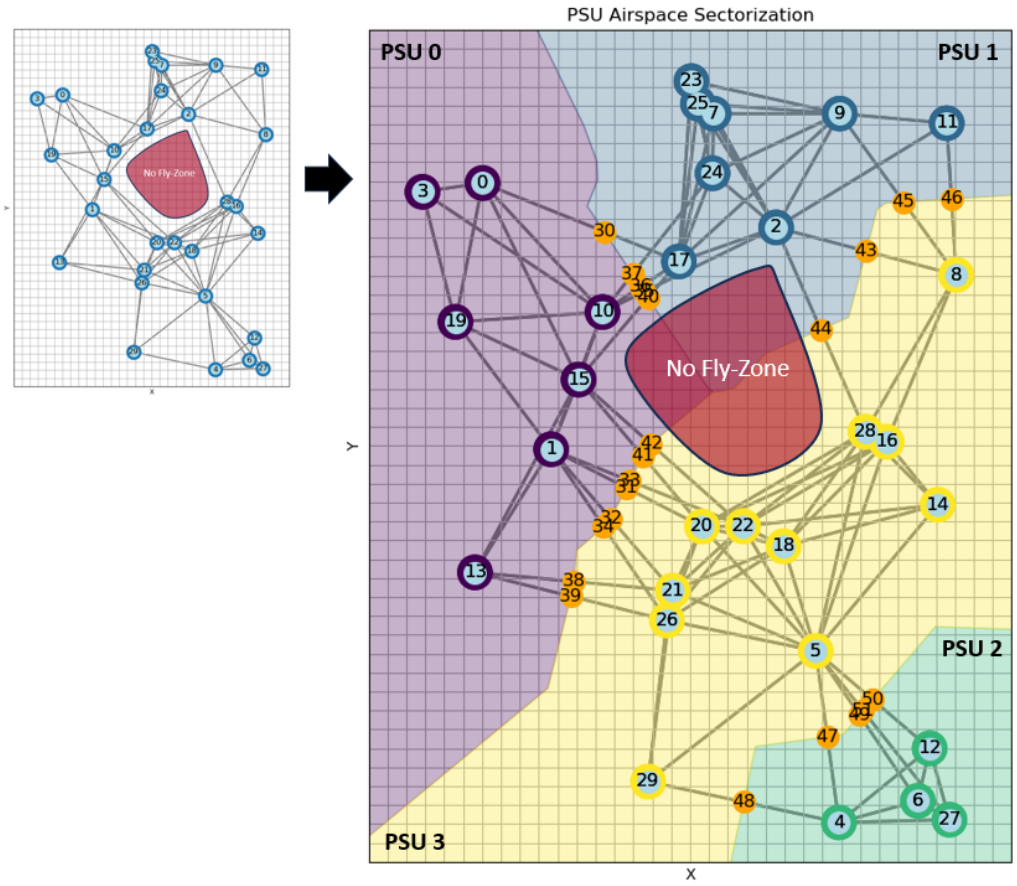


Figure 6.10: Construction of artificial map with airspace sectorization. [grid size: 5 km]

6.5 Simulation Analysis

To evaluate AAM traffic flow management, 150 and 300 AAM flight operations are generated in the artificially created map with the above assumptions and parameters. A total of 150 Monte Carlo simulations are conducted for each scenario: 1. 150 AAM flight operations with distance-based corridor routes, 2. 150 AAM flight operations with weighted/optimized corridor routes, 3. 300 AAM flight operations with distance-based corridor routes, and 4. 300 AAM flight operations with weighted/optimized corridor routes. The performance of both centralized and distributed PSU environments are examined and analyzed for comparison in terms of runtime, objective costs, and departure and airborne delay.

Figure 6.11 compares the percentages of delayed flights for 150 and 300 flights. Mean and standard deviation values were calculated for both centralized and distributed settings based on 150 simulations of optimized/weighted corridor routes each. The error bars represent the average plus or minus one standard deviation. In the distributed AAM traffic management scenario, there was an average increase of less than 1.05% in the number of delayed flights for both 150 and 300 flights compared to the centralized system. As anticipated, the percentages of delayed flights were higher for 300 flights compared to 150 flights in both centralized and distributed systems. Figure 6.12 presents the comparison of average delays by vehicle type and service priority type for both 150 and 300 flights in both centralized and distributed PSU systems. Across all vehicle types, 300 flights experienced more delays than 150 flights in both centralized and distributed systems. In the distributed system, vectored thrust and lift + cruise vehicle configurations exhibited greater average delays compared to multicopter. This is attributed to their extended ranges, increasing the likelihood of flight routes traversing multiple PSUs and resulting in increased average delays in departure and airborne delay to negotiate entry/exit times at transitioning PSUs. Regarding service priority types, 150 flights in the centralized system indicated that express service (i.e., second highest priority) experienced the least average delay, followed by medical and regular, respectively. For 300 flights, medical service (i.e., highest priority) experienced the least average delay, followed by express and regular services, respectively. However, the trend was consistent in the distributed system. For both 150 and 300 flights, regular service experienced the highest average delay, followed by express and medical services. In our simulation, we assigned weights of 1, 2, 3 to regular, express, and medical service priority types, respectively. The results suggest that the weight of medical service should be significantly increased relative to the other two types in both centralized and distributed systems. That will ensure that medical service consistently has the least departure and airborne delay

The runtime and objective cost comparisons are shown in Figure 6.13 for both 150 and 300 flights across centralized and distributed systems. The average runtime shows that centralized system has the longest duration for both 150 and 300 flights. This is because the distributed nature of individual PSUs allows them to manage a fraction of the total AAM flights handled by the centralized system. Notably, in the case of 300 AAM flight operations, individual PSUs generated coordinated AAM traffic solutions between 1.4 to 30 times faster compared to the time required for centralized management to generate a global solution. A similar trend was observed in objective costs, with individual PSUs having significantly lower costs compared to the centralized system. Furthermore, among the distributed PSUs, PSU 3 had the highest runtime and objective cost. This outcome can be attributed to the substantial number of vehicles operated within PSU 3, along with its extensive corridor

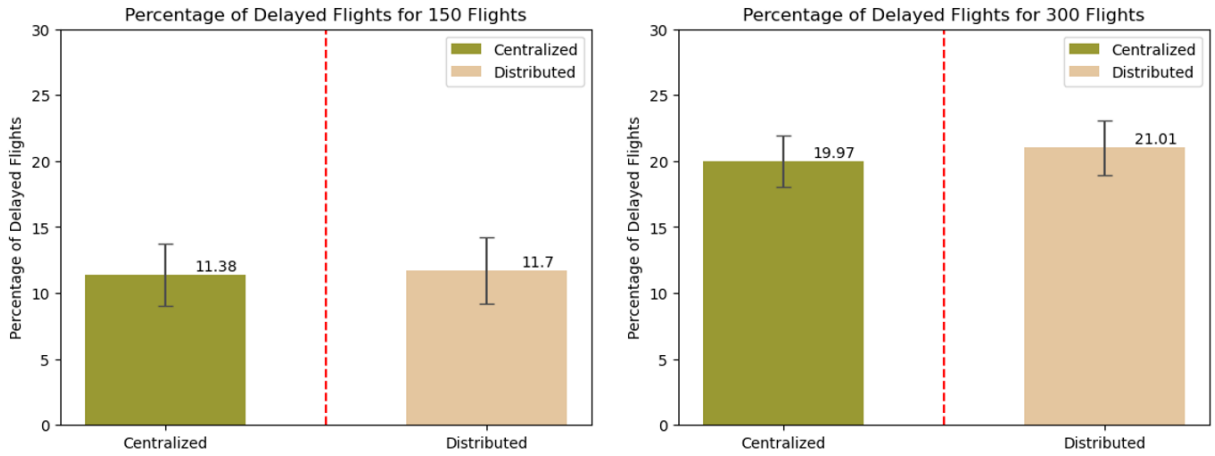


Figure 6.11: Comparison of Delayed Flight Percentages: 150 vs. 300 Flights.

