8-18-2022 11:00 AM

Fully Autonomous UAV Exploration in Confined and Connectionless Environments

Kirk P. Vander Ploeg, *The University of Western Ontario*

Supervisor: Haque, Anwar, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

Unmanned aerial vehicles (UAVs) can provide automated approaches to remote data collection, inspection, and exploration. When doing so, these tasks are accompanied by commercial gains and safety benefits. While UAVs can automate these tasks in some environments today, their size, resource requirements, and communication dependencies prevent them from exploring many other types of environments. These barriers are especially prevalent in areas that deny wireless communication and are too small for larger-bodied UAVs. In this work we present a novel exploration planner capable of overcoming these barriers and operating in otherwise inaccessible environments. To achieve this, we present a new approach to frontier detection and mapping which enables exploration and scales to nano sized UAVs. We prove the viability of this solution through real-world experimentation at WING research labs. This exploration software accommodates the extreme resource constraints of the small UAVs required to fly in confined spaces. The presented strategy is truly autonomous with no dependency on communication with external systems and no prior knowledge of the exploration space. To the best of our knowledge, the presented prototype can explore the smallest spaces that have yet to be reached by connectionless and autonomous UAVS. This claim is supported by the demonstration of real-world testing as our prototype achieved full exploration of several challenging environments.

Keywords

UAV, autonomous drone, UAV exploration, underground inspection, inspection in confined spaces, GPS denied exploration

Summary for Lay Audience

Autonomous Unmanned Aerial Vehicles (UAVs) are a rapidly developing technology that is influencing many fields today. An exciting and commercially viable application of this technology is autonomous exploration and inspection of spaces that are inaccessible to humans. These spaces come in a variety of forms presenting varying constraints and affordances to the flight of drones. In some of the most constrained environments, we observe the denial of wireless communication between a drone and external systems. If access to the environment is not available before a mission, the UAV is required to make complicated navigational decisions onboard during flight. Such an environment can also feature narrow passages placing restrictions on the physical size of the drone. This in turn limits the computing power, battery size, and payload capacity available. The lighter weight also makes the drone less stable in windy conditions or when propellers cause gusts of air to reflect off nearby obstacles and back towards the drone. The ability to inspect small spaces despite these factors will provide new business opportunities while promoting health and safety. In eliminating the need for expensive human inspection, companies can automate inspection tasks. In doing so, they also eliminate human exposure to potentially hazardous working conditions.

In this work, we introduce strengths and weaknesses of various types of UAVs. We then examine components that enable autonomous navigation in previously unknown environments, from low level flight dynamics to path planning for exploration. By reviewing existing solutions for autonomous exploration, we distinguish the barriers preventing existing approaches from moving to smaller connectionless spaces. With our motivation to overcome these barriers, we present a novel exploration strategy that enables small-bodied UAVs to explore previously unknown, confined, and connectionless environments. To operate in this environment, we present a novel frontier-based exploration method capable of navigating in spaces less than 50cms wide. We demonstrate our design onboard a commercially available nano-sized UAV. We evaluate this prototype on multiple real world test beds imitating different types of environments. To the best of our knowledge, this prototype explores more confined environments than any other fully autonomous solution existing today.

Acknowledgments

I would like to extend my gratitude to my supervisor Dr. Anwar Haque, who guided me through the last two years of research. His support gave me confidence in our work and his willingness to accommodate new ideas and research goals enabled the work presented in this thesis. Beyond our academic work, Dr. Haque has given me valuable insights from his industry experiences and partnerships which have helped to orient myself as I move into the next stages of my career. Working with him was a pleasure, and I hope to continue working together in the future.

Next, I would like to thank my partner and best friend Danielle Havord-Wier for her support, humour, patience, and dedication during all chapters of my academic journey. From my application to an undergraduate program to the final submission of this thesis, she has always been the first one I go to for help and advice. I am forever grateful for her encouragement, perspective, and for giving me the confidence to go for it.

Finally, I would like to acknowledge one of the earliest influences during my undergraduate degree, Kenton Dang. Kenton was and will always be a light that reminds me to seize the moment, maintain perspective, and stay true to who I am. His friendship and support carried me through far more than Calculus 1000. I will continue to cherish our memories that bring a smile to my face during life's challenging moments.

Table of Contents

List of Tables

List of Figures

Glossary of Terms

WING - Western Information and Networking Group
uUAV - Small Unmanned Ariel Vehicles
MAV - Micro Ariel vehicles
NAV – Nano Ariel Vehicles
PAC – Pico Ariel Vehicles
SD - Smart Dust
VTOL -Vertical Takeoff and Landing
HTOL – Horizontal Takeoff and Landing
GNSS – Global Navigation Satellite Systems
GPS – Global Positioning Systems
TOA – Time of Arrival
UWB – Ultrawide Band
VLC – Visible Light Communication
LED – Light Emitting Diode
PD – Photodiodes
IMU – Inertial measurement unit
LK – Lucas Kanade
LiDAR Light Detection and Randing
FOV – Field of View
RTT – Rapidly Exploring Random Tree
NBV – Next Best View
ESDF – Euclidian Signed Difference Function
AEP – Autonomous Exploration Planner
VV – Visibility Volume
BFS – Breadth First Search
MCU – Micro-Controller Unit
FPU – Floating Point Unit
DSP – Digital Signal Processing
DMIPS – Dhrystone Million Instructions per Second
SRAM – Static Random Access Memory
RTOS – Real Time Operating System
PID – Proportional integral derivative
OAR – Obstacle Avoidance Radius

Chapter 1

1 Introduction and Motivation

Unmanned Aerial Vehicles (UAVs) will continue to change how we conduct business across many sectors and countless applications in the coming years. These systems, commonly referred to as drones, are becoming more accessible as advancements in their technology have uncovered more opportunities while driving their cost down. With this, it is expected that over 800 000 commercial drones will be registered by 2024 in the United States alone [1]. Many of these drones will be used for data collection, remote inspection, and surveillance. These applications have already become widespread across industries like agriculture, civil development and maintenance, defense, mining, entertainment, and many more. In the next decade, many new applications will arise as we find ways to bring UAVS to previously inaccessible spaces like cluttered smart cities or confined underground areas. In these areas, automated UAV flight is not currently possible as they deny the environmental affordances that drones depend on today. When we find solutions to these limitations, drones will be capable of reaching more remote areas where human traversal is not feasible. Aside from the commercial opportunities from the automation that such solutions will provide, they will also eliminate the health and safety concerns in data collection tasks that expose humans to hazardous environments.

1.1 Applications of Drones

To help understand the significance of drones today, we present several prevalent applications that leverage this technology. Despite first being a strictly military tool, UAVs have now been adopted by many civil sectors. In agriculture, we see their use in crop management by automating the monitoring of things like disease or invasive species. Crops can be maintained with automated chemical sprays and by watering with intelligent UAV systems. Soil samples can be quickly gathered to better understand their composition. Sensors embedded within the ground can collect data which is then transmitted to a drone during a flyover [2]. In the transportation industry, we see leaders like Amazon and DHL create new delivery strategies for small packages [3] and postal services trialing new

delivery techniques with UAVs [4]. Ambulance drones have also been used for medicine and blood sample transportation [2]. Military use is still prevalent today as drones continue to carry out reconnaissance missions with a focus on target acquisition and surveillance [5]. They are also heavily used in both ground combat and counter-aircraft tactics [6]. Drones used to assist in infrastructure development have become widespread and are expected to dominate the market value of UAV services at $45 billion (USD) [2]. In other applications, drones are used for the inspection of construction zones and can help to identify hazardous materials or safety violations. They can be used to inspect hard-to-reach areas like under bridges or near traffic [7]. Not only can they collect data for later inspection, but they can also employ vision-based methods to inspect these structures on the fly [8]. In search and rescue operations, drones become a tool to rapidly cover search areas and, in the presence of a disaster, can provide safe transportation of survival gear and medicine [5]. By deploying connected drones overhead, UAVs provide a mobile signal node to help ground crews [2]. In the mining industry, ariel data collection is used for safety assessments, planning, and inventory management. Deep underground mining shafts can be explored and mapped remotely while technicians wait safely on the surface [9].

Though this list is not exhaustive, it provides a snapshot of the many uses of drones for remote data collection that are already being done today. While keeping in mind that these applications are limited to environments and conditions that the current UAV systems can support, enabling UAVs to operate in even more environments, under more constraining conditions, will bring new opportunities and impact many industries.

1.2 Limitations of Drones

UAVs have limitations that must be considered in any mission or application. Aside from legislation that may limit where a drone can operate or how it must be piloted, hardware-related constraints should be considered in the design of any application.

The ability to have powerful hardware onboard a drone scales with its physical size. The larger the drone, the larger and heavier processors can be. In many cases, a small form factor is required where processors and memory must be physically small enough to fit onboard. This is a limiting condition as drones use memory and processing resources to

make decisions about where to travel next and to build representations of their surroundings. Furthermore, putting energy-demanding processors onboard a small UAV will greatly reduce flight time. If communication channels are available, it is often more suitable to offload processing to an external system. There are many cases where a UAV must communicate with either a pilot, base station, another drone, or a sensor. In all these cases, the environment must have the capacity to allow for a high bandwidth communication channel to pass through unaltered.

A UAV is constantly drawing energy from a battery to remain in flight and carry out all operations. Complicated missions require time to complete, and some surveillance operations require continuous flight. Drone flight time is still limited to less than 30 minutes in many types of drones. Increasing battery capacity can extend flight time but has a diminishing return as the battery gets heavier [10]. When operating in cold environments, batteries are less efficient and flight time is reduced even further [11]. Energy consumption increases with the number of orientation and position changes as the motors need to change the velocity of propellers. In some cases, wireless charging or quickly swapping batteries can be a solution but when this is not possible, flight planning, scheduling, and optimization can help.

These restrictions have an immediate impact on the drone's performance and capabilities. Research has addressed many ways to deal with these limitations, but there are still environments that are not conducive to flight.

1.3 Thesis Contribution

Many UAV based applications have use in environments that are presently unreachable with today's UAV technology. Specifically, the advantages of autonomous inspection, data collection, and surveillance can be extended to areas that have not been reachable before. In this work, we study the extension of autonomous flight into extremely restrictive and constraining environments. We will look at the critical components of autonomous flight to gain an understanding of where and how it is limited. With this background, we will then explore the current solutions and where their capabilities end. We use this background knowledge to inform our design of a new, alternative approach to autonomous exploration.

We then take the next step in drone autonomy by presenting a solution for autonomous flight in extremely constrained spaces and proving its viability through real world testing on a small drone. The contributions are listed as follows:

- From a grounded robot to an autonomous ariel agent, the algorithms and components enabling autonomous flight are reviewed. As these are presented, we address how flight environments can restrict the resources available to these components.
- With the review complete, we study the prevailing research and commercial solutions that bring us to the edge of what is possible using UAVs for autonomous exploration in connectionless, confined spaces. Here, we identify the key barriers that are preventing these solutions from being capable of moving to more constrained spaces.
- We present a novel solution to explore extremely confined spaces with a UAV that has no dependencies on communication, can be run onboard a resource-constrained UAV, and can explore previously unknown environments. To the best of our knowledge, when compared to existing our prototype can explore more confined and environmentally constrained spaces. This makes it the first viable option for many new types of exploration environments previously inaccessible to UAVs.
- To prove the functionality of our new exploration planner, we implement it on a real UAV and test it on several real-world testbeds with varying types of environments and obstacles.

