Systems & Mission Management

GBDP Final Report – UAM II

Jean-Michel Alvarez
12174
May 18, 2021
Word count: 5344

Executive Summary

The following report outlines the design strategy for the avionics systems & mission management element of HopAir's UAM aircraft. A fly-by-wire system is used in HopAir's 5-seater, octarotor, battery-electric aircraft. It is currently in early design stages, with the company hoping to begin production by 2028. Its primary use will be for commercial intercity travel within a 130 mile range where travel by car, boat or rail would be more time-consuming, unpleasant, expensive, and unpractical.

Avionics systems are the backbone of all modern aircraft, artificial satellites, and spacecraft. Their purpose is to provide flight-critical information to the pilot, whether they are on board or piloting remotely. This includes monitoring the health of individual components, power distribution, collision avoidance, mission management, communications, situational awareness, autopilot, control system actuation, emergency procedures, and many other responsibilities.

The main processing systems will be an autopilot computer, a flight control computer, and a mission computer. Each will have its own processing unit and be connected to each other via industry-standard RS232 serial cables. A central ethernet switch, directly interfacing with each computer, will also provide extremely fast data transmission between the sensors and computers, also interfacing with a C2 data link. The computers will be at least doubly redundant, with the possibility of having additional redundancy integrated within them for enhanced safety. This loose coupling allows for semi-independent evolution, crucial for the design, maintenance, and future upgrading of the avionics system. The mission computer will handle pre-flight mission planning, in-flight mission management, mission repair and replanning, as well as post-flight mission analysis and debriefing. This computer will most likely require the fastest processor as it will have to perform nonlinear state estimation using an extended Kalman filter, as well as constantly interrogate each component to identify its state and make modifications to mission and route plan in the case of an anomaly.

An electro-optic and forward-looking infrared camera, as well as a Synthetic Aperture Radar, mounted below the aircraft, will be the "eyes" of the aircraft. With integrated video processing and direct interfacing with the mission computer and C2 data link, Ground Control will have a live video feed streamed from the aircraft, and the mission computer will be able to scan the footage and detect potential obstacles such as other aircraft, bird strikes, and adverse weather. It will then directly be able to make modifications to the mission and route plan if necessary, alerting Ground Control about the amendments via the state-of-the-art C2 data link.

An ADS-B transponder that doubles as a taillight will also be mounted to the aircraft. An improvement to the more traditional TCAS technology, ADS-B does not require a specific and extensive suite of electronics in both aircraft, also providing increased communications accuracy between the relevant aircraft and Ground Control.

Novel mission management algorithms were also written for path-planning and emergency landing zone selection. Their design process illustrates HopAir's software methodology regarding future algorithm development. Rigorous hardware- and software-in-the-loop testing will be carried out to identify the weakest or more unreliable links of the system and either multiply redundancies or replace the component altogether.

Nomenclature

ADS-B	Automatic Dependent Surveillance-Broadcast
ASIC	Application-Specific Integrated Circuit
ATR	Austin Trumbull Radio
AWG	American Wire Gauge
C2	Command and Control
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
DSA	Detect, Sense and Avoid
DSP	Digital Signal Processor
EFL	Emergency (Forced) Landing
EKF	Extended Kalman Filter
FAA	Federal Aviation Authority
FCS	Flight Control System
FLIR	Forward-Looking Infrared
FMEA	Failure Mode and Effects Analysis
FPGA	Field-Programmable Gate Array
GCS	Ground Control Station
GPS	Global Positioning System
HWIL	Hardware-In-The-Loop
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
JAUS	Joint Architecture for Unmanned Systems
LQR	Linear Quadratic Regulator
LTR	Loop-Transfer Recovery
MAC	Medium Access Control
MIMO	Multiple Input, Multiple Output
MMS	Mission Management System
NDI	Nondevelopmental Item
PMU	Power Management Unit
RPA	Remotely Piloted Aircraft
RPS	Remote Pilot Station

SAR	Synthetic Aperture Radar
SHF	Super-High Frequency
SWaP	Size, Weight and Power
SWIL	Software-In-The-Loop
SISO	Single Input, Single Output
SSD	Solid-State Drive
TCAS	Traffic Alert and Collision Avoidance System
UAM	Urban Air Mobility
UAS/UAV	Unmanned Aerial System/Vehicle
CBM	Condition-Based Maintenance
CCPS	Center for Chemical Process Safety
CFRP	Carbon Fibre Reinforced Polymer
CMM	Coordinate Measurement Machine
COTS	Commercial Off-The-Shelf
EPSRC	Engineering and Physical Sciences Research
FAL	Final Assembly Line
GKN	GKN Aerospace
KPI	Key Performance Indicator
MPC	Manufacturing Planning and Control
MRP I	Materials Requirement Planning
MRP II	Manufacturing Resource Planning
MTO	Make-To-Order
MTS	Make-To-Stock
NPP	New Part Planning
QC	Quality Control
UAM	Urban Air Mobility
VTOL	Vertical Take-Off and Landing
WIP	Work-In-Process

Contents

Introduction

Many avionics components are flight-critical and as such, their hardware and software interfacing must be rigorously checked for compatibility. The report that follows outlines the systems & mission management technology required for the UAM aircraft flight operations, identifies potential COTS components, and evaluates their interoperability and reliability, ensuring safety standards such as multiple redundancies and health monitoring systems are implemented and fully functioning.

1. Overview

Most components will need be commercial-off-the-shelf (COTS), making them a mature part of the system and a nondevelopmental item (NDI) [1]. This allows for more resources to be invested into the design and development of their physical and digital interfaces, and later, into algorithm design and rigorous software- and hardware-in-the-loop (SWIL/HWIL) testing.

The components will be able to fit into an ATR (Austin Trumbull Radio) box fitted into the nose of the aircraft, where they will be powered by a Power Management Unit (PMU) directly linked to the battery system [2]. An overview of the system is shown in Fig. 1.

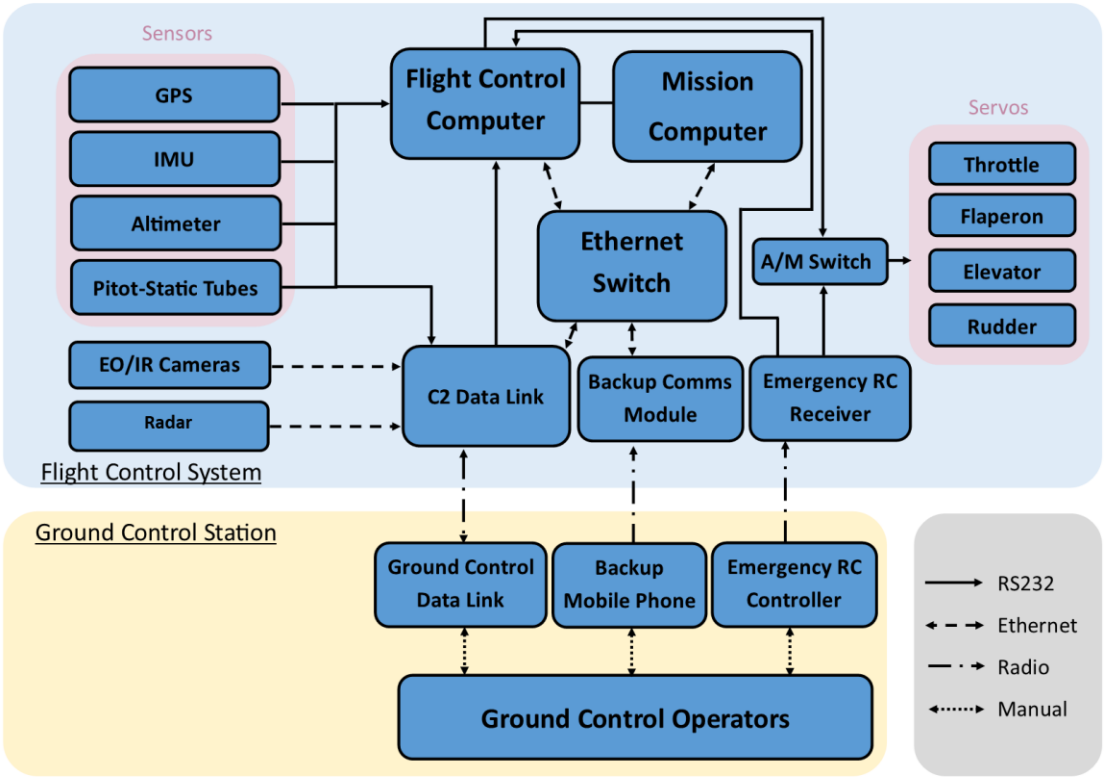


FIGURE 1: OVERVIEW OF AVIONICS SYSTEM

The main priorities for designing the avionics system are as follows:

- A clear and robust flow of information

- A total avionics weight of 0.5-1% of MTOW (similar to commercial airliners [3]) and power requirement of under 3kW.
- An uninterrupted stream of information from the sensor suite
  - Camera capable of at least 480p 30 fps footage
  - A radar range of over 10 km
  - Reliable and accurate position and state estimation
- Multiple redundancies throughout the system
- Seamless integration with control systems and landing gear
- Offload as much computation and analysis onto the onboard computers as possible to reduce operator workload
- Detailed novel algorithm design

2. On-board hardware

The flow of information in the avionics subsystem is illustrated in Fig. 2, roughly flowing from left to right. Each element of this diagram will be delved into more detail in subsequent sections of this report.

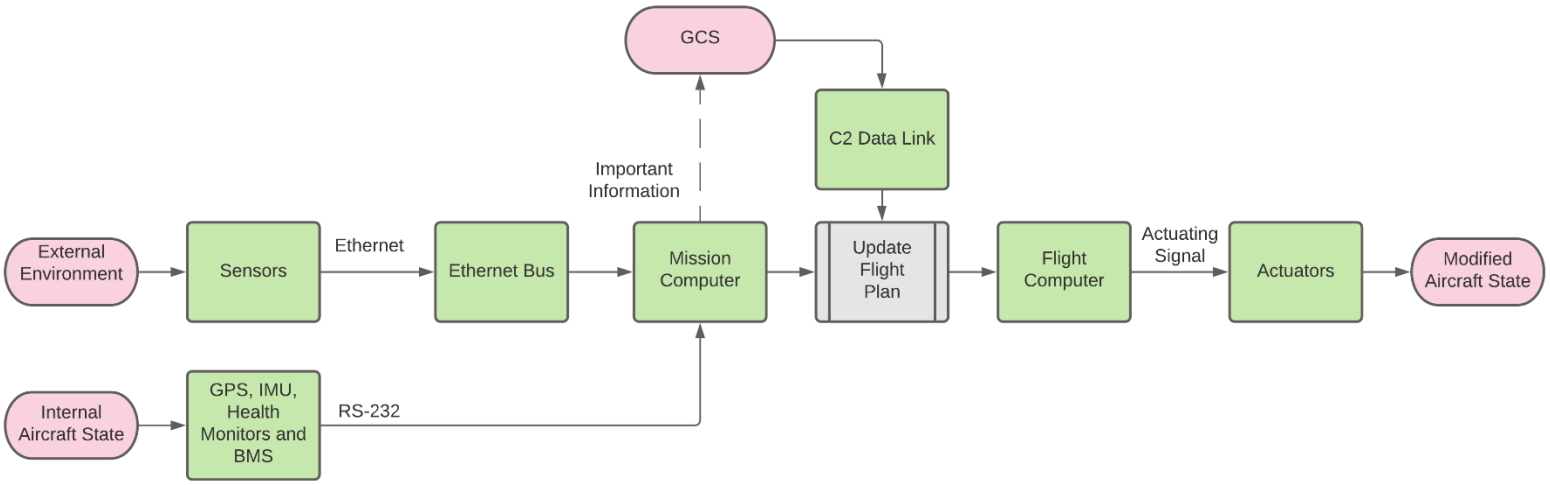


FIGURE 2: FLOW OF INFORMATION

2.1 Processing Units

A central part of any fly-by-wire system and automatic or autonomous operations is the processor. For mainly safety purposes, the Flight Control Computer and Mission Computer will be physically separate, and double redundant. These will operate according to the *frontseat-backseat* driver paradigm, commonly used in many autonomous robotic systems [4]. To summarise, the *frontseat* driver is the Flight Control System (FCS), which includes the autopilot and avionics and related controller hardware relevant to direct flight control. The *backseat* driver is the Mission Computer, a more deliberative system that is responsible for longer-range planning, relating the information it receives about the internal and external state of the aircraft to the overall mission [5]. The idea behind this paradigm is to decouple low-level control from high-level autonomous operations.

The processors used for these computers can be FPGA, ASIC, DSP, or microprocessor, such as 555 and PowerPC [1]. In most avionics systems, a PC/104 system is used, consisting of a CPU board, power supply board, and multiple peripheral boards, such as a data acquisition module or a GPS receiver [6]. This is because their rigidity, reliability, and SWaP characteristics are unparalleled in the

computer subsystem industry [7]. Virtually every UAV COTS avionics component is made to interface with them, making their presence within the avionics system incontrovertible.

2.1.1 Flight Control

The flight control computer is the second flight critical computer system in the aircraft. It is responsible for basic autopilot functions, such as cruise control, and therefore is directly linked to the servo motors controlling the actuation of the control surfaces.

Classical control theory, the oldest and most established control method, concerns linear time-invariant single-input single-output (SISO) systems [8]. This is why it is appropriate for low-level autopilot tasks such as attitude stabilization, longitudinal control and lateral control [9]. This is done using a combination of PID controllers, phase lead and lag compensators [10]. Possible control design techniques are LQR/LTR and H_∞. LQR/LTR separates the controller into a Kalman filter observer then a state feedback controller, whereas H_∞ is a frequency domain method of designing controllers for systems that exhibit uncertainty in terms of the system model or the operational environment [11]. For this application, an Extended Kalman Filter (EKF) can also be used for data fusion from the GPS, IMU, dead reckoning system, and radar altimeter [2] [12] [13].

"Loose coupling" is another philosophy behind the UAM's avionics architecture. A design methodology where subsystems are not tightly integrated but coupled in a manner allowing for at least semi-independent evolution has been found to be crucial for designing highly complex systems [14].

2.1.2 Mission Computer

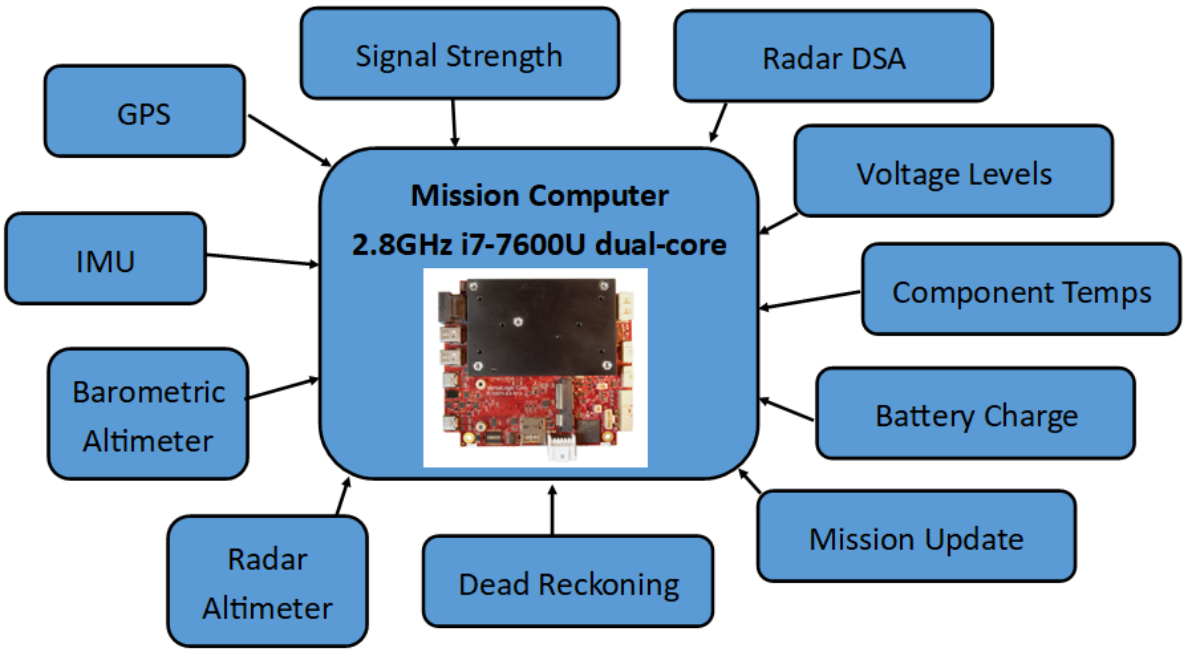


FIGURE 3: OUTLINE OF MISSION MANAGEMENT SYSTEM

The Mission Management System (MMS) shown in Fig. 3 has the following responsibilities [15]:

- Mission planning
    - Task/resource Allocation

    - Motion Planning
    - Airworthiness checks
    - Risk Assessment
- Mission Execution
    - Route Planning
    - Navigation
    - Intelligent Decision Making
- Mission Monitoring
    - Mission Progress Evaluation
    - Situational Awareness
    - Contingency
    - Anomaly Detection
- Mission repair/re-planning
    - Re-tasking
    - Resource reallocation
    - Re-routing
- Post-Mission Analysis
    - Log Downloading
    - Inspection
    - Maintenance

As a result of this extensive list of responsibilities, the mission computer will have the highest processing burden of any processor onboard. Attempts to adequately size this processor in terms of clock speed and number of cores have been made in the section on novel algorithms further in the report, where timings were taken and related to real-world scenarios. However, these timings are rough approximations at best and final processor selection should only be made after extensive testing of the entire MMS on a variety of different processors.

It has been shown that mission plan repair has clear benefits over mission replanning in most cases [16]. Adaptive mission planning, a process involving the detection of events, the effects the events have on the mission plan, and the execution of a response will be critical to mission management. To enable this, a status monitor agent and mission plan adaptation agent will be implement within the MMS. The former considers the internal and external state of the aircraft, and classifies events according to their priority and their nature (critical or incipient). The latter is more complicated, executing partial plans based on the input from the status monitor agent and revalidating mission status [16].

In the event of loss of communication, the operator is immediately alerted and the aircraft is automatically tasked with searching for the radio beam and re-establishing contact as soon as possible, or otherwise switching to a different radio frequency band if the radio link is duplexed [2].

2.1.3 Autopilot

The dream of the modern UAM designer is for the air taxis flying above us to be completely autonomous, intelligently manoeuvring the skies safely and efficiently, making objectively superior decisions to that of human pilots by appropriately taking into account the state of its internal components, the aircraft itself, and its surroundings. Meystel and Albus outline the requirements of a successfully autonomous system by defining intelligence as: “the ability of a system to act appropriately in an uncertain environment, where an appropriate action is that which increases the probability of success, and success is the achievement of behavioural sub-goals that support the

system's ultimate goal" [17]. It is clear that for full autonomy to be operational and widely accepted, the system must be capable of making decisions considered objectively superior to that of a human pilot, with any human input being fully eliminated from the decision-making process.

A clear distinction must first be made between autonomy and automation: automatic systems are designed such that the machine exactly follows the programmed commands, with no decision-making process whatsoever. Autonomous systems successfully recognize different situations and makes one decision out of many possible ones. The subsequent action sequence will have been chosen based on the current internal and external situation together with programmed criteria and rules [15].