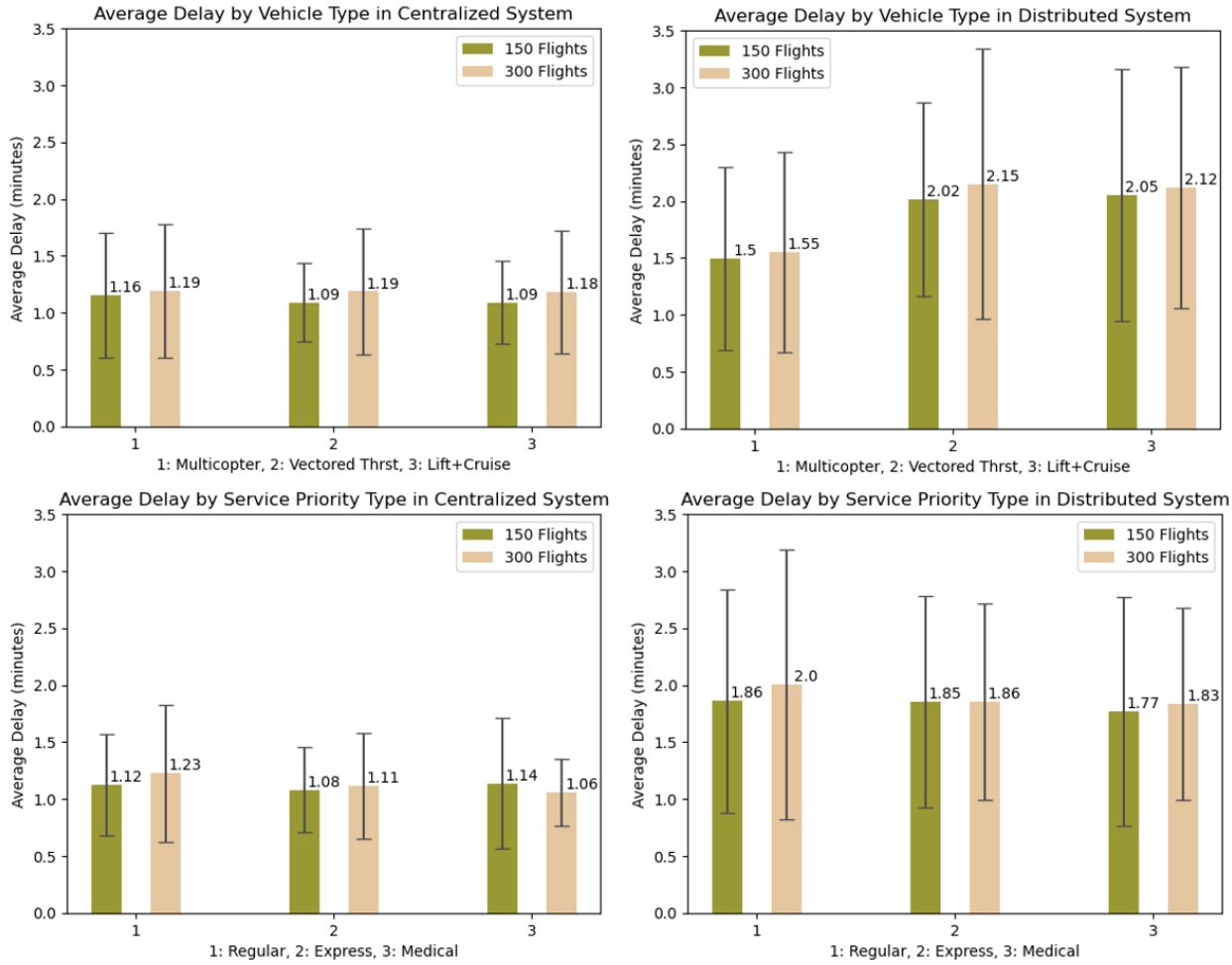


Figure 6.12: Comparison of Average Delays by Vehicle Type and Service Priority: 150 vs. 300 Flights.

network and a higher frequency of spatial corridor conflicts. Refer to Figure 6.10 that shows the PSU 3 corridor network. These findings show the critical influence of factors such as the volume of AAM vehicle operations, corridor network size, and the occurrence of spatial corridor conflicts on both runtime and objective costs. The comparison of average ground and airborne delays for 150 and 300 flights is illustrated in Figure 6.14. Across both centralized and distributed systems, increasing ground and airborne delays are observed with a higher volume of vehicles. Furthermore, the distributed system resulted sub-optimal solutions, but the total delays incurred by each PSU are not significantly higher compared to the centralized system.

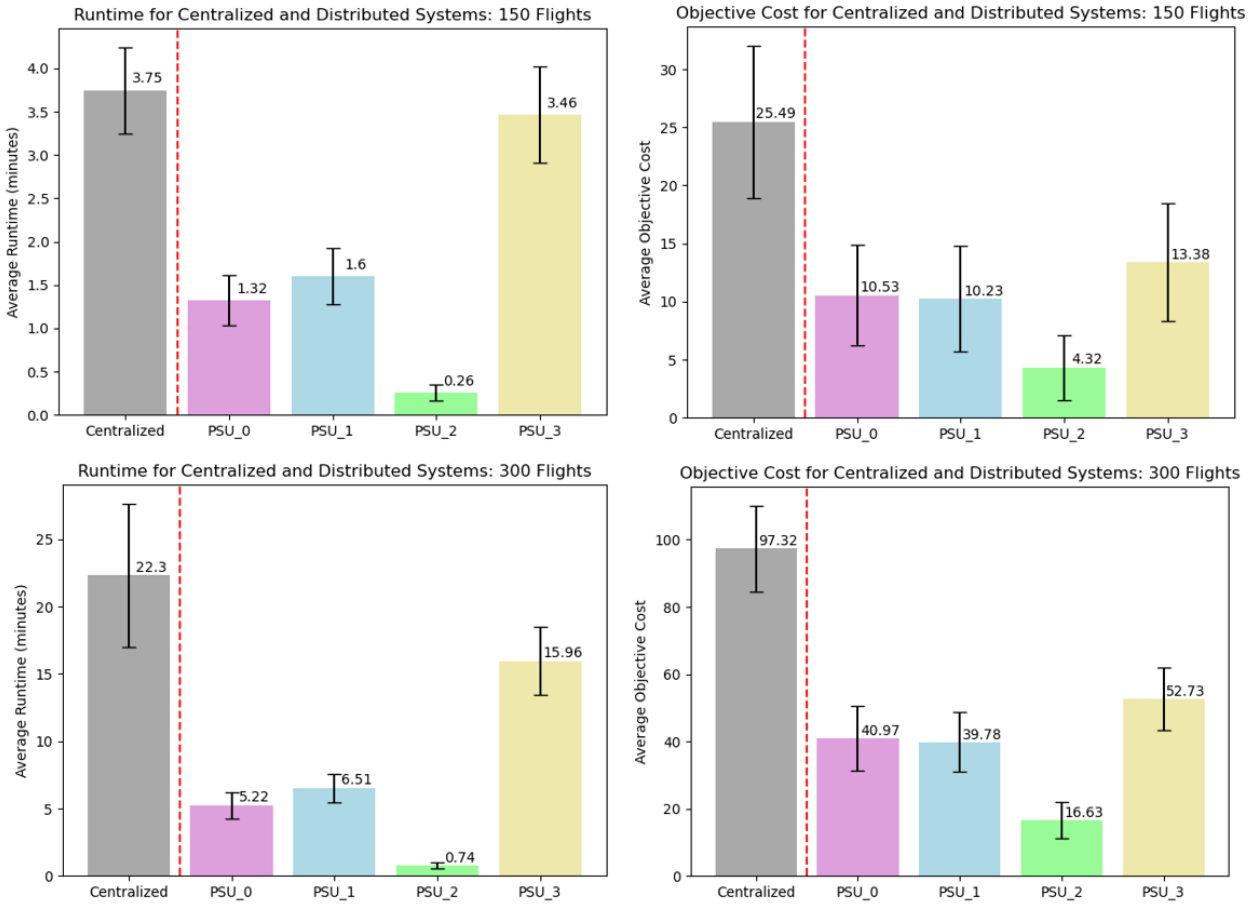


Figure 6.13: Runtime and Objective Cost Comparisons: 150 vs. 300 Flights.