1.4 Thesis Blueprint

The chapters are organized as follows: after a brief review and classification of available UAV systems, Chapter 2 will serve as an end-to-end review of autonomous exploration with UAVs. We start from low levels of quadcopter dynamics and end up at the high-level algorithms running on board which enable autonomous navigation decisions. We conclude with exploration strategies seen in literature and practice today. In Chapter 3 we review the work and solutions related to autonomous flight specifically in conditions where traditional localization, mapping, and navigation techniques are not feasible. Using this review as a guide, we identify our contribution amongst the landscape of other autonomous exploration

techniques. Chapter 4 presents a new solution to autonomous flight, which allows a drone to explore in resource constrained settings. In this design we overcome the current barriers to autonomous flight in confined, connectionless, and unknown environments. In Chapter 5 we prove the functionality of our design by implementing it on a real-world platform testing it against multiple real-world testbeds. To conclude, in Chapter 6 we summarize our results and suggest further work which can allow our design to become even more robust with the ambition to expand its use to even more spaces.

Chapter 2

2 Background

In designing a system capable of autonomous exploration, it is helpful to first understand the hardware and software modules on which the later examined exploration algorithms can function. Reviewing these underlying components will provide a better understanding of how our prototype functions and the design choices that went into building it. First, we will take a broad look at drones and their various forms. Before developing any software, an appropriate platform must be identified based on the intended application of the system. We base this choice on the factors specified in the following section to isolate a class of UAVS that will perform well in our targeted environment. We then pick one configuration from this class and focus our attention on the components and algorithms suitable for this configuration.

2.1 Drone Types and Classifications

Drones can range from large UAVs capable of travelling thousands of kilometers in a single flight to small bio-inspired systems weighing less than 5g [12]. The classification of a drone is typically determined by weight, size, flight range, and application type. In literature there are inconsistencies in the specific values of these variables that distinguish classes form each other. Despite this, classification systems generally include the following five classes listed from largest to smallest: 1) Large Unmanned Ariel Vehicle 2) Small Unmanned Ariel Vehicles (uUAV) 3) Micro Ariel Vehicle (MAV) 4) Nano Ariel Vehicle (NAV) 5) Pico Ariel Vehicle (PAV) and 6) Smart Dust (SD). Figure 1 shows the division of these classes according to wingspan and weight [12]. Regardless of these divisions and their associated name, any such Fixed Wing vehicle is often simply referred to as a UAV and classification details are present when necessary.

Amongst each of these classes, further categorization can separate drones by their configurations of wing/rotor type, the number of rotors, take-off and landing direction, as well as lift style. It is important to note that these categories of drones can be seen across multiple classes (sizes) of drones. For example, a single rotor Vertical Take Off and

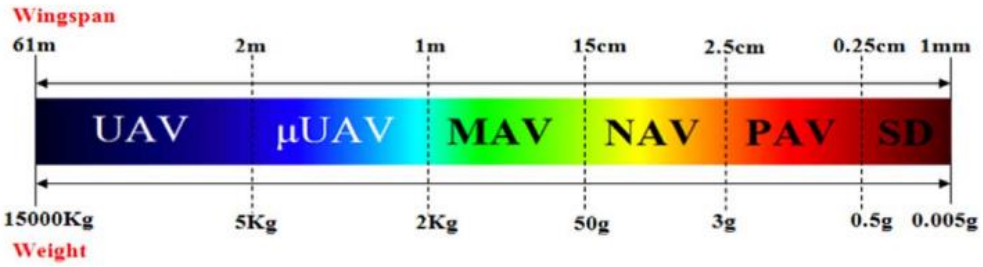

Figure 1 Drone class spectrum based on length of wingspan and weight [12]

Landing (VTOL) helicopter weighing over 1000kg can be classified as a UAV while a smaller helicopter weighing just 4kg can be classified as a MAV. Despite being in different classes, both have a similar configuration and can be categorized as helicopters. Each configuration has distinct advantages over the others and should be chosen to reflect the needs of an application. Here we will review the most common ones.

2.1.1 Fixed Wing

Fixed wing drones generate lift using rigid wing structures with an accompanying rotor. Like airplanes these drones have horizontal take-off and landing (HTOL) capabilities. These drones have the capability to quickly travel long distances with large payloads in a single flight before having to find a new energy source [13]. These advantages make this category of aircraft ideal for long surveillance or other data collection applications. Despite the advantages, kinematic and dynamic constraints of these systems limit their usage in any environment requiring tight turns or limited take-off and landing space. Fixed wing aircrafts are also more likely to experience the effects of air turbulence when compared to multi-rotor drones. Their inability to hover can also limit the amount of attention which can be given to a specific view during flight [13]. Some of these drawbacks are remedied in the hybrid category: Fixed-Wing Hybrid VTOL. This category combines the advantages of fast, long flights, with the ability to hover and take off in a vertical fashion. Despite these capabilities, Fixed Wing Hybrid VTOL crafts do not excel in forward flight to the degree that regular Fixed-Wing drones do, nor do they have the steady hovering capabilities of multi-Rotor drones. An example of a large, fixed wing drone can be seen in Figure 2.

2.1.2 Rotary Wing

Rotary wing drones use rotating blades to generate upward thrust. This can be a single blade with an accompanying system (propeller, turning wing, etc.) to change the drone's orientation, but they are more commonly configured with at least four separate rotors. One such example is seen in Figure 2. By controlling the angular velocity of these rotors, the drone can rapidly change its orientation and translational direction. Multi-rotor UAVs offer the ability to fly in every direction and the dynamics of flight are well established. These advantages are accompanied by the ability to carry a moderate payload and set of sensors offering favorable conditions for shorter term transportation applications. Given their VTOL ability, they are easy to deploy and require very little space to take off. Flight times are limited by battery capacity and the slow speed relative to fixed wing drones.

2.1.3 Flapping Wing

Taking inspiration from nature, Flapping Wing UAVs mimic animal and insect flight where thrust and lift are generated by flapping various arrangements of wings [14]. Wings can be arranged as a single pair (monoplane), two offset pairs of wings (tandem), or two pairs overlapping each other (biplane) [12]. In particular, the translational and rotational motions of hummingbirds and dragonflies have been the subject of research to develop flapping wing UAVs [14]. These systems introduce new complexities due to the aerodynamics of the light and flexible wings compared to other types of drones [12]. This category shows promise for an increase in efficiency if the low turbulent aerodynamics observed in insects can be exploited [15]. This advantage is also accompanied by maneuverability and rapid transition between a hover state and forward flight [16]. However, wide scale use in real world applications has not yet been observed given that commercial availability remains limited. An example of a small flapping wing drone can be seen in Figure 2.

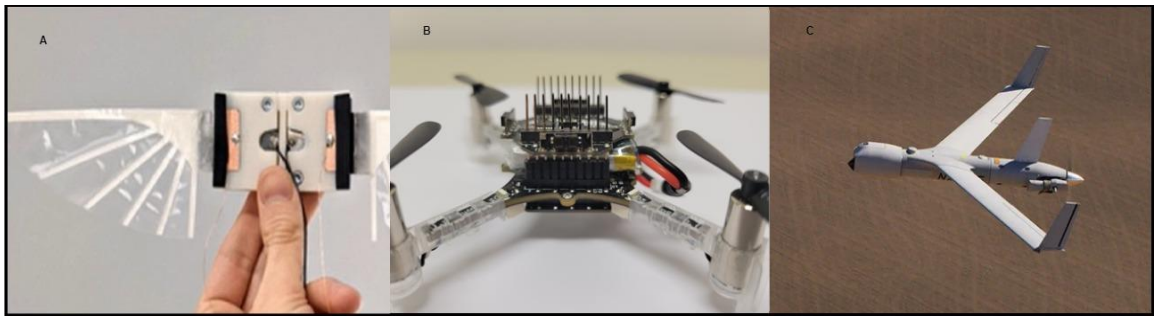

Figure 2 Three examples of drone configurations. A) A monoplane flapping wing drone. [64] B) A rotary wing UAV. C) A Fixed wing UAV [65].

2.2 Quadcopter Dynamics

Multirotor NAVs present a class and configuration that are most suitable for exploration in confined areas given their agility, size, and commercial availability. For the remainder of this study, we concentrate our work on these NAVs as they show the most promise in overcoming environmental barriers. In such an arrangement a drone consists of four propellers mounted on motors which attach to the body of the drone. Each adjacent propeller spins in opposite directions. These four propellers generate a thrust force which is combined to accelerate the drone. To change the orientation of the drone, propellers will spin at different speeds. When discussing the dynamics and/or state of a drone two reference frames with three perpendicular axes (x ,y, z) are used. In the World/Inertial Frame gravity dictates the direction of the z axis while the x and y axis are perpendicular to it. The Body Frame has its origin at the center of mass of the drone with the z axis being the direction of the propeller's axis of rotation. The x and y axes run along the arms of the drone normal to the z axis. The two reference frames are shown in Figure 3. If the two frames are aligned at takeoff, the conversion of a vector from the body frame to another is accomplished by multiplying it with a rotation matrix R and translating the result. The rotation matrix is constructed as a sequence of three rotations about each of the x, y and z axes.

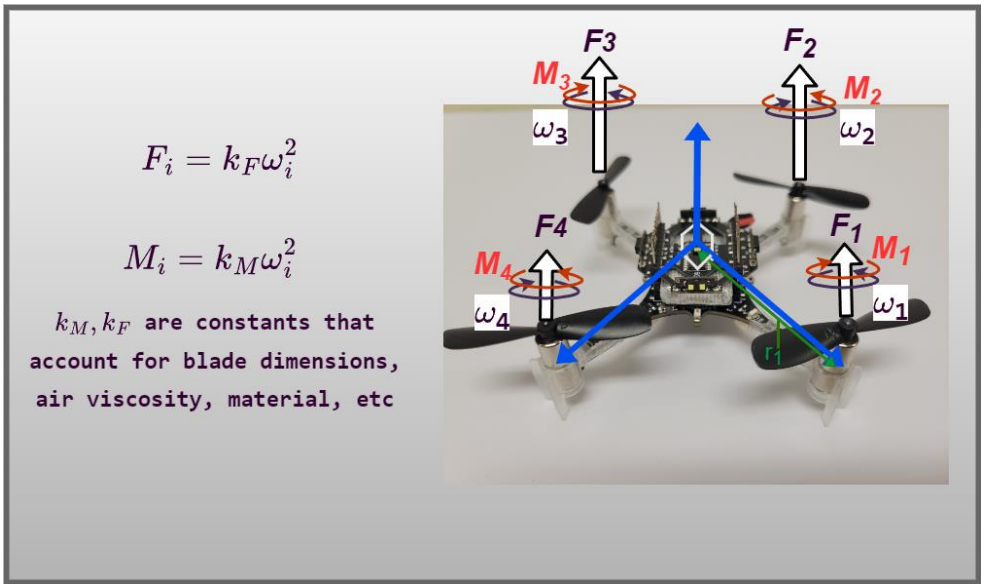

Figure 3 Visual representation of Moments and Thrust forces for each propeller i = {1, 2, 3, 4}. Here omega represents the angular velocity for a propeller.

