University of Cincinnati

Date: 2/11/2019

I, Matthew J Dechering, hereby submit this original work as part of the requirements for the degree of Master of Science in Mechanical Engineering.

It is entitled:

Traffic Management of small-Unmanned Aerial Systems in an Urban Environment

Student's name: **Matthew J Dechering**

This work and its defense approved by:

Committee chair: Manish Kumar, Ph.D.

Committee member: Kelly Cohen, Ph.D.

Committee member: David Thompson, Ph.D.

32947

Traffic Management of small-Unmanned Aerial Systems in an Urban Environment

A thesis submitted to the

Graduate School
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in the Department of Mechanical Engineering
of the College of Engineering and Applied Sciences

by

Matthew John Dechering

Master of Science, University of Cincinnati, 2019
Bachelor of Science, The Ohio State University, 2012

2019

Committee Chair: Dr. Manish Kumar

Abstract

Traffic Management of small-Unmanned Aerial Systems in an Urban Environment

by

Matthew John Dechering

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in the Department of Mechanical Engineering

University of Cincinnati
February 2019

Unmanned Aerial Systems Traffic Management (UTM) and Sense-And-Avoid (SAA) are important topics as the number of civilian s-UAS applications grows. Civilian applications are due to grow by leaps and bounds by 2021. These applications include Potential areas of application include emergency management, law enforcement, infrastructure inspection, precision agriculture, package delivery, and imaging/surveillance. With this increased low-altitude air traffic, algorithms capable of automating s-UAS trajectory planning are important. This thesis examines two algorithms for handling s-UAS traffic in urban environments, including path planning and SAA elements. The first algorithm operates in a 2D domain, uses Mixed Integer Linear Programming (MILP) to provide Sense-And-Avoid features, and uses A* as a top level path planner. In this approach, MILP provides in-depth optimization of the vehicle's trajectory by modelling its dynamics. The top-level A* reduces the work MILP has to do to model the entire path, by reducing the problem to a set of MILP optimizations within separate finite horizons. This combined algorithm produced 2-dimensional solutions with satisfactory separation for up to 35 s-UAS in a 16000 m^2 area. The second algorithm uses A* as a path planner, and uses a combination of priority and re-routing to resolve 3D trajectory conflicts. It provides a faster, but less thorough solution to UTM problems. It uses A* to route each s-UAS in three dimensions, and then steps through each route until a conflict is detected. When a conflict is detected, it is resolved by re-routing the s-UAS of lower priority around the higher priority s-UAS. The A* with re-routing algorithm provided satisfactory separation for up to 50 s-UAS in areas up to 466556 m^2 in size and volumes 56 m tall. For

30 s-UAS in a 5184 m^2 by 56 m volume, a solution was found with no more than 10 re-routes for a single s-UAS. Both algorithms are analyzed using randomly generated s-UAS missions and terrain data for the greater Cincinnati area. The algorithms provide adequate separation results at minimal cost to traffic timing. The grid-based simulations, while simple to set up, are computationally expensive. Future work would include better models of the airspace with pre-optimized routes that function like roads. A TIN would allow MILP to expand in optimization to 3D, further improving routing.

Thesis Supervisor: Manish Kumar

Title: Associate Professor

To my loving parents, and their unending support.

Acknowledgments

Manish Kumar

Mohammadreza Radmanesh

Hans Guentert

Matthew Dechering

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

s-UAS	Small UAS
UAS	Unmanned Aircraft Systems
FAA	Federal Aviation Administration
NAS	National Airspace
UTM	Unmanned Traffic Management
SAA	Sense-And-Avoid
SARP	UAS ExCom Science and Research Panel
VMD	Vertical Miss Distance
HMD	Horizontal Miss Distance
AGL	Above Ground Level
UAM	Urban Air Mobility
MILP	Mixed-Integer Linear Programming
ADS-B	Automated Dependent Surveillance-Broadcast
TOF	Time-Of-Flight
WAVE	Wireless Access in Vehicular Environments
GPS	Global Positioning System
IMU	Inertial Measurement Unit
RTK	Real-Time-Kinematic
ATC	Air Traffic Controller
ASP	Airspace Service Provider
NASA	National Aeronautics and Space Administration
V2V	Vehicle to Vehicle
SWAP	Size, Weight, and Power
BVLOS	Beyond Visual Line of Sight
ODOT	Ohio Department of Transportation
ANSP	Air Navigation Service Provider
PKI	Public Key Infrastructure
LAANC	Low Altitude Authorization and Notification Capability
USGS	United States Geological Survey
TIN	Triangular Irregular Network

Chapter 1

Introduction

Small UAS (s-UAS) have recently generated a lot of interest in civilian domains due to their potential to revolutionize several applications. These areas of application include emergency management, law enforcement, infrastructure inspection, precision agriculture, package delivery, and imaging/surveillance. The FAA expects an increase in number of unmanned flights, between 162% to 432% by 2021 [2]. Several challenges will face safe operation of UAS, specifically in terms of traffic management, with the increased numbers of drones expected to be in the air (FAA predicts at least 2.75 Million units by 2021, up from 1.10 million units). Safe separation needs to be maintained between unmanned systems and other aircraft, unmanned systems and the ground, and unmanned systems and stationary objects. Safe integration of UAS into the National Airspace System (NAS) involves disciplinary areas that include recent technologies (sensing, command, control, and communications) and regulations. Major challenges in integrating s-UAS in NAS include Unmanned Traffic Management (UTM), and Sense-And-Avoid (SAA). UTM means keeping s-UAS traffic safe and efficient. SAA means the specific methods by which s-UAS detect and avoid obstacles, each other, and other aircraft. This thesis explores two solutions to unmanned traffic management in an urban environment. Both solutions use a top-level algorithm to manage long-distance routing and obstacle avoidance, and a bottom-level algorithm to provide more refined obstacle avoidance and conflict resolution.

1.1 Problem Statement

In this project, N s-UAS are operating in an urban environment. Each s-UAS has a mission that consists of a start waypoint, a goal waypoint, and any number of intermediate waypoints. The s-UAS are in communication with a centralized UTM service which gives them detailed path instructions that lead to them accomplishing their mission. When planning a path for a s-UAS, the UTM service takes into account the s-UAS's current position and velocity, information on the terrain, and trajectories for all other s-UAS in the area the UTM service covers. The UTM service has two major objectives: maintaining safety for s-UAS and providing optimal paths for efficient traffic flow. It maintains safety by keeping safe separation distances between all s-UAS, and between s-UAS and the terrain. It provides optimal paths by minimizing path length and/or minimizing acceleration, depending on the method used.

1.1.1 Regulations for s-UAS in Urban Environments

The FAA defines Sense-And-Avoid (SAA) as the capability of a UAS to remain *well clear* from and avoid collisions with other airborne traffic, in the context of UAS [3]. The UAS ExCom Science and Research Panel (SARP) published a recommendation for s-UAS which defines *well clear* as a 250 ft. Vertical Miss Distance (VMD), and a 2000 ft. Horizontal Miss Distance (HMD). Unlike the recommendations for large UAS *well clear*, no time-based definitions of *well clear* were proposed. [4]. Under the FAA's small UAS rule, remote operation is allowable in G-class airspace under 400 ft. AGL, to provide a buffer between s-UAS operations and manned airspace. Following these definitions, traffic in G-class airspace is quite limited. Only two s-UAS can operate within the same vertical column. In un-congested rural areas, these rules would be effective at meeting *well clear* requirements while permitting effective use of airspace, but in congested urban environments these could severely limit airspace, especially between buildings taller than 400 ft. that are separated by less than 150 ft. Since such areas would be impassible with current standards, if urban s-UAS become widespread, highly urban areas will likely be limited to highly

maneuverable s-UAS. The FAA and several major companies have suggested UTM strategies for various environments. These are all developments towards Urban Air Mobility (UAM), a safe and efficient system for urban UAS services.

1.2 Solution Overview

In this thesis, two UTM solutions are presented. Both involve a top-level route planner and a lower-level Sense-And-Avoid. The top level route planner is similar in both cases, but they differ in their lower-level Sense-And-Avoid methods. Figures 1-1 and 1-2 provide a brief comparison between the two methods.

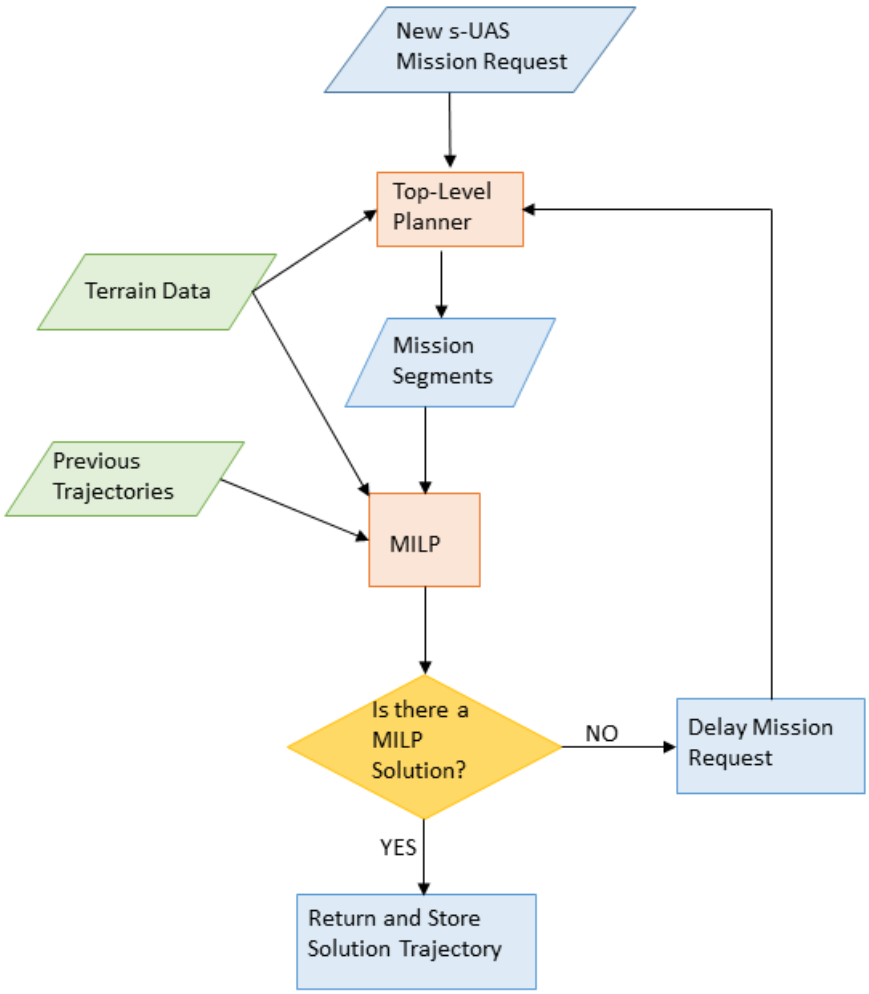

Figure 1-1: Method 1: A*-MILP

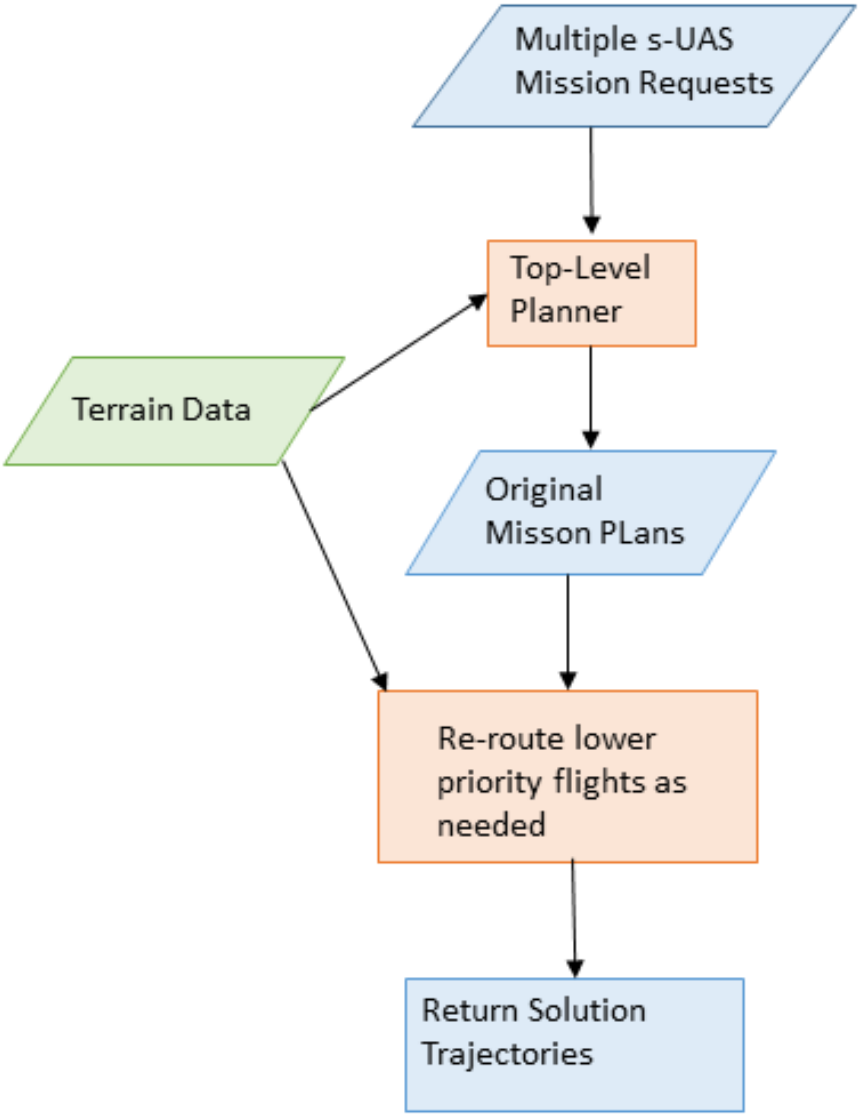

Figure 1-2: Method 2: A*-Priority and Re-routing

1.2.1 Top-level route planner

In both lower-level implementations, route management becomes computationally expensive as the distance covered by the algorithm increases. The top-level algorithm uses an A* search to provide an optimal-length path that safely avoids terrain. A* was chosen because it finds optimal solutions, is reasonably fast, and is easy to implement. This path is then given to a lower-level algorithm. Both lower level algorithms provide SAA functions while avoiding terrain and attempting to

remain as close to the optimal route as possible. There is some variation on how the top level is implemented. In the first method (type 1), s-UAS routing requests are processed in a first-in, first-out manner. In the second method (type 2), s-UAS routing requests are processed simultaneously.

1.2.2 Lower-level SAA type 1: Mixed Integer Linear Programming

In the Mixed Integer Linear Programming (MILP) solution, the path provided by the top level is broken down into segments that fit within a spatial and temporal finite horizon. Then, for each segment, the MILP solution applies a set of dynamic constraints that model how an s-UAS actually moves. The s-UAS is also constrained to be a safe distance from all obstacles within the finite horizon and all pre-existing trajectories of other s-UAS within the finite horizon. The final set of constraints set the s-UAS to visit all waypoints in the current segment in order. Given these constraints, a MILP solver is used to optimize a cost function that incorporates time and acceleration of the s-UAS. This trajectory is returned to the top-level to be used in future trajectory calculations. Because it includes s-UAS dynamics, the MILP method provides a more accurate picture of the trajectory of an s-UAS, but is more computationally intensive than the second method.

1.2.3 Lower-level SAA type 2: Priority and Re-Routing

In the priority and re-routing method, the paths provided by the top-level are traced in time until a conflict is detected. The priority of each s-UAS is calculated in such a way to favor s-UAS with lower remaining battery life with higher priority or s-UAS that are nearer to their goal with higher priority. When a conflict is detected, an algorithm similar to the one used in the top level is called, except it re-routes the lower priority s-UAS away from the conflict and towards its goal. It repeats this until no conflicts exist, then resolves tracing the new paths. This loop continues until all paths are resolved successfully.

1.3 Organization of Thesis

Chapter 2 explores existing s-UAS technologies, existing UTM solutions, and proposed UTM architecture. Chapter 3 discusses possible methods to model an urban airspace and the methods by which the model used in this thesis was made. Chapter 4 describes the top-level algorithm and the transition between it and lower-level algorithms. Chapter 5 describes the MILP (type 1) lower level algorithm. Chapter 6 describes the priority and re-routing lower-level algorithm. Chapter 7 describes the results of each algorithm. Finally, Chapter 8 includes discussion, conclusions, and future work.