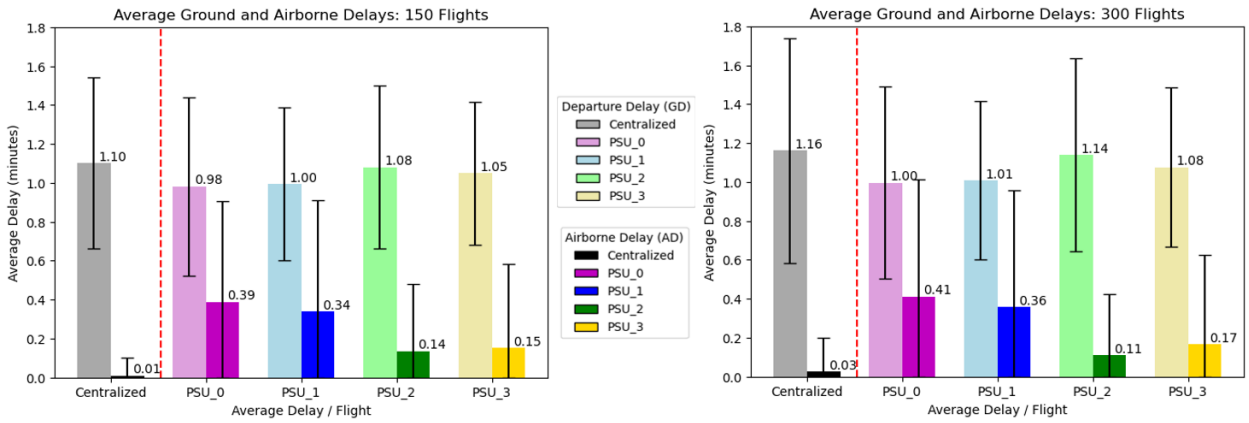


Figure 6.14: Average Ground and Airborne Delay Comparisons: 150 vs. 300 Flights.

Figure 6.15 illustrates the comparison between distance-based and weighted/optimized corridor routes for both 150 and 300 flights. The comparison is based on the average count of total delay duration for delayed AAM flights. In both scenarios, the weighted/optimized route method resulted in slightly fewer spatial conflicts in PSU 0, PSU 1, and PSU 2. However, PSU 3 showed significantly higher spatial conflicts in the weighted/optimized corridor route method for both 150 and 300 flights. This may be attributed to the fact that as more AAM flights seek to avoid congested areas due to highly connected vertiports and high TLOF vertiports, the number of spatial conflicts inadvertently increases as vehicles are routed to detour around these areas, leading to more conflicts among themselves. Consequently, our weighted/optimized corridor route did not outperform the distance-based route method. However, we found that the weighted/optimized corridor route method minimizes the number of PSU transitions, thereby reducing the amount of coordination required between PSUs compared to the distance-based method. This finding underscores the potential of the weighted/optimized corridor route method. This highlights the need to explore the selection between the two route planning methods in congested areas to further reduce flight delays beyond solely relying on the distance-based route method. Lastly, Figure 6.16 presents the runtime comparison for varying numbers of flights in the centralized system. The computation time demonstrates a cubic increase, as indicated by the curve-fitted dash line. This provides insights into the algorithm's scalability. For instance, the equation predicts that coordinating 1,000 flights in the centralized system would require approximately 11 hours to compute the solution. This underscores the minimum lead time required for AAM operators to submit their scheduled AAM flights in advance, facilitating efficient coordination within each PSU.

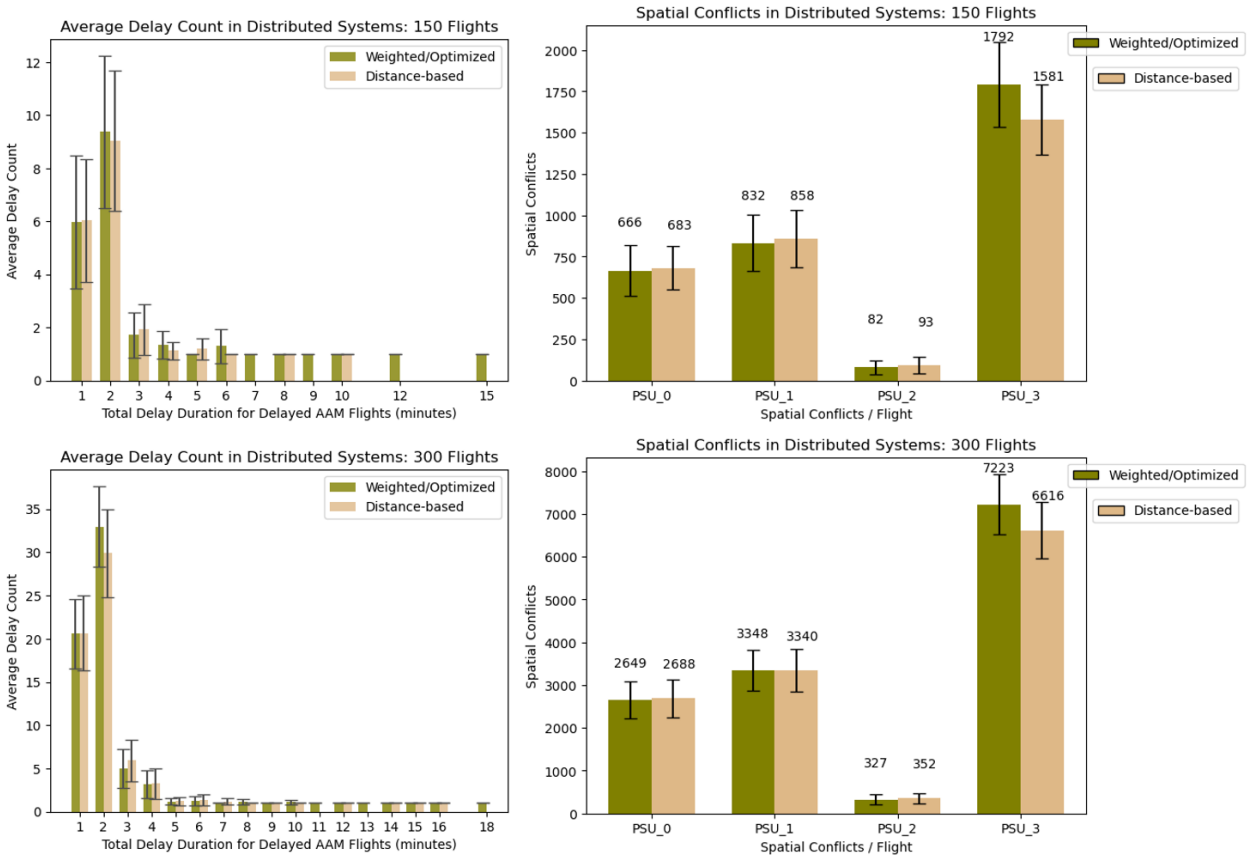


Figure 6.15: Route Method Comparisons: Distance-based vs. Weighted/Optimized Paths.