The four propellers are responsible for producing an upward thrust and a moment in the opposite direction of rotation. Upward thrust from one propeller is quadratic in the angular velocity of the propellor is given by:

$$F_i = k_f \omega_i{}^2$$

Where F_i is the upward force generated by a single propellor and k_f is a constant which is typically found through experimentation and is dependent on factors like air density, a

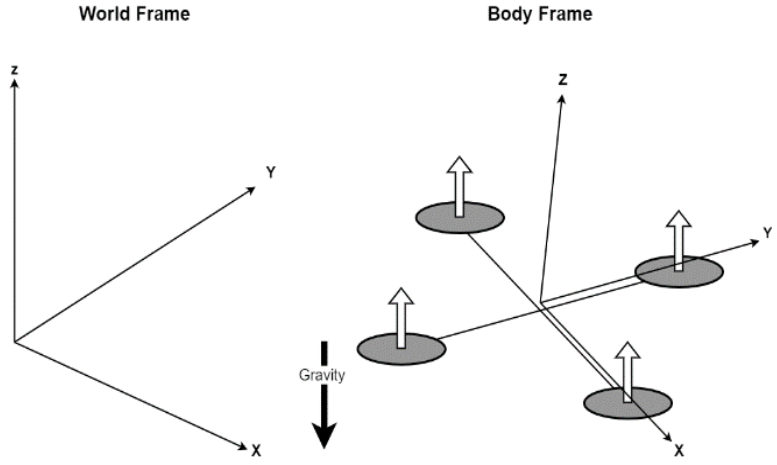

Figure 4 World and Body frames of reference

motor's torque proportionality constant, and the area covered by the propeller, the upward thrust is then the sum of thrust from each propellor in the body frame given here as a vector.

$$t_B = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=0}^{3} F_i \end{bmatrix}$$

The force which causes a propeller to rotate, is accompanied by an equal and opposite reaction force. This creates a moment M in the opposite direction of propeller rotation about the z axis of the body frame.

$$M_i = (-1)^i \, k_m \omega_i{}^2$$

Where again k_m is a constant and i is the rotor number from 0 to 3 and identifies the direction of rotation. Here $(-1)^i$ dictates whether the moment will be clockwise or counterclockwise as adjacent propellors must rotate in opposite directions. Both the moment M and the upward thrust F are shown in Figure 4 as a function of the angular velocity of each propeller. The torque about the z axis in the body frame is then given by:

$$k_m(\omega_0^2 - \omega_1^2 + \omega_3^2 - \omega_0^2)$$

Torque about x and y (roll and pitch), can be derived as:

$$\tau = L\left(k\omega_i^2 - \omega_j^2\right)$$

Where $\omega_i and \omega_j$ are the angular velocity of propellers opposite of each other and L is the distance between the center of a propeller and the center of mass of the quadcopter. Putting these together we have the vector of torques in the body frame τ_B:

$$\tau_B = \begin{bmatrix} L(k\omega_1^2 - \omega_3^2) \\ L(k\omega_0^2 - \omega_2^2) \\ k_m(\omega_0^2 - \omega_1^2 + \omega_3^2 - \omega_0^2) \end{bmatrix}$$

The rotation about each axis in the body frame is shown Figure 5. We now have a simplified model of forces which can be used with linear and rotational equations of motion. When

the propellers begin to spin, the drone will accelerate in some direction. Newton's Second Law suggests that the quadcopters mass multiplied by its acceleration is equal to the sum of all forces acting on it which is shown in the following equation.

$$m\ddot{x} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + Rt_B + F_D$$

Where x is the position vector of the drone we use R to convert the previously defined thrust vector from the body to the world frame. F_D is the force due to drag, and the first addend is the force due to gravity. The Rotational motion is described in the body frame using Euler's equations for rigid body dynamics:

$$I\dot{\omega} + \omega \times (I\omega) = \tau_B$$

Here, ω is the angular velocity vector of the body, I is the inertial matrix and τ_B is the vector of torques in the body frame.

These two equations provide a foundation to model the quadcopter's motion using the angular velocity of the four propellers and the rotation matrix. Using these parameters and equations, state estimation algorithms predict the location, velocity, and orientation of the body. Further details on the derivation of these equations are found in [17].

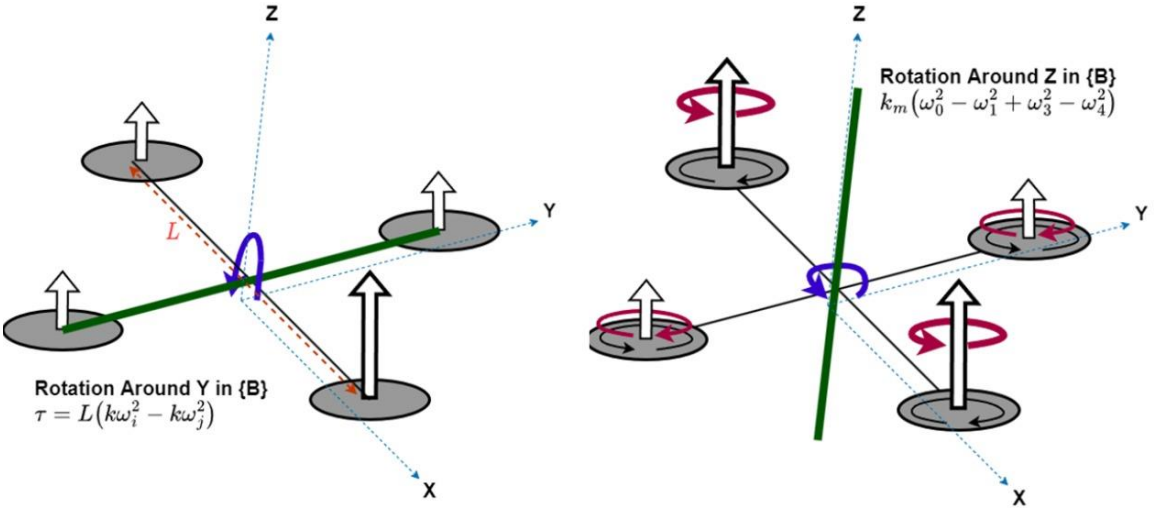

Figure 5 Rotation in the body frame where omega is the angular velocity, L is the length between two propellers, k is a constant accounting for propeller characteristics, air viscosity, etc.

2.2.1 Localization

To enable autonomous agents to make navigational decisions, they require knowledge about their current state within an environment. The process of determining a robot's location and orientation is referred to as localization. In this section we examine several of the most prominent positioning tools which help to enable localization. In practice, more than one of these strategies are used together where noisy data from many sources is combined in a process called state estimation.

2.2.2 External Positioning Tools

Global Navigation satellite systems (GNSS) are some of the most popular positioning tools for UAVs today. They are often recognized for their widespread use in Global Positioning System (GPS). These systems operate by communicating with satellites and measuring the strength and time of arrival (TOA) of signals to infer positioning. For UAVs this may be an appropriate approach in some flight environments, but line of site is required for a reliable signal. Techniques have been developed to help overcome these barriers by extending the technology, as seen in differential GPS or real-time kinematics-GPS, and by developing mm Wave communication channels. In environments that lend themselves to quality GNSS signals, accuracy is within a few centimeters [18]. With the advent of 5G it is expected that a fusion of GNSS and 5G signals can manage decimeter accuracy in dense urban areas where GNSS estimates alone are not viable [19].

When moving indoors, ultra-wideband (UWB) sensor nodes can be deployed in the environment as low power stationary anchors that a robot can communicate with. Again, by using TOA or time of flight data, an agent can determine its position [20]. This implies the availability to access the environment ahead of time to deploy such sensors. For this reason, it may not be ideal for indoor exploration tasks where the environment may be unreachable or unknown ahead of time.

Visible light communication (VLC) offers an alternative localization strategy which employs light emitting diodes (LEDs) and photodiodes (PDs). This low-cost option has the capability to determine distances between an optical sensor and an LED. In [21] existing light sources act as anchors and with a trilateration process, the position of a camera is

determined. Like UWB methods, access to the environment ahead of time is ideal. This allows the communication channel to be modelled ahead of time for robust localization [22].

2.2.3 On-board Positioning Tools

Autonomous agents can also be equipped with on board sensors like inertial measurement units (IMUs) to help localization. By measuring acceleration, altitude, and orientation over time, the current position of a drone can be determined. This works well over a short distance but measurements from these sensors tend to accumulate errors over time [23]. This can cause the state estimation to drift even after a matter of seconds and some correction step is needed [18].

UAVs equipped with optical sensors can also perform localization using vision-based algorithms for positioning. In [24] a survey of vision-based navigation separates the subject into indirect and direct methods. Indirect method extracts feature from images to use as inputs to a localization algorithm. These features should be static relative to the robot's state. Such a vision based optical flow algorithm is deployed on many commercial quadcopters using a downward facing optical camera to detect the flow of the ground beneath. This leverages the Lucas-Kanade (LK) gradient method [25] which assumes that the intensity value of a pixel representing a (x, y) coordinate on the ground below should be invariant to time defined by the following equation:

$$I(x, y, t) = I\left(x + \delta_x, y + \delta_y, t + \delta_t\right)$$

Where I is a function returning the intensity of the image at position (x, y) at time t. δ_x, δ_y, are the shifts of x, y between two frames and δ_t is the change in time between two frames. An example of (LK) is shown in Figure 6. In [26] this was further extended to include depth information and help locate the camera in the z direction. Indirect methods like edge detection using the gradient of an image can be used to find features but require some variation throughout an image. This might not be ideal for tunnel environments with smooth colourless surfaces. To overcome these limitations, direct methods make dense reconstructions of environments to help with localization. By optimizing geometric parameters with intensity data, the camera position and pose can be derived. This is a

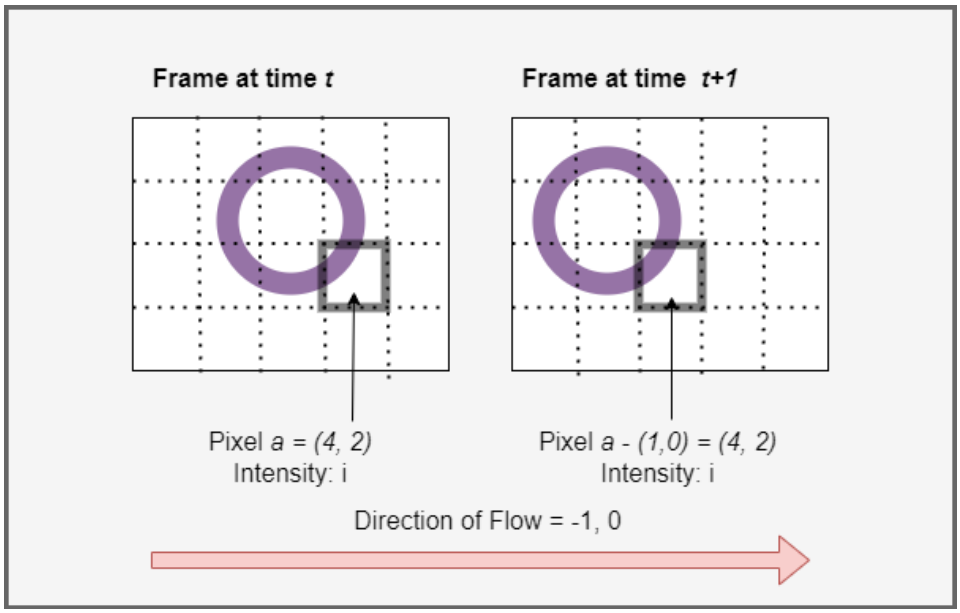

Figure 6 An example of the LK algorithm using two consecutive frames. In the first frame the pixel at (4,2) is measured to have intensity *i*. In the next frame, at time *t+1*, the same intensity is measured at (1,0). This suggest left to right movement.

computationally expensive task but shows promise for high bandwidth connected environments such as smart cities where 5g enables the computation to be offloaded to the edge [27].

In practice any one of the above methods may not be sufficient due to their limitations. To design a more dependable state estimation, many robots will fuse together position data from multiple sources in a process known as Multimodal Sensor Fusion [18]. This allows an agent to consider data form external and internal sources by using signal processing-based filtering algorithms such as the Kalman Filter [28], Extended Kalman Filter, Unscented Kalman filter [29] or Monte Carlo approaches like particle filters [30].

2.3 Environmental Constraints and Affordances

Having introduced UAV applications, classes of drones, and localization methods, we now direct attention to the environments in which drones will operate. We address the impacts that an environment can have on a drone and how these considerations can present barriers to autonomous flight. We also consider some ways in which assumptions about the environment can be leveraged as prior knowledge in some navigation strategies.