2.2 Sensor Suite

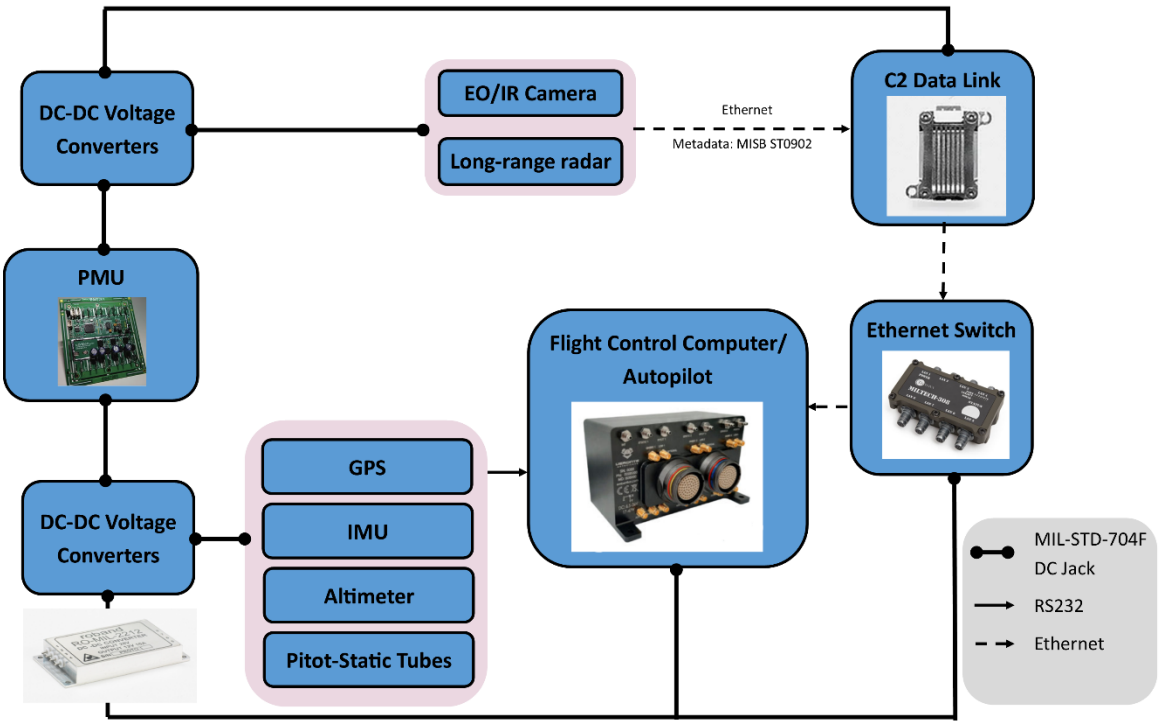


FIGURE 4: SENSOR SUITE AND INTERFACING

Cameras and radars are vital for the perception of the environment and the situational awareness (SA) of the autopilot [15]. A majority of the sensor data is processed onboard, so that fewer analysts are required to support each UAV. The system can therefore highlight areas of interests to the analysts, allowing them to focus on the most important aspects of their job and potentially ease pressure on downlink-related architecture. This is why sensors using integrated video processing and encoding, transmitting global motion information rather than motion vectors derived from the video footage, are preferred due to their favourable SWaP characteristics [18].

The entirety of the sensor suite will use Ethernet interfacing and protocols (see Fig. 4), as modern imaging devices output increasingly high resolution and frame rate data and thus require data transfer speeds in the order of Gbps. A camera capable of daylight video capture and white-hot long wave infrared video capture will be used, with a toggle depending on the environmental conditions [2]. Only the minimum required frames per second and resolution will be used to reduce strain on

data links [9]. The camera will be able to rotate 360° and tilt +45°/-85° though most of the time it will be facing forward as the aircraft is unlikely to receive a sudden impact to its rear [19].

Due to the camera's flight-critical role, a trustworthy manufacturer, well-known in the aviation industry and offering competitive products, had to be chosen. The TASE™ 350 (see Fig. 5), a medium-end option from Collins Aerospace's TASE Imaging Systems line, was chosen for this task. It is capable of high resolution (1280 x 720) relative to its low weight and power consumption (3.2 kg and 25W). It is also compatible with a multitude of interfaces, such as RS-232, CAN, and Ethernet [19]. Due to the storage size of live video feeds, it is likely that Ethernet will be used to connect it to the mission computer, which will then analyse it and transmit salient features to GCS.

FIGURE 5: TASE 350 EO/LWIR CAMERA (LEFT) AND SAR (RIGHT)

For the on-board radar, the more conventional radome configuration was initially explored, due to its common use in military applications. This consists of a radar covered by an RF-transparent nose cone. However, after close investigation, the 50% added weight generated from active electronically scanned array (AESA) radars (commonly found in radomes) relative to an SAR mounted below the aircraft was deemed to not be worth the increase in radar range [20]. Indeed, this is where military radar applications exceed civil ones.

It was decided that the aircraft would instead feature Synthetic Aperture Radar (SAR) technology, as it provides finer spatial resolution than conventional beam-scanning radars and reduced weight (at the expense of imaging range) [21]. This will be incredibly useful for navigating dynamic, congested, urban airspace. The well-known radar systems company IMSAR was chosen. The NSP-5, whose performance lies between the more compact NSP-3 and the higher performance NSP-7, was chosen for the task. Relative to the aircraft it adds little additional weight, yet has a 24 km SAR Imaging Range (at 1m resolution) and uses less power than AESA radars, at 130 W [22]. This radar will interface with the system via Ethernet cables and protocols. Although it will protrude from the bottom of the aircraft, it has the very favourable aerodynamic profile of a rounded cylinder (see Fig. 5), and the Aerodynamics team has already included a parasitic drag factor of 1.25, in which both the camera and radar's aerodynamic profiles are taken into account [23].

2.3 Communications

The radio communications systems are known to have a strong effect on power budget (see App. C) [9]. These communications are most likely to occur in the SHF band of 5-10Ghz, depending on the

data rate, Line-of-sight requirements, and frequency availability, which all need to be assessed in further detail. C2 data links have multiple capabilities [24]:

- Override Flight of RPA.
- Override/Monitor Detect and Avoid system on RPA.
- Support RPA/RPS handover, Flight Data Recording etc.
- Provide RPA health and status.
- Relay voice/data.

They are therefore very likely to be used in HopAir's UAM aircraft. It should also be noted that in terms of DSA system interoperability with other aircraft, the industry is shifting from the traditional Traffic Alert and Collision Avoidance System (TCAS) to the improved ADS-B [25]. This is because TCAS requires a transponder onboard the other aircraft, which is not particularly useful for smaller UAVs and birdstrikes, but also because the full suite of electronics related to TCAS can cost between $25,000 and $150,000 [25].

2.4 Data Bus

A data bus is a computer subsystem that allows for the transferring of data from one component to another. These components can be connected to computer memory, CPUs, and other avionics components. Various types of data buses have a range of bandwidths, protocols, and requirements on which electronic control units (ECUs) they can be directly connected to. Characteristics of common data busses currently used in aeronautical and aerospace applications are listed in Table 1.

TABLE 1: COMPARISON OF COMMON SPACE AND AVIATION DATA BUSSES

MIL-STD-1533	**I^2C**	**CAN bus**	**ARINC 429**	**RapidIO**	**SpaceFibre**
1 Mbps	400 Kbps	5 Mbps	100 Kbps	10 Gbps	20 Gbps
Up to 31 remote terminals (each one can be a router connected to multiple subsytems)	1 master – 112 slaves	Middle ground in terms of performance, power consumption and reliability	Similar to CAN bus	Extreme performance	Improvement on SpaceWire in FDIR
Linear bus, protected against subsystem short circuit	Very low power equipment, single wire	Introduced by Robert Bosch, used and developed by Airbus	Most widely used data bus standard in aviation	Used in mobile phone infrastructure and cellular towers	Fewer logic gates than RapidIO
High reliability: Dual/Triple/Quadruple redundant configurations available	Used in cube sats	Bi-directional 64-bit bus	Uni-directional 19-bit bus		

The main contenders for non-serial and non-ethernet connections are the CAN bus and ARINC 429. This protocol is likely to be used for auxiliary power units and other miscellaneous equipment that lack serial or ethernet interfacing capabilities. However, as a result of its bi-directionality, higher data

rate, and higher data width, it was decided that the CAN bus would be a more optimal choice for transmitting data in the aforementioned cases. An example CAN Bus wiring diagram is shown in Fig. 6.

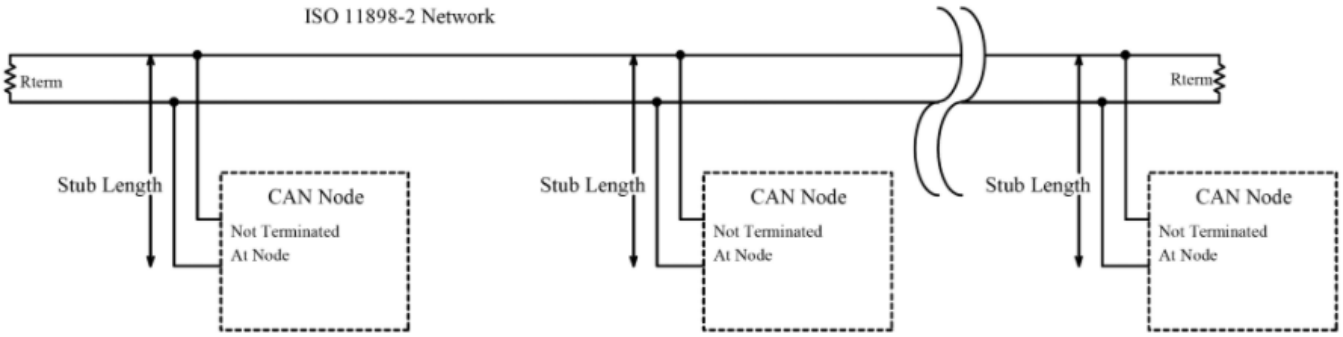


FIGURE 6: EXAMPLE CAN BUS WIRING DIAGRAM [46]

Another crucial responsibility of the data bus will be combining the aircraft's measured state parameters from the GPS and IMU and sending them to the flight computer, where an extended Kalman filter (EKF) will be used to error-proof the sensor inputs by filtering out noise. Traditionally, an avionics system will have more redundant data busses than redundant power supplies, as the former are more likely to fail and require backup [26].

2.5 Power

At least 2 software-controlled power management units (PMUs) will be onboard distributing power to the various avionics systems. They will provide vital information about voltage levels and current consumption by each electronic component, regulating these and reporting any anomalies via the C2 data link. This will also help protect from spikes in the voltage, and the pair will be redundant so that if one of the PMUs fail, the other will be able to supply power where it is needed. They also have a buzzer for power supply-related alarms [27]. Their usefulness is showcased further in the FMEA later in the report. An estimated 28 VDC will be inputted to the PMU, although this has yet to be confirmed by the Powerplant specialist. This estimate enables standardised 28VDC to 6/12VDC conversion via the use of easily-sourced COTS components (see App. C).

2.6 Wiring

Wiring harnesses, made of thermoplastic materials, will be used to protect the cables from the environment and to promote efficient cable management among the avionics engineers. This will not only improve assembly times, but also make it easier to diagnose problems and maintain the aircraft in the future as each cable will have clear and visible endpoints.

Most of the wiring will be either 18 or 22 AWG, approximately 1mm in diameter, as this is common in industry [26]. The alternator leads and battery leads are likely to be #10 (⌀2.6 mm) and #2 (⌀6.5 mm), respectively. Based on these cable thicknesses and common wiring weight percentages of MTOW in similarly-sized commercial aircraft, the weight of the cables in HopAir's UAM aircraft is expected to be in the order of 10kg [28].

Several good-practice wiring techniques will be encouraged and even enforced, such as color-coding power as yellow or red, and return as black. Blue, green and white can be used for serial and ethernet applications, as well as other non-standard applications.

3. Weight and Power Calculations

According to App. C, the total weight of the components will be approximately 17.5kg. This mass represents 0.49% of the aircraft's MTOW. This percentage is similar to that of commercial airliners, such as the B-747-100 and the C-5A [3]. The total power requirement was calculated to be approximately 200W, although this figure will require adjustment once the landing gear and actuating systems, such as tiltrotors and doors, undergo final component selection and are included in the weight breakdown. The Battery Management System (BMS) chosen can provide enough wattage for every component of the avionics system.

4. On-Board Software

An important aspect of complex avionics design is the amount of on-board and off-board computation [5]. A design priority is clearly to have as much of the computation and analysis occur onboard to avoid a potential "data deluge" for operators [29]. It is the aim of the avionics systems designer that salient details and behaviours are highlighted to the operators to ensure their workload is reduced [5]. As less data is transferred, the severity of a cyberattack is reduced, data fusion efficiency is increased as irrelevant data is erased, and the autopilot response time is decreased as it only receives information that may alter the flight course.

Examples of software protocols expected to be used in the avionics system:

- Medium Access Control (MAC) protocols will ensure optimal usage of available frequencies [9].
- Joint Architecture for Unmanned Systems (JAUS) as an interoperability standard between components [30].
- Solid State Disk (SSD) technology over standard high drives for vibration resistance and processing speed [31].

The basis of the EKF will be the aircraft state parameter vector $\hat{\boldsymbol{x}}$ shown in Fig. 7, which will be measured by both the GPS and IMU onboard:

$$\hat{\boldsymbol{x}} = [x_E\ y_E\ z_E\ V\ u\ v\ w\ \alpha\ \beta\ \psi\ \theta\ \phi\ a_x\ a_y\ a_z\ p\ q\ r]^T$$

	State variable	Notes
Position in Earth coordinates	x_E	Relative to a fixed longitude, μ
	y_E	Relative to a fixed latitude, λ
	z_E	Relative to the Earth's surface
Airspeed	V	Total airspeed
	u	Airspeed along x axis (body)
	v	Airspeed along y axis (body)
	w	Airspeed along z axis (body)
Aerodynamic angles	α, Angle of attack	
	β, Angle of sideslip	
Euler angles	Ψ, Heading angle	Relative to North, positive in West direction (counterclockwise)
	θ, Pitch angle	Relative to horizontal, positive up
	φ, Roll angle	
Accelerations in body axes	a_x	
	a_y	
	a_z	
Angular rates in body axes	p	Roll rate
	q	Pitch rate
	r	Yaw rate
Geographic position	λ, Latitude	Can be related to y_E
	μ, Longitude	Can be related to x_E

FIGURE 7: AIRCRAFT STATE VARIABLES FOR USE IN KALMAN FILTER [1]

5. Mission Management

The purpose of the mission management system is to monitor the aircraft's current state, forecast its future state, and resolve current and possible future problems or conflicts. Potentialities that can affect mission success and require mission management are [32]:

- Unanticipated Targets
- Uncertain Conditions
- Changing Mission Requirements

The management of these potentialities requires Situational Awareness on behalf of the aircraft. This involves being fully aware of the following critical entities [33]:

- Terrain
- Airspace boundaries
- Adverse weather
- Other aircraft
- Tall buildings
- Navigation aid locations and types
- Runways
- Radio frequency zones

Mission management can be broken down in flight phases, in each of which a different set of priorities and considerations govern general decision-making [32]. As mission management is clearly a very broad term, this report will delve deeper into two selected areas of the field, to showcase the software-level design methodology which will then be applied to the other areas of mission management in later design stages.

As the greatest risk from a prospective consumer's point of view regarding UAM aircraft is the risk of collision with buildings or other aircraft in congested urban airspace, path-planning and collision avoidance in a dynamic 3D environment will be the first area of focus. Additionally, the most common cause of UAV mission failure being emergency procedures at 26% [34], Emergency Landing Zone Identification will be the second mission management-related area this report will focus on.

5.1 Path Planning & Collision Avoidance

Route planning will initially be done during the mission planning phase, where the optimal route will have been selected and displayed on the pilot's screen. This can be done by direct control, inputting speed/heading/altitude instructions or destination coordinates [2]. Research has also been done into the more complex autonomous route planning, taking into considerations elements such as aerodrome-related no flight zones and acoustic constraints such as densely populated areas [35]. Taking this into account will severely reduce the impact of UAM noise pollution on public perception.

Once regulations for autonomous UAM aircraft are introduced, autonomous UAM development is expected to drastically increase [36]. However, one can already begin considering the standards and regulations for Detect, Sense and Avoid (DSA) Collision Avoidance for UAVs put forth by the FAA in F2411-04 [37]. In essence, the autonomous UAM aircraft must be 'equal or better than the theoretical see and avoid capability of a human pilot' and be as reliable as a flight-critical system [38]. The quantification of a human pilot's ability to detect an aircraft and successfully avoid it is beyond the scope of this report but suffice it to say that autonomous UAM aircraft will need to have superior vision and response time to a world-class pilot on his best day [2].

5.1.1 Path-Planning Literature Review

The research undertaken in this section revolves around the aircraft's ability to find the optimal path between two points. When the path is clear, aircraft have the privilege of travelling directly from point A to point B, without much need for computation or optimisation, especially over short distances. However, Class B airspace can become very congested [39]. In addition, the UAM aircraft by definition will need to traverse urban areas consisting of tall buildings and no-fly zones. Combining the importance of minimising battery energy expenditure and navigating a constantly evolving airspace ecosystem is clearly very important. Unmanned Traffic Management (UTM) systems of the future will have to integrate seamlessly with existing ATC (see Fig. 8).

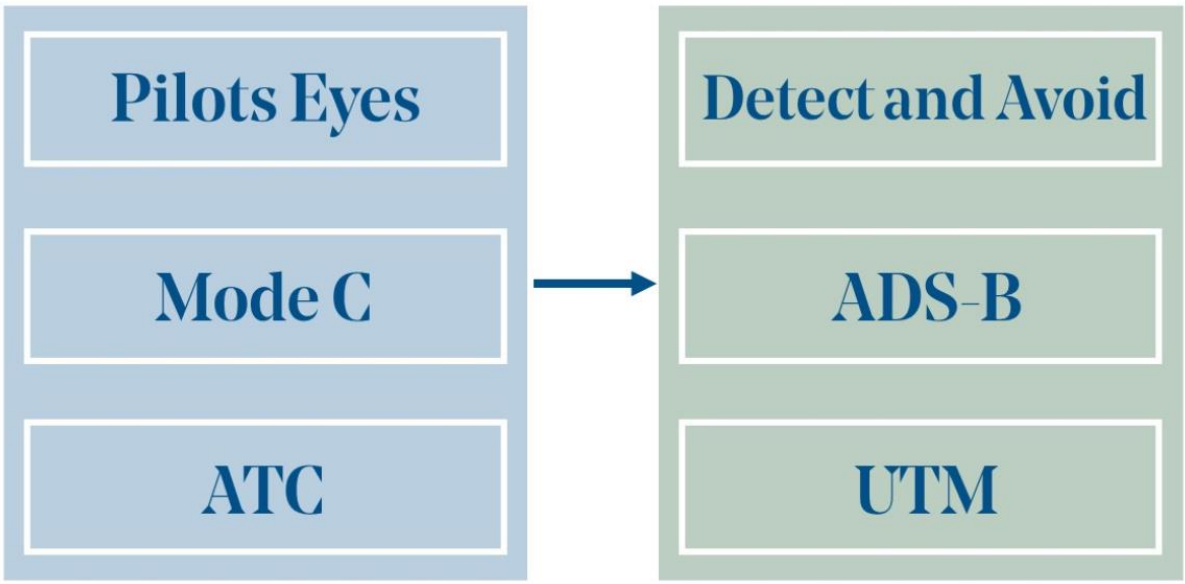


FIGURE 8: AIRSPACE REGULATIONS TRANSITION REQUIRED TO ACCOMMODATE UA

In this chapter, a fast replanning path-finding philosophy was chosen and written in MATLAB to simulate the behaviour of the UAM aircraft in typical congested airspace conditions. Studying the code's performance and results will allow for a deeper analysis of the mission computer's tasks and provide insight into the sizing of a processor for the mission computer.