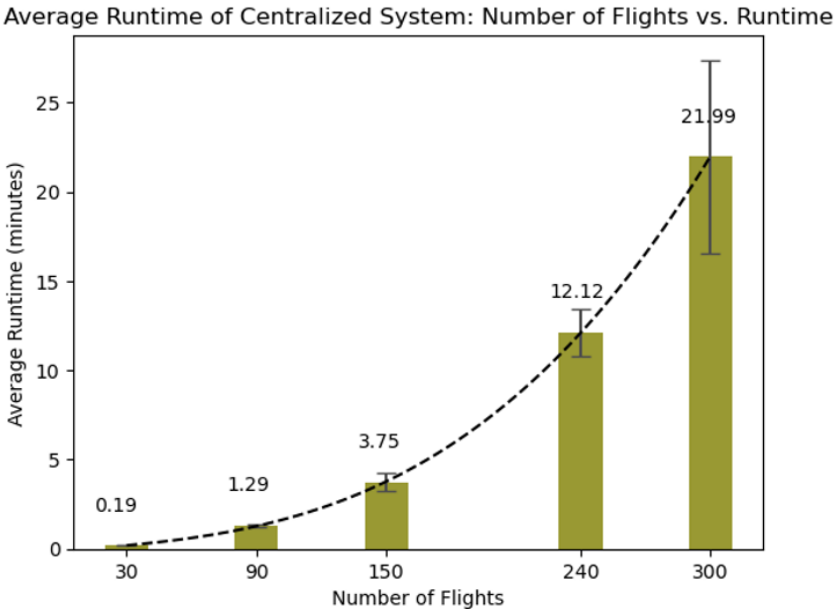


Figure 6.16: Runtime vs. Number of Flights.

6.6 Conclusions and Future Work

Our research offers solutions for both centralized and distributed AAM traffic management, taking into account the diverse stakeholders involved in AAM architecture. The contributions of this chapter are as follows: 1. a method to effectively sectorize low-altitude urban airspace is developed. 2. centrally planning AAM routes is modeled considering the limited capacities of corridors and vertiports. We compare two different corridor route planning methods and discuss how those methods can be combined for re-routing method. 3. A bi-level optimization architecture is formulated, solving distributed AAM traffic management. This involves MIP and cooperative game theoretic approach. We use an artificially created map to conduct Monte Carlo simulations, evaluating 150 and 300 AAM flight operations across three vehicle configurations and three service priority types.

Our research provides valuable insights into the performance of centralized versus distributed AAM network management strategies. We found that the centralized system consistently exhibits longer runtime for both 150 and 300 flights compared to individual PSUs within the distributed system (i.e. 1.4 to 30 times longer). Our analysis indicates that the distributed system may lead to sub-optimal solutions, but the total delays incurred by each PSU are not significantly higher compared to the centralized system. These findings underscore the importance and potential of scalable distributed AAM traffic management strategies in addressing the evolving demands of urban airspace. By providing a comprehensive technical framework, our research informs decision-making processes in the development and implementation of AAM traffic management strategies.

In future work, the stochastic optimization approach will be explored to address uncertainties in estimated times of arrivals, as well as robust optimization concerning the worst-case scenarios. Additionally, we plan to incorporate real map data with demand forecasts based on geography, population, and general aviation traffic to enhance the accuracy of AAM traffic management systems. By introducing auction mechanisms, we plan to incorporate the payment systems of each AAM vehicle operator within each PSU. Simulation analysis can be further expanded to investigate the impact of different weight ratios (γ) for airborne and departure delays across various AAM vehicle types, as well as the impact of time-dependent TLOF vertiport capacities. Lastly, we will explore rerouting strategies and implement dynamic airspace sectorization for PSUs to further optimize AAM traffic management.

CHAPTER 7

Conclusion

7.1 Conclusion

In the history of aviation and transportation, significant milestones have transformed how people travel. The advent of AAM heralds another such milestone, promising to revolutionize our daily commute and provide unprecedented convenience and flexibility in airborne package delivery and data collection. This dissertation is dedicated to addressing future challenges in AAM to contribute to the realization of safe, efficient, and robust AAM operations.

The operational landscape of AAM differs significantly from both commercial/general aviation and ground transportation. AAM vehicles will navigate low-altitude airspace with vertical take-off and landing capabilities, operating at speeds faster than ground vehicles and with a higher frequency of take-offs and landings than commercial/ general aviation. With vertiports located within cities, safety, efficiency, and resilience must be paramount considerations in the early stages of AAM architecture. To this end, our focus has been on developing geofencing algorithms, contingency landing management systems, and AAM traffic management architectures.

Geofencing algorithms play a crucial role in managing low-altitude airspace by separating flyable and no-fly zones. Our algorithms construct static and durational geofences for both obstacles and flight trajectories, providing solutions for vehicle path planning and first-come, first-served strategic deconfliction. Five flight planning solutions are generated for each vehicle (i.e., constant cruise, terrain follower, go-around obstacles, wait until durational obstacle disappears, and a combination of go-around and constant cruise). Then, computational geometry and polygon decimation processes are employed to rapidly generate conflict-free geofence map data. Our algorithms find flight planning solutions with minimum energy and distance costs for each vehicle flying in low-altitude urban airspace. Our research addresses the optimization of vehicle-specific flight trajectory geofence sizing and the development of spatially efficient geofencing algorithms for climb and descent maneuvers, as well as for

multi-agent UAS teams or swarms. We use vehicle dynamics, sensor characteristics, and wind models to generate statistically-guided geofence sizes with a three-sigma safety level on a full-size AAM aircraft model. A parallelepiped geofence for climb/descent and containment geofences for multi-agent UAS teams further minimize geofence volumes. Based on our Monte Carlo simulation results, we conclude that direct geofenced flight routes have shorter lengths than fixed corridor routes. Geofencing can safely protect aircraft in both direct routing and corridor based airspace designs.

Safety is paramount in AAM operations, necessitating robust contingency management systems. Despite the multiple layers of redundancy and safety features in AAM vehicles, unforeseen off-nominal conditions can arise unexpectedly. Therefore, the development of a contingency landing management system, such as our Assured Contingency Landing Management (ACLM) system, is imperative. ACLM integrates mathematically-provable controllability and reachability logic to rapidly identify when a new landing solution is needed. Contingency landing plans generated along the planned flight route during preflight preparation minimize the response time. ACLM references a landing site database with prepared and unprepared landing site options. A hexacopter simulation case study with rotor and/or battery degradation scenarios illustrates ACLM functionality for low-altitude flight in Manhattan, New York City. Assurance is based on use of mathematically sound algorithms plus trusted map and model information. ACLM executes in milliseconds for our case studies thanks to the use of pre-planned data and parallel threading.

From a network perspective, efficient management of AAM operations is critical due to the projected increase in flight volume, surpassing traditional air traffic control capacities globally. Without a dedicated AAM traffic management architecture in place, significant departure and airborne delays are inevitable. Vertiport limitations in handling lift-offs, touch-downs coupled with corridor throughput constraints exacerbate the challenge. Safety concerns arise from spatial conflicts among AAM flights within designated corridors. Our research addresses these challenges comprehensively by offering solutions for both centralized and distributed AAM traffic management. We optimize low-altitude urban airspace by sectorizing it and designing vehicle-type and service-priority-aware traffic management systems, accounting infrastructure constraint and equity. Additionally, a distributed AAM traffic management framework is formulated, incorporating cooperative game theory and MIP. Our findings provide insight into the performance disparities between centralized and distributed AAM network management strategies. We believe our research can inform decision-making processes in the development of effective AAM traffic management strategies.

7.2 Future Directions

For future work, there are several directions for further analysis and development based on our research findings. First, geofence algorithms can be extended to generate databases for varying wind conditions across different city models, considering factors such as building layouts and terrains. Investigating the influence of different wind conditions and vehicle types on geofence buffer sizes would provide valuable insights into optimizing low-altitude airspace.

Our proposed ACLM architecture is tailored for multicopter systems with more than four rotors. ACLM can be further adapted for other vehicle configurations, such as a lift + cruise configuration, which may have distinct operational constraints. Integrating ACLM with visual sensor data could enhance the robustness of ACLM. Furthermore, sampling-based path planning methods can be explored for alternative real-time contingency path planning, as such methods are known for effectiveness in dynamic environments.