Chapter 2

Background

2.1 Existing Technologies

This section focuses on an exhaustive market survey on existing state-of-the art technologies available for achieving SAA and UTM. Limitations and capabilities of each of these technologies have been catalogued in the previous report. Technologies include different onboard sensor modalities, ground-based sensors (radar and cameras), Automated Dependent Surveillance-Broadcast (ADS-B), communication devices and protocols, and computing. Onboard sensors include intruder and obstacle detection (LIDAR, RADAR, vision, thermal vision, Time-Of-Flight (TOF) cameras) and localization and position measurement (GPS, inertial measurement units (IMU)). Communication devices and protocols include cellular networks (LTE), satellite-based communication, and Wireless Access in Vehicular Environments (WAVE). Computing includes the capabilities of onboard computers for data processing and path planning and auto-pilots for lower-level control.

2.1.1 ADS-B

Automatic Dependent Surveillance-Broadcast (ADS-B) is a prominent part of the future of aviation situational awareness. It is a surveillance system which an aircraft broadcasts its altitude, airspeed, and GPS locations to ground stations and other aircraft via a dedicated radio frequency. The major advantage for ADS-B (or a similar system) on a s-UAS is its ability to provide SAA

coverage to areas where traditional ground-based radar has difficulty (such as at low altitudes) [5]. Additionally, ADS-B has a faster update rate of 1 second versus radar's 5-12 seconds. Recent advances in ADS-B to allow it to be produced with small enough SWAP for a s-UAS. Table 2.1 shows a summary of prominent ADS-B solutions for the s-UAS market. Most ADS-B systems are currently too expensive for the UAS community. uAvionix has a particularly strong offering with their small, lightweight line of ping ADS-B transceivers. ADS-B is not a perfect system due its vulnerability to electronic attack [6]. It should be paired with additional on-board sensors to detect birds and other unexpected obstacles.

Table 2.1: ADSB products for s-UAS

Product	Refernce	Input Power (W)	Weight (g)	ADS-B in	ADS-B out	Size (mm)	Internal GPS
ping2020	uAvionix	0.5	20	yes	yes	25 x 39 x 12	ping2020i
ping1090	uAvionix	0.5	20	yes	yes	25 x 39 x 12	ping1090i
XPS-TR	Sagetech		100	no	yes	89 x 46 x 18	no
XPG-TR	Sagetech		100	no	yes	89 x 46 x 18	yes
MXS	Sagetech	15	150	yes	yes	84 x 64 x 19	available

2.1.2 LIDAR

LIDAR, a common obstacle detection sensor, is a laser rangefinder which scans a path radially to detect objects. LIDAR is popular due to historical low cost with respect to other conventional aviation technologies such as radar. It is available in planar scan 2D-scan and 3D-scan variations. The 3D scan is more expensive, and typically heavier, but provides coverage a 3 dimensional space rather than a 2 dimensional plane. A 3 dimensional scan is important for detecting small targets as well as targets that do not approach from the UAV's 2 dimensional plane of travel. A 2D lidar will not be able to detect objects that approach from below or above the UAV unless the UAV pitchs or rolls the LIDARs detection plane. A study presented a 3D-Scan LIDAR and algorithm

that can detect intruders at 88.4232 ft. per microsecond, which is sufficient to meet the FAA Equivalent Level of Safety requirement [7]. LIDAR operates ideally in overcast conditions and against reflective targets, but struggles in bright conditions and against non-reflective targets. A list of LIDAR products available for s-UAS is shown in table 2.2.

Table 2.2: LIDAR products for s-UAS

Product	Type	Weight (g)	Range (m)	Power (W)
Hokuyo 3D-LIDAR YVT-X002	3D Scan	750	50	8.4
Hokuyo UST-10LX	Planar Scan	130	30	3.6
Hokuyo UTM-30LX	Planar Scan	210	30	8.4
Hokuyo UTM-30LX-EW	Planar Scan	210	30	8.4
Hokuyo UTM-30LX-F	Planar Scan	210	30	8.4
Hokuyo UXM-30LX-EW	Planar Scan	800	30	7.2
Hokuyo UXM-30LXH-EWA	Planar Scan	1200	80	7.2
Ibeo LUX	Planar Scan	900	200	10
Ibeo LUX 8L	Planar Scan	1000	200	10
Ibeo LUX HD	Planar Scan	1000	120	10
Ibeo miniLUX	Planar Scan	450	40	7
lightware SF40/C	Planar Scan	229	100	4.5
Quanergy M8	3D Scan	800	150	15
Quanergy M8-1	3D Scan	900	200	18
Quanergy S3	3D Scan	N/A	150	N/A
Quanergy S3-Qi	3D Scan	N/A	150	N/A
RIEGL VQ-480-U	Planar Scan	7500	950	55
RIEGL VUX-1UAV	Planar Scan	3750	920	60
Scanse Sweep	Planar Scan	120	40	3.25
Spectrolab SpectroScan 3D	3D Scan	2018	20	30
Velodyne HDL-32E	3D Scan	2000	100	12
Velodyne Puck Hi-res	3D Scan	830	100	8
Velodyne Puck LITE	3D Scan	590	100	8
Velodyne PUCK VLP-16	3D Scan	830	100	8

2.1.3 RADAR

Radar is a proven technology in aerospace and is a powerful Sense-And-Avoid tool. The main challenge with radar for s-UAS is achieving low enough SWAP to be feasible while maintaining a usable cost. There have been a few companies that have released small, lightweight radar systems over the past two years. While there are not as many products on the market as LIDAR, those that are have good statistics. Table 2.3 shows strong on-board radar for s-UAS.

Table 2.3: RADAR products for s-UAS

Product	Range (m)	Accuracy (m)	Weight (g)	Power (W)	Update rate (Hz)
Aerotenna Sharp 360	40	0.22	243	2.5	80
Aerotenna Sharp Patch	120	0.22	43	1.25	90
Echodyne MESA SSR	750	3.25	1250	45	2
Echodyne MESA-DAA	750	3.25	817	35	1
Fortem DAA-R20	1500	0.0508	464	60	8
IMST DK-sR-1200e	307	0.6	280	4.5	10-200
Integrated Robotics IRIS Sensor	66	1.24	360	4.5	3.4

2.1.4 Cameras and TOF

Vision and thermal vision sensors have advantages of low cost and offer the ability to pilot the drone via camera. Thermal vision has the advantage of being applicable in low-light conditions. Systems often need to include multiple cameras to fulfill SAA FOV requirements (±110° azimuth, ±15° elevation) [8].

Time-of-Flight (TOF) cameras operate on a similar concept to LIDAR: it emits light and measures the time for the light to reflect [9]. Unlike LIDAR, it emits light as a single pulse rather than a scan. They are relatively expensive and perform poorly in the same conditions as LIDAR, but can come bundled with a camera, effectively being 2 sensors in one.

2.1.5 Ground-Based Monitoring

In addition to ADS-B and onboard sensors, there is a desire to monitor the airspace from the ground using ground-based radar stations. Ground radar is challenging due to cover the low altitudes s-UAS operation. Radar shadow caused by hills and buildings are more common. Ohio's Ground-Based Detect and Avoid system at Springfield-Beckley Airport fuses three FAA surveillance to provide a 95% probability to track and avoid all aircraft at 500 ft. AGL and above. While useful for keeping UAS out of manned airspace, this does not allow the system to track s-UAS in their typical operating zone. Gryphon Sensors' Skylight system includes a precision avoidance radar that detects s-UAS at 10 km and general aviation at 27 km, a spectrum sensor with up to 5 km of range, and a camera with 3 km detection range [10]. It provides good coverage of the low altitude operation area, but its limited range means more radar stations would be needed to cover an area than typical air traffic control radar. There are similar low-altitude radar solutions, which are summarized in table 2.4.

Table 2.4: Ground-Based Tracking Methods

Ground Unit	Company	Type	s-UAS detection limit	RADAR	ADS-B
Skylight	Gryphon Sensors	non-cooperative detection	10 km	yes	yes
Harrier DSR-200	Detect	non-cooperative detection	4.8 km	yes	yes
SharpEye^{TM} SxV	Kevin Hughes	non-cooperative detection	1 km	yes	
pingStation	uAvionix	ADS-B Station	241 km		yes
FlightHorizon GCS	Vigilant	situational awareness, C2 solution		yes	yes

2.1.6 GPS and IMU

The ability of a s-UAS to maneuver is dependent on its ability to self-localize. For self-localization, most s-UAS carry a GPS and an Inertial measurement unit (IMU). Low cost hobby GPS units have a positional accuracy of ± 7.8 meters [11]. The GPS used for uAvionix's ping ADS-B has an integrated system with a 95% horizontal accuracy bound of 3 meters [12]. Furthermore, Real-Time-Kinematic (RTK) GPS has a positional accuracy of 1 centimeter, but it requires a

calibrated connection with a base station [13]. IMUs measure acceleration and rotation rate of the s-UAS. Integrating these twice to find position can compounds the errors registered by these sensors, so filtering is often employed. GPS and IMUs are either directly supported by most autopilot boards, or are integrated with the onboard ADS-B system through sensor fusion.

2.1.7 Onboard Computing

To operate safely and autonomously, s-UAS use an autopilot and other onboard computing resources. The ArduPilot Mega, Pixhawk, and PX4 are popular open-source autopilots used by hobbyists and academia. Yuneec's Typhoon H is capable of intelligently avoiding obstacles at low altitude [14]. The Intel Falcon 8+ is an inspection, surveying, and mapping drone that is capable of up to 1 millimeter accuracy in terms of data collection, indicating it has tightly controlled flight performance [15]. DJI's Guidance system boasts up to 0.05 m positioning accuracy and 0.04 m/s positioning accuracy, which is powerful, but it is currently optimized to perform under 65 feet AGL [16]. Qualcomm's Snapdragon is a powerful all-in-one onboard computer and autopilot solution [17]. Table 2.5 shows an overview of autopilot solutions.

Table 2.5: s-UAS autopilots

Company	AutoPilot	Source Control	Waypoint Navigation	ADS-B Compatibility
3DR	APM 2.8	Open Source	Yes	Yes
	PixHawk Mini	Open Source	Yes	Yes
AeroQuad	AeroQuad v2.2 Kit	Open Source	Optional	No
DJI	A2	Proprietary	Yes	Yes
	Naza-M Lite	Proprietary	Optional	Yes
	Naza-M V2	Proprietary	Yes	Yes
	Wookong	Proprietary	Yes	Yes
Emlid	Navio2	Open Source	Yes	Yes
Erle Robotics	Erle Brain 3	Open Source	Yes	Yes
	Erle-Brain PRO	Open Source	Yes	Yes
Feiyu Tech	FY-40A	Unknown	No	No
	FY-41AP	Unknown	Optional	No
	FY-41AP Lite	Unknown	Unknown	No
	FY-DOS	Unknown	Unknown	No
	Panda2	Unknown	Yes	No
Free Flight	FF Auto Balance Controller	Unknown	No	No
HobbyKing	AfroFlight Naze32 Rev6 Acro	Unknown	Unknown	No
	KK 2.1 HC	Open Source	Unknown	No
Hobbyking / Crius	All In One PRO	Open Source	Optional	No
	MultiWii Lite	Open Source	Unknown	No
	MultiWii SE	Open Source	Optional	No
Holybro	Pixfalcon	Open Source	Yes	Yes
Hoverfly	HoverflyPRO	Unknown	Optional	No
Intel	Intel Aero	Open Source	Yes	Yes
Intrinsyc	Snapdragon Flight Autopilot	Open Source	Yes	Yes
LibrePilot	CopterControl/Atom	Open Source	Unknown	No
	Revolution	Open Source	Unknown	No
MiKroKopter	Flight-Ctrl ME 2.1 Complete	Unknown	Yes	No
mRobotics	Pixracer R14	Open Source	Yes	Yes
MultiWiiCopter	iNav Sirius^{TM} AIR3 F3 SPI	Open Source	Add-on	No
ProfiCNC	Pixhawk 2.1 Cube	Open Source	Yes	Yes
QuadroUFO	UAVX-ARM32 Full Sensors	Open Source	Unknown	No
Range Video	RVOSD 6	Unknown	Yes	No
SmartAP	3.0 Pro	Unknown	Yes	No
	4 Set	Open Source	Yes	No
Viacopter / FlyDuino	AutoQuad v6.6	Open Source	Yes	No

2.1.8 Communication

To operate successfully in the national air space, a s-UAS must maintain communication with an ATC and/or an airspace service provider (ASP). Prominent Technologies include Wireless Access in Vehicular Environments (WAVE), cellular networks (LTE), and satellite communication. WAVE, also known as IEEE 802.11p, IEEE1609.xx, or dedicated short-range communication [18], is mostly marketed for automotive applications, but could be used for s-UAS. Cellular vehicle-to-vehicle communication was defined in March 2016 by 3GPP release 14 [19]. Qualcomm's Snapdragon X16 LTE Modem is strong, capable of 75 Mbps, and operates with LTE category 16 downlink and category 13 uplink [20]. Qualcomm has also completed a study where it showed reliable connection of a drone to an LTE network in G-class airspace [21]. Satellite communication is more expensive and slower than LTE and WAVE, but it does not require infrastructure to be used in remote locations [22]. This makes it a viable back-up in case where the first two are unavailable. Table 2.6 lists WAVE products available for s-UAS. Table 2.7 lists LTE products available for s-UAS.

Table 2.6: s-UAS WAVE Solutions

Product
WaveCombo
LocoMate
SnapDragon

Table 2.7: s-UAS lte products

Product
Snapdragon
HUAWEI E3272
AES-ATT-M14A2A-IOT-ADD-G
Quectel Raspberry pi kit

2.2 Existing Solutions

Several novel architectures for UTM have been proposed that provide details about the command and control concept of operations. These include the Google and Amazon's different Airspace Service Provider concepts [23] [24] [25]. NASA also has many advances and simulations for UTM operations [26]. Table 2.8 shows NASA's Technical Capability Level Roadmap. [1] Rock-

well Collins also has made advances with their UAS services. [27] This thesis examines solutions provided by industry, government, and academia to evaluate their respective effectiveness and feasibility in terms of technologies available.

Table 2.8: NASA TCL Levels [1]

Capability 1	Capability 2
• Airspace volume use notification • Over unpopulated land or water • Minimal general aviation traffic in area • Contingencies handles by UAS pilot • Enable agriculture, firefighting, infrastructure monitoring	• Beyond visual line-of-sight • Tracking and low density operations • Sparsely populated areas • Procedures and rules-of-the-road • Longer range application
Capability 3	**Capability 4**
• Beyond visual line-of-sight • Over moderately populated land • Some interaction with manned aircraft • Tracking, vehicle-to-vehicle, internet connected • Public safety, limited package delivery	• Beyond visual line-of-sight • Urban environments, higher density • Autonomous vehicle-to-vehicle, internet connected • Large-scale contingencies mitigation • News gathering, deliveries, personal use

As the airspace becomes more congested, determining permission to access airspace becomes

more important. Amazon's best-equipped, best-served model [24] is a straightforward implementation that only allows s-UAS to fly where they can maneuver and perform SAA functions effectively. This will mean that new s-UAS will need to have their capabilities rated, a service which most federal s-UAS test sites already perform. Furthermore, cooperative SAA requires a well-defined communications protocol. Considering the weaknesses of ADS-B, a well-defined ADS-B like system will be one of the next major steps in terms of in terms of inter s-UAS communication. Google's public key infrastructure is one example that would be more secure than the current unencrypted system. [23] With the large amount of data produced by and needed to handle high-volume s-UAS flights, major computational services like Rockwell Collins' WebUAS^{TM} [27] will play a role in the future of s-UAS management. Airspace monitoring, flight abortion/re-routing, and separation assurance will be key features. Airspace service providers interfacing with the FAA UAS data exchange are the current direction that flight management is headed. They are currently very useful because to the FAA, because it can outsource the automation of flight management to several partners. The challenge will be having all partners acting cooperatively in the same airspace. Not all partners use the same airspace model for obstacles, but they do share the FAA's flight data, which is how traffic is currently managed.

The UTM architecture proposed by NASA is shown in figure 2-1. [28] Here, the Air Navigation Service Provider (ANSP) provide traditional air traffic management (ATM) services. UTM is separate from and complementary to UTM; UTM operators and stakeholders conduct and provide support for s-UAS operations independent of the ANSP's scope of influence, but not in isolation of it. Interactions between the two are coordinated by the Flight Information Management System (FIMS), a central cloud-based component that also acts as a broker of information between UTM stakeholders. UAS Service Suppliers (USS) that meet minimum requirements for functionality, quality of service, and reliability. USS then support missions by UAS operators. Connections and communications are internet-based and built on industry standards and protocols.