A multitude of algorithms have been developed for the purpose of 2D computation of an optimal path through a gridworld with fixed obstacles since Djikstra's original algorithm for this purpose [40]. These are listed in Table 2.

Most of the algorithms are descendants or optimisations of each other, rarely "reinventing the wheel" but rather scaling down the effort required. It is thus unsurprising that the pseudocodes behind these algorithms can have significant amounts of overlap. Nevertheless, a few popular ones in the fields of robotics, artificial intelligence, and game development are listed below with a few notable merits and demerits.

However, a proven, 3D path-finding algorithm that can process both fixed and moving obstacles has yet to publicly released. This has been attempted using an A* algorithm.

TABLE 2: COMPARISON OF POPULAR PATH-PLANNING ALGORITHMS

Potential Field	Floyd-Warshall	Genetic Algorithm	A* Algorithm
Artificial Potential field assigned to every point in the world	Subset of Dynamic Programming	Evolutionary algorithm inspired by the process of natural selection	"Estimated cost to target" is a heuristic
Goal node has lowest potential and starting node has highest potential, obstacles and no-fly-zones also having high potential, so the UAV is pulled towards goal whilst obstacles push UAV away.	Finds shortest path between all pairs of vertices	Does not scale well with complexity [41]	Popular in games such as Pacman and Tower defense
Poor optimization in narrow passages, dynamic environments, and symmetrical obstacles	Improvement upon Dijkstra's and Bellman-Ford algorithms		Relies heavily on heuristics

The fundamental equation behind A* and its descendants is:

$$f(n) = g(n) + h(n)$$

Where *g(n)* is the path cost to get from the start node to node *n*, and *h(n)* is the heuristic (or approximation) representing one of the breakthroughs behind the revolutionary algorithm. It is the Euclidian (as opposed to Manhattan) distance behind two points, obtained using the Pythagorean theorem. Fig. 9 shows the static A* algorithm that the following section builds off of.

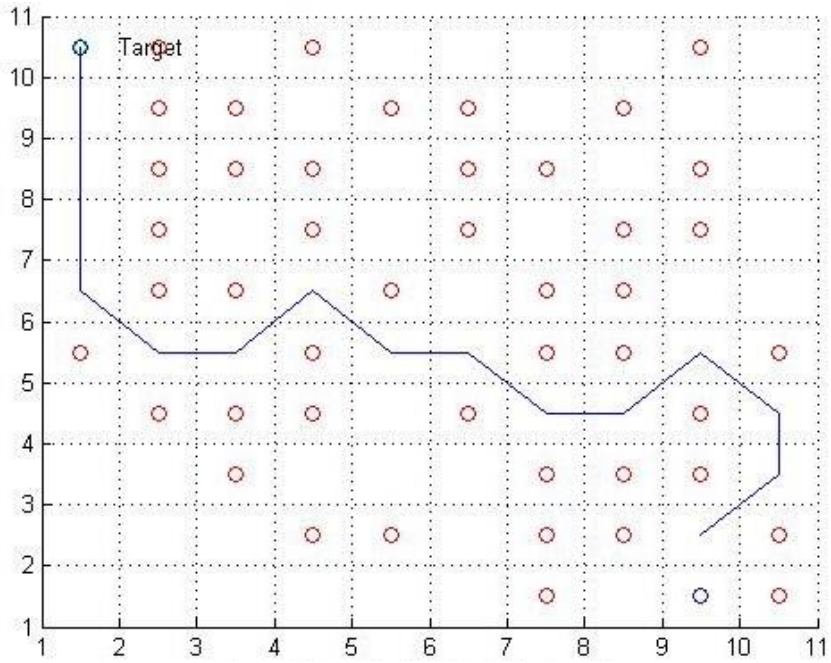


FIGURE 9: 2-D STATIC A* ALGORITHM OPTIMAL PATH EXAMPLE

5.1.2 Novel Path-Planning Algorithm Design

The algorithm design process was first established, listing major design steps in the following order:

- 2D A* with fixed obstacles
- 3D A* with fixed obstacles
- 3D A* with moving obstacles
- Creation of GUI and presets

Fig. 10 shows flowcharts of both the initial 3D A* algorithm with fixed obstacles and the subsequent A* algorithm with moving obstacles, of which one of the blocks is the original A* algorithm (Run *A_Star1_3D.m*). In essence, a path is found, the aircraft and obstacles are moved by a timestep, if an obstacle obstructs the path, static A* is run again from the UAM's location, until the target is reached.

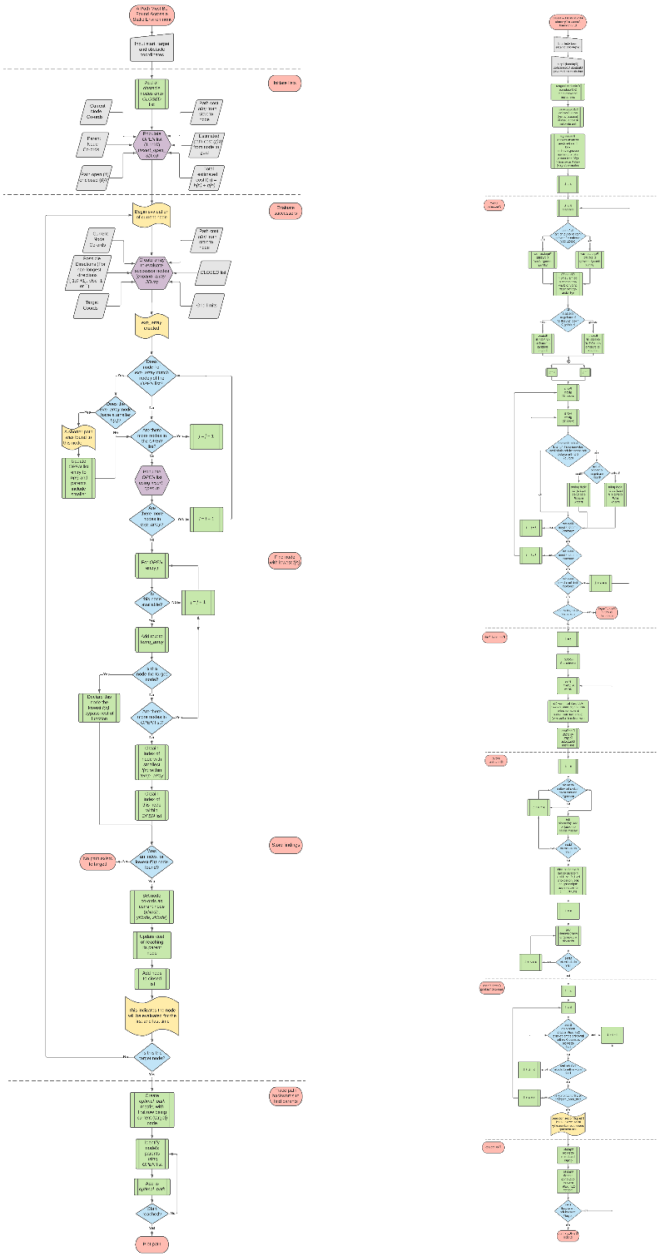
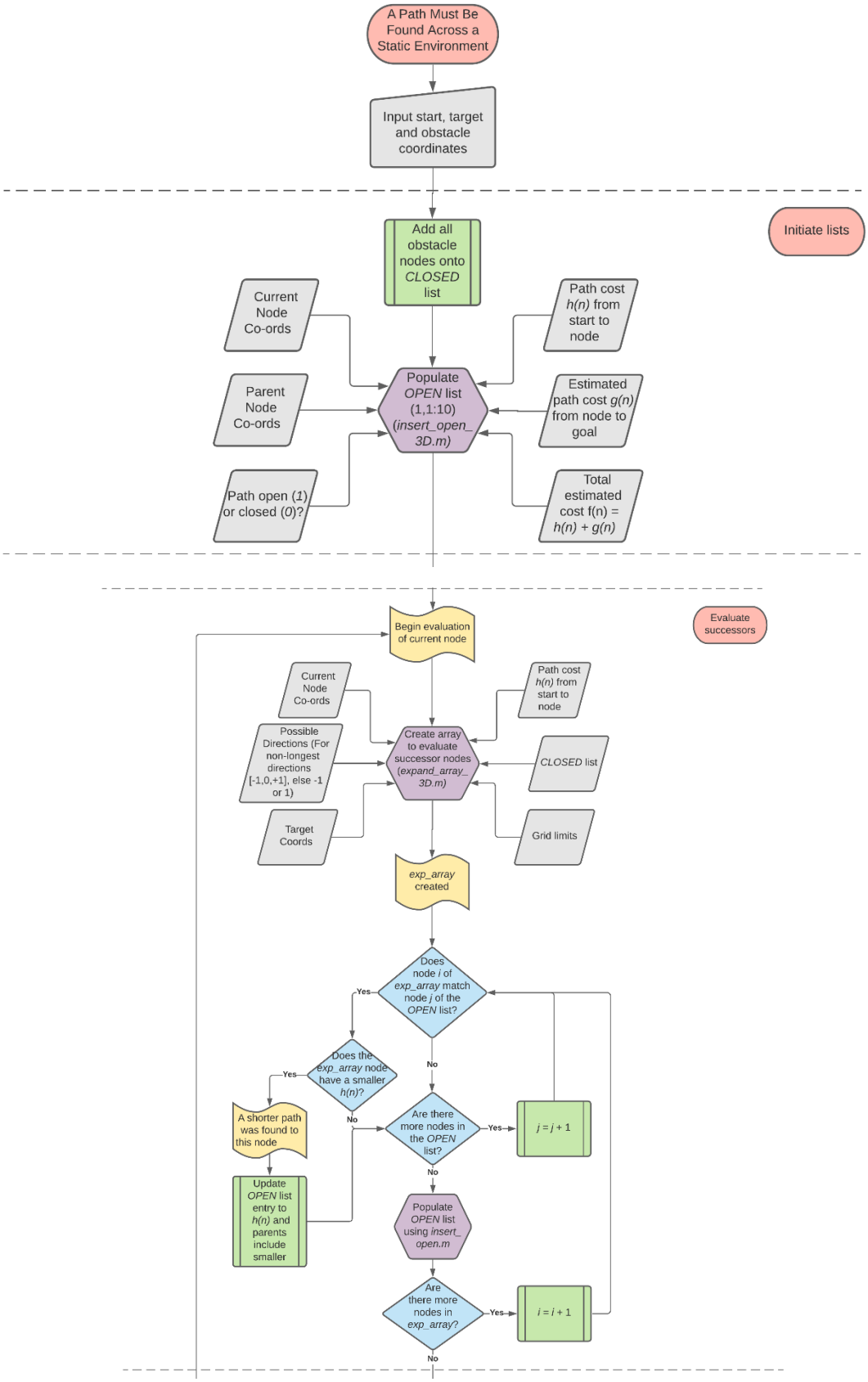
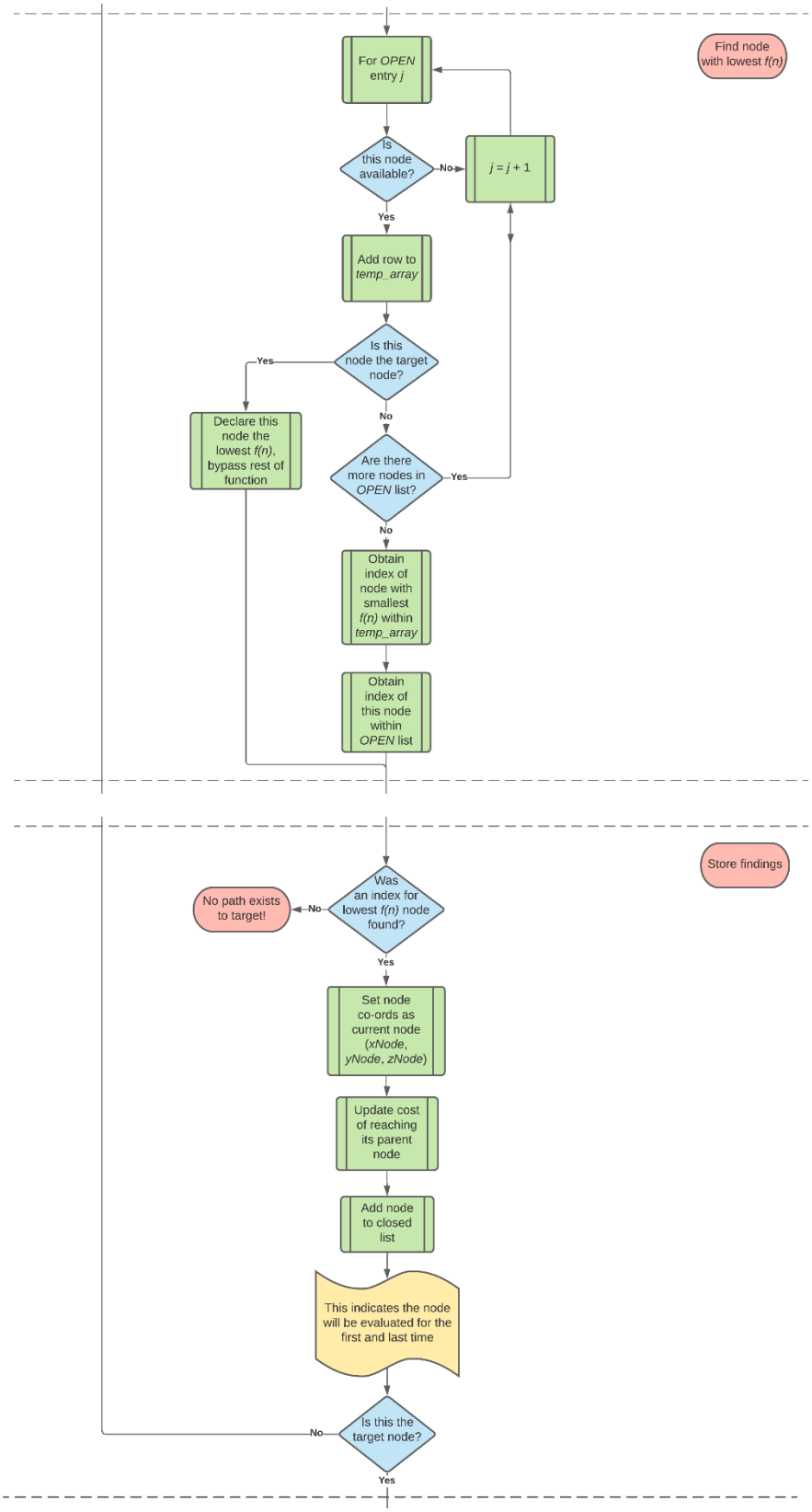


FIGURE 10: A* FLOW CHARTS

The following figures show closeups of the static A* flow chart on the left of Fig. 10.




Find node with lowest f(n)
For OPEN entry j
Is this node available?
No
j = j + 1
Yes
Add row to temp_array
Is this node the target node?
Yes
Declare this node the lowest f(n), bypass rest of function
No
Are there more nodes in OPEN list?
Yes
No
Obtain index of node with smallest f(n) within temp_array
Obtain index of this node within OPEN list
Store findings
Was an index for lowest f(n) node found?
No
No path exists to target!
Yes
Set node co-ords as current node (xNode, yNode, zNode)
Update cost of reaching its parent node
Add node to closed list
This indicates the node will be evaluated for the first and last time
Is this the target node?
No
Yes