In the context of autonomous flight, one of the most crucial properties of an environment is the allowance of wireless communications. The more reliable this communication is, the more effectively a system can off-load the computation required to compute paths, map the environment, or localize itself. As a result, many applications depend on a stable communication channel with some external system. For example, in many precision agriculture applications, it is expected that a GPS signal can be maintained to help a UAV position itself above a crop. A reliable, high bandwidth communication channel with a base station can also be used to send large amounts of image data to process and inspect the crop.

Contrasting an open-air environment, the example in [31] features a confined tunnel environment where a reliable GNSS signal is not feasible. In this case the constraints of the environment force any UAV mission to operate with greater levels of autonomy. No longer can external systems help to support the drone during the mission. However, this work does provide an example of an environmental affordance in that the authors assume the tunnel to be a symmetric cylinder. This assumption is used to help a drone orient itself at the center axis of the tunnel and avoid collisions. Assumptions can be geometric properties of an environment but can also come in other forms. Another such example is in [32] where the presence of footpaths in a forest are assumed. This helps the authors develop a vision-based obstacle avoidance tool within the forest. Instead of only focusing on avoiding trees, the UAV actively searches for footpaths which would indicate a collision free path. Prior assumptions like this can be used to improve existing navigation strategies but are not dependable alone or in the presence of dynamic obstacles.

Many localization strategies depend on sensors to help with localization or other aspects of a mission. The performance of these sensors can be heavily impacted by an environmental variable. In civil applications, such as bridge inspection, an unstable GNSS signal will become a barrier to autonomy. For instance, as the UAV passes under the bridge the signal may be degraded or lost entirely as mentioned in [33]. Here the authors investigated alternative localization strategies, like using optical flow, to step in when the flight environment impacts the GNSS quality. These alternatives expose new environmental requirements such as the presence of light for passive sensing. In a case

where vision-based algorithms are employed for any part of the mission, the entire application will depend on the presence of visible light. Even in the case where a drone can produce its own energy source it could encounter conditions such as dust or rain which will affect sensor readings.

Another important consideration for an environment is the physical limitations in terms of flight space relative to the size of the drone's collision box. Indoor and underground environments often have narrow passages which must be navigated. These factors limit the class and configuration of drones that can be used. Large UAVs or those with HTOL configurations are not feasible in these environments. This also means there will be trade-offs between flight time and payload capacity. Cluttered environments also present challenges to the stability of a flight as turbulence generated from propellers can deflect off nearby surfaces interacting with the drone. This can have significant impacts on the stability of a lightweight body [34].

When designing an autonomous exploration strategy, the constraints of the environment may not be known ahead of time. To create practical solutions, applications must address and generalize to as many environmental conditions as possible. Knowledge of the environment ahead of time can help to reduce navigation complexity but should be used with caution in the presence of dynamic obstacles.

2.4 Autonomous Navigation

In the previous sections many of the lower-level components which support trajectory generation algorithms, or path planning algorithms, were introduced. These higher-level algorithms must be able to communicate with the lower-level components and function based on the assumption that components like that state estimator or controller modules are accurate. In turn, these components will be given commands from a path planning algorithm, determine the amount of power to distribute to each propeller, and control the drone to move to a certain point. For example, a higher-level navigation algorithm could generate a point in 3D space and pass it to the lower-level components through a commander interface. The controller subsystem will see these commands and cause the drone to respond appropriately and move towards this goal. A state estimation module will

provide feedback to the controller so adjustments can be made as necessary. The navigation algorithm could consult the state estimator to determine where the drone is, and create a new trajectory based on that information. The separation between the high and low levels of navigation allows path planners to focus on generating a quality set of points for the drone to travel without worrying about how a drone will get to each point. Moving forward with the discussion of autonomous navigation it will be assumed that the lower levels of navigation are accessible and reliable.

In many cases autonomous navigation can be split in two parts, short term dynamic obstacle avoidance and the longer-term path planning. In short term avoidance, we deal with previously unknown obstacles and avoid collisions due to imperfect state estimation. These dynamic responses to potential collisions provide collision free movement along a more global and long-term path. With many different arrangements of sensors, a wide variety of perception techniques are available to detect these obstacles. As [35] points out, short-term obstacle avoidance can be viewed as a process of perception and a collision prevention maneuver. This process is highly reactive and is therefore also affected when facing dynamic obstacles. Global path planning requires more resources as it must search for safe paths in a potentially large area. If there is previous knowledge of the environment, its obstacles are static, and a drone has near perfect state estimation and controller systems, global path planning alone could provide a collision free path.

2.4.1 Perception and Mapping

When generating collision free trajectories an accurate mapping of obstacles and free space is required. Before discussing mapping techniques, we consider how an agent can perceive its environment so that a map can be constructed. Many sensors available in the market today can be characterized as either passive or active. Passive sensors will use energy from a different source to acquire data. Most often this energy is in the form of visible light used by optical cameras, or infrared light for thermal images. For example, when perceiving the environment with a camera, light is reflected into the camera, but the camera sensor is not responsible for creating the light [35]. A limitation to these types of sensors is their reliance on an energy source within the environment. Active sensors will use their own source of energy and typically emit something into the environment which it will then measure. One

Sensor	Mode	Accuracy	Weather Condition	Light Sensitivity	Range	Sensor Size	Processing Requirement	Power Required
LiDAR	Active	High	Low Dependency	No	Medium	Small	Low	Medium
Radar μ-wave	Active	High	Not dependant	No	Long	Large	Low	High
MMW	Active	High	Dependant	No	Long	Small	Low	Medium
Ultrasonic	Active	Medium	Partial Dependency	No	Short	Small	Low	Medium
Thermal or IR	Passive	Medium	High Dependency	No	Medium	Small	High	Low
Camera	Passive	Medium	High dependency	Yes	Short	Small	High	Low

Table 1: A comparison of the key properties of several commercially available sensors for UAV perceptions [10]

example is Light Detection and Ranging (LiDAR) sensors like the one used in [36] where a 940 nm laser is emitted and its reflection off an object is then detected. The time of flight of the photons can be measured to determine the distance of the object. This type of sensor has the advantage of being more informative in environments lacking external energy sources but can require more power to emit its own energy. Both passive and active categories include sensors that vary in their range and Field of View (FOV) which both have an influence on a mapping techniques performance. Table 1 shows a comparison of several sensor types typically found onboard drones.

When trying to perceive whether space is open or occluded, an effective mapping technique is needed to build and store a representation of the environment. This representation can then be consulted by path planning algorithms to find safe trajectories through an area. In this section some of the popular mapping techniques are presented. We limit most of the discussion to mapping techniques that do not make assumptions about the shape or geometric properties of obstacles but acknowledge that these assumptions can be leveraged to generate effective models. For example, in [31] the environment is assumed to be a single axis cylindrical and symmetric pipe. This prior knowledge allows the authors to construct a parametric representation of the pipe by dividing it into smaller segments and using range data to learn each of the segment's geometric parameters. They can then be combined to represent the entire environment. While approaches like this one can be effective for specific structures, we focus on mapping techniques that generalize many types of environments.

Cell-based mapping methods, employ a technique which divides an area into an equally spaced grid that discretizes the environment. A cell corresponding to one of the discrete spaces will have an associated variable representing whether that space is free, occupied, or unknown. As the number of cells used within an area increases, so does the resolution of the map. If representing one cell requires the same amount of memory regardless of the size of space it represents, this presents a trade-off between the total area mapped, and the resolution of the map. In practice, environments will have large sections of occupied or unoccupied space. Quadtrees, and their 3D analog Octrees, are structures that adapt regular cell-based maps to exploit this property. These maps use a hierarchical tree to represent discrete sections of an area. A single tree node can represent a large space. If that space can be represented by a single value (free, occupied, or unknown), it does not have to be divided further. When representing a two-dimensional map with a quadtree as in Figure 7, the root represents the entire space. If the space is all free or all occupied, we assign the appropriate value (usually 1 or 0) to the root. However, if the area has both occupied and free space, we create four children representing the four quadrants of the space. Each of these children is treated as the root of another quadtree representing their quadrant. A tree only grows to the resolution necessary to summarize all space within a square, and only grows where needed. Using these adaptive structures reduces the memory requirements compared to traditional cell-based approaches. The occupancy value of a cell can be retrieved in $O(log(n))$ where n is the resolution of the map and techniques are available for constant time neighboring cell retrieval [37]. A simplified example of a quadtree mapping and the corresponding data structure is seen in Figure 7.

By using the corners of obstacles in the environment, visibility graphs can map an area by assigning nodes to vertices and connecting them with edges. Edges are only added between nodes separated by free space. In [38] a visibility graph was used to generate collision free shortest paths to a point. Despite their widespread use, visibility graphs are demanding in processing and memory requirements when used for 3D mapping. Attempts have been made to reduce these limitations by using fewer obstacles seen in [39] [40]. These maps are useful only when there is previous knowledge about the environment and may need short, additional path planning if the environment is dynamic. An example is seen in Figure 8.

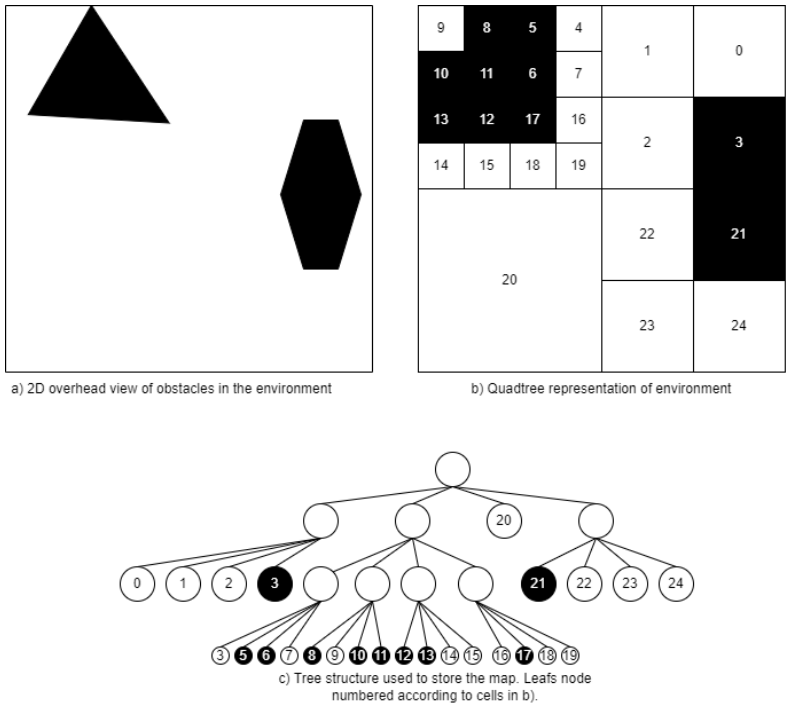

Figure 7 Adaptive cell-based mapping using a quadtree. a) A 2D view of the environment. b) the cell-based representation of the space, where white is free and black is occupied. c) The associated tree data structure.

In Figure 8, another alternative approach to mapping the environment is also shown. The map is known as a Voronoi diagram and in this technique, an area is represented as polygons, each enclosing one obstacle. These polygons are shaped in such a way that within any given polygon, the closest obstacle will be the obstacle that is associated with the polygon. Paths can be determined by following the lines along which two polygons meet. On these lines, objects are equal distances away. Although a path can be generated quickly, it may be suboptimal, and like visibility graphs, previous knowledge of the area is required.