In the domain of AAM traffic management, exploring stochastic optimization approaches can address uncertainties in estimated arrival times. Also, facilitating robust optimization will mitigate worst-case scenarios. Furthermore, integrating real map data with demand forecasts based on geographical, demographic, and general aviation traffic factors would enhance the accuracy of AAM traffic management systems. Finally, incorporating payment systems for individual AAM vehicle operators into AAM traffic management research could introduce economic factors into the design of AAM traffic management strategies such as pricing mechanisms to optimize airspace utilization and incentivize efficient flight behaviors.

These directions of future research hold the potential to further advance the safety, efficiency, and scalability of AAM operations in urban environments.

BIBLIOGRAPHY

[1] NASA. NASA UAM Concept of Operations. https://www.nasa.gov/directorates/armd/aosp/amp/, 2023.

[2] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[3] Joseph Kim and Ella Atkins. Airspace geofencing and flight planning for low-altitude, urban, small unmanned aircraft systems. *Applied Sciences*, 12(2):576, 2022.

[4] Jeremy Castagno, Cosme Ochoa, and Ella Atkins. Comprehensive risk-based planning for small unmanned aircraft system rooftop landing. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1031–1040. IEEE, 2018.

[5] Yuneec. Yuneec Typhoon Hexacopter. https://yuneec.online/typhoon-h-plus/, 2023.

[6] FAA-NextGen. A New U.S. DOT Volpe Center-FAA Thought Leadership Series. Transformation: Urban Air Mobility Concept of Operations. https://www.volpe.dot.gov/events/transformation-urban-air-mobility-concept-operations, 2023.

[7] Varon. Varon Vehicles: UAM Concept. https://varon.aero/concept/, 2023.

[8] SMG-Consulting. AAM Reality Index: Vehicle Types. https://aamrealityindex.com/aam-reality-index, 2023.

[9] Mckinsey and company. AAM Reality Index: Vehicle Types. https://www.mckinsey.com/~/media/mckinsey/industries/aerospace%20and%20defense/our%20insights/perspectives%20on%20advanced%20air%20mobility/airmobilitypdf.pdf, 2022.

[10] Akshay Mathur and Ella M Atkins. Design, modeling and hybrid control of a quad-plane. In *AIAA Scitech 2021 Forum*, page 0374, 2021.

[11] DIVYA Joshi. Drone technology uses and applications for commercial, industrial and military drones in 2020 and the future. *Dec*, 18(2019):7, 2019.

[12] Malik Doole, Joost Ellerbroek, and Jacco Hoekstra. Estimation of traffic density from drone-based delivery in very low level urban airspace. *Journal of Air Transport Management*, 88:101862, 2020.

[13] Madhukar Mayakonda, Cedric Y Justin, Akshay Anand, Colby J Weit, Jiajie Wen, Turab Zaidi, and Dimitri Mavris. A top-down methodology for global urban air mobility demand estimation. In *AIAA AVIATION 2020 FORUM*, page 3255, 2020.

[14] Jonathan Gomez. 7 Urban Air Mobility Companies to Watch. https://www.greenbiz.com/article/7-urban-air-mobility-companies-watch#:~:text=With%20about%20200%20companies%20involved,as%20the%20end%20of%202022., 2021.

[15] George Price, Douglas Helton, Kyle Jenkins, Mike Kvicala, Steve Parker, Russell Wolfe, Felix A Miranda, Kenneth H Goodrich, Min Xue, Karen Tung Cate, et al. Urban air mobility operational concept (opscon) passenger-carrying operations. 2020.

[16] Kenneth H Goodrich and Colin R Theodore. Description of the nasa urban air mobility maturity level (uml) scale. In *AIAA Scitech 2021 forum*, page 1627, 2021.

[17] Federal Aviation Administration. Advanced Air Mobility Implementation Plan. https://www.faa.gov/sites/faa.gov/files/AAM-I28-Implementation-Plan.pdf, 2023.

[18] Mia N Stevens and Ella M Atkins. Generating airspace geofence boundary layers in wind. *Journal of Aerospace Information Systems*, 17(2):113–124, 2020.

[19] Mia Stevens and Ella Atkins. Geofence definition and deconfliction for uas traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):5880–5889, 2021.

[20] Akshay Mathur, Karanvir Panesar, Joseph Kim, Ella M Atkins, and Nadine Sarter. Paths to autonomous vehicle operations for urban air mobility. In *AIAA Aviation 2019 Forum*, page 3255, 2019.

[21] FAA_D. Press release—us department of transportation issues two much-anticipated drone rules to advance safety and innovation in the united states.

[22] Parker D Vascik and R John Hansman. Constraint identification in on-demand mobility for aviation through an exploratory case study of los angeles. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3083, 2017.

[23] Christabelle Bosson and Todd A Lauderdale. Simulation evaluations of an autonomous urban air mobility network management and separation service. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3365, 2018.

[24] Minghong G Wu, Andrew C Cone, Seungman Lee, Christine Chen, Matthew W Edwards, and Devin P Jack. Well clear trade study for unmanned aircraft system detect and avoid with non-cooperative aircraft. In *2018 Aviation Technology, Integration, and Operations Conference*, page 2876, 2018.

[25] Parimal Kopardekar, Joseph Rios, Thomas Prevot, Marcus Johnson, Jaewoo Jung, and John E Robinson. Unmanned aircraft system traffic management (utm) concept of operations. In *AIAA Aviation and Aeronautics Forum (Aviation 2016)*, number ARC-E-DAA-TN32838, 2016.

[26] Tao Jiang, Jared Geller, Daiheng Ni, and John Collura. Unmanned aircraft system traffic management: Concept of operation and system architecture. *International journal of transportation science and technology*, 5(3):123–135, 2016.

[27] David P Thipphavong, Rafael Apaza, Bryan Barmore, Vernol Battiste, Barbara Burian, Quang Dao, Michael Feary, Susie Go, Kenneth H Goodrich, Jeffrey Homola, et al. Urban air mobility airspace integration concepts and considerations. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3676, 2018.

[28] Aleksandar Bauranov and Jasenka Rakas. Designing airspace for urban air mobility: A review of concepts and approaches. *Progress in Aerospace Sciences*, 125:100726, 2021.

[29] SESAR. U-space Concept of Operations. https://www.sesarju.eu/sites/default/files/documents/u-space/CORUS%20ConOps%20vol2.pdf, 2019.

[30] Cristina Barrado, Mario Boyero, Luigi Brucculeri, Giancarlo Ferrara, Andrew Hately, Peter Hullah, David Martin-Marrero, Enric Pastor, Anthony Peter Rushton, and Andreas Volkert. U-space concept of operations: A key enabler for opening airspace to emerging low-altitude operations. *Aerospace*, 7(3):24, 2020.

[31] Nader Samir Labib, Grégoire Danoy, Jedrzej Musial, Matthias R Brust, and Pascal Bouvry. Internet of unmanned aerial vehicles—a multilayer low-altitude airspace model for distributed uav traffic management. *Sensors*, 19(21):4779, 2019.

[32] Qixi Fu, Xiaolong Liang, Jiaqiang Zhang, Duo Qi, and Xiujun Zhang. A geofence algorithm for autonomous flight unmanned aircraft system. In *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pages 65–69. IEEE, 2019.

[33] Mia N Stevens, Hossein Rastgoftar, and Ella M Atkins. Geofence boundary violation detection in 3d using triangle weight characterization with adjacency. *Journal of Intelligent & Robotic Systems*, 95(1):239–250, 2019.

[34] Guodong Zhu and Peng Wei. Low-altitude uas traffic coordination with dynamic geofencing. In *16th AIAA aviation technology, integration, and operations conference*, page 3453, 2016.

[35] Elie Hermand, Tam W Nguyen, Mehdi Hosseinzadeh, and Emanuele Garone. Constrained control of uavs in geofencing applications. In *2018 26th Mediterranean Conference on Control and Automation (MED)*, pages 217–222. IEEE, 2018.