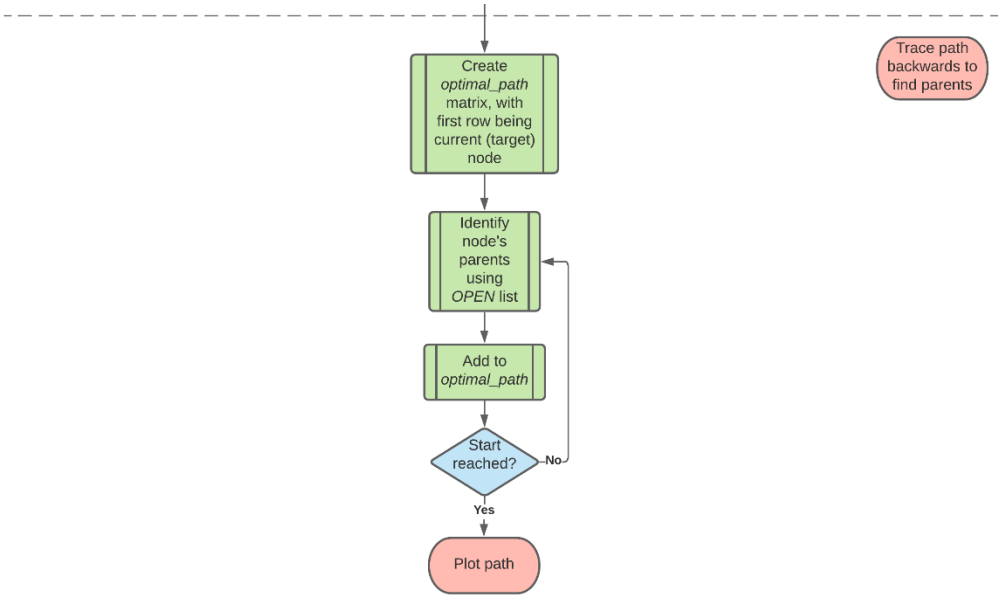
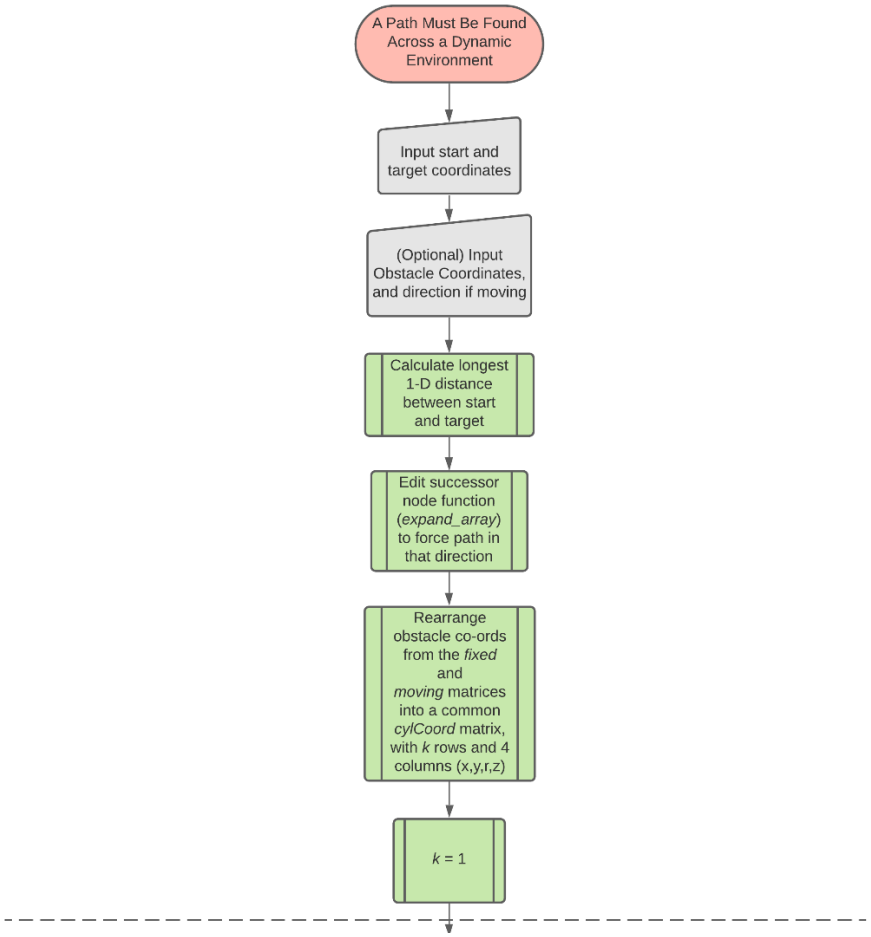
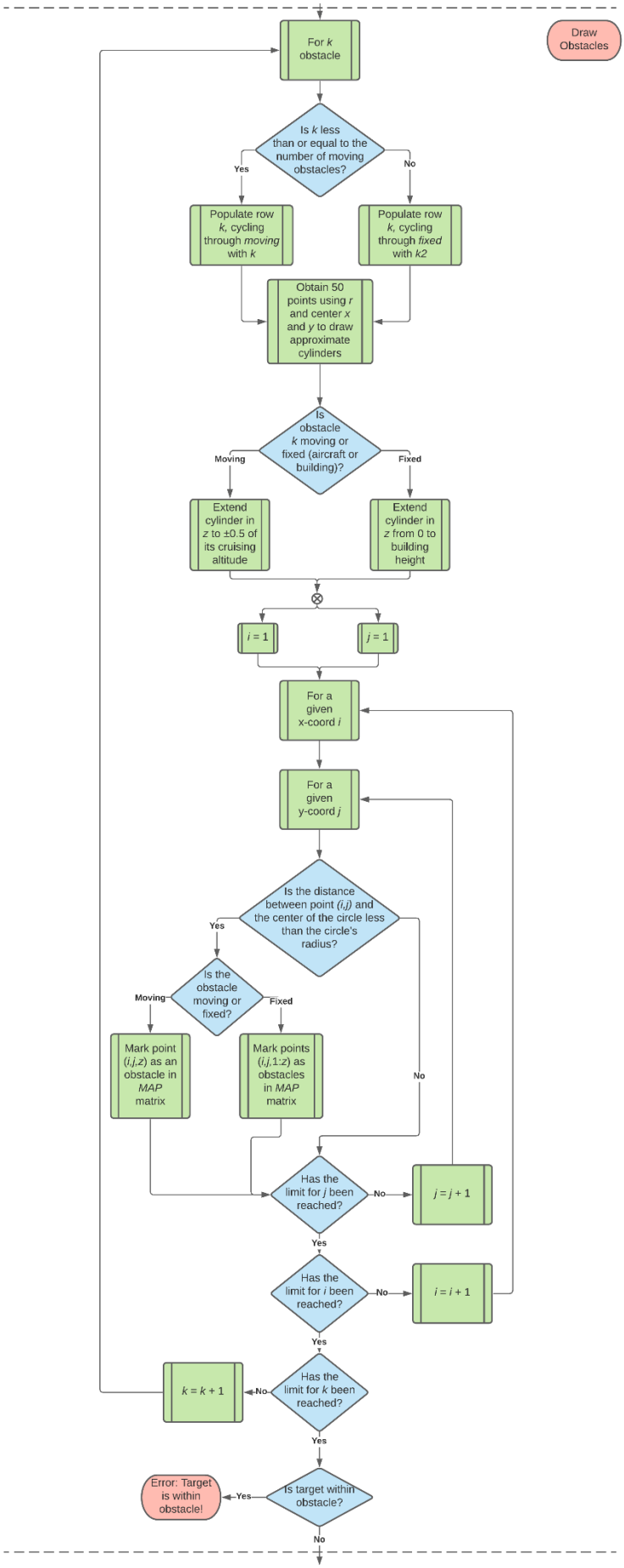
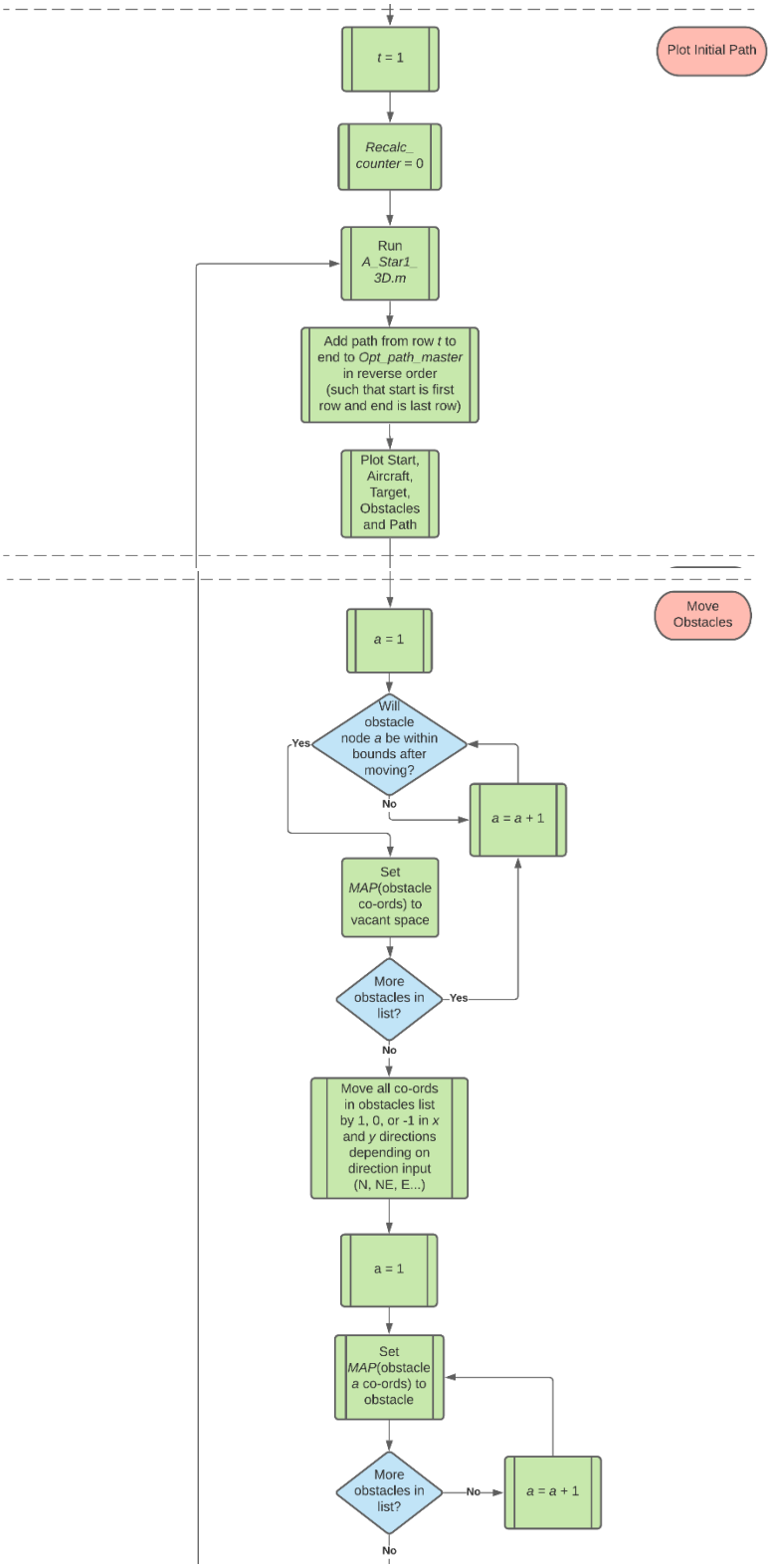


FIGURE 11: FINAL STEP FOR *A_STAR1_3D.M*

The next set of figures, leading up to Fig. 12, relate to the moving environment, i.e. the *A_Star1_3D_moving.m* program (flow chart on the right of Fig. 10).




Draw Obstacles
For k obstacle
Is k less than or equal to the number of moving obstacles?
Yes
No
Populate row k, cycling through moving with k
Populate row k, cycling through fixed with k2
Obtain 50 points using r and center x and y to draw approximate cylinders
Is obstacle k moving or fixed (aircraft or building)?
Moving
Fixed
Extend cylinder in z to ±0.5 of its cruising altitude
Extend cylinder in z from 0 to building height
i = 1
j = 1
For a given x-coord i
For a given y-coord j
Is the distance between point (i,j) and the center of the circle less than the circle's radius?
Yes
No
Is the obstacle moving or fixed?
Moving
Fixed
Mark point (i,j,z) as an obstacle in MAP matrix
Mark points (i,j,1:z) as obstacles in MAP matrix
Has the limit for j been reached?
No
j = j + 1
Yes
Has the limit for i been reached?
No
i = i + 1
Yes
Has the limit for k been reached?
No
k = k + 1
Yes
Is target within obstacle?
Yes
Error: Target is within obstacle!
No


Plot Initial Path
t = 1
Recalc_ counter = 0
Run A_Star1_ 3D.m
Add path from row t to end to Opt_path_master in reverse order (such that start is first row and end is last row)
Plot Start, Aircraft, Target, Obstacles and Path
Move Obstacles
a = 1
Will obstacle node a be within bounds after moving?
Yes
No
a = a + 1
Set MAP(obstacle co-ords) to vacant space
More obstacles in list?
Yes
No
Move all co-ords in obstacles list by 1, 0, or -1 in x and y directions depending on direction input (N, NE, E...)
a = 1
Set MAP(obstacle a co-ords) to obstacle
More obstacles in list?
No
a = a + 1
No

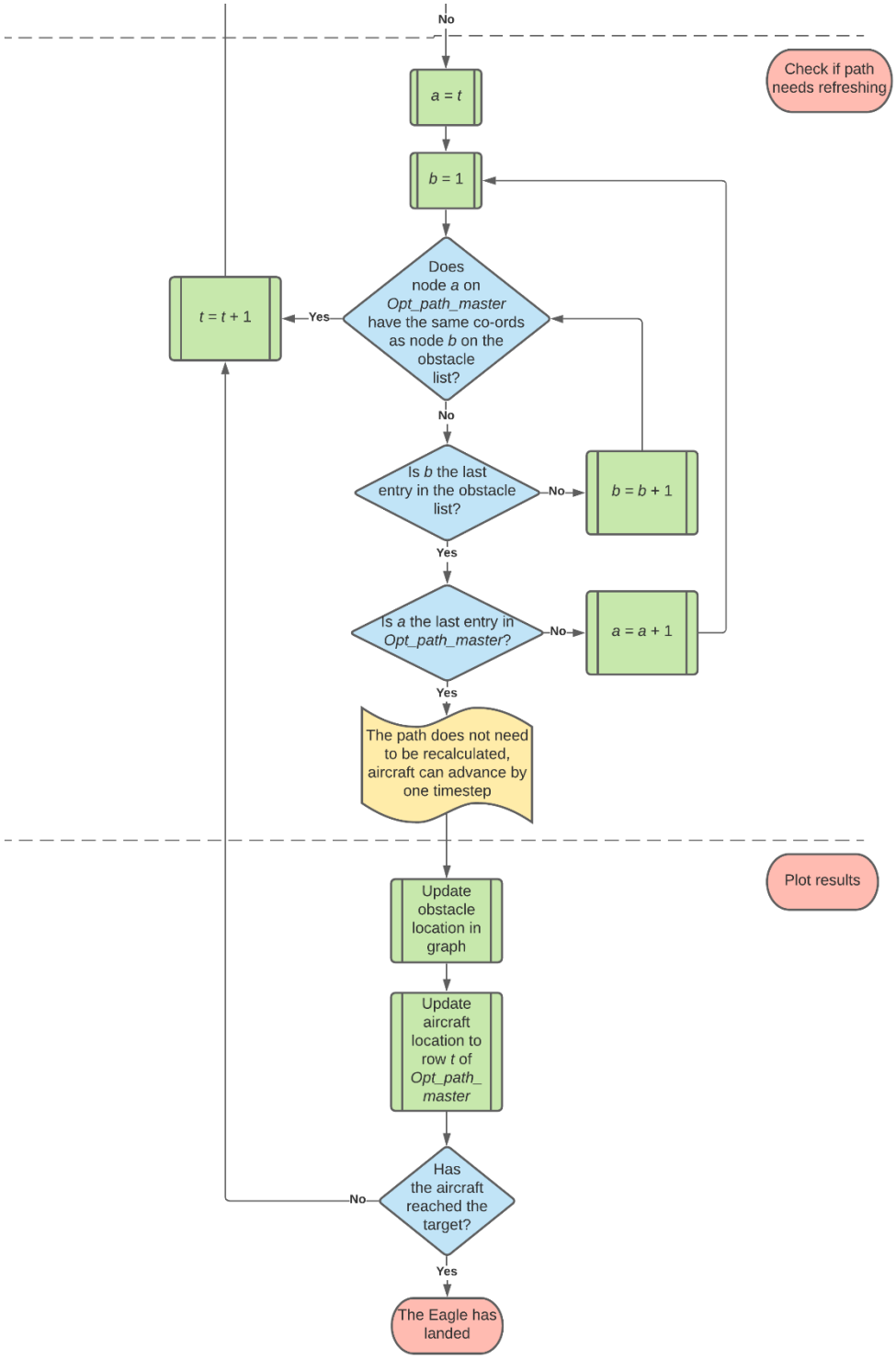


FIGURE 12: FINAL STEP FOR *A_STAR1_3D_MOVING.M*

As the algorithm was successively extended and optimised iteratively, the computational effort required to calculate the optimal path increased. The most notable increases, it was discovered, occurred not when adding obstacles (fixed or moving), but rather whilst increasing the size of the gridworld. A change in size from 10x10x10 to 100x100x100 overloaded the testing processor, which is notably larger than the mission computer's processor onboard the aircraft, and thus had to be scaled down to the more manageable 50x50x50 gridworld used in the final version of the algorithm.

It is clear that an even more simplified environment will have to be tested in-flight, along with a variety of timestep lengths.

A unit length of the 50x50x50 gridworld can therefore be said as representing approximately 2 miles. A start node and target node is initially inputted, and obstacles are inputted as cylinders of variable location, radius, and a corresponding height, interpreted differently by the algorithm if the obstacle is fixed (such as a building) or moving (such as an aircraft). For the former, the height inputted is the height of the building from sea level; for the latter, the height corresponds to the cruising altitude of the aircraft, of one unit of thicknesses (resembling a "disc"). This corresponds to international vertical and horizontal separation minima laws between aircraft (approx. 5 NM horizontally and 0.16 NM vertically) [42].

The optimal path was calculated for similar environments for each version of the program. The results are shown in Table 3. These results were obtained on a desktop computer with a 12-core 4.2Ghz clock speed AMD Ryzen 3900X processor. This information can be used to narrow down the potential range of on-board processors prior to testing and final processor selection.

TABLE 3: A* ALGORITHM VERSION HISTORY

Version	Category	Improvement	Time to execute (s)	Factor
1	Fundamentals	2D 10x10 Grid, fixed obstacles	0.28	
2	Search Area	Increased to a 3D, 10x10x10 grid	0.32	1.14
3	Obstacles	Moving obstacles, path reset after each timestep	0.47	1.47
4	Obstacles	Fixed and moving obstacles simultaneously	0.53	1.13
5	Search Area	Increased to a 50x50x50 Grid	117	221
6	Path reset	3D Grid, fixed and moving obstacles, reset only if path crossed	49	0.42
7	Direction	Backwards travel disallowed	34	0.69
8	Direction	Backwards and lateral travel disallowed	13	0.38
9	Interface	GUI created		

As mentioned before, when increasing the gridworld from 1,000 nodes to 125,000 nodes in version 5, the processing time increases by a factor nearly double that of the nodes (125x more nodes, 221x more processing time). Finding an appropriate resolution and range combination in the modelling of the surrounding environment will be paramount in obtaining practical results in a realistic time frame. This difficulty is mitigated by reducing the number of directions the path can take from any given node from 26 in version 6 to 9 in version 8 (see Fig. 13). *direction.m* was written to infer the direction of travel from the positions of the *start* and *end* points and, separately, handle obstacle directions.

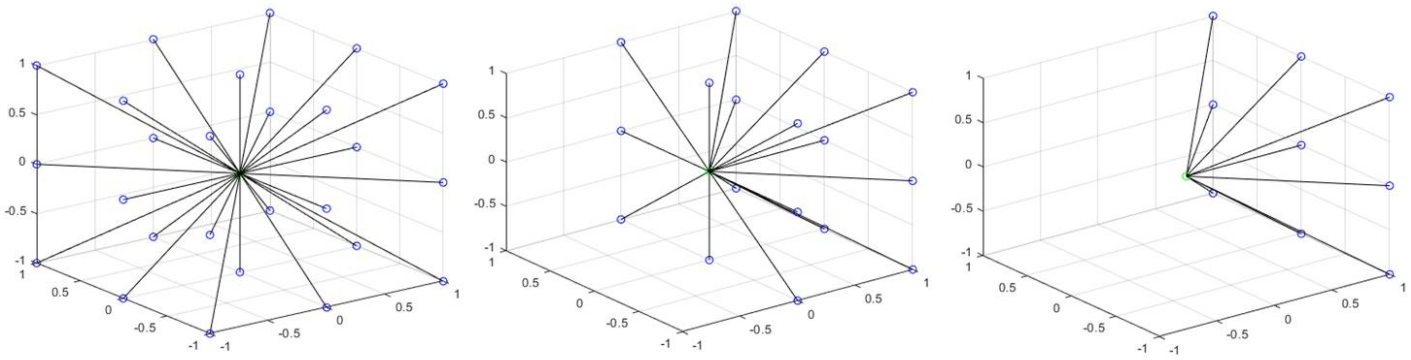


FIGURE 13: POSSIBLE PATH DIRECTIONS IN VERSIONS 6, 7, AND 8 (ASSUMING A TRAVEL DIRECTION OF +X)

A variety of presets were created to simulate real-world scenarios that UAM aircraft may face. Images of the GUI and preset results are pictured below, in Figs. 14-17. Readers are encouraged to download the applet and run it from their own computers using Matlab R2021a or above[1]. The user can select and edit presets, as well as create entirely new environments from scratch. These features demonstrate the modularity and flexibility of the algorithm at the heart of the flight-critical path-planning mission management function.