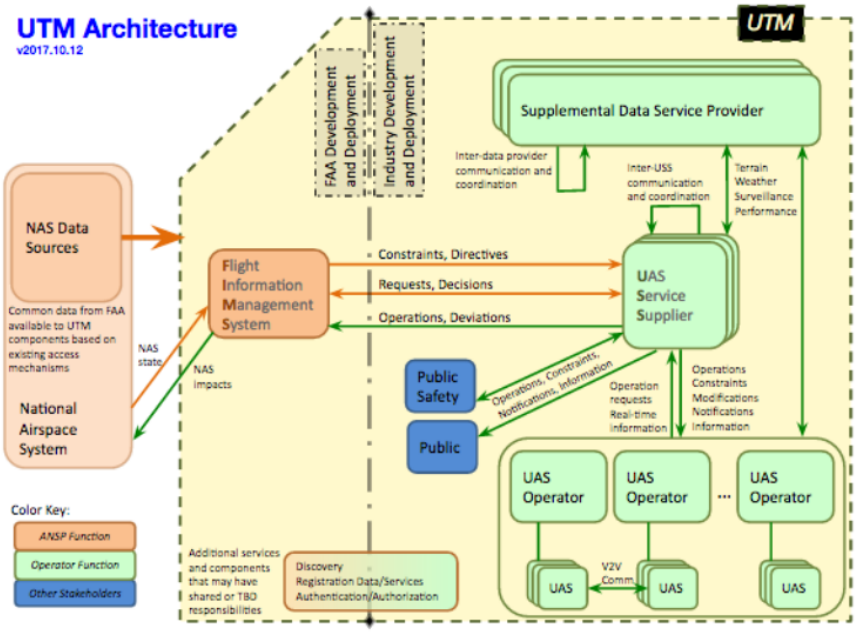

Figure 2-1: NASA Proposed UTM Architecture [28][1]

With this centralized architecture, Vehicle to Vehicle (V2V) communication can play a key role in SAA and conflict resolution. SAA applications often rely on knowledge of an s-UAS's position and velocity and knowledge of intruder position and velocity. [5] More detailed knowledge of intended maneuvers of the intruder are also seen as helpful. [5] ADS-B or an ADS-B like V2V communication system that also provides information on intended maneuvers (i.e. turns) would therefore be the most needed form of V2V communication. In the absence of any V2V communication, SAA must be provided by onboard sensors and/or coordination through possibly multiple USS. For an example of two UAS with separate operators subscribing to separate USS, the coordination would go through communication between the separate USS.

S-UAS situational awareness is set to grow by leaps and bounds soon, because of the expedient decrease in the cost of low-SWAP sensors. After the initial survey of sensor technologies and UTM solutions, we began investigating algorithms for traffic management. The accuracy of the

[1]Reproduced from [28] as per the copyright policy on: `https://www.nasa.gov/multimedia/guidelines/index.html`

sensors means completely cooperative vehicles can operate near one another. Intruders still pose a problem, despite some models being equipped with sensors sufficient to anticipate anything that their maneuverability allows them to avoid. Initially, package delivery in an urban environment was considered because large numbers of package delivery s-UAS pose a challenging problem in a sky where traffic between s-UAS is codified to the extent of other vehicles i.e. conventional aircraft or cars. The mission definition was easy to define for package delivery s-UAS. The algorithms covered in this thesis will be effective for package-delivery s-UAS or any s-UAS with a similar mission definition. The mission definition for a package delivery service leaving a depot and travelling to N destinations is: take off from depot, travel to destination 1, perform delivery, travel to the next destination (for n destinations), then return to the depot.

2.3 UTM Architecture

Amazon's strategy follows the best equipped, best served model. [24] This means a UAV's permission to access an airspace is decided based on how well equipped the UAV is for that airspace. To operate BVLOS and/or in the most challenging airspace, a UAV needs five equipage elements: geospatial data for safe separation from known hazards, online flight planning and management, a reliable internet connection, collaborative vehicle-to-vehicle Sense-And-Avoid, and non-collaborative sensor-based Sense-And-Avoid. To implement this model, Amazon wants to partition the airspace under 500ft. AGL into areas of different required levels of safety. [25] These partitions are not one-size fits all: a remote area with less air traffic has a lower required level of safety than an urban area. Under 200 feet AGL is the low speed, localized traffic area. It is used for terminal non-transit operations, such as surveying and inspection. Vehicles with low equipage are restricted to areas in this airspace where they meet the required level of safety. Between 200 and 400 feet AGL class G airspace is the High-Speed transit zone. It is meant for transit operations and requires a higher required level of safety. They are allowed access in emergencies only. In addition to an area's required level of safety being dependent on how urban it is, the aviation authorities are

also expected to designate predefined low-risk locations, limited areas with a lower required level of safety. Because load on Air Navigation Service Providers (ANSP) is proportional to number of UAVs in the air, amazon wants ANSPs to delegate some of their responsibilities to automation, although they will remain the central offline authority. Amazon envisions several distributed operators controlling fleets that must coordinate with each other via established protocols and services and vehicle-to-vehicle communication.

Google's strategy focuses licensing and information exchange. [23] They discuss operations under 500 feet AGL class G airspace, and hope for future allowances for UAS in lower altitude airport airspace. They expect UAS to give way to manned aircraft via ADS-B, and envision an ADS-B like system for UAS-UAS collision avoidance. The Airspace service providers will license aircraft for operation and provide separation and planning to the UAS via cellular networks. They want Project Wing to have its own service provider, and expect other operations will have their own service providers. The ASP is the interface between the UAS operators and air traffic control. The ASP maintains a database of Temporary flight restrictions, no-fly zones, weather, obstacles and terrain, traffic, and flight plans. It will also take data from airspace authorities and data sources, including NOAA, FAA, ATCs, and weather data sources. Google emphasizes all pilots, aircraft, and operators having some form of traceable ID (pilot license, aircraft registration, or operator registration). A UAS ID system would be scalable, allows for authentication, and gives traceability. It wants to do this using the proven public key infrastructure (PKI). The PKI is a security process where a participant creates a public/private key pair and shares one with the registration authority. The registration authority verifies the ID of the participant and provides this verification to the certificate authority. The certificate authority uses this information to provide the participant with a signed certificate, which is a secure encapsulation of the participant's identification data. This certificate will then be used whenever a participant submits a flight plan request to the ASP, who verifies the certificate with a verification authority. The ASP signs off on the flight plan (assuming no conflicts exist). The Airspace participant can now use this signed plan for operations. For the future, Google wants ASPs to be allowed to operate as manned aviation does today in uncontrolled

G-Class airspace. ASPs need to be open and collaborative with each other and air traffic control authorities. Google also wants the ADS-B ruling for 2020 amended to also apply to helicopter flying below 500 feet AGL over populated areas, as these are the manned aircraft s-UAS will encounter most often. The public key infrastructure is one step that is more secure than the current unsecured ADS-B communications.

Considering the expansion of s-UAS in the near future, Rockwell Collins speaks about its WebUAS™ services, which are a set of secure cloud-based services for UAS operators to interface together and with air traffic control. [27] It emphasizes computational engines which are separate from the services infrastructure, allowing an operator to select the engine elements computational engines and services they need to operate well when setting up a server. It supports FAA System Wide Information Management, and is flexible to further standardization. WebUAS™ supports third-party engines because its computational engines are separate from its infrastructure, and dedicates servers to fulfill specific needs. WebUAS™ uses AviNet (a secure global network for airlines and airports) to interface to FAA systems, and uses the same FAA approved firewalling it uses with all aviation customers. WebUAS™ iis based on national centralized servers that are peers to and under the same constraints as Aircraft Situation Display servers. It will support future expansion by adding regional servers. WebUAS™ has dedicated servers for ATC tower interfaces, other industry partners, and large operators of UTM. WebUAS™ supports a number of services for BVLOS operations. WebUAS™ is able to generate many service packages by combining a wide selection of specialized servers and computational engines. It includes a flight plan authorization engine that is able to make recommendations instead of simply accepting or rejecting the plan. A list of engine elements available for service customization include: a measure of how well airspace is monitored, the ability to abort or reroute flights in emergencies, real-time separation assurance, specific industry requirements, collaborative decision making for all types of service providers, graphic depiction of recommended routes, and a measure of the proactive prediction and warning capabilities of the engine.

NASA, Google, and Amazon all refer to an Airspace Service Provider (ASP) in their models.

An ASP is a company that provides separation and planning services and interface with FAA data. The beginnings of ASP's are seen in the FAA UAS data exchange, an umbrella agency for partnerships between the FAA and industry to facilitate the sharing of airspace data. [29] Its first partnership is the Low Altitude Authorization and Notification Capability (LAANC), an application through which the FAA may authorize operations under the small UAS rule. It allows operators to interact with maps and provide automatic notification and requests to the FAA. Users can apply through a part 107 process, or through an approved UAS service provider. Approved UAS service providers include AirMap, Project Wing, and Skyward.

AirMap allows users to plan flights with a phone or web app. [30] Paths may be constructed from points, lines, and areas. The app also provides a list of keep-out zones to make low altitude flight planning easy. AirMap does not include topographic information or building information, but does allow users to set the height of their flights. Project Wing claims to safely manage complex flight paths across multiple drones, and was tested with drone delivery to residential yards. The Project Wing UTM platform ensures a route that is clear from buildings, terrain, obstacles and participating aircraft. [31] Skyward has an interactive airspace map that allows users to view flight restrictions and mark hazards. [32] They also have a map of elevations and obstacles driven by LATAS flight planning software. For all of these UAS service providers, communication will be highly important as the skies become more congested. For now, communication is done through the LAANC posting flight notifications.

2.4 Literature Survey

There are several advances that have been made in the areas of SAA and UTM. Collision avoidance has been examined based on distributed dynamic optimization and causal analysis [33]. It generates trajectories that are globally valid and not limited to the 2-UAS scenario. It uses solution clustering to reduce the problem size and fires a sense-and-avoid function whenever the trajectories become too close. It models the trajectory of each UAS as discrete waypoints with a

discrete kinematic model and minimizes an objective function to resolve conflicts. It additionally uses causal analysis to limit the number of conflict resolutions necessary

NASA has made several advances in the development of UTM algorithms. ICAROUS [34] is a software architecture meant to build safety-central, autonomous s-UAS applications. It provides stand-off distances from obstacles and intruders to ensure safe s-UAS operations. There is also a path-planning algorithm to enable well-clear low altitude BVLOS operations [35]. In this algorithm, a heuristic is used to limit decision tree growth, the decision space is searched with an RRT (Rapidly exploring random tree) [36], and DAIDALUS [37] is used to check for well-clear, and PolyCARP [38] is used to provide obstacle avoidance. These algorithms work in real-time for a generic UAS architecture.

Collision detection is seen with several algorithms. In a survey [38] on unmanned vehicle collision algorithms, several categories are discussed. In geometric approaches the turning-away angle is optimized (path optimality vs safety). Optimized trajectory approaches use path-planning algorithms to determine an optimal path while still maintaining safety. Bearing angle-based approaches are simpler, but have synergy with a pure visual system. Guidance and control algorithms for s-UAS use GPS=based waypoint navigation. [39] Adaptive autopilots have more robust performance than pure PID-based controllers. LIDAR is a resource for localization and mapping, which can be used for obstacle detection and avoidance. [40] It has been shown to work in the 2D case with 3D data, even in GPS-denied environments. Stereo vision is also an autonomous collision avoidance resource. [41] It has been shown to effectively give resolution solutions in low altitude scenarios.

In Feasibility of Varying Geo-Fence around an Unmanned Aircraft Operation based on Vehicle Performance and Wind [42], DSouza et al describe a method for determining appropriate geo-fence size for a UAV based on its performance parameters and wind data. Their geofences were smaller than the 30m geofences which have been used as a standard in prototypes, but they did not account for position uncertainty from GPS or other position sensors. Despite this, their models still showed satisfactory performance in real-world wind data: they achieved 15m horizontal and 5m vertical

geofence with basic vehicle parameters and algebraic-geometric equations of motion, and less than 5m of geofence with a gain-scheduled PID controller. Their wind data came from three sources: the NOAA High Resolution Rapid Refresh (HRRR) model, which has the best spatio-temporal resolution of 15 minutes and 3 kilometers. HRRR provides horizontal components recorded at 10 and 80 m AGL. They derived a maximum vertical component with data from the California State University-Mobile Atmospheric Profiling System. For the PID design, they used real wind data and an OpenFOAM flow simulation for a simplified building. For their algebraic-geometric geo-fence, their vehicle parameters were maximum vehicle airspeed and time from detection of disturbance to vehicle recovery. Control time was not provided by any authority, and was estimated to be 1 second. Worst-case wind data from HRRR was used, and the geo-fence was set at the point of farthest divergence from the path, which was calculated using algebraic-geometric equations of vehicle motion over the time till vehicle recovery. Their result indicated a larger geofence is required for a UAV at higher speed and/or in stronger wind. For their PID controlled geo-fence, they simulated the vehicles control dynamics, and set the size of the geofence as the maximum deviation from the desired path. They simulated their vehicle moving through a path given flow data from OpenFOAM. Their controller was a gain-scheduled PID controller with simplified equations of motion as the plant. The gain scheduling was done with a gradient-free artificial bee colony genetic optimizer. They modelled an AscTec Pelican quadrotor, and their model could maintain less than 5m of error in the simulated flow.

NASAs UTM simulation capabilities and lab environment are extensive. [43] The airspace operations lab at NASA AMES is equipped to enable test capabilities at each TCL. Their outward-facing TCL2 release is available to properly credentialed users. The users can access it through a variety of Clients. Their python client is the primary client for live flight testing and demonstration. It interfaces with the mission planner ground control station to create operational plans and volumes based on waypoint definitions. It also establishes a web connection to the UTM research platform, allowing the operator to submit plans and receive feedback. The multi-aircraft control system (MACS) provides users with the ability to develop custom, map-aided flight profiles and

operational volumes, and is capable of simulating flight of UASs according to developed profiles while sending position reports. It allows any number of flights to be operated in autonomous and manual modes, and interfaces with google earth to display the flights. Furthermore, the research lab has simulation and support activities. Their flight test support is based on the mission planner software, a software for making UAV flight profiles. The initial flight profile is done with mission planner, and the software simulates the flight to see if it is feasible. It also outputs a waypoints file, which is used to calculate operational volumes, altitudes, and the duration for each. The simulation services include concept developing and testing, which allows for simulated flights in addition to the real-world flight tests, which are limited by safety and logistics. MACS is the primary software used for these services. The simulation platform also allows for live, virtual, constructive activities, which means integrating simulated flights and services with live flights and the types of environments allows researchers to evaluate features that would be impractical to test in a live setting.

NASAs AMES research center also includes a Human/Systems component. [1] It focusses on API-based coordination of UAS operations and services into a research software environment. The research software is used to test and evaluate increasingly complex operations. Their lab includes multiple test areas and a distributed network of client stations, servers and display media. Although the entire facility can be leveraged for UTM, there are generally two main test areas used for simulation and demonstration. The test set-up can simulate multiple environments for multiple clients simultaneously, and there is a test area for concept and capability demonstrations. There are large displays to show currently running simulations, and the infrastructure is designed to reflect components of an outward-facing system. It also tests the scalability of future environments. Test bed and rapid prototyping for human/systems interfaces and procedures are required for field demonstrations and LVC testing with partners. The system includes many human-friendly interfaces to support the research teams needs.

In Rapid Trajectory Prediction for a Fixed-Wing UAS in a Uniform Wind Field with Specified Arrival Times, [44] Ishihara et al discuss a method to determine flight plan feasibility given an

operator submitted flight plan, wind forecast, and vehicle parameters. This feature would be part of a high-level UTM system where the client submits a flight plan, the clients identity is validated by authorities, static airspace constraints are checked, dynamic capabilities including vehicle capabilities and weather are checked. If these are satisfactory, the plan is accepted. They formulated the problem with a 4D trajectory, a forecast from HRRR for a given position and time. All wind fields are modeled as uniform at the measured point. Each segment between two waypoints with a kinematic model and a cost function based on the integral of bank controls and the distance between current state and segment goal. The model is non-dimensionalized with respect to the length of the current segment and the velocity magnitude for the current segment. The model finds the inputs to minimize the cost function given the desired waypoints and wind data. In the higher-level system, they assume verified vehicle identification provides the vehicles maximum, minimum, and cruise velocities, as well as maximum turn rate. The dynamic simulation generates an error metric for each waypoint given distance between simulated aircraft and waypoint at client supplied time of arrival. The high-level model would reject a plan if any waypoint exceeds an acceptable error threshold. In the results of their simulation, they showed that their model could find an optimal velocity if the cruise velocity was infeasible. They also simulated 24 trials of a trajectory for 4 days of weather data, where the system decided to accept or reject a trajectory based on weather data. On the windiest day (average wind speed 5.88 m/s), the system rejected 14 of the 24 flight plans, and on the calmest day (average wind speed 3.46 m/s) it rejected 6.