One of the most efficient ways of representing an area for navigation is to use a topological map. Like maps seen in a subway system, this map is created by connecting nodes with edges that represent a path. This map scales well with the size of an environment and can be used to efficiently calculate paths between nodes. As this map is simplified, geometric properties of obstacles or free space are not fully specified and distances between points

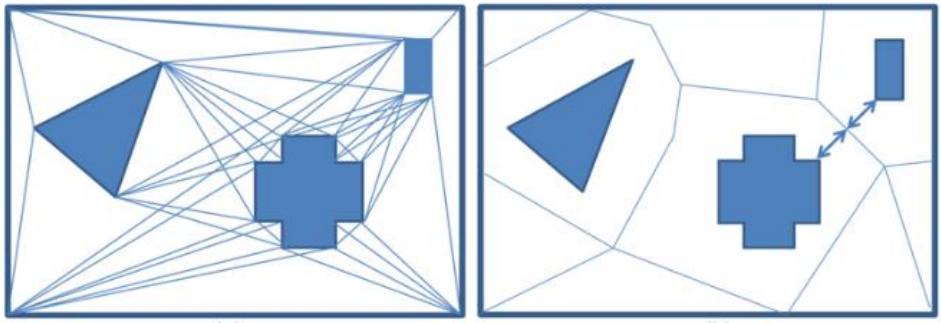

Figure 8 A visibility map (left) and a Voronoi diagram (Right) mapping of a 2d environment [67].

may not be preserved in the map. Instead, their usage relies on an agent's ability to detect when they are at a specific node. The efficiency of this mapping technique is met with a large dependence on a robot's ability to know its location within the environment [18] and to have robust short-term obstacle avoidance. An example of this map is shown in Figure 9.

Figure 9 A topological map showing a crude representation of available paths in a space [69].

2.4.2 Obstacle Avoidance and Path Planning

Autonomous flight is only useful to any mission if it can efficiently plan a route to and from the necessary coordinates to carry out a task. With previous knowledge of the environment, this process can be done offline. However, any applications involving the exploration of unknown environments will require online planning to generate new

trajectories. While a drone is flying along a given trajectory, it may also be at risk of colliding with dynamic obstacles which were unknown at the time of mapping. Using onboard sensors, shorter term collision avoidance can be introduced. Collision avoidance and path planning can be either a reactive or deliberative approach. In reactive planning, a loop of data acquisition and reactive control is used to keep the drone a safe distance from any obstacles, static or dynamic. With reactive planning, the drone can end up in a local minimum where it cycles loops without ultimately reaching its goal. On the other hand, a deliberative planning approach will use some form of mapping of the surroundings to determine a collision free path. This approach alone cannot accommodate safe flight in dynamic environments and if such flight space is demanded by an application, some form of a hybrid approach will be needed [36]. Typically, this will involve a path planning technique followed by some form of rerouting or replanning if a dynamic obstacle is encountered. Here we review some of the most common path planning techniques.

Geometric path planning approaches consider the geometry of both obstacles and the drone ensuring a safe distance between the two. Paths travelling through free space are inferred by the geometry of obstacles. By creating geometric models of the obstacles in the environment, such as with a Voronoi diagram or Visibility graph, an optimal solution can be found using graph-based searching algorithms. Often this involves some cost estimate of distance and how aggressive a maneuver may be required to prevent a collision and stay on a potential trajectory. These maneuvers are most commonly a change in speed or heading of the drone [41] which can be directly calculated from the angle of a turn in a path.

Most planning algorithms rely on randomly sampled points which connect and eventually sample a goal region. In [42] Rapidly Exploring Random Trees (RRTs) were first introduced. Using this method, a tree is grown within a model of an area by generating random points and attempting to grow the tree from an existing node in the tree that is closest to a goal. If there are no obstacles between a newly generated point and its nearest neighbour in the tree, a new node and edge are created. This tree is rooted at the starting coordinates and continues until a goal region is reached. Probabilistic Roadmaps first presented in [43] are another tool in which a tree is grown with random samples of the

configuration space. In this technique, the randomly generated points become nodes of a graph. Each newly generated point is compared to its nearest neighbours in the tree. If a certain criterion is met, a new edge and node are added to the graph and the tree has grown. Eventually, a dense map of the entire space is created. As new goal regions are needed, new paths can be created without regrowing the tree serving well in dynamic spaces. Building on RRT, in [44] Rapidly Exploring Random Tree star (RRT*) were introduced. This approach optimizes RRT paths by checking all neighbours within a radius and choosing to grow only from the most optimal one. Optimal is defined as the shortest path to the root and not necessarily the closest node. After a node is added, other nodes in the area are checked to see if connecting to the newly added node would provide a more optimal path. Figure 10 shows an example of a path found using and RRT.

Artificial Potential Fields are methods that assign an attractive or repulsive force to obstacles and drones to direct the drone away from dangerous areas and towards a goal [45]. In constructing an Artificial Potential Fields an attractive force is placed at the goal, while repulsive forces are put around obstacles. If a robot comes within a predefined distance of an object, the field presents a repulsive force [46]. With optimization-based methods a probabilistic search algorithm is optimized to quickly find a trajectory which is near optimal and avoids any collisions. These methods usually rely on previous knowledge of the obstacles in a space and require additional short-term obstacle avoidance if dynamic obstacles may be present. If this knowledge is available paths can be computed efficiently but path generation may suffer from becoming stuck in a local minimum in the presence of complex or multiple obstacles. Recent work has been conducted to help avoid these issues in [46].

Sense and avoid methods are lightweight techniques in which an individual agent detects an obstacle and reacts accordingly [35]. These are short-term methods which react to the environment as it is sensed. With this technique, there is no requirement of previous knowledge. Geometric algorithms can span into this category when using obstacles to follow but are often more sophisticated in that they consult either a global map or share information between agents and external systems. In this category, we also see wall-follow algorithms such as the collection of bug algorithms in [47].

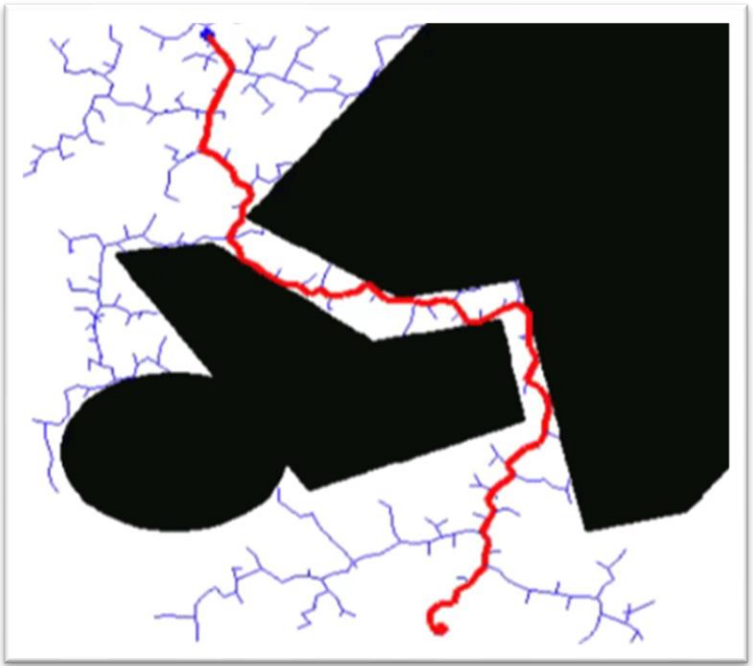

Figure 10 Safe path generation with RRT. A tree is randomly grown from the starting point (blue) until it reaches the goal area (red). Once the tree intersects this area a path is generated(Red line) [66].

2.4.3 Exploration

If an agent's goal is to explore an area of which there is no previous knowledge, it must have a way to develop goal states to travel to. These goal states should motivate the drone to move to an area that will expand the amount of explored space while safely travelling through the environment. While doing this, important considerations are the time it takes to explore an area and the power required to move to new states. If an exploration technique requires repeated rapid changes in the robot's pose, it must frequently draw power to accelerate the angular velocity of propellers and change its orientation. Algorithms that find worthwhile sections of the space to explore can are divided into frontier-based exploration [48] or sample-based exploration [49] strategies.

Frontiers were introduced in [49] as defining the border between explored regions and unexplored regions within a space. If the exploration space is discretized into cells, as is done in cell-based mapping, a frontier cell would be a free cell that is adjacent to a cell

whose occupancy is unknown. This approach works by searching for frontier cells at the borders between previously explored space and unknown space. Under the assumption that open and accessible space is contiguous, any path to unexplored space must cross one of these frontier borders. After finding the frontiers, one is chosen as the goal and a path is generated. This method depends on the efficiency of detecting frontiers, a process that can be computationally expensive depending on the mapping strategy used and the dimensionality of the environment.

Sample based exploration strategies find goal regions by randomly sampling possible configurations in the exploration space. This strategy was motivated by early work in [50] which sampled viewpoints around a structure to find a Next Best View (NBV) point to help map the structure. Newer strategies have extended the sampling strategy to augment a random tree to not only path plan, but to find quality views which would increase knowledge of the environment. In [51] a random tree is grown in the environment and each sampled point added to the tree is also evaluated based on how much unexplored space is visible from the point.

Other explorations strategies have also been proven useful under certain conditions. For example, it has been shown in [52] that in a maze-like environment, a wall-following bug algorithm can produce better results than a frontier approach. In [53] the use of frontier exploration was combined with sampling methods. Frontier exploration was used to generate a global path and while the drone was on a path to a global frontier, it sampled nearby configurations. If a configuration with high information gain was sampled, the algorithm would check if going to that configuration was efficient. If so, the global path would be altered to include the sampled area.

Chapter 3

3 Related Work

With a preliminary study of autonomous drone exploration, we now turn attention to notable contributions in the area. The research and strategies discussed in this chapter present the current state-of-the-art solutions. These solutions differ in their implementation of state estimation, perception, mapping, path planning, and exploration, but all aim to develop online trajectories which eventually cover an entire space. We first review solutions found in research before turning towards commercial UAV exploration options. Many of these solutions expand on leading exploration methods described in Chapter 2 and may be generalized to many types of environments. These are not necessarily complete solutions in that some may not form a fully autonomous system, but all will help to provide a layout of where the current technology lies.

3.1 Dynamic Exploration Planner (DEP)

The authors of [54] propose a multi-query dynamic exploration planner (DEP) capable of exploring a previously unknown environment while avoiding both static and dynamic obstacles. To map the environment, an adaptive cell-based octree is used. A PRM is constructed for exploration and an incremental sampling strategy continues to improve node coverage of areas already mapped, adding nodes to the PRM. By storing the utility of each node, the reconstruction of a path in the presence of dynamic obstacles is possible. The entire process is broken down into four stages.

In stage one, a PRM is created by sampling free space configurations which are close to the UAV and then sampling the free space globally. The purpose of sampling nearby areas first is to ensure better coverage of recently observed areas which are not yet covered by the PRM. If strictly global sampling was used, areas already covered by the PRM would become too dense as more nodes were added. This sampling process continues until a saturation threshold is met. Sampled points are evaluated for their proximity to existing nodes and if they are not too close to a neighbour they are converted into nodes and added to the PRM.

In stage two, Nodes are evaluated and updated by the number of unknown visible voxels which would fall into a sensors FOV when at the location of the node. Unknown voxels are categorized as normal voxels, surface voxels, or frontier unknown voxels, depending on their neighbours. Each of these categories receive a different weight in the evaluation of nodes. Surfaces are weighted the strongest (given that they are the most important), followed by frontier unknown voxels, and then normal unknown voxels. The weighted sum of these voxels provides the information gain for a node. It would be inefficient to repeat this process for every single node in the PRM as many would be unchanged from the previous iteration. Rules are applied to determine which nodes to evaluate based on Euclidian distance to the drone's previous trajectory. Those nodes which are very close to the previous trajectory are deemed to have very low exploration utility and so their information gain values are set to zero and not re-evaluated.

Stage three is responsible for trajectory generation that minimizes exploration time by analyzing the possible trajectories through the PRM and evaluating their information gain rate. This is a measurement of the expected information gain over time travelled along the trajectory. Trajectories are first found by looking at the node gains and selecting those that have high information gain. These nodes are gathered into a set and for each one a path is generated using graph search algorithms. These paths make up the set of trajectories to evaluate.