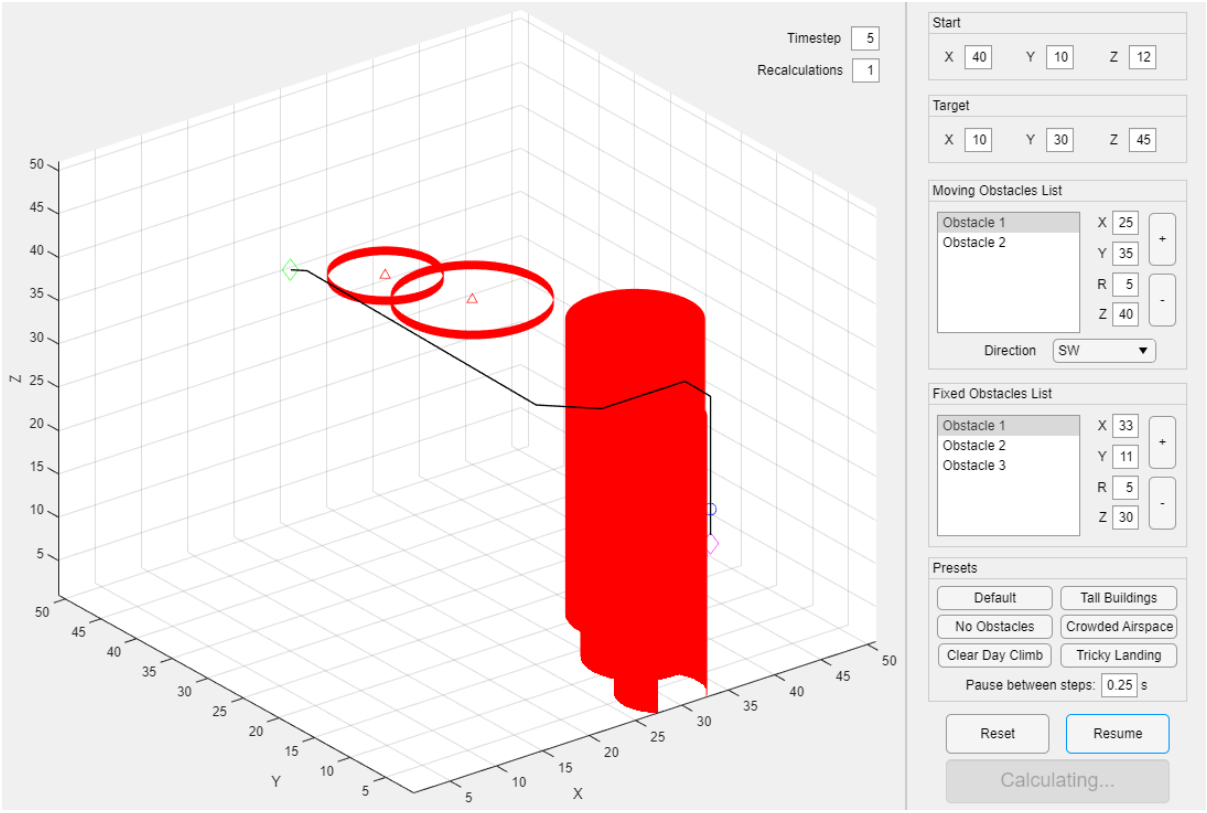


FIGURE 14: CLEAR DAY CLIMB

[1] The link is https://www.dropbox.com/s/tecaqg3fa84ci5f/JeanMichelAlvarez_AStar_EFL_Code.zip?dl=0 and will be available throughout the assessment period of this report. Run AStarApp.mlapp to begin. If there is a problem with the link or the files please do not hesitate to contact me on my university email address.

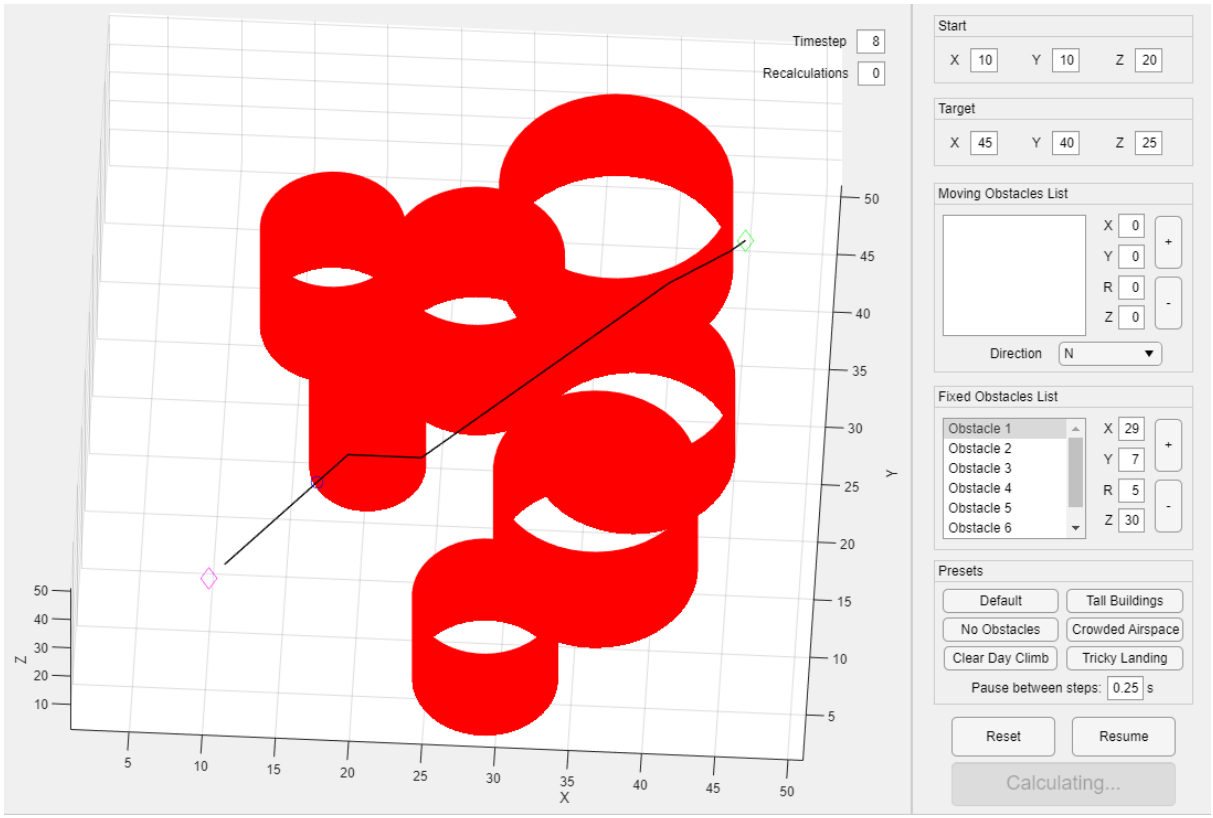


FIGURE 15: TALL BUILDINGS

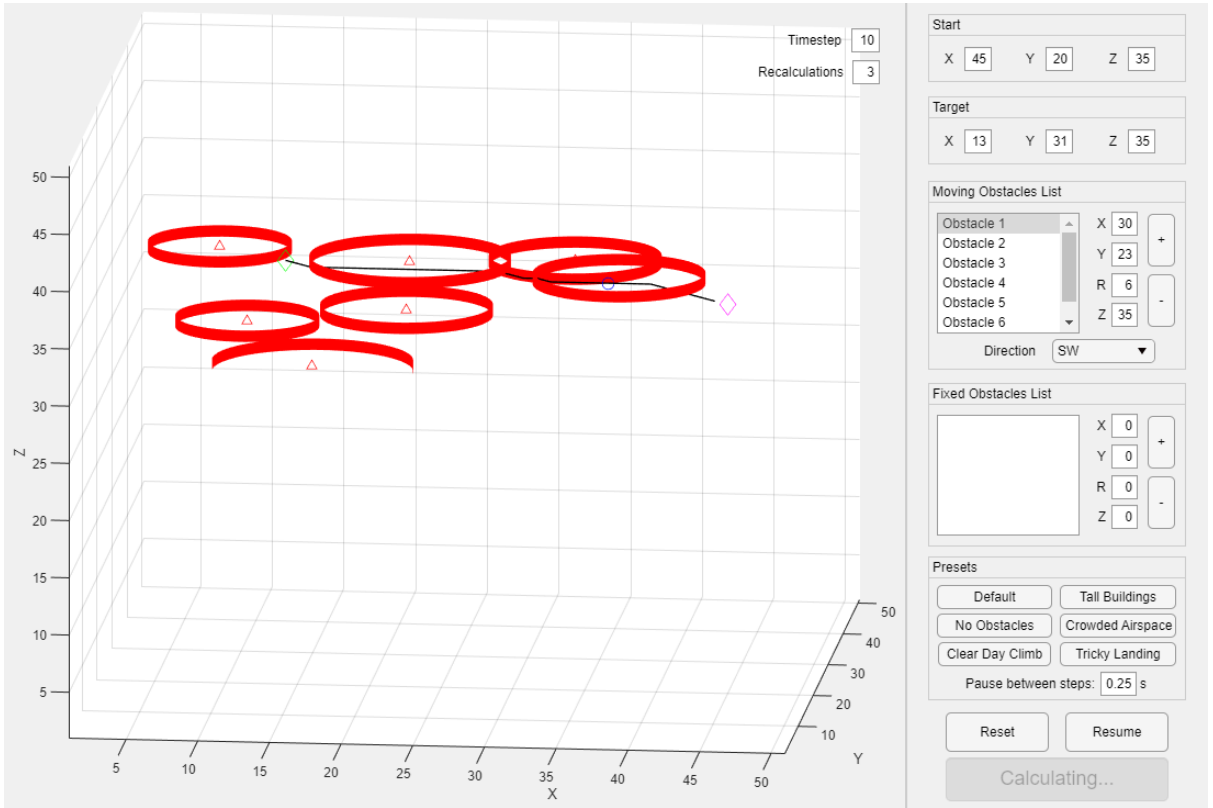


FIGURE 16: CROWDED AIRSPACE

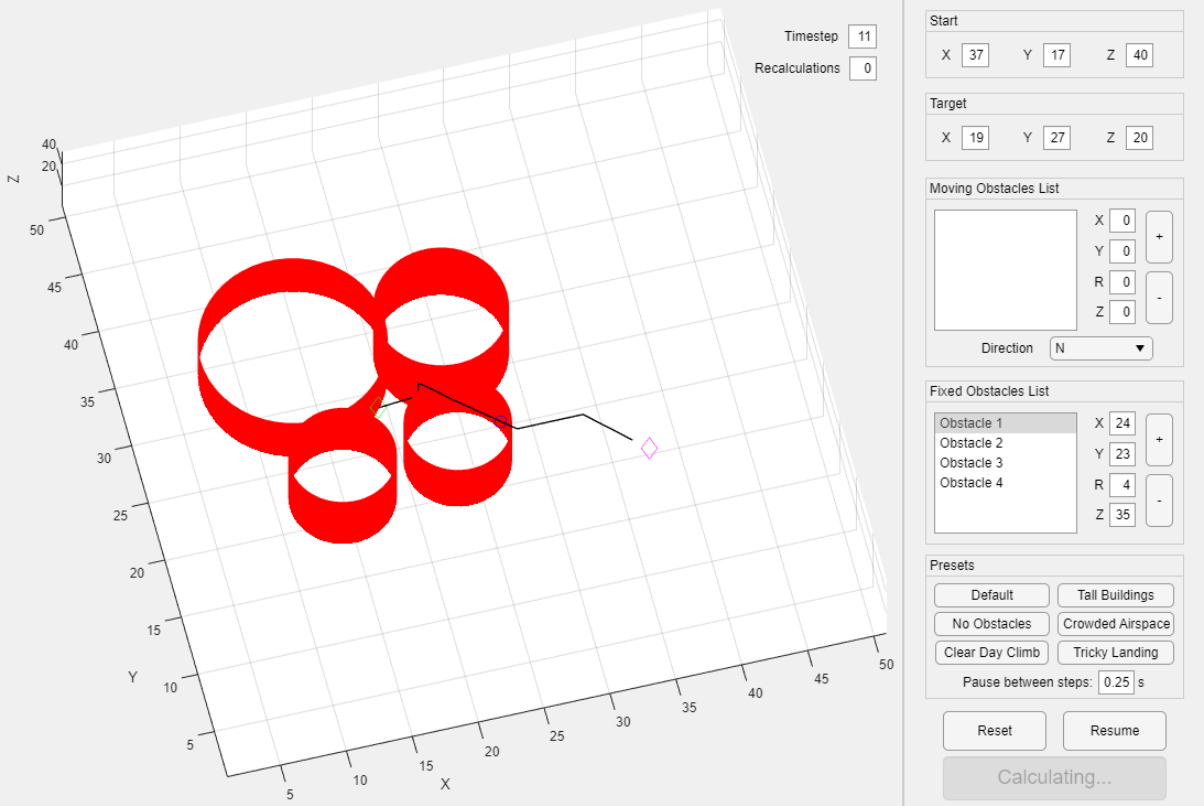


FIGURE 17: TRICKY LANDING

5.2 Emergency Landing Zone Identification

Another essential element of mission management is the ability to execute an emergency (forced) landing (EFL). The aircraft can enter this process for a variety of reasons and in a variety of different states, ranging from full propulsion and control systems available, to both being offline. For the sake of brevity these states, affecting the manner in which the aircraft approaches the landing zone, will not be delved into. One element of this process remains constant however: the initial, flight-critical priority of identifying the landing zone.

An algorithm was written to fulfil this purpose, to be ran on the mission computer when necessary. A signal would first be sent to the EO camera via Ethernet to rotate to the gimbal's limit, -85° to the horizontal. The 5° incline is assumed to be insignificant (although this will not be the case for future iterations of the algorithm where terrain topography through camera depth analysis is ascertained). A frame is then taken from the live video feed and sent to the EFL zone-finding program. The image is reduced to grayscale, and an edge detection algorithm is used to convert the image into an identically-sized logical array, where 0 represents space and 1 a line. Multiple edge detection algorithms are evaluated and compared. The most notable of these is the Canny edge algorithm, developed by John Canny in 1986, whose original publication has been cited over 37,000 times. The first three images chosen for testing (see Figs. 18-20) are from routes that are likely to be popular with UAM aircraft according to the team's sales forecast [43], with the fourth image serving as a more extreme example (see Fig. 21).

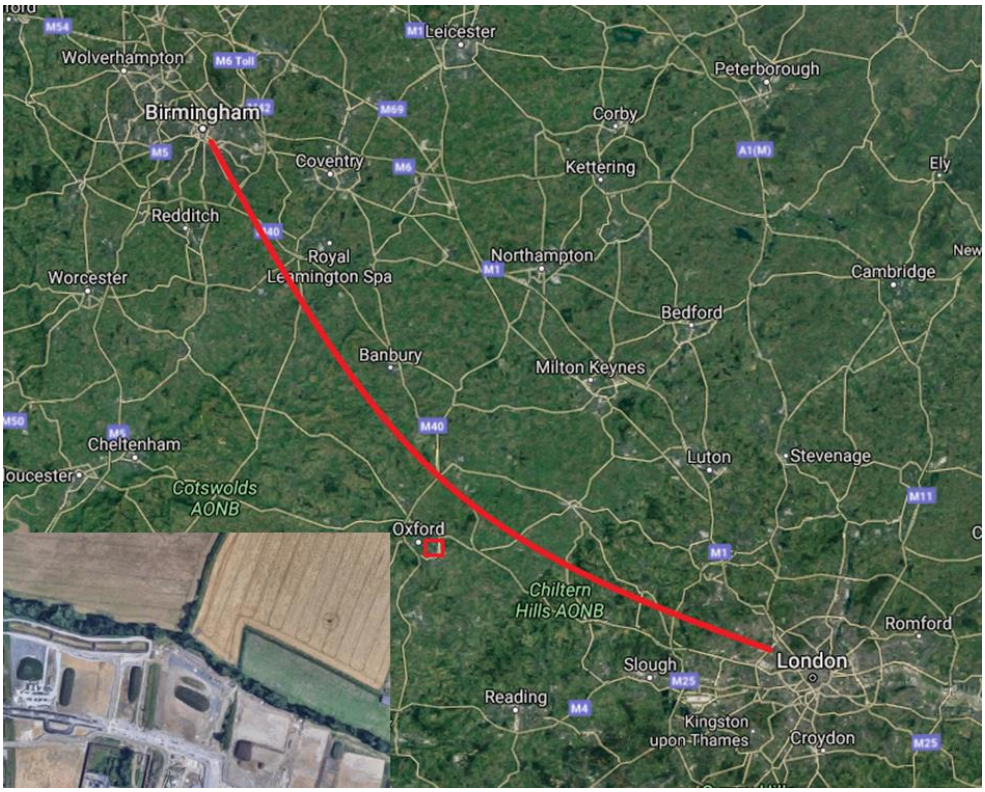


FIGURE 18: IMAGE 1, BIRMINGHAM – LONDON

FIGURE 19: IMAGE 2, BRUSSELS - THE HAGUE

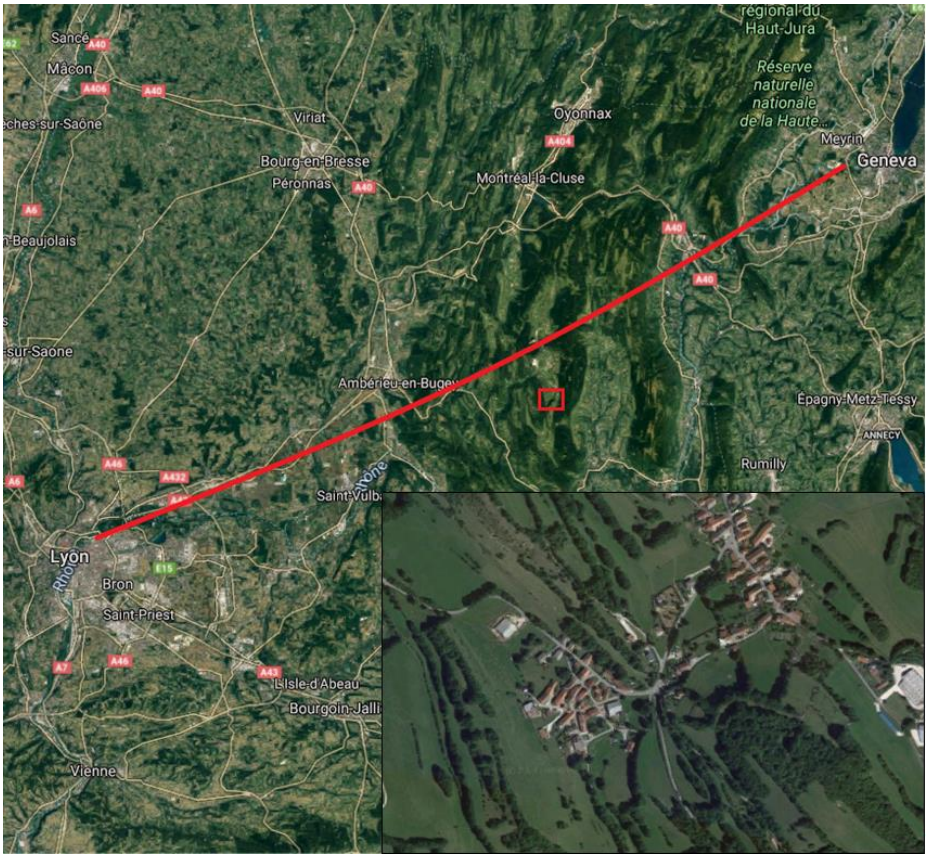


FIGURE 20: IMAGE 3, LYON – GENEVA

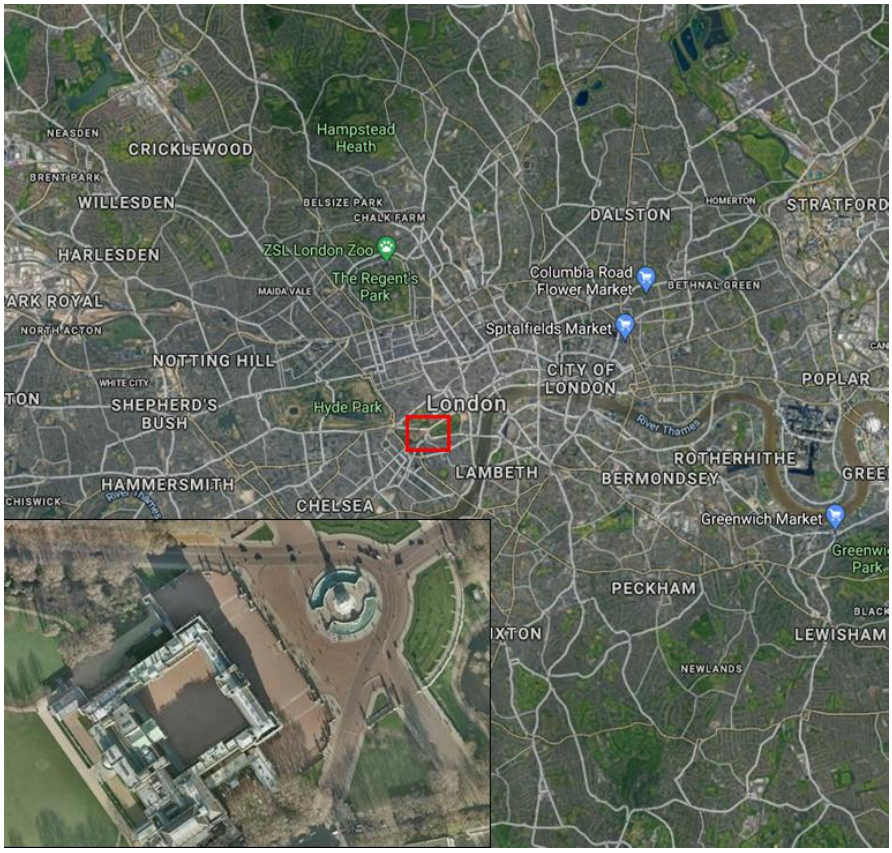


FIGURE 21: IMAGE 4, BUCKINGHAM PALACE

The results of the testing are tabulated in Table 4, and Image 1's edge detection results are illustrated in Fig. 22. All timings were taken on the same computer and in the same session to ensure repeatability.