[36] Marcus Johnson, Jaewoo Jung, Joseph Rios, Joey Mercer, Jeffrey Homola, Thomas Prevot, Daniel Mulfinger, and Parimal Kopardekar. Flight test evaluation of an unmanned aircraft system traffic management (utm) concept for multiple beyond-visual-line-of-sight operations. In *12th USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)*, 2017.

[37] Evan T Dill, Steven D Young, and Kelly J Hayhurst. Safeguard: An assured safety net technology for uas. In *2016 IEEE/AIAA 35th digital avionics systems conference (DASC)*, pages 1–10. IEEE, 2016.

[38] Joseph T Kim, Akshay Mathur, Nicholas Liberko, and Ella Atkins. Volumization and inverse volumization for low-altitude airspace geofencing. In *AIAA AVIATION 2021 FORUM*, page 2383, 2021.

[39] Thomas Prevot, Joseph Rios, Parimal Kopardekar, John E Robinson III, Marcus Johnson, and Jaewoo Jung. Uas traffic management (utm) concept of operations to safely enable low altitude flight operations. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016.

[40] JU SESAR. European drones outlook study unlocking the value for europe. *Siebert, JU, Nov*, 2016.

[41] Leonid Sedov and Valentin Polishchuk. Centralized and distributed utm in layered airspace. In *8th International Conference on Research in Air Transportation*, pages 1–8, 2018.

[42] Jungwoo Cho and Yoonjin Yoon. How to assess the capacity of urban airspace: A topological approach using keep-in and keep-out geofence. *Transportation Research Part C: Emerging Technologies*, 92:137–149, 2018.

[43] Tamraparni Dasu, Yaron Kanza, and Divesh Srivastava. Geofences in the sky: herding drones with blockchains and 5g. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 73–76, 2018.

[44] Hansol Yoon, Yi Chou, Xin Chen, Eric Frew, and Sriram Sankaranarayanan. Predictive runtime monitoring for linear stochastic systems and applications to geofence enforcement for uavs. In *International Conference on Runtime Verification*, pages 349–367. Springer, 2019.

[45] Mia N Stevens, Hossein Rastgoftar, and Ella M Atkins. Specification and evaluation of geofence boundary violation detection algorithms. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1588–1596. IEEE, 2017.

[46] Mica R Endsley. Design and evaluation for situation awareness enhancement. In *Proceedings of the Human Factors Society annual meeting*, volume 32, pages 97–101. Sage Publications Sage CA: Los Angeles, CA, 1988.

[47] Mica R Endsley and Mark D Rodgers. Situation awareness information requirements analysis for en route air traffic control. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 38, pages 71–75. SAGE Publications Sage CA: Los Angeles, CA, 1994.

[48] Savita A Verma, Spencer C Monheim, Kushal A Moolchandani, Priyank Pradeep, Annie W Cheng, David P Thipphavong, Victoria L Dulchinos, Heather Arneson, Todd A Lauderdale, Christabelle S Bosson, et al. Lessons learned: using utm paradigm for urban air mobility operations. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2020.

[49] Mia N Stevens and Ella M Atkins. Layered geofences in complex airspace environments. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3348, 2018.

[50] Mia N Stevens and Ella M Atkins. Multi-mode guidance for an independent multicopter geofencing system. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, page 3150, 2016.

[51] Kevin Weiler. Polygon comparison using a graph representation. In *Proceedings of the 7th Annual Conference on Computer Graphics and interactive Techniques*, pages 10–18, 1980.

[52] Jack Sklansky. Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, 1(2):79–83, 1982.

[53] Eric Haines. Point in polygon strategies. *Graphics Gems*, 4:24–46, 1994.

[54] Kai Hormann and Alexander Agathos. The point in polygon problem for arbitrary polygons. *Computational geometry*, 20(3):131–144, 2001.

[55] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59–69, 2014.

[56] Anthony Stentz. Optimal and efficient path planning for partially known environments. In *Intelligent unmanned ground vehicles*, pages 203–220. Springer, 1997.

[57] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 1997.

[58] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[59] Frank Lingelbach. Path planning using probabilistic cell decomposition. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 1, pages 467–472. IEEE, 2004.

[60] Yong Koo Hwang, Narendra Ahuja, et al. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32, 1992.

[61] Brendan Burns and Oliver Brock. Sampling-based motion planning using predictive models. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 3120–3125. IEEE, 2005.

[62] Anton Semechko. Decimate 2d contours/polygons, 2018. Available online: https://github.com/AntonSemechko/DecimatePoly (accessed on 3 Jan 2021).

[63] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[64] Han-Pang Huang and Shu-Yun Chung. Dynamic visibility graph for path planning. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2813–2818. IEEE, 2004.

[65] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 7(4):12–18, 2008.

[66] Cosme A Ochoa and Ella M Atkins. Urban metric maps for small unmanned aircraft systems motion planning. *arXiv preprint arXiv:2102.07218*, 2021.

[67] Muneendra Kumar. World geodetic system 1984: A modern and accurate global reference frame. *Marine Geodesy*, 12(2):117–126, 1988.

[68] John Hershberger and Subhash Suri. An optimal algorithm for euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

[69] Justin R Rufa. *Location-Based Sensor Fusion for UAS Urban Navigation.* PhD thesis, University of Michigan, 2014.

[70] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[71] Israel Lugo-Cárdenas, Gerardo Flores, Sergio Salazar, and Rogelio Lozano. Dubins path generation for a fixed wing uav. In *2014 International conference on unmanned aircraft systems (ICUAS)*, pages 339–346. IEEE, 2014.

[72] Thomas R Yechout. *Introduction to aircraft flight mechanics*. AIAA, 2003.

[73] Frank L Lewis and FL Lewis. *Optimal estimation: with an introduction to stochastic control theory*. Wiley New York, 1986.

[74] Derek R Nelson, D Blake Barber, Timothy W McLain, and Randal W Beard. Vector field path following for small unmanned air vehicles. In *2006 American Control Conference*, pages 7–14. IEEE, 2006.

[75] Randal W Beard. Embedded uas autopilot and sensor systems. In *Unmanned Aircraft Systems*, pages 231–248. Wiley, 2017.

[76] Steven R Hanna, Michael J Brown, Fernando E Camelli, Stevens T Chan, William J Coirier, Olav R Hansen, Alan H Huber, Sura Kim, and R Michael Reynolds. Detailed simulations of atmospheric flow and dispersion in downtown manhattan: An application of five computational fluid dynamics models. *Bulletin of the American Meteorological Society*, 87(12):1713–1726, 2006.

[77] Thomas Prevot, Joseph Rios, Parimal Kopardekar, John Robinson III, Marcus Johnson, and Jaewoo Jung. Uas traffic management (utm) concept of operations to safely enable low altitude flight operations. 06 2016.

[78] Christopher D Barkey, Joseph T Kim, and Ella M Atkins. Space efficient airspace geofence volume sizing. In *AIAA AVIATION 2023 Forum*, page 3859, 2023.

[79] Franco P. Preparata and Se June Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, 1977.

[80] Sesar Joint Undertaking. European drones outlook study - unlocking the value for europe. 2016.

[81] Nader Samir Labib, Grégoire Danoy, Jedrzej Musial, Matthias R. Brust, and Pascal Bouvry. Internet of unmanned aerial vehicles—a multilayer low-altitude airspace model for distributed uav traffic management. *Sensors*, 19(21), 2019.

[82] Marcus Johnson, Jaewoo Jung, Joseph Rios, Joey Mercer, Jeffrey Homola, Thomas Prevot, Daniel Mulfinger, and Parimal Kopardekar. Flight test evaluation of an unmanned aircraft system traffic management (utm) concept for multiple beyond-visual-line-of-sight operations. 06 2017.

[83] Savita Verma, Spencer Monheim, Kushal Moolchandani, Priyank Pradeep, Annie Cheng, David Thipphavong, Victoria Dulchinos, Heather Arneson, Todd Lauderdale, Christabelle Bosson, Eric Mueller, and Bogu Wei. Lessons learned: Using utm paradigm for urban air mobility operations. pages 1–10, 10 2020.