In the final stage, the trajectory is optimized using Euclidian signed difference function (ESDF)-based optimization. The objective is to reduce average trajectory execution time and distance. During this process, a safety distance from obstacles is also considered and the trajectory is altered where necessary.

The DEP was tested in simulation using a UAV equipped with a camera and intel i7 7700HQ 2.4 Ghz. To benchmark against other leading exploration methods the exploration time, path length, and computational time where evaluated. Receding-Horizon Next Best View [51], The frontier approach from [48], and Autonomous Exploration Planner (AEP) [55] were also tested in the same simulation environments. Table 2 summarizes the results averaged over 10 runs for each planner.

Simulation Description	Planner	Exploration Time Average (minutes)	Total Path Length Average (Meters)	Computational Time Average (Minutes)
Café Environment 20x10x3 meters	DEP [54]	4.77	43.12	0.17
	RH-NBV [51]	7.12	76.80	0.5
	Frontier [48]	6.44	56.11	0.37
	AEP [55]	5.48	59.16	0.37
Maze Environment 20x20x3 meters	DEP [54]	17.75	146.44	1.05
	RH-NBV [51]	31.44	271.60	8.10
	Frontier [48]	34.74	330.10	2.01
	AEP [55]	23.23	200.65	2.44
Office Environment 40x30x3 meters	DEP [54]	31.84	318.58	2.45
	RH-NBV [51]	48.21	421.8	12.13
	Frontier [48]	38.14	253.63	12.29
	AEP [55]	37.46	327.78	6.64

Table 2 DEP simulated testing results and comparison to other leading planners [54]

3.2 Receding Horizon Next-Best-View Planner

Expanding on RRT path planning, the authors in [51] combine exploration into the creation of the RRT structure. This is achieved by randomly sampling within known free space and evaluating whether the sampled point would provide view into unexplored space. When a quality view is sampled, the RRT is used to find a path using a graph search algorithm. When this path is found, the robot will execute only the first step in the trajectory. After travelling along this one edge in the graph the process starts over. The outcome of doing this replanning every iteration is that the horizon of unknown space is typically pushed further away from the drone.

The exploration method relies on an Octree structure for mapping of the environment and tracking known free, known occupied, as well as unknown space. Perception is done with a stereo camera, but other sensors could be employed. To begin a random tree is grown by sampling within free space that is rooted at the drone's current configuration. A node n is assigned a gain value given by the sum of unmapped volume which could be explored at nodes along the branch terminating at n. Sampling continues until a predetermined number of samples are generated. After sampling is complete, the node with the highest information gain is selected as the next trajectory. The branch along this trajectory is saved and used to

reinitialize the random tree in the next iteration. The drone executes the first step along the selected trajectory completing one iteration. The tree is regrown, and the process continues until samples are no longer providing views with a positive information gain.

The complexity of this approach is dependent on the volume being explored and the resolution of the octree. The computational cost is of the following order.

$$\left(N_T \log(N_T) + N_T/N_T \log(V/r^3) + N_T\left(d_{max}^{planner}/r\right)^4 \log(V/r^3)\right)$$

Where N_T is the number of nodes in a tree, r is the resolution of the octree, V is the volume of space being explored, and $d_{max}^{planner}$ is the sensor range. The full derivation of this equation is found in [51].

This algorithm was tested in three environments, two simulated and one real world. Under the simulated environments, an AscTec Firefly hexacopter MAV is used with a stereo camera. The environments include an apartment space and a space containing a bridge. The results of total exploration time, and computation time were compared with the frontier approach from [56] adapted to the same environments. This comparison showed an increase in exploration rate across both simulated environments. The frontier method failed to explore the space in the larger bridge environment due to the computation time required for detecting frontiers. In the real world testing a closed room with scaffolding against one wall was used. Again, an AscTec Firefly Hexacopter MAV was chosen to perform the task. This drone was equipped with Visual-Inertial sensor providing stereo imagery coupled with IMU data. Results of the experiments are summarized in Table 3.

3.3 LiDAR-Based Stabilization, Navigation and Localization

In [57] the limitations of vision-based localization in GNSS denied spaces were addressed by developing a pose estimation system using 2D LiDAR scans. Instead of depending on heavy 3D scans, this solution is suitable for small MAVs with lower payload capacities. By building a robust localization system for MAVs autonomous exploration in cluttered environments can be achieved. Results from this work show

Environment	Exploration Method	Total computation time (minutes)	Exploration time (minutes)	Notes
Simulate Space containing bridge: 50x26x14 meters	RH-NBV [51] Frontier [56]	9.4 1670.1 (aborted)	43.8 1660.4 (aborted)	Frontier approach did not complete exploration task.
Simulated Apartment Space: 20x10x3 meters	RH-NBV [51] Frontier [56]	0.25 1.39	8.37 7.83	Frontier based had lower total area explored
Real world room with scaffolding: 9x7x2 meters	RH-NBV [51]	0.19	4.22	Frontier comparison unavailable

Table 3 RH-NBV planner testing results in two simulated environments and one real world experiment.

promise as the ability to reduce state estimation drift was evident in the piloted exploration of confined indoor environments.

Equipped with a LiDAR sensor, this solution is presented as a pipeline architecture starting with the LiDAR data at each timestep and flowing into a Kalman filter for final state estimation. The incoming data is fed into two modules. In the first stage, two sequential LiDAR scans are aligned to find the rate of change of the state vector. Separately, the LiDAR scan is also sent to a global matching module where scans are matched against a global map. This map is implemented as an octree. To avoid performing the expensive matching process over the entire map, only a small region around the current location is used.

Results show that the state estimation functioned well both indoors and outside where fewer obstacles lead to a smaller number of data points on LiDAR each scan. Testing indoors with a roughly 4m by 6m rectangular trajectory revealed the drift in the trajectory to be limited to less than 40cms. This was achieved without any loop closing methodology to reaffirm previous state estimates in past states.

3.4 History-Aware Autonomous Exploration

In [58] a History-Aware Autonomous Exploration planner was tested on an MAV equipped with a visual inertial sensor for depth perception and localization. This solution uses a record of previous states that the robot has been in to seed a RRT in a multistep fashion. By leveraging this history of this planner, a UAV can first locally optimize the robot's orientation with respect to the unexplored space.

First, the agent generates samples in its immediate vicinity. These samples are treated as potential NBVs but if no quality views are found, the sampling continues from a state in the agent's history log. This state is recognized as one that still has potentially valuable configurations in its near vicinity. If sampling fails again, the entire space is sampled. For each sample, N discrete orientations of the drone are used at that position to evaluate the information gain. When a quality view is found the RRT is used to generate a path. Trajectory optimization is performed to limit the number of orientation changes and length of path. Testing was carried out on a small and large maze in simulation with an Intel i7-4700MQ 2.4Ghz. When compared against RHNBV approach it showed faster exploration time in both scenarios. Testing was also performed in a real-world environment in a semi-autonomous fashion. First a safety pilot would map some of the space before allowing the UAV to take over. A supervisor also approved trajectories as they were generated.

3.5 Industry Solutions

Commercial UAV exploration solutions are growing in popularity as companies develop systems that can explore more constraining environments. For example, the Autonomous Control Systems Laboratory (ACSL) provides an inspection solution for infrastructure in GPS denied areas by combining visual and LiDAR data to localize a UAV and safely navigate moderately confined spaces [59]. The Elios 2 by FlyAbility uses a spherical cage around the drone to prevent collisions. With powerful lighting and sensors, it can be piloted through GPS denied indoor environments [60]. Exyn Technologies have equipped their own product, an Exyn Aero drone, with various vision and LiDAR sensors to perform remote inspection in GPS denied environments [61].

3.6 Research Gap

After a consideration of the related work, Table 4 provides a summary to help identify the deficiencies which prevent the available solutions from moving into more confined, connectionless spaces. A review of this table shows that there has been significant progress in the ability to explore connectionless, dark, constrained environments but each solution has some characteristic preventing its flight in our target environment. This discussion brings us to the edge of autonomous capabilities and helps to define what is required to push UAVs to operate in more environments. After looking at the related work, we observe a trade of the capabilities and size of the drone. Physically larger drones have been able to combine all the necessary components of autonomous flight as their higher carrying capacity can accommodate the necessary hardware. These solutions have yet to be presented on NAVs with small physical dimensions with limited resources. On the other hand, the smallest UAVs still have environmental dependencies or are not yet fully autonomous, and thus still require connection to a piloting operator/system.

When considering work in the area, we outline two available options that will help to overcome the barriers of autonomous flight that are prohibiting existing solutions from moving to more confined and connectionless spaces. First, the development of more efficient mapping, localization, and/or explorations techniques. If scalable maps that can be managed on smaller processors become available, these can be leveraged for localization and exploration using the existing algorithms in place today. Another approach is to reduce the coupling between mapping, localization, and exploration. Exploration may then be carried out without requiring such informative maps or localization. This would eliminate the need for detailed and computationally expensive mapping, giving a higher tolerance to imperfect sensor data or state estimation which may perform poorly in certain conditions. Another way to bring UAVs to more confined spaces, could be to find new solutions that enable a high bandwidth connection between a base station and a drone as it travels through otherwise connection-denied areas. If a solution can be found, the expensive localization and mapping can still be offloaded to a device with more resources. The navigation commands can then be generated and sent back to the UAV to be carried out.

Method	Testing	Autonomy	UAV Size (mm)	Mapping Type	Scalable to Connectionless NAVs
DEP [54]	Simulated	Autonomous	400x400x300	Octree 20cm resolution	No
RH-NBV [49]	Real World	Autonomous	500x500x500	Octree 20cm Resolution	No
History Aware Autonomous Exploration [55]	Real World	Semi-Autonomous	500x500x500	undisclosed	No
Lidar Localization [58]	Real World	Piloted	undisclosed	Octree 20cm Resolution	No
ACSL [59]	Real World	Autonomous	654x1173x1067	Undisclosed	No
Elios 2[60]	Real World	Piloted	400 (sphere)	Undisclosed	No
Exyn Aero[61]	Real World	Autonomous	883x886x520	Undisclosed	Unknown
Presented Approach	**Real World**	**Autonomous**	**100x100x40**	**Topological**	**Yes**

Table 4 A comparison of related research and commercially available navigation and/or exploration strategies. Each cell is colored according to the attributes ability to scale to small UAVS operating in confined, connectionless spaces. Red shows that an attribute is preventing the solution from scaling, yellow is used for unavailable data, and green shows an attribute that does scale. The proposed solution presented in chapter 4 is along the bottom.

3.6.1 Novelty of Proposed Solution

To enable our design to reach new environments, we focused on reducing the dependence of a detailed, cell-based map found in most other solutions. To accomplish this, we designed an alternative approach to exploration which leverages the concept of frontiers but does not require searching over discrete sections of the exploration space running in constant time, regardless of how large the flight space is. This allows our solution to operate on extremely limited hardware resources. In eliminating the computational costs of inserting, deleting, and storing data in a cell-based map, resources are preserved for other

tasks like stabilization, control or for higher level applications like infrastructure inspection. In the absence of a detailed map, we provide robust obstacle avoidance supported by reactive short term obstacle avoidance as well as globally planned paths through free space. These paths are easily generated from the computationally inexpensive, graph-based map. By leveraging these key attributes, our exploration technique can operate fully autonomously in environments yet to be reached by other state of the art solutions. As table 4 shows, our solution is the first to meet all the following requirements:

1. Full autonomy will all decisions being made on board the UAV
2. Small frame UAV less than 15x15x10 Centimeters
3. Scalable Mapping techniques capable of operating on resource depleted systems
4. Zero connectivity with external systems
5. Proof of functionality in real world test cases

As proven in Chapter 5, the presented approach has overcome the existing barriers and introduced a new way for UAVS to explore previously unknown environments. Upon reviewing the existing literature and commercial options we believe this design has enabled our prototype to be capable of systematically exploring the smallest, connectionless environments yet to be reached by autonomous UAVs.