TABLE 4: EDGE DETECTION ALGORITHM TIMINGS

	Canny	Prewitt	Sobel	Roberts	Approxcanny
Image 1	0.104105	0.103010	0.096833	0.096204	0.099178
Image 2	0.119695	0.103855	0.101089	0.106149	0.101406
Image 3	0.110831	0.099028	0.104259	0.100214	0.126735
Image 4	0.117547	0.099126	0.097021	0.133336	0.295548
Average (s)	0.113045	0.101255	0.099801	0.108976	0.155717

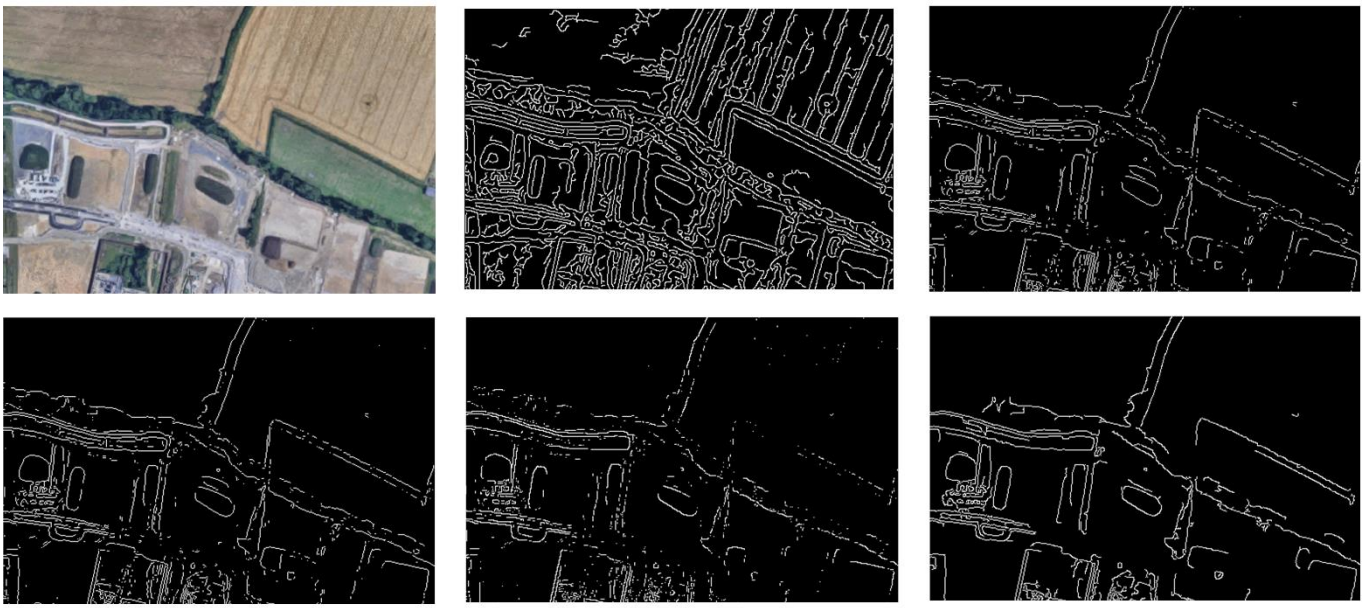


FIGURE 22: FROM LEFT TO RIGHT, TOP TO BOTTOM: ORIGINAL IMAGE, CANNY, PREWITT, SOBEL, ROBERTS, APPROXCANNY

The next part of the process is the identification of the landing zone from the logical array. This is done by attempting to fit a circle of a certain diameter within the image. If a landing zone is found, the circle will enlarge and the algorithm will try to find another space for it. If no landing zone is found, the algorithm will shrink the circle and run it again. This ensures the landing zone with the most clearance is selected for the next step. The results from this step are illustrated in Fig. 23.[2]

[2] See footnote [1] for EFL code link to test the algorithm on any given image.

FIGURE 23: LANDING ZONE IDENTIFICATION (CANNY)

In the case of Image 1, the selected landing zones from two separate edge detection algorithms are overlayed. These are from the Canny (smaller circle) and Sobel (larger circle) algorithms, the former arguably the most accurate, and latter the most time-efficient according to Table 4. Cross-referencing this with the results depicted in Fig. 19 shows that the Canny edge detected wheat field rows as being obstacles, whilst Sobel's "corner-cutting" actually produced a more advantageous and sensible result, having picked a wider field further away from man-made objects. This demonstrates the need for an additional layer of decision-making, for the purpose of selecting the most appropriate edge-finding algorithm for a given situation.

5.3 Novel Algorithm Future Improvements

It was shown that the two aforementioned algorithms yield satisfactory results in their respective rough approximations of crucial real-world scenarios. However, as is the case with every other piece of software existing within the digital ecosystem, it is never truly "done", and should be endlessly honed, optimised, and improved until a fundamentally different way of looking at the problem is discovered and the program must be retired permanently for another to take its place.

It is therefore of interest to state the next logical steps in the direction in which these algorithms must be developed to further increase their chances of gaining relevance and future use in the UAM sector. A few of these steps are tabulated below in Table 5.

TABLE 5: POTENTIAL ALGORITHM IMPROVEMENTS

A* Path-Planning	EFL Identification
Add battery charge indicator which would deplete during flight and alert the operator/GCS if charge insufficient to accomplish mission, and automatically proceed to EFL Identification.	Aircraft able to identify radius of circle required based on height, then work off "candidate" system. If none found, decide whether to continue search or shrink landing radius.
Consider wind-speed along path and vary path slightly to avoid flying upwind.	Consider proximity to high reflectivity (man-made) objects.
Add various paths and speeds for moving obstacles rather than one single direction and speed.	For each candidate, assign risk level for human casualties, external property damage, and aircraft damage.
Apply novel Dynamic A* algorithms (also known as D* and D*-Lite) to compare timings.	Use a Canny-Deriche detector for increased accuracy [44].
Perform calculations for gridworlds of varying resolutions, with a unit length representing anywhere from 0.1-10 NM.	Add a Dubins Path to calculate gliding trajectory based on minimum turn radius to land the aircraft with minimal downward force.

See Appendices D-J for the code behind each of these algorithms.

6. Safety

A major role of the avionics system in any aircraft is to provide safety and redundancy throughout all flight operations. Redundancies and management units have been implemented into the physical architecture when possible, mitigating the risks of individual component failure.

6.1 Initial Risk Assessment

Table 6 shows the initial risk assessment that was undertaken during the preliminary design stages of the avionics system. Severity (S) represents the magnitude of the impact on the aircraft and its passengers. Likelihood (L) is how often the issue is likely to occur. Both of these elements are scored on a scale of 1 to 5, with 5 being very severe/very likely. Risk Rating (RR) is obtained by multiplying S and L, and aids in the identification of high-priority risks.

TABLE 6: PRELIMINARY RISK ASSESSMENT AND MITIGATION

Item	Category	Risk	Risk Rating			Control Measures	Residual Risk		
			S	L	RR		S	L	RR
1	Alarm Systems	Components operate outside of	4	4	16	Effective cooling measures	4	1	4
2	Communications Systems	Loss/Jamming of communications	4	4	16	Pre-installed landing zone identification and landing system; circle around and scan for signal to reengage with GCS	1	2	2
3	Computer Systems	Processing Units Overloaded	5	3	15	Size processor speed correctly to task; testing	5	1	5
4	Control Systems	Inability to actuate control surfaces	4	4	16	Redundant servo motors and wiring to flight control computer	4	1	4
5	Environment	External agent intrusion (humidity water dust salt temperature)	3	4	12	Aircraft to be stored in regulated, enclosed conditions; Visual Inspection prior to flight.	3	2	6
6	General	RFI/EMI Interference	5	3	15	Adequate EMI shielding of components (Faraday cage etc)	5	1	5
7	General	Total loss of control during testing	5	3	15	Remotely activated flight termination system (deadman output) integrated into flight computer and GCS software	5	1	5
8	Powerplant	Loss of battery power	5	3	15	Switch to emergency power supply; distribute power only to flight-critical systems	1	2	2
9	Sensor Systems	Camera/radar fails to detect possible collision	5	2	10	Rigorous testing; monitoring of camera feeds by GCS	4	1	4
10	Sensor Systems	Pitot Static Tubes icing, water intrusion, insects, dust and debris	3	3	9	Covered when on ground with red bags marked "remove before flight"	2	1	2

Although this assessment helps provide insight during the very early stages of the design process, it is clearly not nearly comprehensive enough to support later stages of design. A separate tool, such as FMEA, must be used to appropriately identify and prioritise risks.

6.2 Failure Mode and Effects Analysis (FMEA)

FMEAs are a vital tool for identifying all possible failures in a design. This process analysis tool is used to identify the modes in which something might fail, and the consequences of these failures. This will allow the Systems team to efficiently eliminate or reduce failures by starting with the highest-priority ones according to the tables below.

The scope of this analysis is restricted to avionics-related equipment within HopAir's UAM aircraft. It is extremely important that this document be a living document, continually edited and updated throughout the aircraft's development. For subsequent design phases, a contribution from a multidisciplinary FMEA team will provide more perspective and depth to this crucial document and be beneficial to the aircraft's design as a whole. It should be noted that this list is clearly non-exhaustive as there are virtually limitless ways in which a complex system such as an aircraft's avionics can fail, but it can undoubtedly help create contrast between serious and insignificant risks.

- SEV: Severity (on a scale of 1 to 10, 10 being severe)
- OCC: Occurrence (on a scale of 1 to 10, 10 being frequent occurrence)
- DET: Detection likelihood (on a scale of 1 to 10, 10 meaning detection is highly unlikely)
- RPN: Risk Priority Number (higher number = higher priority)

Process Step	Potential Failure Mode	Potential Failure Effect	SEV	Potential Causes	OCC	Current Process Controls	DET	RPN	Action Recommended
Processing Units	Flight Computer Miscalculation	Erroneous information fed to servo motors	7	Error in Flight Computer code	8	SWIL and HWIL testing	3	168	Integrated circuit (IC) to compute the same step twice or more and compare outcome
	Error in mission plan	Erroneous information fed to Flight Computer	7	Human error (inputting wrong destination)	8	Cross-referencing of on-board and GCS mission directives	3	168	Accelerate progress towards seamless integration of user-inputted destination and mission plan
	Excessively high CPU usage	Limited/Slowed Processing Functionality	6	Incorrectly sized processor	5	SWIL and HWIL testing	2	60	Ensure CPU usage remains below 80% during processor-intensive flight phases
	Processor Failure	Loss of CPU functionality	7	Overheating, power cut	3	Redundant computers activated by independent CPU health monitors	5	105	Test health monitors actively, potentially increase redundancies
Printed circuit board (PCB)	Latchup in CMOS	IC short circuit	8	Single event upset, i.e. heavy ions or protons	3	Fast detection and power cycling	6	144	Add layer of insulating oxide (such as Silicone) and have a separate tap connection for each transistor
	Foreign Substance (Moisture, dust, chemicals) near PCBs	Loss of PCB functionality	9	Excessive exposure of internal subsystems during maintenance	4	Conformal coating	5	180	Apply thicker coat (100-250 µm) to frequently-maintenanced PCBs

Process Step	Potential Failure Mode	Potential Failure Effect	SEV	Potential Causes	OCC	Current Process Controls	DET	RPN	Action Recommended
Printed circuit board (PCB)	Chemical Fluid Leakage	Loss of PCB functionality	9	Manufacturing Error	4	Visual Inspection of PCBs during maintenance	7	252	Chemical swab testing during maintenance to detect leaks quicker
	Component barrier breakage	Internal elements exposed to oxygen and humidity, accelerating ageing and failure	5	Ageing	4	Visual Inspection of PCBs during maintenance	7	140	Review component lifespans more often, adjusting accordingly
	Excessive Thermal Stress	Weakened/broken solder joints, Loss of PCB functionality	7	Exposure to heat and humidity, Manufacturing errors	5	Visual Inspection of PCBs during maintenance, testing	6	210	Research manufacturer's client testimonials prior to ordering
Software	Software Bug During Startup	Delayed take-off	4	Buggy code	5	Code review, testing and debugging by multiple software engineers	4	80	If problems persist, replace engineers
	Software Bug After Takeoff	Flight mission abort	9		5		3	135	
CAN Bus	Data Bus failure	Loss of functionality of various subsystems, processors disconnected from each other	8	Short circuiting/loss of continuity of CAN-H/CAN-L lines to each other, or excessive radio interference	4	CAN Bus health monitor, redundancies	4	128	Increased redundancies
Sensors	Wrong/Missing Input	Incorrect flight information processed by system	7	Miscalibration of sensor system	6	Multiple data sources, such as GPS + IMU	2	84	GCS to cross-reference AHRS with independent aircraft tracker

Process Step	Potential Failure Mode	Potential Failure Effect	SEV	Potential Causes	OCC	Current Process Controls	DET	RPN	Action Recommended
Sensors	Unidentified obstacle	Collision	10	Malfunctioning radar/camera	4	Multiple visual sources to cross-check external environment information	2	80	GCS to cross-reference known building and aircraft locations in proximity to aircraft
Actuators/ Landing Gear	Valve Failure	Actuator Unresponsive	9	Worn out valve stem, seized up packing, obstruction in mechanism, excessive torque	3	System can handle up to 50% of actuators failing	3	81	Increase redundancies, attempt to operate valve manually to faster identify source of actuator issue
	Inadequate operation of servo motors	Control surface/landing gear unable to displace to correct location	8	Servo motor failure	2	Vetting of actuator manufacturer/flight heritage research	3	48	Life cycle testing prior to implementation
Battery	Over-voltage	Low temperature operation, non-uniformities in cell elements, BMS fault	8	Lithium Plating/Overheating	3	BMS (health monitor)	4	96	Backup BMS to verify voltages
	Under-voltage	Over-discharging/ excessive storage leading to breakdown of electrode materials	8	Anode copper current collector dissolved into electrolyte leading to short circuit between electrodes	3	BMS (health monitor)	4	96	
	Low Temperature Operation	Chemical reaction rates decrease (Arrhenius Law) leading to reduced power handling capacity	7	Long-term storage in cold environment	5	BMS (health monitor)	4	140	Store electronics/aircraft in temperature-controlled facilities

Process Step	Potential Failure Mode	Potential Failure Effect	SEV	Potential Causes	OCC	Current Process Controls	DET	RPN	Action Recommended
Battery	High Temperature Operation	Thermal runaway occurs in stages of varying intensity, ultimately leading to a class B flammable liquid fire	10	Inproper cooling of batteries	3	Multiple temperature sensors across battery cells	3	90	Backup temperature alarms
	Fails to provide adequate power	At best: graceful degradation (Preventing catastrophic failure by limiting functionality)	7	One of the above battery failure modes	4	BMS (health monitor)	3	84	Above measures
Communications	Loss of radio signal	Aircraft status becomes unknown by GCS	7	Aircraft flies out of range	1	Alert operator, search for radio beam/switch to different radio frequency	2	14	N/A
	Radio Frequency Jam	Aircraft status becomes unknown by GCS	6	Hostile radio jamming attack	2	Switch to different radio frequency	2	24	Autonomous Emergency Landing Procedure
Wiring	Cable Sheath Degradation	Wire failure	6	Ageing	5	Visual Inspection	3	90	Replace cable sheaths anticipating degradation
	Mechanical Failure	Wire failure	7	Cable damaged during installation or during maintenance	5	Testing	3	105	Replace cables anticipating degradation

Process Step	Potential Failure Mode	Potential Failure Effect	SEV	Potential Causes	OCC	Current Process Controls	DET	RPN	Action Recommended
General	Overheat	Component failure	8	Improper cooling of components	4	Multiple temperature sensors across avionics system	4	128	Backup temperature alarms
	Manufacturing Error	Component failure	7	Improper manufacture of components	6	Testing	6	252	More thorough COTS manufacturer vetting process
	Excessive Ground Resonance	Spontaneous aircraft disassembly	10	Poorly designed/assembled rotors or landing gear	1	Thorough balancing of rotors, landing gear horizontal movement test	3	30	Specific ground resonance testing

Conclusions

The design for the systems & mission management element of an eVTOL UAM aircraft was outlined. Existing technologies were researched and analysed, and their merits evaluated compared to other similar technologies. Safety elements such as redundancies and health monitoring were discussed when relevant. The main design considerations, SWaP characteristics and interoperability of each component were illustrated using functional block diagrams, flow charts, tables, and other figures. The design of novel mission management algorithms was also described and code the made available to the assessors for testing. Weight and power targets of 0.5% of the MTOW and 3 kW respectively were met.

Acknowledgements

Throughout the writing of this dissertation I received a great deal of support and assistance. I would firstly like to thank my technical assessors, Dr. Pejman Irvani and Dr. Martin Fullekrug, for their insightful feedback relating to the interim Phase 2B report. It helped me discern what I needed to focus on for the remainder of the project. Dr. Irvani also provided invaluable knowledge throughout the semester and helped spark genuine curiosity about a field I had little prior knowledge of.

I would also like to thank my Project Manager, Sebastien Defauw, for ensuring the team was always on the same page, and for beginning every Teams meeting with a smile and a plan. And finally my team, who made collaboration and teamwork during a quarantine a breeze, and without whom this report would not have been possible.

References

[1] J. Gundlach, Designing Unmanned Aircraft Systems - A Comprehensive Approach, American Institue of Aeronautics and Astronautics (AIAA), 2012.

[2] R. Austin, Unmanned Aircraft Systems: UAVS Design, Development and Deployment, Wiley Publishing, 2010.

[3] A. Filippone, "2.2 Weight," in *Aerospace Engineering e-Mega Reference*, Elsevier Science, 2009, pp. 31-32.

[4] M. R. Benjamin, P. Newman, H. Schmidt and J. J. Leonard, "A Tour of MOOS-IvP Autonomy Software Modules," MIT Computer Science and Artificial, 2009.

[5] C. Pippin, "Integrated Hardware/Software Architectures to Enable UAVs for Autonomous Flight," in *Handbook of Unmanned Aerial Vehicles*, Springer, 2015, pp. 1725-1746.

[6] PC/104, "What is PC/104? - A Proposed Extension to IEEE-P996," 2008. [Online]. Available: https://pc104.org/hardware-specifications/pc104/. [Accessed 15 March 2021].

[7] M. Southworth, "PC/104 and UASs: The beauty of the system," 2021. [Online]. Available: http://smallformfactors.mil-embedded.com/articles/pc104-uass-beauty-the-system/. [Accessed 15 March 2021].

[8] W. Zhong, Duality System in Applied Mechanics and Optimal Control, Kluwer Academic Publishers, 2004, p. 283.