[84] Tamraparni Dasu, Yaron Kanza, and Divesh Srivastava. Geofences in the sky: Herding drones with blockchains and 5g. 11 2018.

[85] Evan T. Dill, Russell V. Gilabert, and Seth S. Young. Safeguard. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–8, 2018.

[86] Giovanni Legnani, Federico Casolo, Paolo Righettini, and Bruno Zappa. A homogeneous matrix approach to 3d kinematics and dynamics—i. theory. *Mechanism and machine theory*, 31(5):573–587, 1996.

[87] Jeff Holden and Nikhil Goel. Fast-forwarding to a future of on-demand urban air transportation [white paper], 2016. Available online: https://www.uber.com/elevate.pdf/ (accessed on 17 June 2021).

[88] Matthew Osborne, Jennifer Lantair, Zain Shafiq, Xingyu Zhao, Valentin Robu, David Flynn, and John Perry. Uas operators safety and reliability survey: Emerging technologies towards the certification of autonomous uas. In *2019 4th International Conference on System Reliability and Safety (ICSRS)*, pages 203–212. IEEE, 2019.

[89] Prashin Sharma and Ella Atkins. Experimental investigation of tractor and pusher hexacopter performance. *Journal of Aircraft*, 56(5):1920–1934, 2019.

[90] Prashin Sharma and Ella Atkins. Prognostics-informed battery reconfiguration in a multi-battery small uas energy system. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 423–432. IEEE, 2021.

[91] Hafiz Zeeshan Iqbal Khan, Jahanzeb Rajput, and Jamshed Riaz. Reconfigurable control of a class of multicopters. In *2020 European Control Conference (ECC)*, pages 1632–1637. IEEE, 2020.

[92] Hussein Hamadi. *Fault-tolerant control of a multirotor unmanned aerial vehicle under hardware and software failures*. PhD thesis, Compiègne, 2020.

[93] Mo Chen and Claire J Tomlin. Hamilton–jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:333–358, 2018.

[94] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[95] Anayo K Akametalu, Claire J Tomlin, and Mo Chen. Reachability-based forced landing system. *Journal of Guidance, Control, and Dynamics*, 41(12):2529–2542, 2018.

[96] Pedro F. A. Di Donato, Sweewarman Balachandran, Kevin McDonough, Ella Atkins, and Ilya Kolmanovsky. Envelope-aware flight management for loss of control prevention given rudder jam. *Journal of Guidance, Control, and Dynamics*, 40(4):1027–1041, 2017.

[97] Ella M Atkins, Igor Alonso Portillo, and Matthew J Strube. Emergency flight planning applied to total loss of thrust. *Journal of aircraft*, 43(4):1205–1216, 2006.

[98] Pedro FA Di Donato and Ella M Atkins. Optimizing steady turns for gliding trajectories. *Journal of Guidance, Control, and Dynamics*, 39(12):2627–2637, 2016.

[99] Joshua E. Baculi and Corey A. Ippolito. *Onboard Decision-Making for Nominal and Contingency sUAS Flight*. AIAA, 2019.

[100] Parker C Lusk, Patricia C Glaab, Louis J Glaab, and Randal W Beard. Safe2ditch: Emergency landing for small unmanned aircraft systems. *Journal of Aerospace Information Systems*, 16(8):327–339, 2019.

[101] Sanjiban Choudhury, Vishal Dugar, Silvio Maeta, Brian MacAllister, Sankalp Arora, Daniel Althoff, and Sebastian Scherer. High performance and safe flight of full-scale helicopters from takeoff to landing with an ensemble of planners. *Journal of Field Robotics*, 36(8):1275–1332, 2019.

[102] Irene M Gregory, Newton H Campbell, Natasha A Neogi, Jon B Holbrook, Jared A Grauer, Barton J Bacon, Patrick C Murphy, Daniel D Moerder, Benjamin M Simmons, Michael J Acheson, et al. Intelligent contingency management for urban air mobility. In *Dynamic Data Driven Applications Systems: Third International Conference, DDDAS 2020, Boston, MA, USA, October 2-4, 2020, Proceedings 3*, pages 22–26. Springer, 2020.

[103] Hyeonwoong Lee, Seung-Hyun Park, Hak-Tae Lee, Bomi Park, and Jae-Hyun Han. Lost c2 link contingency procedures for seoul tma and assessment on safety and controller workload. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–6. IEEE, 2020.

[104] Prashin Sharma, Benjamin Kraske, Joseph Kim, Zakariya Laouar, Zachary Sunberg, and Ella Atkins. Investigation of risk-aware mdp and pomdp contingency management autonomy for uas. *arXiv preprint arXiv:2304.01052*, 2023.

[105] Prashin Sharma, Benjamin Kraske, Joseph Kim, Zakariya Laouar, Zachary Sunberg, and Ella Atkins. Risk-aware markov decision process contingency management autonomy for uncrewed aircraft systems. *Journal of Aerospace Information Systems*, pages 1–15, 2024.

[106] Joseph Kim, Prashin Sharma, Ella Atkins, Natasha Neogi, Evan Dill, and Steven Young. Assured contingency landing management for advanced air mobility. In *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, pages 1–12. IEEE, 2021.

[107] J Castagno and E Atkins. Map-based planning for small unmanned aircraft rooftop landing. In *Handbook of Reinforcement Learning and Control*, pages 613–646. Springer, 2021.

[108] Eduardo D Sontag. Kalman's controllability rank condition: from linear to nonlinear. In *Mathematical system theory*, pages 453–462. Springer, 1991.

[109] Quan Quan. *Introduction to multicopter design and control*. Springer, 2017.

[110] Robert F Brammer. Controllability in linear autonomous systems with positive controllers. *SIAM Journal on Control*, 10(2):339–353, 1972.

[111] Guang-Xun Du, Quan Quan, Binxian Yang, and Kai-Yuan Cai. Controllability analysis for multirotor helicopter rotor degradation and failure. *Journal of Guidance, Control, and Dynamics*, 38(5):978–985, 2015.

[112] Majd Saied, Hassan Shraim, Benjamin Lussier, Isabelle Fantoni, and Clovis Francis. Local controllability and attitude stabilization of multirotor uavs: Validation on a coaxial octorotor. *Robotics and Autonomous Systems*, 91:128–138, 2017.

[113] D Subbaram Naidu. *Optimal control systems*. CRC press, 2002.

[114] Angel Kirchev. Battery management and battery diagnostics. In *Electrochemical energy storage for renewable sources and grid balancing*, pages 411–435. Elsevier, 2015.

[115] Joseph Kim, Nicholas Liberko, and Ella Atkins. Airspace geofencing volume sizing with an advanced air mobility vehicle performance model. In *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, pages 1–8. IEEE, 2022.

[116] Justin R Rufa and Ella M Atkins. Unmanned aircraft system navigation in the urban environment: A systems analysis. *Journal of Aerospace Information Systems*, 13(4):143–160, 2016.

[117] Federal Aviation Administration. Urban Air Mobility (UAM), Concept of Operations. v2.0, US Department of Transportation. Office of Nextgen, 2023. https://www.faa.gov/sites/faa.gov/files/Urban%20Air%20Mobility%20%28UAM%29%20Concept%20of%20Operations%202.0_0.pdf/, 2023.

[118] Wisk. Concept of Operations for Uncrewed Urban Air Mobility. https://wisk.aero/wp-content/uploads/2022/09/Concept-of-Operations-for-Uncrewed-Urban-Air-Mobility.pdf, 2022.

[119] SESAR-Joint-Undertaking. European atm master plan: Roadmap for the safe integration of drones into all classes of airspace. *SESAR Joint Undertaking: Brussels, Belgium*, 2018.

[120] NASA. Air traffic management for low-altitude drones. *NASA: Washington, DC, USA*, 2018.

[121] D Geister and B Korn. Concept for urban airspace integration dlr u-space blueprint. *German Aerospace Center-Institut of Flight Guidance*, 2017.