Chapter 4

4 Methodology

We now present our novel approach to the exploration of a connectionless, previously unknown, and confined space by using a frontier exploration method without relying on a detailed global map for path planning. This exploration strategy is designed for extremely resource constrained systems where computation and decisions are done on board. With no dependence on a feature map, this solution is highly scalable. In our approach, the exploration space is covered by continuously detecting and moving towards unexplored areas like other frontier-based methods. Unlike existing frontier-based algorithms this strategy finds new frontiers in constant time on the fly while it completes trajectories set by a path-planner in a previous step. Our work is tested using a drone with six lightweight LiDAR sensors for environmental perception. Localization is done by fusing IMU data with optical flow from a downward facing camera. Furthermore, by assuring that the heading of the drone is facing the direction of travel this solution can be extended to collect data with other types of sensors where the field of view may be limited to a single direction. After the exploration is complete, or the power resource is depleted, our system will plan a path back to area that it was deployed. The outer loop of the presented algorithm iterates until no frontiers are left to explore. In the following sections the major components of each iteration are discussed.

4.1 LiDAR Based Frontier Detection

The discovery of new regions to explore is centered around a frontier detection approach. As we aim to create a system that can explore confined spaces, we need to consider the limited memory available to the small-bodied drones needed to explore small spaces. A cell-based map is typically used in exploration strategies but using this mapping technique causes a restriction on scalability. If we wish to explore variable sized space with a cell-based map, it must accommodate any size of space reachable to the drone and would have to dynamically grow during the mission. The memory required to do this, even with adaptive cell-based maps, quickly outgrows the available memory of NAVs, especially when extending to 3D. To avoid depending on existing exploration solutions that leverage

maps with these constraints, we propose an alternative approach to detecting frontiers. By doing this, we no longer require a detailed cell-based map and can use alternative and more memory efficient approaches.

To change our approach from finding frontiers using the traditional cell-based map search, we introduce the idea of a Visibility Volume (VV). A VV is a region of space where every point within that region has straight line visibility to every other region. If we consider all the reachable space in an environment, we can be confident that it is in at least one of these volumes of space. Now if we add the condition that no two visibility volumes overlap, we can divide up the space into discrete volumes. For example, in Figure 11 we have divided the space into 5 of these volumes using the red dotted lines. These divisions are of course arbitrary and could have been divided in many other ways. With the entire exploration space divided into VVs which do not overlap, we can rephrase the exploration problem as two smaller subproblems. First, when the UAV is in one VV how can it be sure to detect and move to all adjacent volumes. Second, how can we fully explore the VV that the drone is presently in. With these two problems solved, we can start at any volume and eventually visit all reachable space. This is the motivation for our frontier detection algorithm and in the coming paragraphs we will see how this algorithm tackles both subproblems.

To address the first subproblem we use the assumption that we will be exploring a confined space. We therefore expect that many obstacles will fall into a sensor's FOV at any given time. In other words, the UAV will never be too far from a wall, ceiling, floor, or other

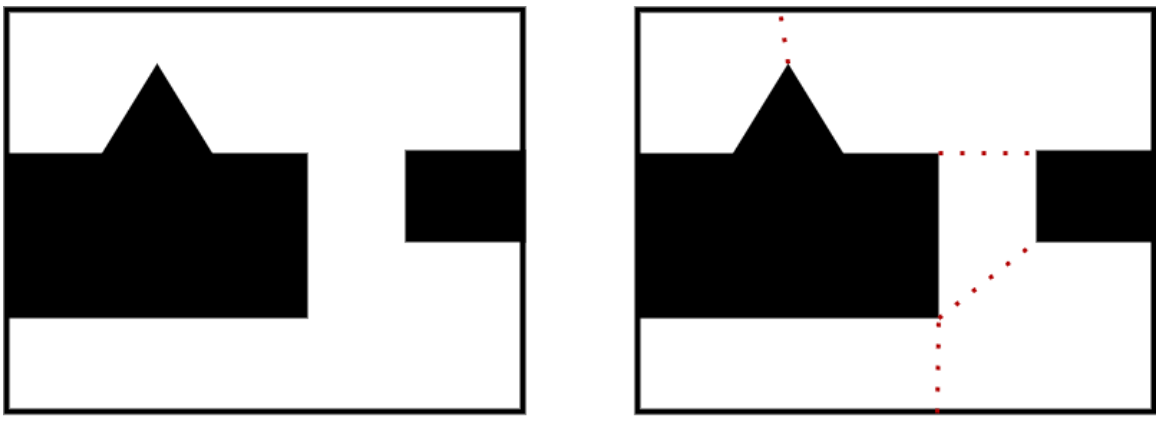

Figure 11 The division of a 2d space into Visibility Volumes that do not overlap. From within any volume, a drone must be able to detect adjacent volumes, and be able to fully explore the volume it is in.

structure in the environment. This assumption is reasonable for our work as more open spaces would accommodate a wireless signal to and from a base station for which more resource demanding solutions already exist. This assumption affords our system the ability to have reliable depth information in multiple directions from six LiDAR sensors.

In the same way that a human could extend their arms and feel around in the dark to avoid obstacles, our UAV uses its sensors to navigate. Instead of arms, a UAV can use laser sensors which function well even in dark spaces and can detect obstacles that are up to four meters away. We can then track the readings from each of these sensors over a short period of time, recording the most recent readings. By finding the gradient of these readings, we can detect when a sudden change occurs. This sudden change reflects an edge of some structure in the environment. Moving back to the analogy of a human navigating in the dark, if one were to run their hand along a wall while walking, this individual would perceive the wall to be a certain distance away. If they continue and suddenly the wall ends, their hand will slip off the corner. This would suggest that an opening exists in the direction of their arm and that a new passage exists. On a UAV we can sense in many directions at once with small sensors readily available today. In each direction, we employ this edge-detection-like method to find new passages to explore. The algorithm in Figure 12 describes the process for one sensor in the first part of the if statement. In this simple

Algorithm 1: Frontier Detection

Input: readings: *a list of the last 2 sensor readings in order of most recently observed*
state: *the robots current state in the exploration space*
thresh: *The change that determines an edge*
maxDistance: *Sensor readings beyond this distance indicate a free space frontier in the direction of the sensor heading*
Output: Coordinates and type of a candidate frontier or None if DNE

```
x; //vector point in the exploration space
type; //Open space or edge frontier
if |readings[0] - readings[1]| > thresh then
    x ← locateFrontier(state, readings[i]); //calculate position of frontier
    type ← edge;
    return x, type;
else
    if readings[0] > maxDistance then
        x ← locateFrontier(state, maxDistance); //calculate position
          of frontier
        type ← open;
        return x, type;
    end
end
return None;
```

Figure 12 Frontier Detection in a single direction at one timestep.

implementation of our proposed frontier detection method, only the last two samples are needed from a sensor. The threshold for selecting a frontier is used to determine what degree of change would indicate a new frontier. A small change could simply point to a variation in a structure that does not expose a new region to explore. To keep exploration efficient, this threshold variable should be large enough such that minor changes in an obstacle are not detected as edges. We use the absolute value of the change to cover the case where the distance of a sensor reading suddenly drops or increases as both cases could signal a new region to explore. Figure 13 shows an example of both scenarios. Returning to the first subproblem of exploration, we now have a way to detect an adjacent visibility volume from within the volume the drone is presently occupying.

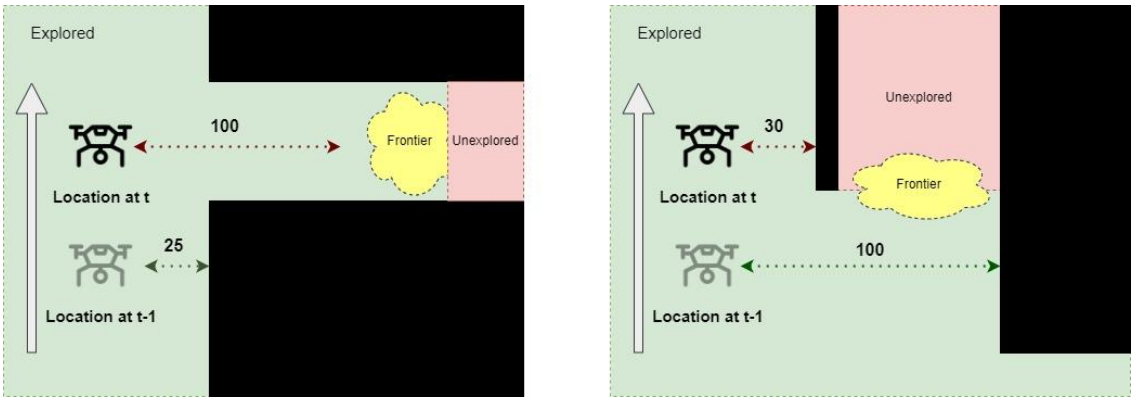

Figure 13 Frontier detection in two scenarios. The UAV is moving in the direction of the white arrow. The drone can be seen at two subsequent timesteps. At each timestep the dashed line shows the sensor reading. A yellow cloud shows the frontier which is found.

While an abrupt difference in sensor range will detect many frontiers, it has limitations and would not be able to generate frontiers without some sort of structural variation. An example of this limitation is if the drone found itself in a large open area or along a narrow corridor. This is where the subproblem of exploring the entirety of the volume of space the drone is presently in. To address this second subproblem, the second part of the presented algorithm checks to see if the sensor reading shows that there is open space extending far enough away to warrant exploration in that direction. For example, if the sensor is reaching its maximum range, it means there could be unexplored space beyond it. Even if a reading is not at the sensor's maximum range, it might still be worth travelling to as there could be

a new region to explore that is only detectable from that point. This case can be seen in the first diagram of Figure 14. The frontier detection in open regions is covered in the second half of the algorithm shown in Figure 12. When detecting frontiers, the unique characteristics combined with sensor readings are used to place the frontier at a safe distance from detected obstacles so that travelling to a frontier will not result in collision. The frontiers are then passed to a topological map building module described in the next section. This frontier finding algorithm is not resource demanding and runs in constant time. It can be applied to multiple sensors at each timestep to cover every direction. It is also important to note that this method will find candidate frontiers and they are first tested for proximity to other existing frontiers before being added to the map. As discussed in the coming sections, it is the responsibility of the graph-based map to manage the criteria for a candidate frontier to be added. By approaching frontier detection this way, the line of site between the drone's current position, and a newly detected frontier also implies that the frontier is likely reachable. If it was not reachable the LiDAR sensor would have detected the obstacle standing in the way and the frontier would not exist.