[9] J. Meyer, F. du Plessis and W. Clarke, "Design Considerations for Long Endurance Unmanned Aerial Vehicles," *Aerial Vehicles,* pp. 443-496, 2009.

[10] L. Guvenc, B. A. Guvenc, B. Demirel and M. T. Emirley, Control of Mechatronic Systems, Institution of Engineering and Technology, 2017, pp. 47-90.

[11] B. L. Stevens, F. L. Lewis and E. N. Johnson, Aircraft Control and Simulation, Third ed., Wiley, 2016.

[12] M. Bryson and S. Sukkarieh, "UAV Localization Using Inertial Sensors and Satellite Positioning Systems," in *Handbook of Unmanned Aerial Vehicles*, Springer, 2015, pp. 434-460.

[13] G. Ducard and H. P. Geering, "Efficient Nonlinear Actuator Fault Detection and Isolation System for Unmanned Aerial Vehicles," *Journal of Guidance, Control, and Dynamics,* pp. 225-240, 2008.

[14] J. W. Dahlgren, "Real Options and Value Driven Design in Spiral Development," in *CCRTS - The State of the Art and the State of the Practice*, 2006.

[15] A. Atyabi, S. MahmoudZadeh and S. Nefti-Meziani, "Current Advancements on Autonomous Mission Planning and Management Systems: an AUV and UAV perspective," *Journal of Annual Reviews in Control,* vol. 46, pp. 196-215, 2018.

[16] P. Patron, E. Miguelanez and Y. Petillot, “Embedded Knowledge and Autonomous Planning: The Path Towards Permanent Presence of Underwater Networks,” *Autonomous Underwater Vehicles,* p. Chapter 9, 2011.

[17] A. M. Meystel and J. S. Albus, Intelligent systems: architecture, design, and control, Wiley Interscience, 2002, Wiley Interscience, 2001.

[18] C. V. Angelino, “Sensor aided H.264 Video Encoder for UAV applications,” in *Picture Coding Symposium (PCS)*, 2013.

[19] CollinsAerospace, “TASE 350 Imaging System,” Cloud Cap Tech (Collins Aerospace), [Online]. Available: https://www.cloudcaptech.com/products/detail/tase-350. [Accessed 28 Feburary 2021].

[20] F. Kunstmann, D. Klarer, A. Puchinger and S. Beer, “Weather Detection with an AESA-Based Airborne Sense and Avoid Radar,” in *2020 IEEE Radar Conference (RadarConf20)*, 2020.

[21] M. Kirscht and C. Rinke, “3D Reconstruction of Buildings and Vegetation from Synthetic Aperture Radar (SAR) Images,” Universität Hannover, 1998.

[22] IMSAR, “NSP-5 Radar System,” IMSAR, [Online]. Available: https://www.imsar.com/radar-systems/nsp-5/. [Accessed 15 March 2021].

[23] D. Gupta, “HopAir UAM Aircraft Phase 3 Aerodynamics Technical Report,” University of Bath, 2021.

[24] M. Neale and D. Colin, “Technology Workshop: ICAO RPAS MANUAL - C2 Link and Communications,” in *Remotely Piloted Aircraft Systems Symposium*, 2015.

[25] L. Mejias, J. Lai and T. Bruggemann, “Sensors for Missions,” in *Handbook of Unmanned Aerial Vehicles*, Springer, 2015, pp. 385-400.

[26] G. Richter, “Aircraft Wiring for Smart People,” Sullivan Acutronic Power Systems, 2008. [Online]. Available: http://www.sullivanuv.com/wp-content/uploads/2014/06/Aircraft-Wiring-Best-Practices.pdf. [Accessed 2 March 2021].

[27] UAVision, “PMU - 305 Power Management Unit,” 2021. [Online]. Available: https://www.uavision.com/pmu-305.

[28] W. Cinibulk, “Aircraft Electrical Wire - Wire Manufacturers Perspective,” Center for Advanced Aviation System Development, 2001. [Online]. Available: https://www.mitrecaasd.org/atsrac/FAA_PI-Engineer_Workshop/2001/aircraft_electrical_wire.pdf. [Accessed 27 April 2021].

[29] C. Dillow, “Can Technology Save the Military From a Data Deluge?,” *Popular Science,* 2011.

[30] SAE, “JAUS Service Interface Definition Language,” AS-4c Information Modeling and Definition Committee, 2010.

[31] P. Crowley and P. Barry, “Embedded Platform Architecture,” in *Modern Embedded Computing - Designing Connected, Pervasive, Media-Rich Systems*, Elsevier, 2012, pp. 72-74.

[32] D. Sullivan, J. Totah, S. Wegener, F. Enomoto, C. Frost, J. Kaneshige and J. Frank, “Intelligent Mission Management for Uninhabited Aerial Vehicles,” NASA Ames Research Center, 2004.

[33] I. McManus, R. Clothier and R. Walker, “Highly Autonomous UAV Mission Planning and Piloting for Civilian Airspace Operations,” in *Proceedings Eleventh Australian International Aerospace Congress, AIAC-11, First Australasian Unmanned Air Vehicles Conference*, Melbourne, Australia, 2005.

[34] T. L. Johnson, H. A. Sutherland, S. F. Bush, W. Yan and C. Eaker, “The TRAC mission manager autonomous control executive,” in *2001 IEEE Aerospace Conference Proceedings*, 2001.

[35] R. D. Falck, D. Ingraham and E. Aretskin-Hartion, “Multidisciplinary Optimization of Urban-Air-Mobility Class Aircraft Trajectories with Acoustic Constraints,” in *AIAA/IEEE Electric Aircraft Technologies Symposium (EATS)*, Cincinatti, OH, USA, 2018.

[36] A. Oztekin and R. Wever, “Development of a Regulatory Safety Baseline for UAS Sense and Avoid,” in *Handbook of Unmanned Aerial Vehicles*, Springer, 2015, pp. 1817-1840.

[37] FAA, “Literature Review on Detect, Sense and Avoid Technology for Unmanned Aircraft Systems,” National Technical Information Service (NTIS), Springfield, VA, 2009.

[38] T. Mildenberger, “See and Avoid,” in *UAV 2007 Paris Conference*, 2007.

[39] B. Vaaben and J. Larsen, “Mitigation of airspace congestion impact on airline networks,” *Journal of AIr Transport Management,* vol. 47, pp. 54-65, August 2015.

[40] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[41] R. Leigh, S. J. Louis and C. Miles, “Using a Genetic Algorithm to Explore A*-like Pathfinding Algorithms,” in *IEEE Symposium on Computational Intelligence and Games (CIG 2007)*, 2007.

[42] ICAO, Doc 4444 - Air Traffic Management, 16 ed., ICAO, 2016.

[43] M. Lemaire, “HopAir UAM Aircraft Phase 3 Sales Forecast Report,” University of Bath, 2021.

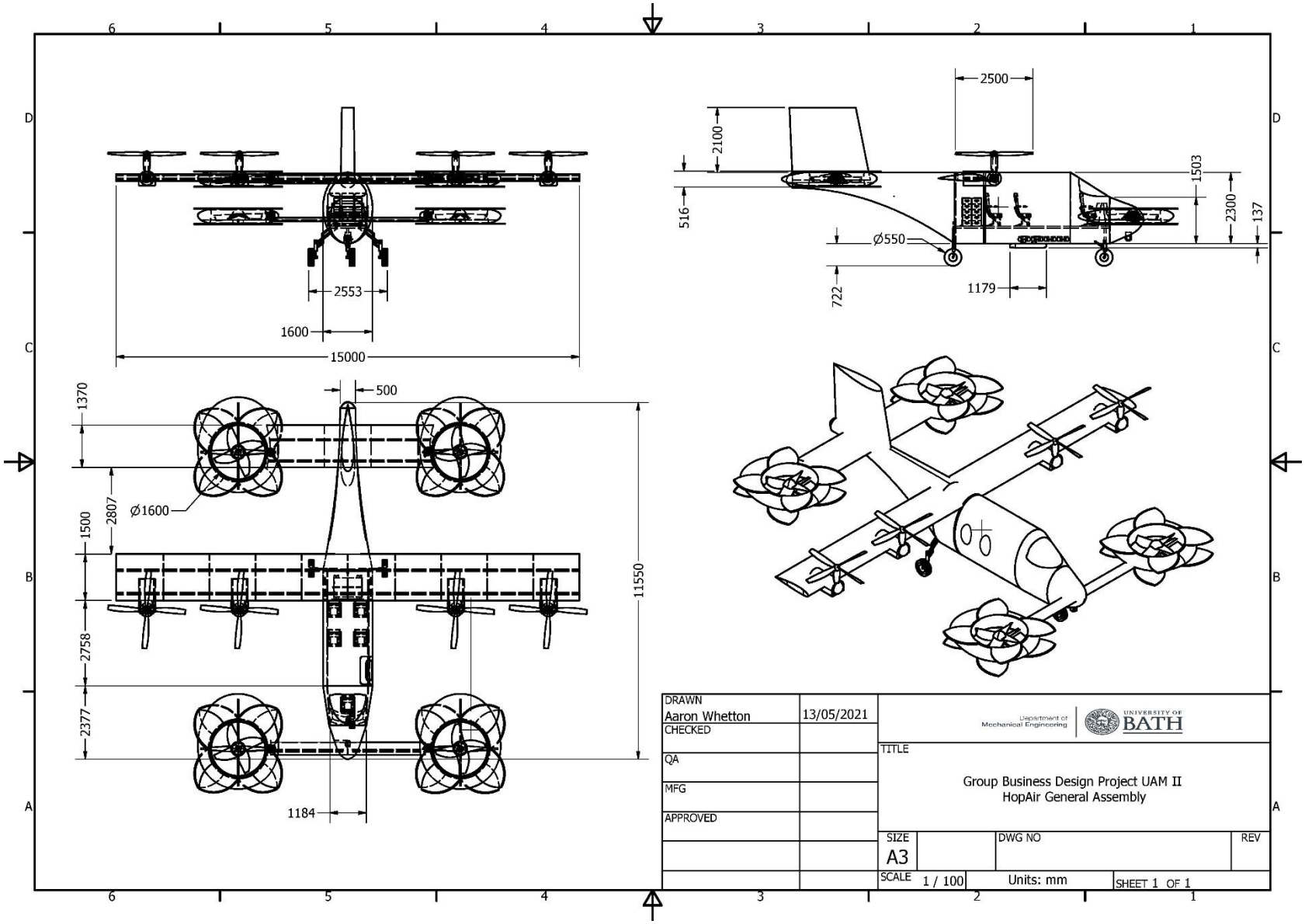
[44] E. Bourennane, P. Gouton, M. Paindavoine and F. Truchetet, “Generalization of Canny-Deriche filter for detection of noisy exponential edge,” in *Signal Processing*, 10th ed., vol. 82, Elsevier, 2002, pp. 1317-1328.

[45] CollinsAerospace, “Piccolo SL,” 2021. [Online]. Available: https://www.cloudcaptech.com/products/detail/piccolo-SL1. [Accessed 16 March 2021].

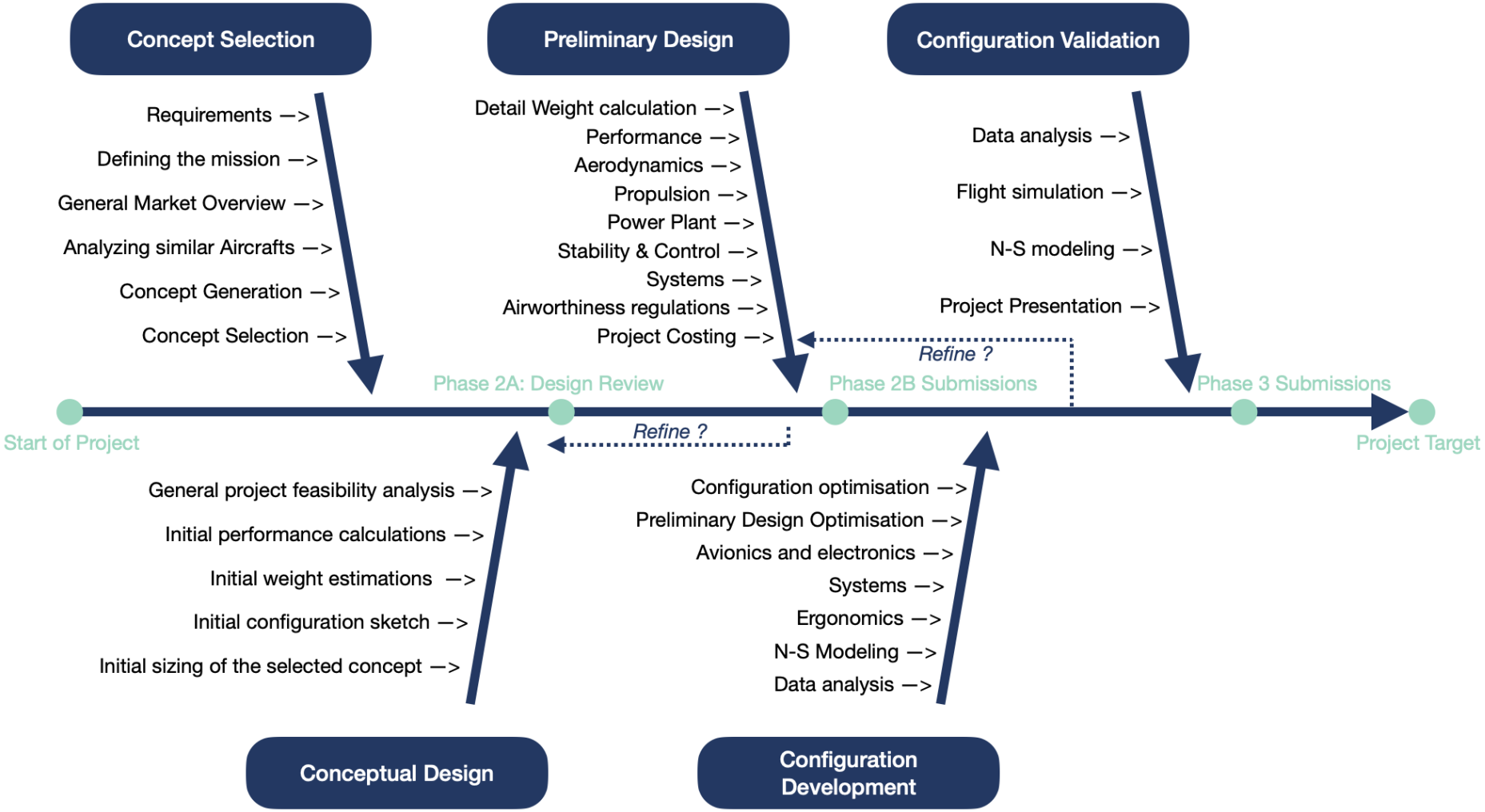
[46] D. S. Fowler, “CAN bus Cable Wiring,” Tek Eye, 2017. [Online]. Available: https://tekeye.uk/automotive/can-bus-cable-wiring. [Accessed 16 May 2021].

Appendices

Appendix A: General Assembly Drawing



Appendix B: Fish-Bone Diagram



Appendix C: Components List

Item Name	Qty	Potential Solutions	Price	Weight	Dims (mm)	Power	Voltage	Ports	Source
GPS	1	truFYX TSO-C145e Class Beta 1 SBAS GPS	$500?	20g	47x8	0.5W	12V	RS-232	truFYX - uAvionix
IMU	1	LPMS-IG1P IMU with GPS Connectivity	632.46	76g	51x45x24	0.4W	12V	USB, RS-232	LPMS-IG1 9-Axis High Precision IMU with Optional GPS Connectivity. (omniinstruments.co.uk)
EO Cameras	1	TASE 350	$1000?	3.2kg	178x267	28W	28V	Std DC Jack, Ethernet	Cloud Cap Technology TASE350 Imaging System | Cloud Cap Technology
Long-range radar	1	IMSAR NSP-5	$2000?	7.5kg	14x118x18	130W	28V	Std DC Jack, Ethernet	NSP-5 – IMSAR
ADS-B Transponder	2	tailBeaconX	$2500	140g	113x105x48	3W	12V	Std DC Jack, RS232	tailBeaconX - uAvionix
C2 Data link	2	uAvionix microLink	$500	16g	31x26x9	1.7W	5V	Connects directly to uAvionix GPS, sky	microLink BVLOS Data Link - uAvionix
Autopilot	1	Veronte OPV UAM Autopilot	$2000?	200g?	100x100x50?	5W	5V	Std DC Jack, RS232	UAM - Urban Air Mobility - Veronte Autopilot - Products | Embention | Embention
Processor	4	Liger VL-EPM-43 PC/104-Plus	$3440	226g	97x91	13W	5V	Various	Liger (EPM-43) Intel Kaby Lake Processor | VersaLogic
Power Management Unit	2	PMU - 305	$1000?	200g?	100x100x10?	100-500W	28VDC In	Std DC Jacks	PMU - 305 | UAVision Aeronautics
Ethernet Switch	2	Techaya MILTECH 308	$500?	140g	82x64x26	2W	5V	Std DC Jack, Ethernet	Ultra Compact Military-Grade Rugged Ethernet Switch, 8 Port (militaryethernet.com)
Altimeter	1	Honeywell Single Antenna Radar Altimeter	$500?	1.9kg	155x72x158	15W	28V	Std DC Jack, RS232	Single Antenna Radar Altimeter (SARA) (honeywell.com)
DC-DC Converter	12	Roband RO-MIL-2212 & 2214	50?	60g	76x38x10	N/A	N/A	Std DC Jack	COTS DC-DC Converters (roband.co.uk)
RS232 Cables	14	Serial/Parallel Male to Female RS232 Cable	$5	50g/m	Various	N/A	N/A	RS232	Serial Cable Assemblies I RS Components (rs-online.com)
Power Cables	16	DC Plug	$2	50g/m	Various	N/A	Up to 28V	Std DC Jack	Dc Power Connectors | Barrel Connectors | RS Components (rs-online.com)
			$30,094	**17.508**		**230W**			