[122] Claude Le Tallec, Patrick Le Blaye, and Moustafa Kasbari. Low level rpas traffic management (llrtm) concept of operation. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3938, 2017.

[123] Karthik Balakrishnan, Joe Polastre, Jessie Mooberry, Richard Golding, and Peter Sachs. Blueprint for the sky: The roadmap for the safe integration of autonomous aircraft. *Airbus UTM, San Francisco, CA*, 2018.

[124] Amazon. Revising the airspace model for the safe integration of small unmanned aircraft systems. *Amazon Prime Air*, 2015.

[125] EmbraerX. FLIGHT PLAN 2030: AN AIR TRAFFIC MANAGEMENT CONCEPT FOR URBAN AIR MOBILITY, author=EmbraerX. https://daflwcl3bnxyt.cloudfront.net/m/f58fb8ea648aeb9/original/EmbraerX-White-Paper-Flight-Plan2030.pdf, 2019.

[126] Hui-Min Huang, Kerry Pavek, Brian Novak, James Albus, and E Messin. A framework for autonomy levels for unmanned systems (alfus). *Proceedings of the AUVSI's unmanned systems North America*, pages 849–863, 2005.

[127] Bruce T Clough. Metrics, schmetrics! how the heck do you determine a uav's autonomy anyway? *NIST Special Publication*, 990:313–319, 2002.

[128] Brian P Hill, Dwight DeCarme, Matt Metcalfe, Christine Griffin, Sterling Wiggins, Chris Metts, Bill Bastedo, Michael D Patterson, and Nancy L Mendonca. Uam vision concept of operations (conops) uam maturity level (uml) 4. 2020.

[129] Arthur Brown and Wesley L Harris. Vehicle design and optimization model for urban air mobility. *Journal of Aircraft*, 57(6):1003–1013, 2020.

[130] W Johnson and C Silva. Nasa concept vehicles and the engineering of advanced air mobility aircraft. *The Aeronautical Journal*, 126(1295):59–91, 2022.

[131] Laurie A Garrow, Brian J German, and Caroline E Leonard. Urban air mobility: A comprehensive review and comparative analysis with autonomous and electric ground transportation for informing future research. *Transportation Research Part C: Emerging Technologies*, 132:103377, 2021.

[132] Aerospace-America. Electric drones and air taxis target the logistical frustration of transporting organs for transplants. https://aerospaceamerica.aiaa.org/electric-drones-and-air-taxis-target-the-logistical-frustration-of-transporting-or
?utm_campaign=AerospaceAmericaAMB&utm_medium=email&_hsmi=294472799&
_hsenc=p2ANqtz--0lJ0ysk-_CBUuwJSRI-GgoRMowEQit2CZnMpGirz0FWdJJK11eIEYEJNkQGjFI5ioXb
utm_content=294472799&utm_source=hs_email, 2023.

[133] Parker D Vascik and R John Hansman. Development of vertiport capacity envelopes and analysis of their sensitivity to topological and operational factors. In *AIAA Scitech 2019 Forum*, page 0526, 2019.

[134] Alba Agustín, Antonio Alonso-Ayuso, Laureano F Escudero, Celeste Pizarro, et al. Mathematical optimization models for air traffic flow management: A review. 2010.

[135] Cheng-Lung Wu and Robert E Caves. Research review of air traffic management. *Transport Reviews*, 22(1):115–132, 2002.

[136] Amedeo R Odoni. The flow management problem in air traffic control. In *Flow control of congested networks*, pages 269–288. Springer, 1987.

[137] Guodong Zhu and Peng Wei. Pre-departure planning for urban air mobility flights with dynamic airspace reservation. In *AIAA Aviation 2019 Forum*, page 3519, 2019.

[138] Nelson M Guerreiro, George E Hagen, Jeffrey M Maddalon, and Ricky W Butler. Capacity and throughput of urban air mobility vertiports with a first-come, first-served vertiport scheduling algorithm. In *AIAA Aviation 2020 Forum*, page 2903, 2020.

[139] Steve Paul, Jhoel Witter, and Souma Chowdhury. Graph learning-based fleet scheduling for urban air mobility under operational constraints, varying demand & uncertainties. *arXiv preprint arXiv:2401.04851*, 2024.

[140] Xuxi Yang and Peng Wei. Scalable multi-agent computational guidance with separation assurance for autonomous urban air mobility. *Journal of Guidance, Control, and Dynamics*, 43(8):1473–1486, 2020.

[141] Huy Trandac, Philippe Baptiste, and Vu Duong. Airspace sectorization with constraints. *RAIRO-Operations Research-Recherche Opérationnelle*, 39(2):105–122, 2005.

[142] Nichakorn Pongsakornsathien, Suraj Bijjahalli, Alessandro Gardi, Angus Symons, Yuting Xi, Roberto Sabatini, and Trevor Kistan. A performance-based airspace model for unmanned aircraft systems traffic management. *Aerospace*, 7(11):154, 2020.

[143] Lukas Preis and Mirko Hornung. Vertiport operations modeling, agent-based simulation and parameter value specification. *Electronics*, 11(7):1071, 2022.

[144] Min Xue. Airspace sector redesign based on voronoi diagrams. *Journal of Aerospace Computing, Information, and Communication*, 6(12):624–634, 2009.

[145] Sameer Kulkarni, Rajesh Ganesan, and Lance Sherry. Static sectorization approach to dynamic airspace configuration using approximate dynamic programming. In *2011 Integrated Communications, Navigation, and Surveillance Conference Proceedings*, pages J2–1. IEEE, 2011.

[146] Marina Sergeeva, Daniel Delahaye, and Catherine Mancel. 3d airspace sector design by genetic algorithm. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 499–506. IEEE, 2015.

[147] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[148] Franz Aurenhammer and Rolf Klein. Voronoi diagrams. *Handbook of computational geometry*, 5(10):201–290, 2000.

[149] Chuantao Yin, Shuaibing Zhu, Hui Chen, Bingxue Zhang, and Bertrand David. A method for community detection of complex networks based on hierarchical clustering. *International Journal of Distributed Sensor Networks*, 11(6):849140, 2015.

[150] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[151] Ang Li, Mark Hansen, and Bo Zou. Traffic management and resource allocation for uav-based parcel delivery in low-altitude urban space. *Transportation Research Part C: Emerging Technologies*, 143:103808, 2022.

[152] Maria Joao Alves and João Clímaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1):99–115, 2007.

[153] Dimitris Bertsimas and Sarah Stock Patterson. The air traffic flow management problem with enroute capacities. *Operations research*, 46(3):406–422, 1998.

[154] Dimitris Bertsimas, Guglielmo Lulli, and Amedeo Odoni. The air traffic flow management problem: An integer optimization approach. In *Integer Programming and Combinatorial Optimization: 13th International Conference, IPCO 2008 Bertinoro, Italy, May 26-28, 2008 Proceedings 13*, pages 34–46. Springer, 2008.

[155] Philip J Smith, Amy L Spencer, and Charles E Billings. Strategies for designing distributed systems: case studies in the design of an air traffic management system. *Cognition, Technology & Work*, 9:39–49, 2007.

[156] John Nash. Two-person cooperative games. *Econometrica: Journal of the Econometric Society*, pages 128–140, 1953.

[157] Roger B Myerson. *Game theory*. Harvard university press, 2013.

[158] Cat Hofacker and Alyssa Tomlinson. Building vertiport cities. https://aerospaceamerica.aiaa.org/features/building-vertiport-cities/, 2021.

[159] Electric-VTOL-News. Volocopter VoloCity (prototype). https://evtol.news/volocopter-volocity/, 2023.

[160] Aviation-Week. Joby Aviation S4. https://aerospaceamerica.aiaa.org/features/building-vertiport-cities/, 2023.

[161] Electric-VTOL-News. Beta Technologies ALIA-250. https://evtol.news/beta-technologies-alia/, 2023.