In Multi-Rotor Aircraft Collision Avoidance using Partially Observable Markov Decision Processes [45], Mueller and Kochenderfer present an extension to the FAAs Airborne Collision Avoidance System X (ACAS X) for small multirotor UAVs. ACAS X alerting logic is based on a numeric lookup table optimized with respect to a probabilistic model of the airspace and a set of safety and operational considerations. Their algorithm is formulated in two dimensions, and uses horizontal plane acceleration for resolution maneuvers. They optimized it with respect to a partially observable Markov decision process with dynamic programming, and found optimal parameters for the process with a gaussian process-based surrogate model. They aimed to maintain separation mini-

mums and closely track mission trajectory. They formulated the problem as a partially observable Markov decision process and used the bellman update and dynamic programming to find the optimal policy. Their dynamics included relative range rates between intruder and ownship, velocities of both, and displacement of ownship from desired trajectory. All variables are normalized by ownships forward speed. Their reward function included the inverse square of the range between intruder and ownship, squared distance from desired trajectory and a maximum cost to prevent cost from becoming infinite. The model used states where the dynamics variables were discretized to selected values. The stopping criteria was a combination of maximum error between iterations and the maximum number of optimal action changes between iterations. Since the real-world process is not discrete, and the states are not precisely known, the potential of each state was calculated based on the sigma point sampling technique, and the actual states are interpolated from the discretized states. The Markov decision process was optimized using an objective function that incorperated the minimum separation that was achieved in the top 95% of encounters and the average deviation distance over time for all simulated encounters. A Gaussian process was used to determine the set of reward parameters in the Markov decision process that resulted in the minimum objective function. The optimization was run several times to determine the tradeoff between separation and deviation parameters. All designs were sorted to find designs for which no parameter selection was better, to allow users to select the most optimal design for their situation. They further simulated the algorithm for multiple intruders. The only difference for this scenario was the Markov decision process selected the belief state with the highest minimum weight over all intruders. They implemented uncertainty for ownship and intruder acceleration for optimization and simulation. They also implemented position and velocity uncertainties in simulation. They simulated intruder trajectories with constant nominal speed and with realistic UAV flight data. For a constant-velocity intruder, their results showed optimal designs where desired minimum separation and maximum trajectory deviation formed a clear trade-off. In separation metrics, maximum deviation was found to decrease weakly as uncertainty decreases, and their dynamic encounters had less difference in metrics with uncertainty because the uncertainty came from the model itself. The minimum sepa-

ration distance increased with uncertainty of the intruder because there is a greater chance of the intruder moving away from a collision trajectory.

In [46], Xue and Rios present the need for a faster-than-real-time simulation platform for sUASs and the impacts and importance of key factors for such a platform. The need for fast-time simulation arises because of the low operation altitude, small size, and large scale of anticipated operations. There are several existing traffic simulation systems in three main categories. The first covers multiple aircraft operations and rules such as CTAS, FACET, ACES, and platforms built by independent researchers. The second category includes simulations dealing with conflict detection and resolution, which tend to focus on encounters between only 2 aircraft. The third category covers simulations developed for simulating a vehicles model and control. Because sUAS navigation errors are more sensitive to wind, vehicle speed, and the vehicle control system because of their low altitude, size, and power, dynamic and controller models are important for a sUAS fast-time simulation platform. The navigation system error plays a significant role, but so does the controllers response to disturbances. In the preliminary study, Xue and Rios applied a PD controller to a trajectory model that was implemented with Coriolis terms, the small-angle approximation, and Newton-Euler equations. They examined the controllers response to different cross-winds. Its performance varies greatly, taking over 50 meters to recover in an 8.7 meter/second crosswind. This response is also worse for greater desired ground speeds. Monte Carlo simulation is important because a statistical study of system parameters is necessary to understand and evaluate the safety of future UTM systems. Parameters and uncertainties may involve many sources, including onboard sensors, navigation and communication devices, right of way rules, collision avoidance rules, weather conditions, and vehicle systems. Monte Carlo simulation is an appropriate tool for this because it is fairly independent of problem dimension, has a rate of convergence of O(1/N), and has a percent error determined by number of simulations run for a given confidence interval. In their Preliminary experiments, Xue and Rios had 6 sUASs with flight plans crossing a region with an added cross wind. They performed 1000 Monte Carlo simulations with varying wind speeds, and simulated collision avoidance conducted with vehicle-to-vehicle communication. The simple

collision avoidance rule was avoid vehicles at less than 20 meters, and give way to aircraft on the right. The simulation showed greater chance of loss of separation and greater flight time at higher average wind speed. Three different collision avoidance maneuvers were tested-turn right, turn left, and hover. Right had no collisions, hover had few and lower average flight time than right, and left performed very poorly, causing collisions in 84% of simulations. A for the type of simulation proposed by Xue and Rios, an effective platform needs models that include more than 100 vehicles per scenario, greater than medium-fidelity vehicle models, vehicle controller models, wind effects, a flight duration that is not limited, small UAS models, and collision avoidance. These simulations also need to be able to be capable of Monte-Carlo simulations. The wind effects need to include along-track, cross-track, and vertical wind effects.

In [5], the authors presented a coordinated path-planning with re-routing method for s-UAS in a 2D Voronoi tessellation of airspace. Paths for vehicles are planned using RRT in the local (within cell) level, and an A* algorithm [47] on the high level.

Mixed-Integer linear programming (MILP) has been used in a number of cases to solve s-UAS sense and avoid problems. MILP has been used with s-UAS in several flight formations with dynamic intruder responses based on ADS-B data. The particular method used in this paper is verified to be faster compared to a standard MILP method [48]. Dynamic optimal s-UAS trajectory planning has been shown in the dynamic sense for the NAS. It shows dynamic waypoint navigation and obstacle avoidance [49]. It has also been used for distributed trajectory optimization using a receding horizon strategy [50]. Another method for handling dynamically challenging s-UAS management is using Grey Wolf Optimization with a Distance Based Value Function [51].

Chapter 3

Modelling Urban Airspace

3.1 Airspace Models

To design and test algorithms for traffic management in a three-dimensional urban environment, an accurate model of an urban environment was first needed. For the purposes of algorithm design, there are many ways to model an urban environment, each with their own advantages and disadvantages. This section describes three different options: Sky Tubes, Voronoii Clusters, and a Grid-Based solution.

3.1.1 Sky Tubes

Sky Tubes are a set of 3-dimensional features that are meant to serve as roads for aircraft. Described by [52], a system of sky tubes is composed of three main elements: horizontal tubes, intersections, and vertical elements. While travelling in a horizontal tube or vertical element, an aircraft maintains a following distance between itself and the next vehicle, like a car on a road. The major difference is vertical elements need to be designed to match the needs of aircraft using them; a simple vertical space would be suitable for a multicopter, while a fixed-wing aircraft requires an inclined space. Intersections are areas where different traffic flows intersect. This system greatly simplifies algorithm design because an elements are not placed through static obstacle, and traffic can be modelled similarly to automobile traffic or fluid-based traffic models. Traffic management

becomes a matter of selecting an appropriate intersection management algorithm, route planning algorithm, and a following distance algorithm. However, constructing a system of Sky Tubes requires substantially more effort, as good locations need to be chosen for intersections and the capabilities of vehicles need to be determined to select appropriate vertical and horizontal elements for the terrain, structures, and no-fly zones. Other models simply exclude volumes based on no-fly zones, terrain, and structures.

3.1.2 Voronoi clusters

In the study [5], Voronoi clusters were used to separate airspace into cells that would be occupied by no more than one aircraft at a time. They used this to demonstrate a preliminary path planning and re-routing system. Random Voronoi clusters chosen randomly may be more like a final airspace model, in that permanent obstacles mean navigation nodes may not need to be distributed uniformly, but this study also did not include a height-map. Their work seemed more geared towards open airspace, although some obstacles were included. The grid-based solution can be considered a highly regular version of the Voronoi solution, with no randomness element.

3.1.3 Grid-Based

The area of interest, in the grid based solution, is divided into rectangular prisms of a specified east-west and north-south resolution, with a height chosen to accommodate the size and maneuverability of the aircraft. At each square in the horizontal grid, there are three elements: the elevation, the height of the tallest obstacle, and the airspace ceiling. The elevation determines where the actual ground is located (not including structures like trees or buildings), and influences the airspace ceiling, which is typically in reference to local elevation. The airspace ceiling is the height above which s-UAS are not allowed to fly for operations in the area. A cell is valid if it is above the minimum clearance to elevation and the tallest obstacle in its footprint, below the minimum clearance from the airspace ceiling, and not within any no-fly zone. It was selected because elevation and tallest obstacle height are easily determined from raw elevation and LIDAR data, airspace

ceiling is determined by adding the maximum allowable height above ground level to the elevation, and no additional processing is needed to determine complex features. The trade-off is this results in a graph with far more nodes than Sky Tubes would, meaning more of the burden is on the path-planning algorithms.

3.2 Building a grid-based Airspace Model

To construct a grid-based airspace model, area of interest is selected, then corresponding data for the elevation and structures within the area. Finally, the data was converted into a grid-based model from point-cloud representation.

3.2.1 Selecting an Area of Interest and Sizing the Grid

To model an urban area, the Greater Cincinnati Area was selected. The area has several features that make it attractive as a model for urban air mobility. It has hills with over 200ft. of elevation change, meaning the model needed to account for the change in elevation when determining airspace ceiling. Additionally, it has buildings that are taller than the airspace ceiling, with the tallest being Great American Tower at 660 ft. This means that some buildings cannot be flown over and must be flown around.

The horizontal size of the grid was chosen to be 8 meters, because the reliability of most off-the-shelf GPS is 7.8m [11]. This could be reduced if higher accuracy GPS become standard. Furthermore, most s-UAS are capable of holding position control under 1m [16], so it may be appropriate to decrease the grid size if strong positional awareness is shown and the airspace becomes more crowded. For vertical spacing, 8m was chosen so that distance calculations were the same independent of direction. A lower distance could be chosen because vertical keep-away distance is typically far less than horizontal keep-away distance [3].

3.2.2 Selecting Model Data

Most models of heights and elevation are only concerned with ground elevation, not building height. LIDAR data sets, however, do include all structures hit by the LIDAR scan. The USGS had a LIDAR Point Cloud model from the USGS National Map 3DEP Downloadable data collection. This covered the entire area of interest and provided building height data in addition to surface height data. Because it included data relevant to the 3D space of interest, it was used to construct the grid-based model. Since LIDAR is a point-cloud model, rather than a raster, this data needed to be interpreted to fit into the grid based model. Cells were selected in 8m intervals over the entire rectangular area of interest. Elevation data was simpler: points were generated at 8m intervals from the southwest corner of the data, and transformed into latitude and longitude coordinates. Google's elevation service was then queried for the points desired.

3.2.3 Interpreting Model Data

A quick examination of the LIDAR data in MATLAB showed a few anamalous point returns that were clearly sensor errors. These were eliminated by removing outliers that were not within 15 median absolute deviations of the median for that tile. Where tiles overlapped, the average maximum height between the two was used to smooth portions of structures that crossed boreders betweeen LIDAR tiles. For a given cell, the tallest point return within the tile was selected as the height of the structure in that cell. A portion of the resulting model is shown in figure 3-1. A closeup of the model is shown in figure 3-2. A satellite image of the area is shown in figure 3-3, for comparison.

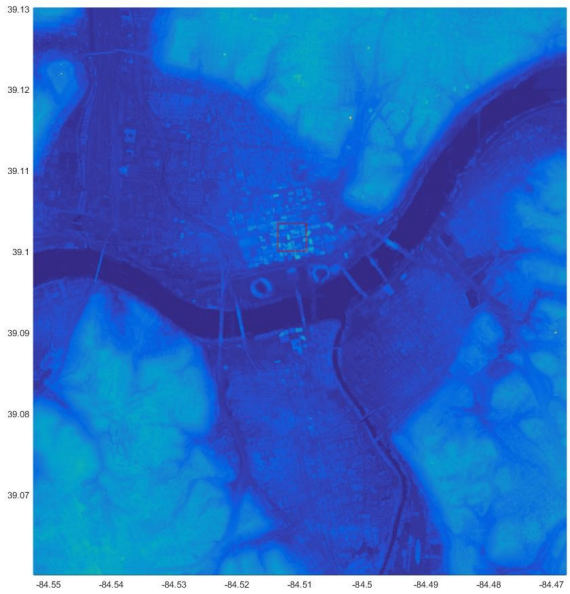

Figure 3-1: Model after interpreting LIDAR data

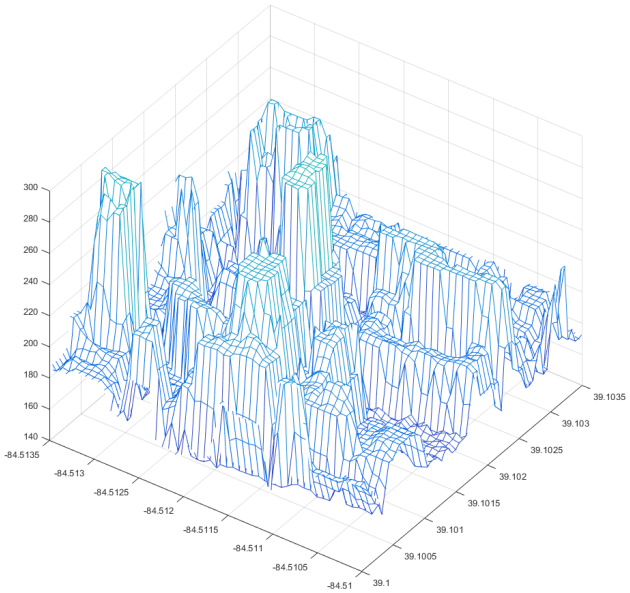

Figure 3-2: Close view of model

Figure 3-3: Satellite view of area modelled

Chapter 4

Top-Level Algorithms

4.1 A*

The first and most revisited top-level algorithm was the A* algorithm. It was chosen for the simplicity of its implementation and ease for interfacing grid-based airspace model. A* is a graph-based search algorithm that keeps track of the cost of its current path and uses a heuristic to select from potential candidates when examining nodes to explore. [47] Because it selects a lower cost path whenever one becomes available, optimallity the solution within the A* graph space is guaranteed. It has been used for top-level graph-based solutions to UTM problems. [5] The graph-based nature of A* made it easy to implement with the grid-based nature of the airspace model, as a grid is a specific type of graph with obvious connectivity.

4.1.1 2D A*

For the Simple 2D A*, nodes are snapped to a grid with 8m spacing and added to the closed set if they are too close to an obstacle. ”Too close” here means that an s-UAS would be less than the MILP safety factor as determined by the s-UAS’s max speed and position uncertainty. To limit the rate at which the tree grows, a visibility criteria of 64m was used. A node was considered visible if it was within a 64m radius of the parent node, and a straight line between the nodes did not come too close to any obstacle. In the simple case, this connectivity was calculated each time A* was

called.

In addition to the simple A*, the algorithm was kept the same, but a preprocessed map was used where all the connectivity of all obstacles was computed before the first call. The criteria for connectivity between nodes was kept the same between A* calls.

For both cases, the same A* algorithm was used. The initial and final coordinates were snapped to the closest node that was not within an obstacle. Pseudocode for this algorithm is shown in pseudocode 1.

Algorithm 1: A* pseudocode

```
startNode.coordingates ⟵ start coordinates
startNode.parent ⟵ no parent
startNode.gScore ⟵ 0
startNode.fScore ⟵ 0
for each other node in nodes do
    node.parent ⟵ no parent
    node.gScore ⟵ 0
    node.fScore ⟵ 0
OpenSet ⟵ new PriorityQueue
ClosedSet ⟵ new set
OpenSet.add(startNode)
ClosedSet.add(invalidnodes)
while not_empty(OpenSet) do
    bestNode ⟵ OpenSet.pop()
    ClosedSet.add(bestNode)
    if bestNode.coordinates == goal coordinates then
        Path ⟵ new set
        currentNode ⟵ bestNode
        while currentNode has a parent do
            Path.prepend(currentNode)
            currentNode ⟵ currentNode.parent
        return path
    for each neighbor of bestNode do
        if neighbor not in ClosedSet then
            tentativeGScore ⟵ bestNode.gScore + distance(bestNode, neighbor)
            if tentativeGScore < neighbor.gScore then
                neighbor.gScore ⟵ tentativeGScore
                neighbor.fScore ⟵ neighbor.gScore + distance(neighbor, goal)
                neighbor.parent ⟵ bestNode
                OpenSet.update(neighbor)
return path unavailable message
```

4.1.2 3D A*

The 3D A* algorithms worked similarly to the 2D algorithms, except expanded to 3 dimensions. Since quadcopters were chosen, paths were planned to be horizontal at any angle or vertical one 8m node at a time. However, the 3D MILP planner was not strong enough to work with a 3D A* algorithm.