Appendix D: A* Algorithm App Script

```
% This is the code that runs when the Calculate button is pressed
%

app.FindPathButton.Enable = 'off';
app.FindPathButton.Text = 'Calculating...';
app.ResetButton.Enable = 'off';

% % INITIALISE % %

%DEFINE THE 3-D MAP ARRAY
MAX_X=50;
MAX_Y=50;
MAX_Z=50;
MAP=2*(ones(MAX_X,MAX_Y,MAX_Z));

% % BEGIN Manual Start, Obstacle, Target Input
% Obstacle=-1,Target = 0,Robot=1,Space=2
start = [app.StartX.Value, app.StartY.Value, app.StartZ.Value];
target = [app.TargetX.Value, app.TargetY.Value, app.TargetZ.Value];
moving = app.MovingObstacles; % x y r z
fixed = app.FixedObstacles; % x y r z
numCyl = (length(fixed)+length(moving))/4;
Obst_i = [];
MAP(start(1),start(2),start(3))=1;
MAP(target(1),target(2),target(3))=0;
xval = start(1);
yval = start(2);
zval = start(3);
Obst_i = [];
cylCoord = [];

dirInput = app.DirectionDropDown.Value;
[obsDirec,uamDirecX,uamDirecY,uamDirecZ] = direction(dirInput,start,target);

app.Paused = 0;

% % CYLCOORD
a=1;
k2=1;
tic
for k=1:numCyl % Obtain obstacles within cylinder
    if k <= length(moving)/4 % Populate top of list with moving cylinder coords
        cylCoord(k,1) = moving(k*4-3);
        cylCoord(k,2) = moving(k*4-2);
        cylCoord(k,3) = moving(k*4-1);
        cylCoord(k,4) = moving(k*4);
    else
        cylCoord(k,1) = fixed(k2*4-3); % Populate bottom of list with fixed
cylinder coords
        cylCoord(k,2) = fixed(k2*4-2);
        cylCoord(k,3) = fixed(k2*4-1);
        cylCoord(k,4) = fixed(k2*4);
        k2=k2+1;
    end

    [x(:,:,k), y(:,:,k), z(:,:,k)] = cylinder(cylCoord(k,3),50); %Define Cylinder
    x(:,:,k) = x(:,:,k)+cylCoord(k,1);
    y(:,:,k) = y(:,:,k)+cylCoord(k,2);
    if k <= length(moving)/4 % if aircraft then thin cylinder at altitude
        z(1,:,k) = cylCoord(k,4)+0.5;
        z(2,:,k) = cylCoord(k,4)-0.5;
    else
        z(2,:,k) = cylCoord(k,4); % if building then cylinder up to altitude
    end
```

```
    for i=1:MAX_X
        for j=1:MAX_Y
            % Grab obstacles near radius of cylinder, add to closed list and plot
            if distance(cylCoord(k,1),cylCoord(k,2),i,j) < cylCoord(k,3)+.3
                if k <= length(moving)/4
                    MAP(i,j,cylCoord(k,4))=-1;
                    Obst(a,1)=i;
                    Obst(a,2)=j;
                    Obst(a,3)=cylCoord(k,4);
                    a=a+1;
                else
                    if isempty(Obst_i)
                        Obst_i = a-1;
                    end
                    MAP(i,j,1:cylCoord(k,4))=-1;
                    a2=a+cylCoord(k,4)-1;
                    Obst(a:a2,1)=i;
                    Obst(a:a2,2)=j;
                    Obst(a:a2,3)=1:cylCoord(k,4);
                    a=a2+1;
                end
            end
        end
    end
end
toc
if isempty(Obst_i) && isempty(cylCoord) == 0
    Obst_i = length(Obst);
end
if MAP(target(1),target(2),target(3)) == -1
    errordlg('Target is within obstacle');
end
% FIND INITIAL PATH %

run A_Star1_3D.m
opsize = size(Optimal_path,1);
for a=1:opsize % Add path to OPM
    Opt_path_master(a,:)=Optimal_path(opsize+1-a,:);
end
recalc_counter = 0;

% PLOT START CONDITIONS %

plot3(app.Graph,0,0,0)
axis(app.Graph,[1 MAX_X+1 1 MAX_Y+1 1 MAX_Z+1])
grid(app.Graph,'on')
hold(app.Graph,'on')

plot3(app.Graph,start(1),start(2),start(3),'md','MarkerSize',10); %% Plot Start
UAM=plot3(app.Graph,start(1),start(2),start(3),'bo','MarkerSize',8); %% Plot UAM
plot3(app.Graph,target(1),target(2),target(3),'gd','MarkerSize',10); %% Plot Target
path =
plot3(app.Graph,Opt_path_master(:,1),Opt_path_master(:,2),Opt_path_master(:,3),'k-
','LineWidth',1); % Initialise Path
if isempty(moving) == 0 || isempty(fixed) == 0
    for k=1:numCyl

cylFig(k)=surf(app.Graph,x(:,:,k),y(:,:,k),z(:,:,k),'FaceColor','r','EdgeColor','no
ne','FaceAlpha',1);
    end
    if isempty(moving) == 0
        centObst =
plot3(app.Graph,cylCoord(1:length(moving)/4,1),cylCoord(1:length(moving)/4,2),cylCo
ord(1:length(moving)/4,4),'r^'); %% Plot airplanes
    end
end
% END INITIALISATION %
```

```
% % BEGIN LOOP % %
t=1;
app.TimestepField.Value = t;

while ( (get(UAM,'XData') ~= target(1) || get(UAM,'YData') ~= target(2))...
        || get(UAM,'ZData') ~= target(3) ) % If target not reached
    for a=1:Obst_i
        if ( (1 <= Obst(a,1)+obsDirec(1)) && (Obst(a,1)+obsDirec(1) <= MAX_X) &&...
                (1 <= Obst(a,2)+obsDirec(2)) && (Obst(a,2)+obsDirec(2) <= MAX_Y) )
            MAP(Obst(a,1),Obst(a,2),Obst(a,3)) = 2;
            Obst(a,1)=Obst(a,1)+obsDirec(1);
            Obst(a,2)=Obst(a,2)+obsDirec(2);
        end
    end
    for a=1:Obst_i
        MAP(Obst(a,1),Obst(a,2),Obst(a,3)) = -1;
    end
    if MAP(target(1),target(2),target(3)) == -1
        errordlg('Target is within obstacle');
    end
    for a=t:length(Opt_path_master)
        for b=1:Obst_i
            if ( (Opt_path_master(a,1) == Obst(b,1) && Opt_path_master(a,2) ==
Obst(b,2))...
                    && Opt_path_master(a,3) == Obst(b,3) )
                start(1) = Opt_path_master(t,1);
                start(2) = Opt_path_master(t,2);
                start(3) = Opt_path_master(t,3);
                run A_Star1_3D.m
                opsize = size(Optimal_path,1);
                t=t+1;
                Opt_path_master(t:t+opsize-1,1)=flip(Optimal_path(:,1));
                Opt_path_master(t:t+opsize-1,2)=flip(Optimal_path(:,2));
                Opt_path_master(t:t+opsize-1,3)=flip(Optimal_path(:,3));
                recalc_counter=recalc_counter+1;
                app.RecalculationsField.Value = recalc_counter;
            end
        end
    end

    % % Plot results

    pause(app.PauseField.Value)
    if isempty(moving) == 0 % If moving cylinders exist
        x(:,:,1:length(moving)/4) = x(:,:,1:length(moving)/4)+obsDirec(1);%% Move
obstacles
        y(:,:,1:length(moving)/4) = y(:,:,1:length(moving)/4)+obsDirec(2);
        for k = 1:numCyl
            set(cylFig(k),'XData',x(:,:,k),'YData',y(:,:,k));
        end
        cylCoord(:,1) = cylCoord(:,1)+obsDirec(1);
        cylCoord(:,2) = cylCoord(:,2)+obsDirec(2);

set(centObst,'XData',cylCoord(1:length(moving)/4,1),'YData',cylCoord(1:length(movin
g)/4,2))
    end

set(path,'XData',Opt_path_master(:,1),'YData',Opt_path_master(:,2),'ZData',Opt_path
_master(:,3)) %% Draw path

set(UAM,'XData',Opt_path_master(t,1),'YData',Opt_path_master(t,2),'ZData',Opt_path_
master(t,3)) %% Move aircraft
    drawnow

    t = t+1;
    app.TimestepField.Value = t;
end
```

```
app.FindPathButton.Value = 0;
app.FindPathButton.Enable = 'on';
app.FindPathButton.Text = 'Find Path';
app.ResetButton.Enable = 'on';
```

Appendix E: Static A* Algorithm Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A* ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
%LISTS USED FOR ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
%OPEN LIST STRUCTURE
%--------------------------------------------------------------------------
%IS ON LIST 1/0 |X val |Y val |Parent X val |Parent Y val |h(n) |g(n)|f(n)|
%--------------------------------------------------------------------------
OPEN=[];
%CLOSED LIST STRUCTURE
%--------------
%X val | Y val |
%--------------
% CLOSED=zeros(MAX_VAL,2);
CLOSED=[];

%Put all obstacles on the Closed list
n=1;%Dummy counter
for i=1:MAX_X
    for j=1:MAX_Y
        for k=1:MAX_Z
            if (MAP(i,j,k) == -1)
            CLOSED(n,1)=i;
            CLOSED(n,2)=j;
            CLOSED(n,3)=k;
            n=n+1;
            end
        end
    end
end
CLOSED_COUNT=size(CLOSED,1);
numPts=CLOSED_COUNT;
%set the starting node as the first node
xNode=start(1);
yNode=start(2);
zNode=start(3);
OPEN_COUNT=1;
path_cost=0;
goal_distance=distance_3D(xNode,yNode,zNode,target);
OPEN(OPEN_COUNT,:)=insert_open_3D(xNode,yNode,zNode,xNode,yNode,zNode,path_cost,goa
l_distance,goal_distance);
OPEN(OPEN_COUNT,1)=0;
CLOSED_COUNT=CLOSED_COUNT+1;
CLOSED(CLOSED_COUNT,1)=xNode;
CLOSED(CLOSED_COUNT,2)=yNode;
CLOSED(CLOSED_COUNT,3)=zNode;
NoPath=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% START ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
while ( ((xNode ~= target(1) || yNode ~= target(2)) || zNode ~= target(3)) &&
NoPath == 1 )
```

```
exp_array=expand_array_3D(xNode,yNode,zNode,path_cost,uamDirecX,uamDirecY,uamDirecZ
,target,CLOSED,MAX_X,MAX_Y,MAX_Z);
 exp_count=size(exp_array,1);
 %UPDATE LIST OPEN WITH THE SUCCESSOR NODES
 %OPEN LIST FORMAT
 %-----------------------------------------------------------------------------
 %IS ON LIST 1/0 |X val |Y val |Z val |Parent X val |Parent Y val |Parent Z val
|h(n) |g(n)|f(n)|
 %-----------------------------------------------------------------------------
 %EXPANDED ARRAY FORMAT
 %--------------------------------
 %|X val |Y val |Z val |h(n) |g(n)|f(n)|
 %--------------------------------
 for i=1:exp_count
    flag=0;
    for j=1:OPEN_COUNT
        if ( (exp_array(i,1) == OPEN(j,2) && exp_array(i,2) == OPEN(j,3)) &&
exp_array(i,3) == OPEN(j,4) )
            OPEN(j,10)=min(OPEN(j,10),exp_array(i,6)); %#ok<*SAGROW>
            if OPEN(j,10)== exp_array(i,6)
                %UPDATE PARENTS,gn,hn
                OPEN(j,5)=xNode;
                OPEN(j,6)=yNode;
                OPEN(j,7)=zNode;
                OPEN(j,8)=exp_array(i,4);
                OPEN(j,9)=exp_array(i,5);
                OPEN(j,10)=exp_array(i,6);
            end%End of minimum fn check
            flag=1;
        end%End of node check
%         if flag == 1
%             break;
    end%End of j for
    if flag == 0
        OPEN_COUNT = OPEN_COUNT+1;

OPEN(OPEN_COUNT,:)=insert_open_3D(exp_array(i,1),exp_array(i,2),exp_array(i,3),xNod
e,yNode,zNode,exp_array(i,4),exp_array(i,5),exp_array(i,6));
    end%End of insert new element into the OPEN list
 end%End of i for
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %END OF WHILE LOOP
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %Find out the node with the smallest fn
  index_min_node = min_fn_3D(OPEN,OPEN_COUNT,target);
  if (index_min_node ~= -1)
   %Set xNode, yNode and zNode to the node with minimum fn
   xNode=OPEN(index_min_node,2);
   yNode=OPEN(index_min_node,3);
   zNode=OPEN(index_min_node,4);
   path_cost=OPEN(index_min_node,8);%Update the cost of reaching the parent node
  %Move the Node to list CLOSED
  CLOSED_COUNT=CLOSED_COUNT+1;
  CLOSED(CLOSED_COUNT,1)=xNode;%Add lowest fn node to closed list and mark as used
on open
  CLOSED(CLOSED_COUNT,2)=yNode;
  CLOSED(CLOSED_COUNT,3)=zNode;
  OPEN(index_min_node,1)=0;
  else
      %No path exists to the Target!!
      NoPath=0;%Exits the loop!
  end%End of index_min_node check
end%End of While Loop
%Once algorithm has run The optimal path is generated by starting of at the
%last node(if it is the target node) and then identifying its parent node
%until it reaches the start node.This is the optimal path
```

```
i=size(CLOSED,1);
Optimal_path=[];
xval=CLOSED(i,1);
yval=CLOSED(i,2);
zval=CLOSED(i,3);
i=1;
Optimal_path(i,1)=xval;
Optimal_path(i,2)=yval;
Optimal_path(i,3)=zval;
i=i+1;

if ( ((xval == target(1)) && (yval == target(2))) && zval == target(3) )
    inode=0;
   %Traverse OPEN and determine the parent nodes
   parent_x=OPEN(node_index_3D(OPEN,xval,yval,zval),5);%node_index returns the
index of the node
   parent_y=OPEN(node_index_3D(OPEN,xval,yval,zval),6);
   parent_z=OPEN(node_index_3D(OPEN,xval,yval,zval),7);

   while ( (parent_x ~= start(1) || parent_y ~= start(2)) || parent_z ~= start(3) )
           Optimal_path(i,1) = parent_x;
           Optimal_path(i,2) = parent_y;
           Optimal_path(i,3) = parent_z;
           %Get the grandparents:-)
           inode=node_index_3D(OPEN,parent_x,parent_y,parent_z);
           parent_x=OPEN(inode,5);%node_index returns the index of the node
           parent_y=OPEN(inode,6);
           parent_z=OPEN(inode,7);
           i=i+1;
   end
end
```

Appendix F: A* Expand Array Sub-Function

```
function
exp_array=expand_array_3D(node_x,node_y,node_z,hn,uamDirecX,uamDirecY,uamDirecZ,tar
get,CLOSED,MAX_X,MAX_Y,MAX_Z)
    %Function to return an expanded array
    %This function takes a node and returns the expanded list
    %of successors,with the calculated fn values.
    %The criteria being none of the successors are on the CLOSED list.
    %

    exp_array=[];
    exp_count=1;
    c2=size(CLOSED,1);%Number of elements in CLOSED including the zeros
    for i= uamDirecX
        for j= uamDirecY
            for k= uamDirecZ
                if ( (i~=j || j~=k) || i~=0 )  %The node itself is not its
successor. This breaks when i = j = k = 0.
                s_x = node_x+i;
                s_y = node_y+j;
                s_z = node_z+k;
                    if ( ((s_x >0 && s_x <=MAX_X) && (s_y >0 && s_y <=MAX_Y)) &&
(s_z >0 && s_z <=MAX_Z) )  %node within array bound
                        flag=1;
                        for c1=1:c2
                            if ( (s_x == CLOSED(c1,1) && s_y == CLOSED(c1,2)) &&
s_z == CLOSED(c1,3) )
                                flag=0;
                            end
                        end%End of for loop to check if a successor is on closed
list.
                        if ( flag == 1 )
                            exp_array(exp_count,1) = s_x;
                            exp_array(exp_count,2) = s_y;
                            exp_array(exp_count,3) = s_z;
                            exp_array(exp_count,4) =
hn+distance_3D(node_x,node_y,node_z,[s_x,s_y,s_z]);%cost of travelling to node
                            exp_array(exp_count,5) =
distance_3D(target(1),target(2),target(3),[s_x,s_y,s_z]);%distance between node and
goal
                            exp_array(exp_count,6) =
exp_array(exp_count,4)+exp_array(exp_count,5);%fn
                            exp_count=exp_count+1;
                        end%Populate the exp_array list
                    end% End of node within array bound
                end%End of if node is not its own successor loop
            end%End of k for loop
        end%End of j for loop
   end%End of i for loop
```

Appendix G: A* Insert Open Node Sub-Function

```
function new_row =
insert_open_3D(xval,yval,zval,parent_xval,parent_yval,parent_zval,hn,gn,fn)
%Function to Populate the OPEN LIST
%OPEN LIST FORMAT
%--------------------------------------------------------------------------
%IS ON LIST 1/0 |X val |Y val |Parent X val |Parent Y val |h(n) |g(n)|f(n)|
%--------------------------------------------------------------------------
%

new_row=ones(1,10);
new_row(1,2)=xval;
new_row(1,3)=yval;
new_row(1,4)=zval;
new_row(1,5)=parent_xval;
new_row(1,6)=parent_yval;
new_row(1,7)=parent_zval;
new_row(1,8)=hn;
new_row(1,9)=gn;
new_row(1,10)=fn;

end
```

Appendix H: A* Obtain Minimum f(n) Sub-Function

```
function i_min = min_fn_3D(OPEN,OPEN_COUNT,target)
%Function to return the Node with minimum fn
% This function takes the list OPEN as its input and returns the index of the
% node that has the least cost
%

 temp_array=[];
 k=1;
 flag=0;
 goal_index=0;
 for j=1:OPEN_COUNT
     if (OPEN(j,1)==1)
         temp_array(k,:)=[OPEN(j,:) j]; %#ok<*AGROW>
         if ( (OPEN(j,2)==target(1) && OPEN(j,3)==target(2)) &&
OPEN(j,3)==target(3) )
             flag=1;
             goal_index=j;%Store the index of the goal node
         end
         k=k+1;
     end
 end%Get all nodes that are on the list open
 if flag == 1 % one of the successors is the goal node so send this node
     i_min=goal_index;
 end
 %Send the index of the smallest node
 if size(temp_array ~= 0)
  [~,temp_min]=min(temp_array(:,10));%Index of the smallest node in temp array
  i_min=temp_array(temp_min,11);%Index of the smallest node in the OPEN array
 else
     i_min=-1;%The temp_array is empty i.e No more paths are available.
 end
```

Appendix I: Emergency Landing Zone Algorithm

```
function [result] = EFL(img,method)
% % This function plots landing zone over 'img' using [edge detection] 'method'
% % Note: Image Processing Toolbox is required to run this code
%

img = imread(img);
bwraw = rgb2gray(img);
bwimg = edge(bwraw,method); % Possible methods are: 'canny', 'sobel', etc...
r = length(bwimg)*.1;
flag = 0; % Solution found flag
decreased = 0; % If radius has been decreased


while flag == 0 || decreased == 0 % Find landing zone with biggest radius
    [circle,r,flag,decreased] = findSpot(bwimg,r,flag,decreased);
end

figure(1)
image(img)
hold on
centerX = (max(circle(:,1))+min(circle(:,1)))/2; % Find centers
centerY = (max(circle(:,2))+min(circle(:,2)))/2;
d=1;
for c=1:length(circle) % Grab landing zone outline
    if distance(circle(c,1),circle(c,2),centerX,centerY) >= .9*r
        outline(d,1:2) = [circle(c,1),circle(c,2)];
        d = d+1;
    end
end
result = plot(outline(:,1),outline(:,2),'go','MarkerSize',5);
```

Appendix J: Emergency Landing Zone Sub-Function

```
function [circle,r,flag,decreased] = findSpot(img,r,flag,decreased)
% % This function finds the optimal landing spot for an image and edge
% % detection method
%

disp(['Trying for R = ', num2str(r)])
circle = [];
a=1;
centerX = r;
centerY = r;
for i = 1:2*r+1 % Create circle of specified radius
    for j = 1:2*r+1
        if distance(i,j,centerX,centerY) <= r
            circle(a,1:2) = [i,j];
            a = a+1;
        end
    end
end

circleReset = circle; % Store top left circle location to shift row down once
a=a-1;                    % circle has moved all the way to the right

for i = 1:(size(img,1)-2*r) % Check each row until circle is tangent with edge
    for j = 1:(size(img,2)-2*r) % Check each column
        next = 0;
        for b = 1:a
            if img(circle(b,2),circle(b,1)) == 1 % If an obstacle is in potential
landing zone
                next = 1;
                break
            end
            if next == 1 % If obstacle found, move circle along
                break
            end
            if b == a % If check completed and no obstacle found in circle
                disp(['Solution found for R = ', num2str(r)])
                flag = 1;
                r = r+1;
                return
            end
        end
        circle(:,1) = circle(:,1) + 1;
    end
    circle = circleReset; % Reset circle to first location
    circle(:,2) = circle(:,2) + i; % Move down to next row
end
disp(r)
r = r - 1;
flag = 0;
decreased = 1;
end
```