4.2 Rapidly-exploring Random Tree

A rapidly-exploring random tree algorithms was considered as an alternative to the 2D A*, and to solve dimensionality problems in the three dimensional case. The main reason for this is RRT is supposed to suffer less from the ”curse of dimensionality” associated with other search algorithms. [35] RRT has also been shown to give consistently sub-optimal solutions, although they may be corrected with additional improvements. [53]. In this implementation, the RRT was a poor fit for the grid-based system, and was not able to take advantage of the same preprocessing as A*. RRT methods typically favor continuous maps, as they generate points on their own, while A* must use a pre-generated set of points or method for generating points. RRT was not selected as a top level algorithm for MILP because initial testing of the implementation had a tree that grew very slowly. The A* algorithm simply matched the data model better.

4.3 Transition between levels

To transition between the A* or RRT algorithm and the MILP algorithms, several steps were used. MILP calculates trajectories, which have a time component, rather than paths, which are just a series of points. Therefore in order for the waypoints presented by A* to be use-able by MILP, the points were waypoints were selected such that all original waypoints and dwell times from the flight request were included, and waypoints were included such that the expected time between any two sequential waypoints was less than but close to 40 times the length of a MILP time step for a vehicle travelling at cruise velocity. MILP algorithm did not have to perform routing duties around large obstacles, but and also did not have to find a path through more waypoints than necessary. This allows the MILP algorithm to be flexible in its solutions while also running at a good speed.

Chapter 5

Lower Level type 1: Mixed Integer Linear Programming Algorithms

5.1 Main Section 1

Mixed Integer Linear Programming, or MILP, is an optimization method that solves problems based on a set of linear inequalities. For the initial algorithm trials, the drone was assumed to be a DJI phantom drone, the obstacle data was taken from a 2007 LIDAR survey of the state of Ohio. Downtown Cincinnati was chosen because it allowed us to prove out avoiding tall buildings relative to ground level while also being able to deal with hills. The initial algorithms were 2D MILP algorithms based on the work of Mohammadreza Radmanesh. [48] [49] An Initial trial showed 10 s-UAS entering an area of airspace at once, which was centered on a tall building. The s-UAS were given priority in first-come, first-served order, and their paths were planned using MILP. Subsequent MILP calls listed the predicted positions of all paths planned using MILP so far. If MILP could not find a solution, the request for airspace was instead delayed until MILP could find a solution.

5.1.1 Existing Work and Inequality Framework

In adapting Radmanesh's work [54] [48] [54] [49] to the urban problem, obstacles are represented by 8m x 8n meter rectangles, where m and n are nonzero integers. Each obstacle has minimum and maximum x and y coordinates. Each vehicle has a minimum and maximum speed, a mass, and a maximum acceleration. Each vehicle submits a flight plan at a specified time with a start point, end point, and K destinations, where K is a nonnegative integer.

MILP functions by solving a set of mixed-integer linear programming equations with the Gurobi package. When called, it is given a time during which the s-UAS is expected to traverse a waypoint or set of waypoints from its initial state and a finite horizon it is expected to stay within during this timeframe, in addition to vehicle, intruder trajectory, and obstacle information.

The first constraint is the initial state, i.e., the position and velocity:

$$x_1 = x_{initial} \qquad y_1 = y_{initial} \qquad v_{x1} = v_{x,initial} \qquad v_{y1} = v_{y,initial} \tag{5.1}$$

Where $x_{initial}, y_{initial}, v_{x,initial}, v_{y,initial}$ are all provided by the flight plan or a previous iteration of MILP. x_1, y_1, v_{x1}, v_{y1} are all of form x_n, y_n, v_{xn}, v_{yn}, where n is the nth timestep, here equal to 1. The finite horizon constraint is given by:

$$x_{initial} - L_{horizon} \leq x_n \leq x_{initial} + L_{horizon} \; \forall \, n \in N_{steps} \tag{5.2}$$

$$y_{initial} - L_{horizon} \leq y_n \leq y_{initial} + L_{horizon} \; \forall \, n \in N_{steps} \tag{5.3}$$

$$N_{steps} = \left\{ n \in Z^+ \, | n \leq ceil\left(\frac{T}{T_d}\right) \right\} \tag{5.4}$$

Where $L_{horizon}$ is the length of the finite horizon and T is the length of time the segment is to be completed in. The constraints for the dynamic model are given by:

$$S_{n+1} = A_d S_n + B_d F_n \; \forall \left\{ n \in N_{steps} | n < ceil\left(\frac{T}{T_d}\right) \right\} \tag{5.5}$$

$$S_n = \begin{bmatrix} x_n \\ y_n \\ v_{xn} \\ v_{yn} \end{bmatrix} \qquad A_d = \begin{bmatrix} 1 & 0 & T_d & 0 \\ 0 & 1 & 0 & T_d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

$$B_d = \begin{bmatrix} \frac{T_d^2}{m} & 0 \\ 0 & \frac{T_d^2}{m} \\ \frac{T_d}{m} & 0 \\ 0 & \frac{T_d}{m} \end{bmatrix} \qquad F_n = \begin{bmatrix} f_{xn} \\ f_{yn} \end{bmatrix} \tag{5.7}$$

where m is the mass of the vehicle, f_{xn} and f_{yn} are the force of the vehicle in the x and y directions, respectively. Acceleration and velocity constraints are given by:

$$f_{xn} sin\left(\frac{2\pi h}{H}\right) + f_{yn} cos\left(\frac{2\pi h}{H}\right) \leq f_{max} \tag{5.8}$$

$$\forall n \in N_{steps},\ h \in Z^+,\ h \leq H \tag{5.9}$$

$$v_{xn} sin\left(\frac{2\pi h}{H}\right) + v_{yn} cos\left(\frac{2\pi h}{H}\right) - v_{min} \leq (v_{max} - v_{min})(1 - b_{speed,h,n}) \tag{5.10}$$

$$\forall n \in N_{steps},\ h \in Z^+,\ h \leq H \tag{5.11}$$

$$v_{xn} sin\left(\frac{2\pi h}{H}\right) + v_{yn} cos\left(\frac{2\pi h}{H}\right) - v_{min} \geq \eta + (-v_{min} - v_{max} - \eta) b_{speed,h,n} \tag{5.12}$$

$$\forall n \in N_{steps},\ h \in Z^+, h \leq H \tag{5.13}$$

$$\sum_{h=1}^{H} b_{speed,h,n} \leq H - 1 \tag{5.14}$$

$$\forall n \in N_{steps} \tag{5.15}$$

Where f_{max} is the maximum force the vehicle can generate, H is an integer chosen to discretize the circle, v_min and v_max are the minimum and maximum velocity of the vehicle, respectively,

$b_{speed,h,n}$ is the binary variable for velocity at time step n in direction h, and η is a very small number. If the minimum horizontal velocity is zero, as in multicopter vehicles, the binary variables $b_{speed,h,n}$ can be excluded, and the constraint on velocity becomes:

$$v_{xn} sin\left(\frac{2\pi h}{H}\right) + v_{yn} cos\left(\frac{2\pi h}{H}\right) \leq v_{max} \ \forall n \in N_{steps},\ h \in Z^+,\ h \leq H \tag{5.16}$$

The constraints for arriving at the waypoints were modified to allow the vehicles to dwell at each waypoint for a specified amount of time. The constraints for this are given by:

$$x_n - x_w \leq r_{min} + M_{big}(1 - b_{w,n}) \qquad \forall n \in N_{steps},\ w \in Z^+,\ w \leq W \tag{5.17}$$

$$y_n - y_w \leq r_{min} + M_{big}(1 - b_{w,n}) \qquad \forall n \in N_{steps},\ w \in Z^+,\ w \leq W \tag{5.18}$$

$$x_n - x_w \leq - r_{min} - M_{big}(1 - b_{w,n}) \qquad \forall n \in N_{steps},\ w \in Z^+,\ w \leq W \tag{5.19}$$

$$y_n - y_w \leq - r_{min} - M_{big}(1 - b_{w,n}) \qquad \forall n \in N_{steps},\ w \in Z^+,\ w \leq W \tag{5.20}$$

$$b_{w,1} = b_{TOA,w,1} \qquad \forall w \in Z^+,\ w \leq W \tag{5.21}$$

$$b_{w,n} = \left[\sum_{t=1}^{n} b_{TOA,w,t}\right] - \sum_{t=1}^{n-1} b_{TOD,w,t} \qquad \forall n \in N_{steps},\ w \in Z^+,\ w \leq W \tag{5.22}$$

$$\left[\sum_{t=1}^{N_{MAX}} n(b_{TOD,w,t})\right] - \left[\sum_{t=1}^{N_{MAX}} n(b_{TOA,w,t})\right] \geq W_{dwell,w} \ \forall w \in Z^+,\ w \leq W \tag{5.23}$$

$$\sum_{t=1}^{N_{MAX}} b_{TOA,w,t} = 1 \qquad \forall w \in Z^+,\ w \leq W \tag{5.24}$$

$$\sum_{t=1}^{N_{MAX}} b_{TOD,w,t} = 1 \qquad \forall w \in Z^+,\ w \leq W \tag{5.25}$$

$$N_{MAX} = ceil\left(\frac{T}{T_d}\right) \tag{5.26}$$

$$r_{min} = \frac{m v_{min}^2}{f_{max}} \tag{5.27}$$

$$\sum_{t=1}^{n} b_{TOA,w+1,t} \leq \sum_{t=1}^{n} b_{TOD,w,t} \qquad \forall n \in N_{steps},\ w \in Z^+,\ w \leq W \tag{5.28}$$

Where x_w and y_w are the coordinates of waypoint w. M_{big} is a number that is large with respect to the finite horizon. $b_{w,n}$ is the binary variable for being at waypoint w at time step n. W is the number of waypoints. $b_{TOA,w,n}$ and $b_{TOD,w,n}$ are binary variables for arriving at and departing from waypoint w at time step n, respectively. $W_{dwell,w}$ is the amount of time the vehicle is instructed to dwell at waypoint w.

To prevent the vehicle's path from intersecting with obstacles or intruders between time steps, a safety factor is used. The Constraints on this safety factor are:

$$-S_{f,x,n} \leq v_{x,n} T_d + \frac{f_{x,n} T_d^2}{2m} + \sigma \leq S_{f,x,n} \qquad \forall n \in N_{steps} \tag{5.29}$$

$$-S_{f,y,n} \leq v_{y,n} T_d + \frac{f_{y,n} T_d^2}{2m} + \sigma \leq S_{f,y,n} \qquad \forall n \in N_{steps} \tag{5.30}$$

Where σ is the minimum safe distance allowed between the vehicle and an obstacle or intruder $S_{f,x,n}$ and $S_{f,y,n}$ are the safety factors in the x direction and y direction, respectively. The equations

to prevent obstacle collisions are:

$$x_n \leq x_{min,o} - S_{f,x,n} + M_{big} b_{obs,o,n,1} \qquad \forall n \in N_{steps},\ o \in Z^+,\ o \leq O \tag{5.31}$$

$$-x_n \leq -x_{max,o} - S_{f,x,n} + M_{big} b_{obs,o,n,2} \qquad \forall n \in N_{steps},\ o \in Z^+,\ o \leq O \tag{5.32}$$

$$y_n \leq y_{min,o} - S_{f,y,n} + M_{big} b_{obs,o,n,3} \qquad \forall n \in N_{steps},\ o \in Z^+,\ o \leq O \tag{5.33}$$

$$-y_n \leq -y_{max,o} - S_{f,y,n} + M_{big} b_{obs,o,n,4} \qquad \forall n \in N_{steps},\ o \in Z^+,\ o \leq O \tag{5.34}$$

$$\sum_{k=1}^{4} b_{obs,o,n,k} \leq 3 \qquad \forall n \in N_{steps},\ o \in Z^+,\ o \leq O \tag{5.35}$$

Where $x_{min,o}$, $x_{max,o}$, $y_{min,o}$, and $y_{max,o}$ are the minimum x, maximum x, minimum y, and maximum y of obstacle o, respectively. $b_{obs,o,n,k}$ with $k = 1, 2, 3, and 4$ are the binary variables for being outside of the minimum x, maximum x, minimum y, and maximum y, respectively, of obstacle o at time step n. O is the total number of obstacles in the finite horizon. A similar set of constraints is used to construct the intruder avoidance.

$$x_n \leq x_{min,i,n} - 2S_{f,x,n} + M_{big} b_{i,1,n} \qquad \forall n \in N_{steps},\ i \in Z^+,\ i \leq I \tag{5.36}$$

$$-x_n \leq -x_{max,i,n} - 2S_{f,x,n} + M_{big} b_{i,2,n} \qquad \forall n \in N_{steps},\ i \in Z^+,\ i \leq I \tag{5.37}$$

$$y_n \leq y_{min,i,n} - 2S_{f,y,n} + M_{big} b_{i,3,n} \qquad \forall n \in N_{steps},\ i \in Z^+,\ i \leq I \tag{5.38}$$

$$-y_n \leq -y_{max,i,n} - 2S_{f,y,n} + M_{big} b_{i,4,n} \qquad \forall n \in N_{steps},\ i \in Z^+,\ i \leq I \tag{5.39}$$

$$\sum_{k=1}^{4} b_{i,k,n} \leq 3 \qquad \forall n \in N_{steps},\ i \in Z^+,\ i \leq I \tag{5.40}$$

Here $x_{min,i,n}$, $x_{max,i,n}$, $y_{min,i,n}$, and $y_{max,i,n}$ are the minimum x, maximum x, minimum y, and maximum y, respectively, of intruder i at time step n. $b_{i,k,n}$, with k = 1, 2, 3, and 4 are binary variables for excluding the vehicle from the area covered by intruder i at time step n. I is the total number of intruders in the timeframe and finite horizon. The cost function for this 2D maneuver has

acceleration and time components. The time component is given by:

$$J_{time} = \sum_{w=1}^{W} \sum_{n=1}^{N_{MAX}} (n-1) b_{TOD,w,n} \tag{5.41}$$

The acceleration component is given by:

$$J_{force} = \sum_{k=1}^{2} \sum_{n=1}^{N_{MAX}} W_{slack,k,n} \tag{5.42}$$

Where $W_{slack,k,n}$ are slack variables constrained by:

$$f_{xn} \leq W_{slack,1,n} \qquad \forall n \in N_{steps} \tag{5.43}$$

$$-f_{xn} \leq W_{slack,1,n} \qquad \forall n \in N_{steps} \tag{5.44}$$

$$f_{yn} \leq W_{slack,2,n} \qquad \forall n \in N_{steps} \tag{5.45}$$

$$-f_{yn} \leq W_{slack,2,n} \qquad \forall n \in N_{steps} \tag{5.46}$$

5.2 Inequalities to move to three dimensions

To modify the MILP equations for three dimensions, equation 5.5 becomes:

$$S_{n+1} = A_d S_n + B_d F_n \ \forall n \in N_{steps},\ n \leq N_{MAX} \tag{5.47}$$

Where:

$$A_d = \begin{bmatrix} 1 & 0 & 0 & T_d & 0 & 0 \\ 0 & 1 & 0 & 0 & T_d & 0 \\ 0 & 0 & 1 & 0 & 0 & T_d \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad S_n = \begin{bmatrix} x_n \\ y_n \\ z_n \\ v_x n \\ v_y n \\ v_z n \end{bmatrix} \tag{5.48}$$

$$B_d = \begin{bmatrix} \frac{T_d^2}{2m} & 0 & 0 \\ 0 & \frac{T_d^2}{2m} & 0 \\ 0 & 0 & \frac{T_d^2}{2m} \\ \frac{T_d}{m} & 0 & 0 \\ 0 & \frac{T_d}{m} & 0 \\ 0 & 0 & \frac{T_d}{m} \end{bmatrix} \tag{5.49}$$

The acceleration and velocity constraints become:

$$V_n^T \eta_{h,j} \leq v_{max} \ \forall n \in N_{steps}, \{h \in Z \,|\, 0 \leq h < H\}, \{j \in Z \,|\, 0 \leq j < H/2\} \tag{5.50}$$

$$F_n^T \eta_{h,j} \leq f_{max} \ \forall n \in N_{steps}, \{h \in Z \,|\, 0 \leq h < H\}, \{j \in Z \,|\, 0 \leq j < H/2\} \tag{5.51}$$

Where:

$$V_n = \begin{bmatrix} v_{xn} \\ v_{yn} \\ v_{zn} \end{bmatrix} \qquad \eta_{h,j} = \begin{bmatrix} cos(\theta_h)cos(\phi_j) \\ sin(\theta_h)cos(\phi_j) \\ sin(\phi_j) \end{bmatrix} \tag{5.52}$$

$$\theta_h = \frac{2\pi h}{H} \qquad \phi_j = \frac{2\pi j}{\frac{H}{2} - 1} - \frac{\pi}{2} \tag{5.53}$$

Unlike the obstacles in the 2D version, the 3D version uses a grid of terrain heights. The first 3 constraints determine whether the vehicle is over a grid square:

$$-\frac{L_{grid}}{2}b_{grid,p,n} - M_{big,x}(1 - b_{grid,p,n}) \leq x_n - x_{grid,p} \qquad \forall n \in N_{steps},\ p \in [1, P] \tag{5.54}$$

$$\frac{L_{grid}}{2}b_{grid,p,n} + M_{big,x}(1 - b_{grid,p,n}) \geq x_n - x_{grid,p} \qquad \forall n \in N_{steps},\ p \in [1, P] \tag{5.55}$$

$$-\frac{L_{grid}}{2}b_{grid,p,n} - M_{big,y}(1 - b_{grid,p,n}) \leq y_n - y_{grid,p} \qquad \forall n \in N_{steps},\ p \in [1, P] \tag{5.56}$$

$$\frac{L_{grid}}{2}b_{grid,p,n} + M_{big,y}(1 - b_{grid,p,n}) \geq y_n - y_{grid,p} \qquad \forall n \in N_{steps},\ p \in [1, P] \tag{5.57}$$

$$1 \leq \sum_{p=1}^{P} b_{grid,p,n} \leq 4 \qquad \forall n \in N_{steps} \tag{5.58}$$

Where L_{grid} is the size of a grid cell, $M_{big,x}$ and $M_{big,y}$ are just larger than the x and y horizons, respectively. $b_{grid,p,n}$ is a binary variable for whether the vehicle is over grid cell p at time step n. $x_{grid,p}$ and $y_{grid,p}$ are the x and y coordinates of the center of grid cell p. P is the total number of grid cells. The vehicle is kept above the terrain by a safety factor:

$$z_n - z_{grid,low,p} \geq b_{grid,p,n}S_{f,z} - M_{big,z}(1 - b_{grid,p,n})\ \forall n \in N_{steps},\ p \in [1, P] \tag{5.59}$$

$$-z_n + z_{grid,high,p} \geq b_{grid,p,n}S_{f,z} - M_{big,z}(1 - b_{grid,p,n})\ \forall n \in N_{steps},\ p \in [1, P] \tag{5.60}$$

Where $z_{grid,low,p}$ and $z_{grid,high,p}$ are the lower and upper limits of grid cell p, respectively. $S_{f,z}$ is the safety factor in the z direction. To find the cost, the slack variables for force were calculated similarly to the 2D version:

$$J_f = \sum_{n=1}^{N_{MAX}} \sum_{k=1}^{3} W_{slack,n,k} \tag{5.61}$$

Where $W_{slack,n,k}$ is the slack variable constrained by:

$$f_{xn} \leq W_{slack,n,1} \ \forall n \in N_{steps} \tag{5.62}$$

$$-f_{xn} \leq W_{slack,n,1} \ \forall n \in N_{steps} \tag{5.63}$$

$$f_{yn} \leq W_{slack,n,2} \ \forall n \in N_{steps} \tag{5.64}$$

$$-f_{yn} \leq W_{slack,n,2} \ \forall n \in N_{steps} \tag{5.65}$$

$$f_{zn} \leq W_{slack,n,3} \ \forall n \in N_{steps} \tag{5.66}$$

$$-f_{zn} \leq W_{slack,n,3} \ \forall n \in N_{steps} \tag{5.67}$$

Initial and final conditions were specified in a similar manner to the 2D version, but as a trial run the start and endpoint were simply fixed, as seen here:

$$S_1 = \begin{bmatrix} x_{initial} \\ y_{initial} \\ z_{initial} \\ v_{x,initial} \\ v_{y,initial} \\ v_{z,initial} \end{bmatrix} \quad S_{N_{MAX}} = \begin{bmatrix} x_{final} \\ y_{final} \\ z_{final} \\ v_{x,final} \\ v_{y,final} \\ v_{z,final} \end{bmatrix} \tag{5.68}$$

The 3D equations were used in several trial runs but were severely hindered by area complexity. This will be further discussed in section 5.2.1.

5.2.1 Area Complexity

Larger finite horizons improve the optimality of solutions, however, they increase the time to solve the MILP optimization problems for several reasons. The first is simply a larger finite horizon in an obstacle-rich environment means more obstacles proportional to the area of the finite horizon. Also, the distance a s-UAS can cover is limited by its top speed given a finite number

of time steps. Therefore, in order to cover a larger finite horizon, the number of time steps in the simulation must be increased. This means there are $N \times M$ equations that must be solved, where N is the number of time steps, and M is the number of equations that must be solved per time step. Therefore, there is an upper limit on the finite that MILP equations can be applied to while still allowing for acceptable and speedy solutions.

To combat this, the finite horizon was set to a value that allows for good solutions in a short amount of time. Because this limits the possible missions that can be flown to a finite horizon, this was expanded by solving MILP equations with waypoints generated by A* in a sequence of appropriate finite areas based on start and end points of each area. A complete description of this process can be found in section 5.

5.2.2 Three-dimensional Complexity

Transitioning MILP into three dimensions was not as successful as its two dimensional counterpart. This was due the computational complexity added from the third dimension. s-UAS motion was discretized as a sphere, rather than a circle, which added an additional set of constraints on top of each set for an individual dimension shown above. Additionally, obstacles were now more complex. Rather than the sets of rectangles used in the two-dimensional case, a binary grid was used to determine which sector the s-UAS was in. In that sector, the s-UAS was constrained to be above the floor for that sector. This makes the number of obstacles much larger than the simplified rectangles used in the two dimensional case. This was all clearly reflected in the number of time steps that could be computed comfortably. The 2D case comfortably managed 60 time steps, while the 3D case struggled to perform more than 10. A less computationally intense solution was pursued because of the difficulties with the 3D complexity.

Chapter 6

Lower-level type 2: Re-routing using A*

6.1 Motivation

Because of the complexities and computational demands associated with full path planning using MILP and a top-level algorithm, exploring a simpler method of three dimensional path planning and traffic management was desirable. Rather than model the full physics of every vehicle, the vehicle motion would be simplified, allowing a greater quantity of vehicles to be simulated over larger areas and time steps. For this purpose, the path planning and re-routing method was used.

6.1.1 Path Planning and Re-Routing

The path planning and re-routing method divides the airspace up into sections, which only one s-UAS may occupy at a time. This follows and expands upon work done in [5]. For the purposes of this study, airspace was divided into 8m cubes. The path planning and re-routing method has several major components. The first, the initial routing algorithm determines a path from a vehicle's starting point to its goal agnostic of the positions of intruders in the airspace. Secondly, the vehicles are allowed to fly their paths until a conflict is detected. When a conflict is detected, a resolution method is used to determine which UAS to re-route solve the conflict. In this case, a priority-based solution was used, but there are other methods, like a market-based solution.

Additionally, a method to resolve traffic conflicts, where two s-UAS are occupying each other's global destination was used to handle edge cases.

6.2 Routing Algorithm

The initial routing algorithms follow a simple 3D A* approach. The inputs are the vehicle position and its global destination. The A* algorithm works as described in 5, but no visibility criteria is used and the s-UAS is only permitted to travel from one 8m cube to one of the cube's neighbors via one of the cube's faces. The s-UAS were not permitted to travel through a neighboring cube if it was occupied by an obstacle. The re-routing algorithm was identical to the initial routing algorithm, but in addition to the obstacle restrictions, the vehicle was not permitted to route through a cube currently occupied by a vehicle or that would be occupied by a vehicle in the next time step. In the priority case, only vehicles that have no option to move or higher-priority vehicles are considered in this manner. The pseudocode for the priority-based solution is shown in pseudocode 2 and 3.

6.3 Conflict Detection

The Sense-And-Avoid algorithm detects conflicts using the current position, current heading and velocity, and planned mission of the s-UAS. For all t in a time window greater than or equal to the stopping time for all vehicles in the airspace, the position is predicted for all s-UAS as in figure 6-1. Outside of more complicated commands, a UAS will attempt to reach the next waypoint by following a linear path. A conflict is detected if two or more UASs have their keep-away volumes intersect at a time t. The safety radius and horizontal separation for each s-UAS is determined from the sum of error in position and velocity measurements, the s-UAS's maneuverability, and the expected maximum deviation from the s-UAS's mission trajectory. Each s-UAS has an initial mission entered by a user. Straight lines between waypoints for this mission may conflict with known obstacles or geofences, as shown in figure 6-2. Because of this, for each pair of waypoints

the A* conflict avoidance algorithm provides a set of waypoints that form a viable path, as seen in figure 6-3. The conflict avoidance algorithm is also called whenever a conflict is detected. Each s-UAS has a priority rating based on its remaining battery life, remaining mission time, and any other factors for mission importance. In a conflict, the lower-priority s-UAS is re-routed to avoid the keep-away volumes of each s-UAS of higher priority.

Algorithm 2: 3D path-planning with re-routing: initial path planning

```
//set up terrain as 8m cubes that are either occupied or not
emptyTerrain ⟵ setupTerrainModel()
//calculate initial routes
for each s-UAS, s do
    path(s) ⟵ AStar(start(s), goal(s), emptyTerrain)
    currentPosition(s) ⟵ start(s)
    currentGoal(s) ⟵ path(s, waypoint = 2)
    currentVelocity(s) ⟵ 0
```

Algorithm 3: 3D path-planning with re-routing: rerouting section

```
//check and resolve conflicts
while any s-UAS hasn't reached its final goal do
    //keep track of positions currently occupied and positions that will be occupied in the
      near future
    reservedTerrain ⟵ emptyTerrain
    //update positions of s-UAS and remove s-UAS that have reached their goal
    for each s-UAS, s do
        if path(s) not complete then
            update currentPosition(s) based on currentVelocity(s)
            update currentGoal(s) based on currentPosition(s) and path(s)
            if currentPosition(s)==end(s) then
                path(s) is complete
            else
                update remaining flight time of s
                determine priority(s) based on nearness to its final goal and remaining flight
                  time
                update the position in reserved terrain occupied by s to inaccesible
        else
            remove s from the model
    //check if the next position will cause a conflict, and resolve it
    Reset Vacate variable used to resolve edge cases
    for s-UAS, in order of priority do
        set desiredVelocity(s) based on currentGoal(s), max speed of s, and currentPostion(s)
        temporaryTerrain ⟵ emptyTerrain for all spaces not neighbors of
          currentPosition(s)
        temporaryTerrain ⟵ reservedTerrain for all spaces that are neighbors of
          currentPosition(s)
        set futurePosition(s) based on currentPosition(s) and desiredVelocity(s)
        //see 6.4 for more details on the following
        resolve traffic jam edge cases
        if futurePosition(s) is inaccassible in temporaryTerrain then
            if no reroute is possible then
                desiredVelocity ⟵ 0
            else
                reroute ⟵ AStar(currentPosition(s), end(s), temporaryTerrain)
                modify path(s) to indicate the re-route
                update currentGoal(s)
                update desiredVelocity(s)
            predict futurePosition(s)
        update futurePosition(s) in reservedTerrain to inaccessible
```

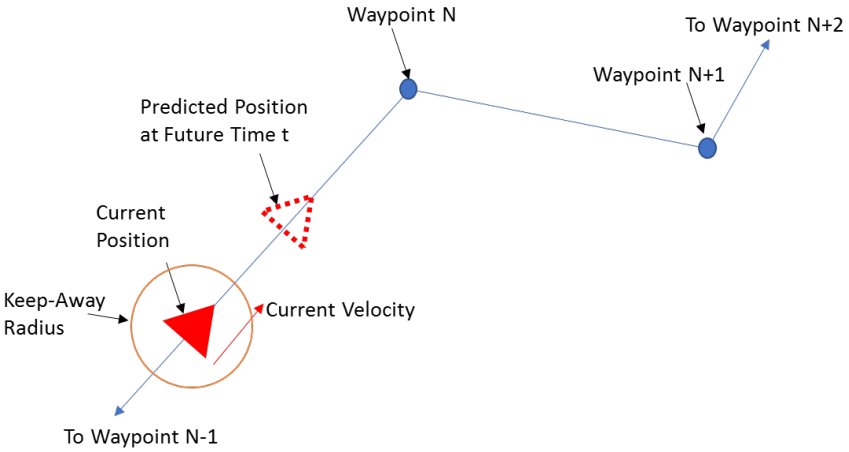

Figure 6-1: Predicted Path for s-UAS

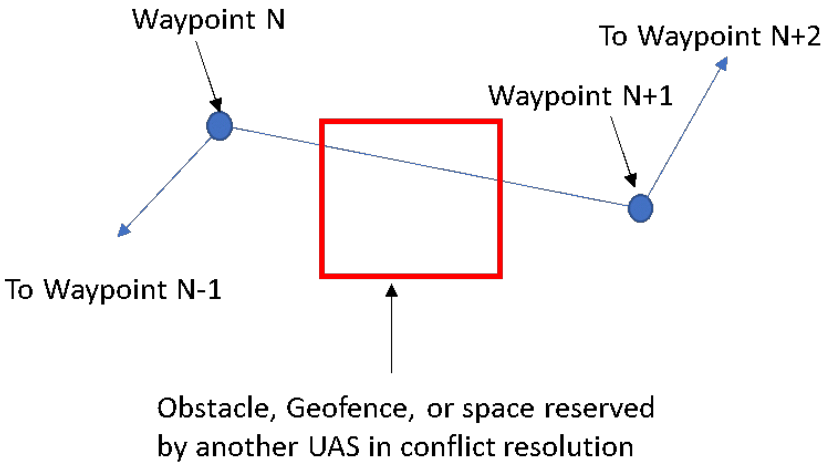

Figure 6-2: Path for s-UAS with an obstacle in the way

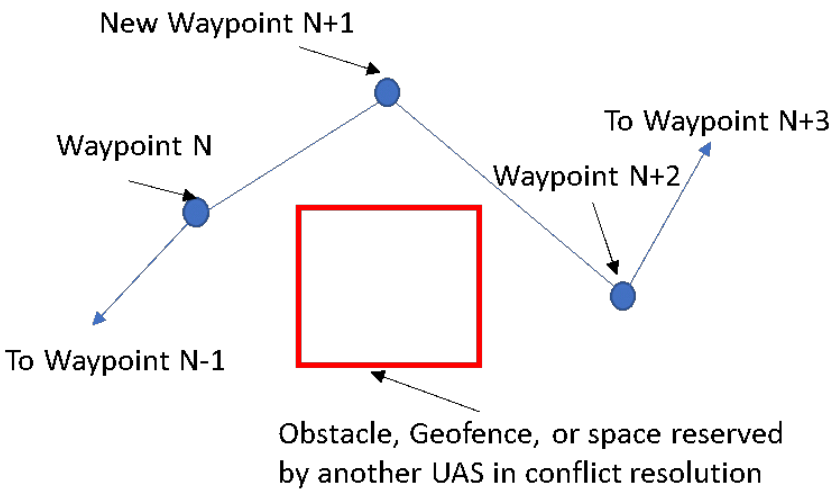

Figure 6-3: Corrected Path for s-UAS

6.4 Resolving Traffic Jams

The re-routing algorithm can encounter a problem if two s-UAS occupy each other's final goals. A* for either s-UAS does not find a solution if used independently, because the final goal position is occupied. To remedy this, an s-UAS will ask for its final goal to be vacated if the final goal is the future position it will occupy next. If a s-UAS is asked to vacate a position, it finds the open neighbor that has the shortest A* path to its final goal, and chooses that as its current goal, updating its desired velocity and future position appropriately. If the vacating vehicle has no accessible neighbors in the temporary terrain, it waits, and asks that its own final goal be vacated if it is occupied. The pseudocode for this is shown below, in pseudocode 4.

Algorithm 4: Resolving Traffic Jam Edge Cases

```
if futurePosition(s) == goal(s)  &  futurePosition(s) is inaccassable in temporaryTerrain
 then
    if NOT vacate(currentPosition(s)) then
        vacate(goal(s)) ⟵ True
        desiredVelocity(s) ⟵ 0
        predict futurePosition(s)
    else
        shortestLength ⟵ infinity
        amendingPath ⟵ empty path
        for each neighbor n of currentPosition(s) do
            if temporaryTerrain(n) is inaccessible then
                continue
            else
                tempPath ⟵ AStar(n, goal(s), emptyTerrain)
                if Length(tempPath) < shortestLength then
                    amendingPath ⟵ tempPath
                    shortestLength ⟵ Length(tempPath)
        if no path was found then
            currentVelocity(s) ⟵ 0
        else
            amend path(s) so from currentGoal(s) on is replaced by amendingPath
            update desiredVelocity(s)
            predict futurePosition(s)
```

Chapter 7

Numerical Simulations And Results

To examine the effectiveness of the algorithms, several tests were run. The MILP lower level algorithm, A* top level, and the priority and re-routing algorithms were all inspected individually.

7.1 MILP results

For all simulations using MILP, the capabilities of the vehicles needed to be specified. The simulations used the vehicle statistics listed in table 7.1. These were similar to the performance characteristics advertised for the DJI phantom.

Table 7.1: Vehicle Statistics used in MILP simulations

vehicle property	value
maximum horizontal velocity	20 m/s
minimum horizontal velocity	0 m/s
mass	1.38 kg
maximum horizontal force	4.9 N
position uncertainty	1.5 m
maximum flight time	1680 s

In all simulations, s-UAS were generated with random mission. These missions consisted of a random start point and random end point. Both points were generated a safe distance away from

all obstacles.

7.1.1 2D MILP without top-level

For the 2D MILP case without a top-level algorithm, the maximum number of s-UAS that could enter a space was examined. For all 2D MILP without the top level s-UAS capacity simulations, a 3584 m^2 area was chosen. s-UAS were added until MILP was unable to generate a solution for three consecutive random missions. The third simulation had an additional complication: after no more missions could be generated for the area at time t=1 second, random missions were generated at time t=2 seconds, until that time period was also congested with s-UAS. This continued until t=n seconds, where n is the time at which no solutions could be found for 5 consecutive time steps. At this point, the airspace could be considered near or at saturation. This was to demonstrate the MILP algorithm's ability to continue to integrate new flight plans as new flight plan requests are made. A summary of the situation for the 2D MILP capacity test is shown in table 7.2.

Table 7.2: Lower-level MILP capacity test

Trial	Area (m^2)	Obstacles	s-UAS simulated	total runtime (s)
simple capacity 1	3584	1	14	88.19
simple capacity 2	3584	1	16	94.92
continuous capacity	3584	1	22	930.7

The continuous capacity trial was stopped earlier than originally intended because the algorithm was taking over 15 minutes to find a route for the 23rd s-UAS. Since this time is far longer than the algorithm is meant to run, this was considered the algorithm's capacity.

The statistics for separation between s-UAS during the capacity simulations are shown in table 7.3. They show that the s-UAS remain safely separated even as the number of s-UAS in the simulation increases.

Table 7.3: Separation Results for the MILP capacity test

Trial	1	2	3
Average s-UAS separation (m)	25.72	27.41	29.78
Minimum s-UAS separation (m)	6.28	6.34	6.50
Standard deviation, s-UAS separation (m)	4.00	0.940	0.0623
Average obstacle separation (m)	31.03	34.22	30.56
Minimum obstacle separation (m)	14.15	13.78	3.239
Standard deviation, obstacle separation (m)	67.10	89.62	115.98

Additionally, the 2D MILP simulation was tested with 2 s-UAS with random paths, each starting at t=1 second, but with increasing operating area and finite horizon. The size of the finite horizons and the time these trials took is shown in table 7.4. Trial 4 was aborted due to long run time, but the inability of the lower level MILP to handle area complexity efficiently is clear.

Table 7.4: Simple MILP distance capacity

Trial	Area (m^2)	Time steps permitted	Obstacles	Total runtime (s)
distance 1	1024	98	0	17.50
distance 2	3584	184	1	32.52
distance 3	12544	344	8	90.52
distance 4	43264	640	30	>250

Finally, it was observed that the lower-level MILP algorithm experienced the most difficulty when it had to take a significant detour between its starting point and its goal. Figure 7-1 shows the start and goal locations for MILP distance trial 4.

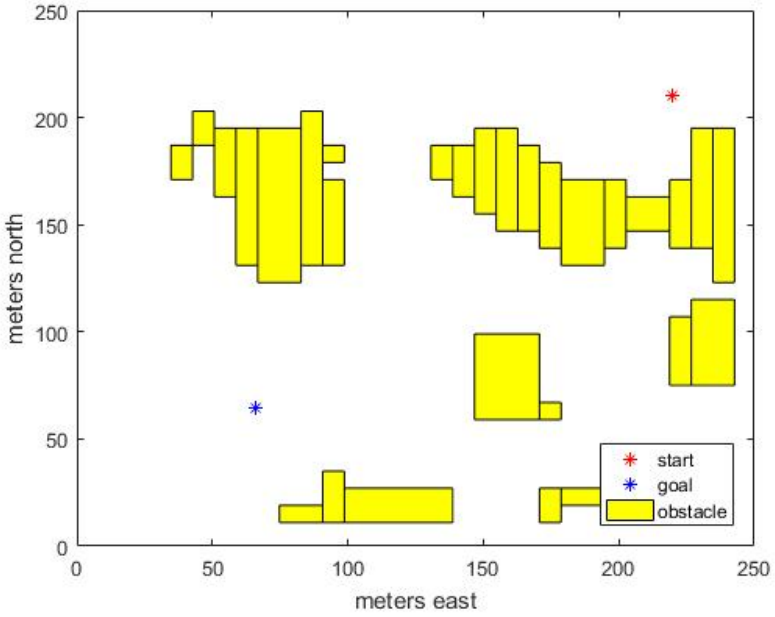

Figure 7-1: MILP distance trial 4

The top level A* algorithm allows the MILP function to better handle both distance and capacity.

7.1.2 2D MILP with top level

In the 2D case, MILP with the top level was capable of handling the same UAV density while also planning longer paths. In several trials, n simulated s-UAS plan to simultaneously leave a base at where they were within 200m of each other. This shows that the system could handle large concentrations of s-UAS in one location. Starting positions were random locations in a 200m square around a center point. Waypoint locations were random locations within 200m of the starting point. This was chosen to make the path of each vehicle relatively unique while still remaining in a nearby area. After reaching its waypoint, each vehicle would then return to its starting point. The system could delay takeoff if no movement was available for the s-UAS. All s-UAS were able complete their missions completely. No s-UAS approached within the keep-away zone of any other s-UAS or any obstacle. Figures 7-2 and 7-3 show two scenarios where MILP

was successful.

Figure 7-2: MILP path planning avoiding an obstacle

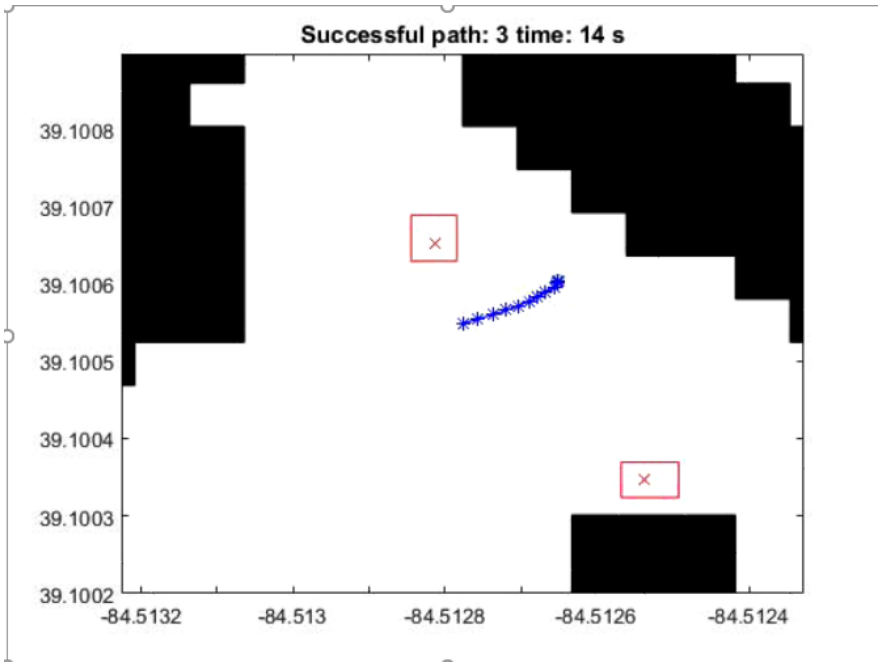

Figure 7-3: MILP path planning avoiding obstacles and other aircraft

Additionally, a MILP trial was done with hexagonal cells with 25m side lengths. The graph's

connectivity was pre-processed to speed up the top level. Statistics for three trial are shown in table 7.5.

Table 7.5: Trials for MILP and A* top level

Trial	1	2	3
Area (m^2)	160000	160000	160000
s-UAS	38	14	35
Obstacles	61	61	61
Average s-UAS separation (m)	184.1	151.7	198.0
Minimum s-UAS separation (m)	6.276	6.420	6.287
Standard deviation, s-UAS separation (m)	75.83	81.40	0.7690
Average obstacle separation (m)	279.8	264.3	288.6
Minimum obstacle separation (m)	41.35	97.03	104.1
Standard deviation, obstacle separation (m)	5785	5540	5900
Top-level runtime (s)	1.087	0.460	0.679
Lower-level runtime (s)	5736.4	802.1	2348.9

7.2 Priority and Re-routing results

The priority and re-routing using the models of simulated downtown Cincinnati Airspace. Trials were performed in a randomly selected 160m by 160m area, with 30 simulated s-UAS participating. In all trials, s-UAS submit their flight plans simultaneously. Waypoints are randomly generated with an even probability anywhere within the area. Five trials were selected and shown in table 7.6. In most cases, each s-UAS was not needed to be re-routed more than once. the outlier was trial 4, where at least one s-UAS required 3 re-routes and a total of 12 re-routes were required. This is still low, considering there were 30 s-UAS in the simulation. Each s-UAS was still able to get to its destination effectively.

Table 7.6: A* with re-routing results for 30 s-UAS in 160m X 160m areas of downtown Cincinnati

Trial	Re-Routes		Number of Time Steps	Re-Routes per Time Step	
	Total	Max per UAS		Total	Max per UAS
1	5	1	68	0.074	0.015
2	1	1	59	0.017	0.017
3	6	2	60	0.100	0.033
4	12	3	72	0.167	0.042
5	4	2	66	0.061	0.030
Average					
	5.6	1.8	65	0.084	0.027

In the following trials, s-UAS are generated like above, except the number of s-UAS and distance were also varied. Table 7.7 shows the parameters for and results of these simulations, which were performed in random areas of the downtown Cincinnati model. Figures 7-4, 7-5, 7-6, and 7-7 show the resulting paths from these trials.

Table 7.7: More A* with priority and re-routing results

Trial	1	2	3	4
Area (m^2)	5184	5184	5184	46656
height	56	56	56	56
s-UAS	30	35	50	50
Average re-routes	2.90	1.03	1.50	0.420
Maximum re-routes	10	5	6	2
Standard deviation, re-routes	2.56	1.32	1.59	0.64
Average s-UAS separation (m)	32.65	41.59	37.43	114.8
Minimum s-UAS separation (m)	8.00	8.00	7.00	8.00
Standard deviation, s-UAS separation (m)	14.89	16.38	14.91	51.47
Average obstacle separation (m)	11.84	50.19	26.03	22.57
Minimum obstacle separation (m)	8.00	18.98	11.58	8.00
Standard deviation, obstacle separation (m)	5.70	21.66	9.99	10.13
top-level runtime(s)	0.960	1.877	2.410	10.322
lower-level runtime (s)	85.06	53.29	80.12	148.97

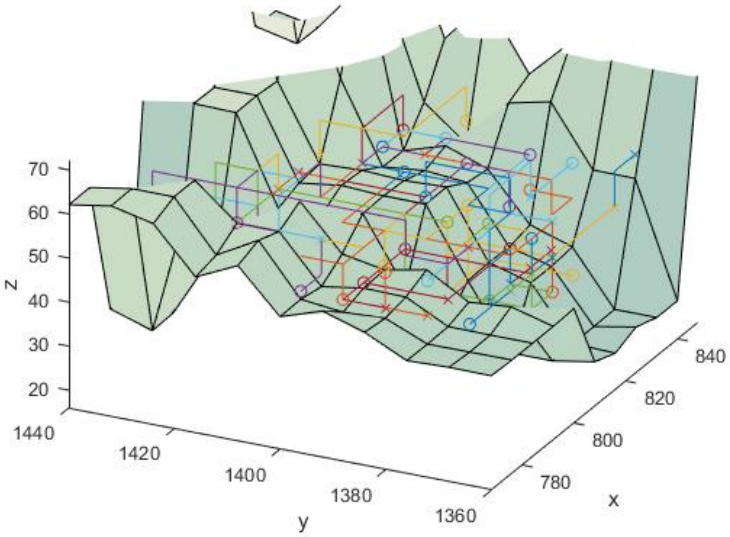

Figure 7-4: Re-routed results, Trial 1

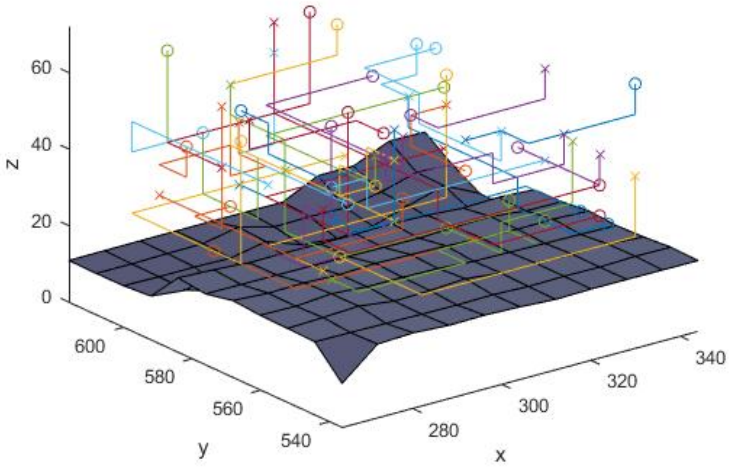

Figure 7-5: Re-routed results, Trial 2

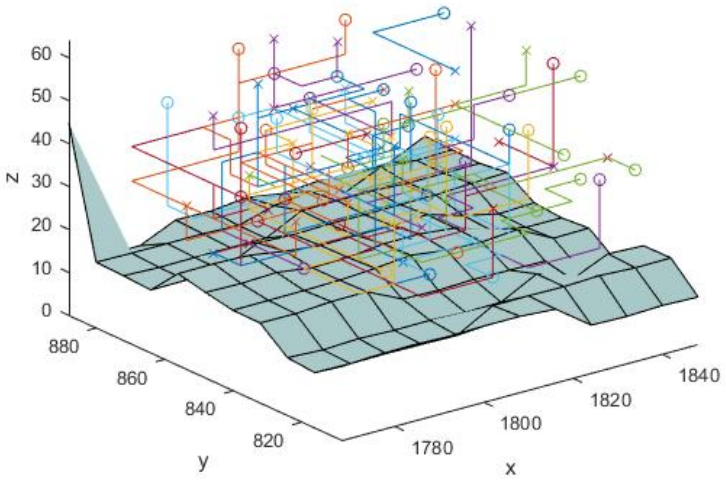

Figure 7-6: Re-routed results, Trial 3

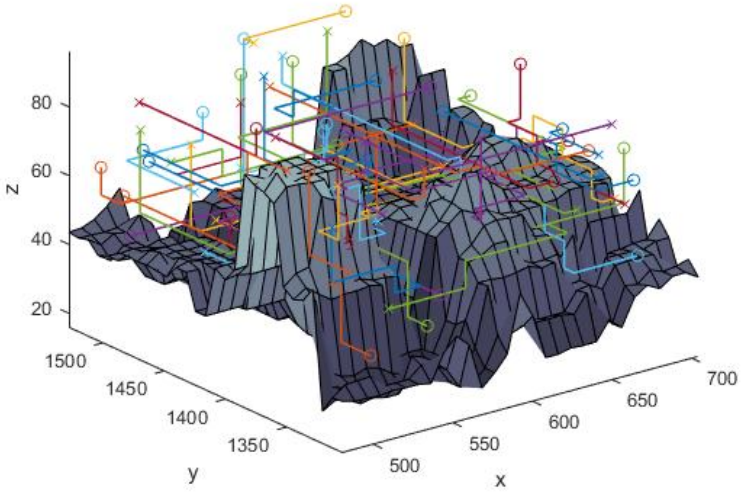

Figure 7-7: Re-routed results, Trial 4

To compare the algorithms across an even playing field, the 2D case was chosen. Three s-UAS were simulated within a 72 m radius around a central point, at a height 64m above the lowest point in nearby terrain. Each vehicle submits a flight plan with the same take-off time-step, but in the

MILP cases, the system solves the trajectory of the vehicles in order UAS 1, then UAS 2, then UAS 3. In the A* with re-routing, the trajectories are solved simultaneously. For each algorithm, the start and end points are the same. Each vehicle has a start point on the edge of a 68m radius circle, and an end point on the point closest to the point opposite it on the circle that was outside of an obstacle and each other vehicle's end point. The start points were attempted to be distributed evenly around the circle, but some adjustments were made when obstacles interfered. The times for MILP without the top level A*, MILP with the top level A*, and the combined re-routing algorithm are listed in table 7.8. Since the re-routing algorithm finds the routes all at once, the average time per UAS is listed for each.

Table 7.8: Comparison Between Algorithm Times

Algorithm:	MILP by itself	MILP with top level	A* w/ re-routing
UAS 1	10.7075	27.4082	9.1775
UAS 2	111.5738	25.8690	9.1775
UAS 3	59.8562	34.8445	9.1775
Average:	60.7125	29.3739	9.1775

Chapter 8

Discussion, Conclusions and Future work

8.1 Discussion

8.1.1 Lower-level MILP

The lower-level MILP simulations show that MILP is able to provide a consistent level of safety while taking into account the dynamics of the s-UASs. At saturation, average s-UAS separation was 29.78, or 1.5 times the distance an s-UAS in the simulation could cover in a second. The longer runtime near capacity indicates that MILP can experience significant slowdown as an airspace becomes increasingly crowded. The distance capacity simulations illustrate the desirability of a top level simulation. MILP provides the fastest results when it does not have to make paths with large detours around obstacles and other s-UAS. The results of adding a top level are further illustrated in the next section.

8.1.2 MILP with A* top level

The top-level A* with MILP results are promising. Pre-processed graphs allow the top-level runtime to be kept low. It runs consistently under a second for the simulated criteria. The lower-level algorithm runs consistently at about 70 seconds per s-UAS for when provided with this top level. This is a great improvement on the runtime of the lower-level algorithm, which could stretch to over 20 minutes when not time-limited if it had to process an s-UAS that had to make a signifi-

cant detour. The results still present good separation between obstacles and s-UAS throughout the runtime.

8.1.3 A* with Priority and Re-routing

The A* with priority and re-routing results show a significant reduction in runtime over the MILP counterparts. This reduction comes, in a large part, from the minimal modelling of the dynamics of the s-UAS flight. However, it does demonstrate an ability to route in three dimensions, which the MILP algorithms always took prohibitively long runtimes to do. It also demonstrates that an individual s-UAS would not need to be re-routed an extremely large amount of times, even if the terrain is complex and the intruders are many.

8.1.4 Surface Model Suitability to MILP and A*

Of note is the decision to model the surface as a grid. For the simple 2D MILP algorithms, reducing the obstacles to a single layer and combining adjacent models allowed the algorithm to run quickly. The pre-processed 2D A* graph largely ignores the problems with obstacle modelling by allowing it to happen offline or mostly offline. When moving to the third dimension, the raster was prohibitively difficult to use with MILP. The much simpler A* with re-routing algorithm managed it much better, but not as well as a simplified, pre-processed graph. A TIN, as noted in chapter 3, would be better suited to the needs of MILP and A* with re-routing because it is a less data-intensive surface model.

8.2 Conclusions

8.2.1 MILP and MILP with A*

MILP is strongest as a SAA technology, because it takes in more information about its surroundings and can better respond to the chaotic events of close encounters and the unforeseen. It

is computationally expensive to model an entire three dimensional mission with, however.

8.2.2 A* with Priority and Re-routing

The re-routing section allows for strong traffic management, even as the airspace becomes crowded and the geometry becomes complex. It does not provide strong SAA procedures, because information on other s-UAS is so limited.

8.2.3 Surface Model Suitability to MILP and A*

The model is not very suitable to MILP and A* in three dimensions, as it becomes computationally cumbersome over large distances.

8.3 Future Work

8.3.1 Improved Models of the City

A TIN (Triangular irregular network) is a better representation that would improve performance for MILP, A*, or any algorithm to generate pre-processed streamlines, by reducing the data overhead.

8.3.2 Improved Pre-processed Maps

Simplifying to an airspace to streamlines would be a better alternative, because then the streamlines could act as a more efficient 3D preprocessed graph, that could be evaluated with A* or a better graph processing algorithm. SAA only then has to care about avoiding other s-UAS (and popup threats) while in a streamline, returning to a streamline when outside of one, and avoiding obstacles while outside of a streamline. One potential way to do this would be to model select random nodes a suitable distance from each other and obstacles, then connect them with straight lines and circular arcs if possible. If this process is continue until the entire airspace is reachable,

a potential set of streamlines has been created. The suitability of this model could be examined using a trial with A* and re-routing, and a most effective set of streamlines for an airspace could be generated using a genetic algorithm.

8.3.3 Combined Pre-processed Maps and Algorithms

With pre-processed streamlines, the A* search could be an effective graph search over long distances, although other graph searches should also be investigated. Rerouting can also be included at the traffic management level. Market-based solutions can also be effective. For the SAA level, MILP, or a comparable algorithm would then also have ideal streamlines to match and navigate with respect to. The algorithms in this thesis could also be adapted to handle a more dynamic case. If each vehicle conducts SAA operations for an appropriate look-ahead time, it can detect when a conflict occurs that necessitates a route change. It then sends this conflict information and a route change request to the routing service, which provides a new route as the vehicle avoids the conflict. The routing service will also respond to pop-up threats and dynamic issues by detecting the effected airspace element in its airspace model and updating it appropriately. Then, for all paths that are passing through the element or will pass through the element, it issues new routes.

Appendices

List of Publications and Presentations

- Dechering, M., Radmanesh, M. and Kumar, M., 2018. Technologies for Integration of small Unmanned Aerial Systems (s-UAS) in National Airspace System (No. FHWA/OH-2018-12).

- Kumar, R., Dechering, M., Pai, A., Ottaway, A., Radmanesh, M. and Kumar, M., 2017, June. Differential flatness based hybrid PID/LQR flight controller for complex trajectory tracking in quadcopter UAVs. In Aerospace and Electronics Conference (NAECON), 2017 IEEE National (pp. 113-118). IEEE.

References

[1] Thomas Prevot, Jeffrey Homola, and Joey Mercer. From rural to urban environments: human/systems simulation research for low altitude uas traffic management (utm). In *16th AIAA Aviation Technology, Integration, and Operations Conference*, page 3291, 2016.

[2] Federal Aviation Administration (FAA). Faa aerospace forecast fiscal years 2017–2037, 2017.

[3] FAA. sense and avoid (saa) for unmanned aircraft system (uas). In *Final Report of the FAA SAA sponsored workshop*, 2009.

[4] Andrew Joseph Weinert. *An information theoretic approach for generating an aircraft avoidance Markov decision process*. PhD thesis, Pennsylvania State University, 2015.

[5] Swee Balachandran, César Munoz, and Maria C Consiglio. Implicitly coordinated detect and avoid capability for safe autonomous operation of small uas. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 4484, 2017.

[6] Donald McCallie, Jonathan Butts, and Robert Mills. Security analysis of the ads-b implementation in the next generation air transportation system. *International Journal of Critical Infrastructure Protection*, 4(2):78–87, 2011.

[7] Christian Carreon-Limones, Andrew Rashid, Phillip Chung, and Subodh Bhandari. 3-d mapping using lidar and autonomous unmanned aerial vehicle. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 1155. American Institute of Aeronautics and Astronautics, 2017.

[8] Aaron Mcfadyen and Luis Mejias. A survey of autonomous vision-based see and avoid for unmanned aircraft systems. *Progress in Aerospace Sciences*, 80:1–17, 2016.

[9] Gonzalo R Rodríguez-Canosa, Stephen Thomas, Jaime Del Cerro, Antonio Barrientos, and Bruce MacDonald. A real-time method to detect and track moving objects (datmo) from unmanned aerial vehicles (uavs) using a single camera. *Remote Sensing*, 4(4):1090–1111, 2012.

[10] Product showcase. http://gryphonsensors.com/products/#product-showcase, 2017. [Online; accessed 27-October-2017].

[11] Navigation National Coordination Office for Space-Based Positioning and Timing. Gps accuracy. https://www.gps.gov/systems/gps/performance/accuracy/, 2017. [Online; accessed 27-October-2017].

[12] uAvionix. products: ping 1090. https://www.uavionix.com/products/ping1090/, 2017. [Online; accessed 27-October-2017].

[13] Gemunu Gurusinghe, Takashi Nakatsuji, Yoichi Azuta, Prakash Ranjitkar, and Yordphol Tanaboriboon. Multiple car-following data with real-time kinematic global positioning system. *Transportation Research Record: Journal of the Transportation Research Board*, 1802(1): 166–180, 2002.

[14] Yuneec. Typhoon h. http://us.yuneec.com/typhoon-h-intel-realsense-technology, 2017. [Online; accessed 25-October-2017].

[15] Intel. newsroom uploads: Intel news fact sheet, intel falcon 8+ system. https://newsroom.intel.com/newsroom/wp-content/uploads/sites/11/2017/07/Intel-Falcon-8-Fact-Sheet.pdf, 2017. [Online; accessed 27-October-2017].

[16] DJI. Guidance. http://www.dji.com/guidance, 2017. [Online; accessed 27-October-2017].

[17] Qualcomm. Snapdragon flight. https://developer.qualcomm.com/hardware/snapdragon-flight, 2017. [Online; accessed 27-October-2017].

[18] Federal Communications Commission. Dedicated short range communications (dsrc) service. https://www.fcc.gov/wireless/bureau-divisions/mobility-division/dedicated-short-range-communications-dsrc-service, 2017. [Online; accessed 29-August-2017].

[19] 3GPP. Release 14. http://www.3gpp.org/release-14, 2017. [Online; accessed 27-October-2017].

[20] Qualcomm. Products: Snapdragon: Modems: 4g-lte: X16. https://www.qualcomm.com/products/snapdragon/modems/4g-lte/x16, 2017. [Online; accessed 27-October-2017].

[21] Qualcomm. Qualcomm technologies releases lte drone trial results: Lte unmanned aircraft systems trial report. https://www.qualcomm.com/news/onq/2017/05/03/qualcomm-technologies-releases-lte-drone-trial-results, 2017. [Online; accessed 27-October-2017].

[22] Imad Jawhar, Nader Mohamed, Jameela Al-Jaroodi, Dharma P Agrawal, and Sheng Zhang. Communication and networking of uav-based systems: Classification and associated architectures. *Journal of Network and Computer Applications*, 84:93–108, 2017.

[23] Inc. Google. Google uas airspace system overview. https://utm.arc.nasa.gov/docs/GoogleUASAirspaceSystemOverview5pager[1].pdf, 2015. [Online; accessed 20-August-2017].

[24] Amazon Prime Air. Determining safe access with a best-equipped, best-served model for small unmanned aircraft systems, 2015.

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

[26] Joseph Rios, Daniel Mulfinger, Jeff Homola, and Priya Venkatesan. Nasa uas traffic management national campaign: Operations across six uas test sites. In *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*, pages 1–6. IEEE, 2016.

[27] George Elmasry, Diane McClatchy, Rick Heinrich, and Boe Svatek. Integrating uas into the managed airspace through the extension of rockwell collins' arinc cloud services. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2017*, pages 1–14. IEEE, 2017.

[28] Joseph Rios and Marcus Johnson. Unmanned aircraft systems traffic management (utm) concepts and architecture overview. *AIAA Aviation Forum*, 2018.

[29] Federal Aviation Administration. Uas data exchange. `https://www.faa.gov/uas/programs_partnerships/uas_data_exchange/`, 2017. [Online; accessed 14-December-2017].

[30] airmap.com. `https://www.airmap.com/`, 2017. [Online; accessed 12-January-2018].

[31] Project wing. `https://x.company/projects/wing/`, 2017. [Online; accessed 12-January-2018].

[32] skyward.io. `https://skyward.io/`, 2017. [Online; accessed 12-January-2018].

[33] Mingrui Lao and Jun Tang. Cooperative multi-uav collision avoidance based on distributed dynamic optimization and causal analysis. *Applied Sciences*, 7(1):83, 2017.

[34] María Consiglio, César Muñoz, George Hagen, Anthony Narkawicz, and Swee Balachandran. Icarous: Integrated configurable algorithms for reliable operations of unmanned systems. In *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*, pages 1–5. IEEE, 2016.

[35] Swee Balachandran, Anthony Narkawicz, César Muñoz, and María Consiglio. A path planning algorithm to enable well-clear low altitude uas operation beyond visual line of

sight. In *Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)*, 2017.

[36] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[37] César Muñoz, Anthony Narkawicz, George Hagen, Jason Upchurch, Aaron Dutle, Maria Consiglio, and James Chamberlain. Daidalus: detect and avoid alerting logic for unmanned systems. *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, 2015.

[38] Anthony Narkawicz and George E Hagen. Algorithms for collision detection between a point and a moving polygon, with applications to aircraft weather avoidance. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, page 3598, 2016.

[39] Elisa Capello, Giorgio Guglieri, and Gianluca Ristorto. Guidance and control algorithms for mini uav autopilots. *Aircraft Engineering and Aerospace Technology*, 89(1):133–144, 2017.

[40] Roberto Opromolla, Giancarmine Fasano, Giancarlo Rufino, Michele Grassi, and Al Savvaris. Lidar-inertial integration for uav localization and mapping in complex environments. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, pages 649–656. IEEE, 2016.

[41] Yang Lyu, Quan Pan, Chunhui Zhao, and Jinwen Hu. Autonomous stereo vision based collision avoid system for small uav. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 1150. AIAA SciTech Forum, 2017.

[42] Sarah D'Souza, Abe Ishihara, Ben Nikaido, and Hashmatullah Hasseeb. Feasibility of varying geo-fence around an unmanned aircraft operation based on vehicle performance and wind. In *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*, pages 1–10. IEEE, 2016.

[43] Jeffrey Homola, Thomas Prevot, Joey Mercer, Nancy Bienert, and Conrad Gabriel. Uas traffic management (utm) simulation capabilities and laboratory environment. In *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*, pages 1–7. IEEE, 2016.

[44] Abraham K Ishihara, Jaewoo Jung, and Joey Rios. Rapid trajectory prediction for a fixed-wing uas in a uniform wind field with specified arrival times. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2018.

[45] Eric R Mueller and Mykel Kochenderfer. Multi-rotor aircraft collision avoidance using partially observable markov decision processes. In *AIAA Modeling and Simulation Technologies Conference*, page 3673, 2016.

[46] Min Xue and Joseph Rios. Initial study of an effective fast-time simulation platform for unmanned aircraft system traffic management. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3073, 2017.

[47] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 3rd. *Essex, UK: Parentice Hall*, page 1152, 2009.

[48] Mohammadreza Radmanesh and Manish Kumar. Flight formation of uavs in presence of moving obstacles using fast-dynamic mixed integer linear programming. *Aerospace Science and Technology*, 50:149–160, 2016.

[49] Mohammadreza Radmanesh, Manish Kumar, Alireza Nemati, and Mohammad Sarim. Dynamic optimal uav trajectory planning in the national airspace system via mixed integer linear programming. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 230(9):1668–1682, 2016.

[50] Razvan-Viorel Mihai and Mirela-Madalina Bivolaru. Cooperative distributed trajectory optimization for a heterogeneous uav formation. In *AIP Conference Proceedings*, volume 2046, page 020061. AIP Publishing, 2018.

[51] Mohammadreza Radmanesh and Manish Kumar. Grey wolf optimization based sense and avoid algorithm for uav path planning in uncertain environment using a bayesian framework. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, pages 68–76. IEEE, 2016.

[52] Dae-Sung Jang, Corey A Ippolito, Shankar Sankararaman, and Vahram Stepanyan. Concepts of airspace structures and system analysis for uas traffic flows for urban areas. In *AIAA Information Systems-AIAA Infotech@Aerospace*, page 0449. American Institute of Aeronautics and Astronautics, 2017.

[53] Fahad Islam, Jauwairia Nasir, Usman Malik, Yasar Ayaz, and Osman Hasan. Rrt*-smart: Rapid convergence implementation of rrt* towards optimal solution. In *Mechatronics and Automation (ICMA), 2012 International Conference on*, pages 1651–1656. IEEE, 2012.

[54] Mohammadreza Radmanesh. *UAV traffic management for national airspace integration*. PhD thesis, University of Cincinnati, 2016.