4.2 Graph Based Mapping and Path Planning

The backbone of our path planning strategy is a growing graph $G = (V, E)$ where V is a set of vertices (or nodes) with an associated coordinate to represent their location in the configuration space. Vertices are also labelled as either a regular node or frontier node. If

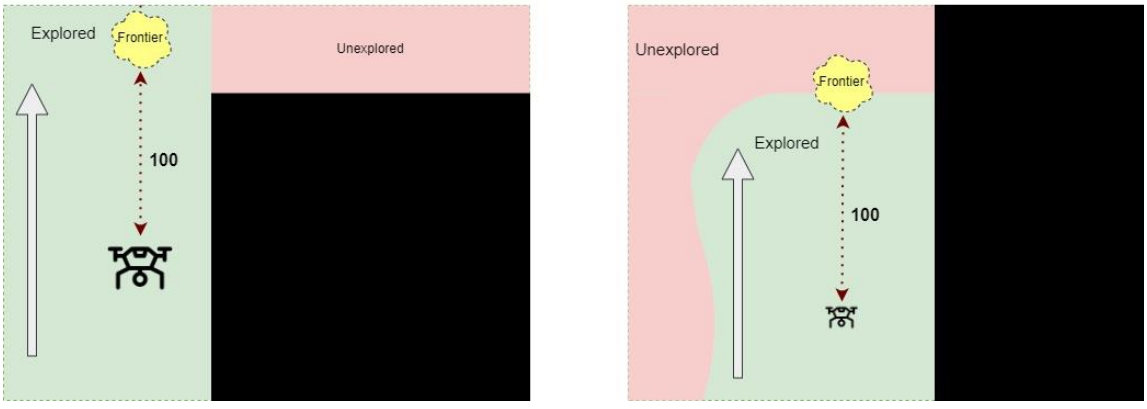

Figure 14 Examples of frontiers found by the second part of the proposed algorithm where open space is encountered. In both images the sensor reading is beyond the predefined max range and a new frontier is generated. This max range parameter controls the density

a vertex is a regular node, it is strictly used for navigation and has a degree greater than one unless it is the last node in a dead-end section of the exploration space. If the edge is a frontier node, it represents a region which must still be explored. The members of set E are edges which connect two vertices together. An edge informs of a free space path between two nodes. This free space is assumed to be a straight line. Initially G is empty but upon takeoff, it creates a node at its starting position and new nodes are added as new frontiers are detected. An example of this map is shown in Figure 15.

While flying, the state estimator of the UAV is used to determine the location of nodes and the drone's position relative to them. Paths are planned as a series of edges through the graph connecting the drone's current location to a frontier. When planning is done, the agent will necessarily be at the location of a regular node. This node is used as the root of a breadth-First search (BFS) of G to determine the lowest cost path to a frontier node. Cost is defined as the sum of the Euclidian distances between nodes along a path. This serves two purposes. First, the closest frontiers will be the quickest to find in the graph making searching quicker. Second, this encourages the drone to finish exploring an area in its immediate surroundings before having to backtrack to a different frontier region preventing unnecessary power consumption from going back and forth between regions. This graph has the added benefit of always maintaining a way to get back to the starting position. When no frontiers are left to explore, a search through the graph for the starting node is carried out, thus providing a collision free route back to a user.

To store the graph, an adjacency list is used. This list can grow along with the size of the space while minimizing the memory requirements. Storing the list requires $O(V + E)$ where V is the number of nodes and E is the number of edges. Finding a path in the graph is also $O(V + E)$, in a case where no path can be found. This indicates that the reachable areas have been explored and a path home can be generated.

When the frontier detection module finds a potential frontier, it is responsible for converting the frontier from the body frame to the world frame and providing the coordinates to the mapping module. With these coordinates we introduce two new nodes to the map. First, a regular node is inserted at the drone's current location. Since the drone

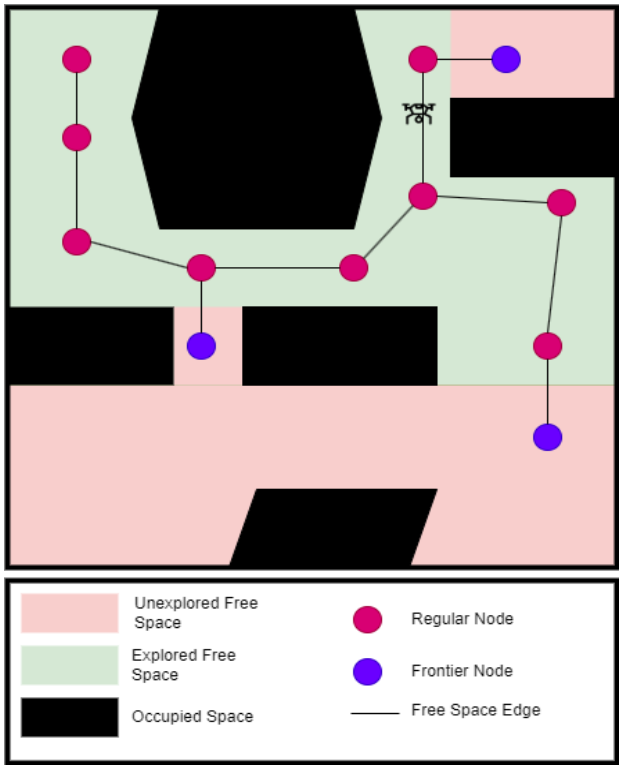

Figure 15 Graph based topological map used to find paths to frontiers in the exploration space. Each node is connected by edges that represent free space.

will most likely be travelling along an edge, this edge is split into two and each connects to the newly generated node. From this regular node, another new edge is introduced extending to a new frontier node created at the coordinates that were passed from the frontier detection algorithm. This process is shown in Figure 16 and demonstrates the update of the map being carried out for one frontier.

4.3 Graph Pruning

During each iteration of the outermost exploration loop, the graph is also maintained to optimize future paths by eliminating unnecessary nodes. This is done by checking the positions of the closet nodes and determining if they are no longer needed. This can occur if the drone has explored an area featuring a loop, or if two paths in the exploration space combine into one. In Figure 17 we see an example of this occurring where the drone has explored part of the space, but where redundant frontiers now exist. In this figure, node 5

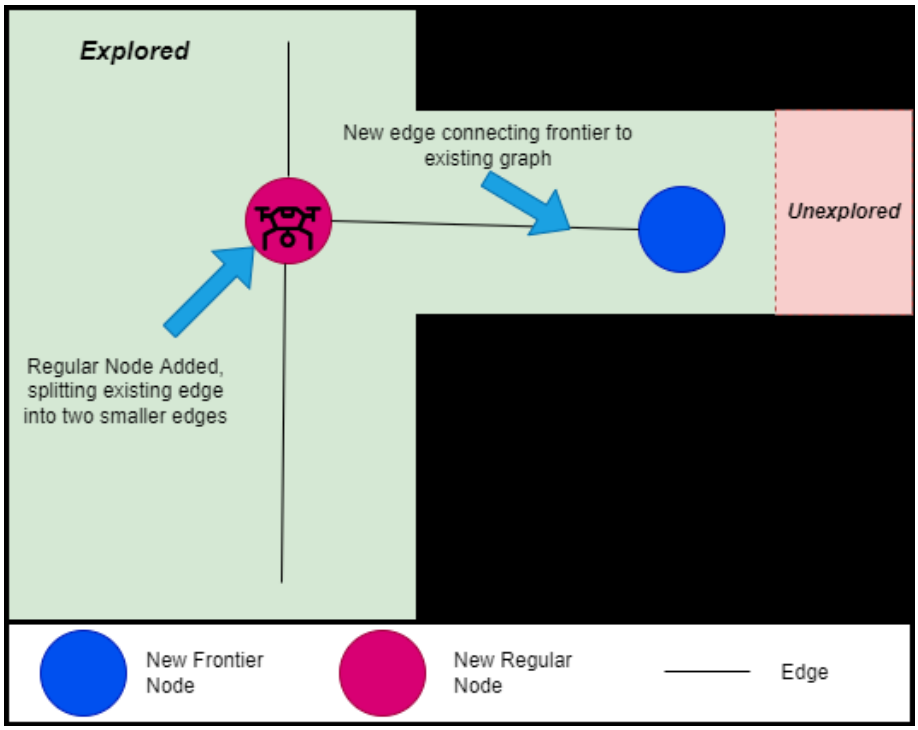

Figure 16 Map update process where the coordinates of the new frontier are generated by the frontier detection module. A new regular node, frontier node, three new edges are needed for this process.

is a redundant frontier node and visiting this node will not contribute to overall exploration. Planning and executing a path to this node will consume resources that could otherwise be used to continue exploring. To avoid these circumstances, at each iteration we check for redundant frontier nodes and use the following pruning scheme.

As the UAV is traveling to a destination frontier node, it may pass by a nearby node. By consulting the graph for other nodes in the proximity, a pruning technique is used to eliminate potentially redundant frontiers that would be of no use to exploration. To remove a node, it must meet the following criteria. First it must be within a set distance of the drone. This is a parameter which can be changed as desired. Next, the node must come within a sensor's FOV. This assures that the node is not behind a wall where more exploration space could exist. Third, the node must have a degree of one and must only be connected to the graph by one edge. If this is not the case, eliminating the node could result in a subgraph being disconnected from the regular graph. This connection may be useful for navigating to another frontier or traveling back to the starting point.

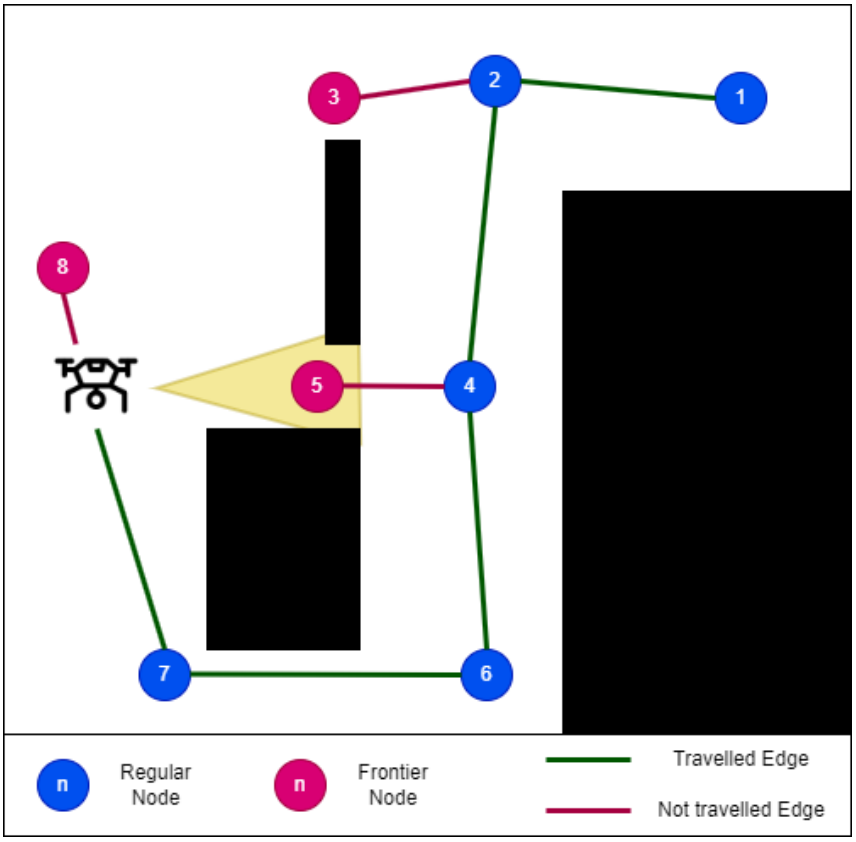

Figure 17 A case where pruning can help to optimize exploration time at no cost to overall exploration. Node 5 is redundant and provides no additional information about the space. It is pruned as the UAV travels from node 7 to 8.

Removing nodes comes at no cost to exploration. If a node is removed because it is too close, but would have still been useful for exploration, then it will immediately be replaced by another frontier. This is done by the frontier detection algorithm which will detect this area as a frontier region and create a new frontier that is placed further away from the drone. This can improve overall exploration efficiency. One such example is seen in Figure 18 where a frontier is removed after falling within a sensors FOV. It was replaced into a section of unexplored space as a new frontier.

4.4 Obstacle Avoidance

On top of the planned trajectories, an extra layer of obstacle avoidance is added. Though the graph can provide the connections between spaces, this alone is not enough to provide

Figure 18 Graph pruning. A node is deleted but then added back in a more optimal location.

collision-free travel between nodes for the following reasons. First, the collision box of the drone is not considered in the generation of nodes. A straight line between two nodes may come too close to an obstacle with which the UAV may collide. Second, without the advantages of an external positioning tool like GNSS or UWB sensors, the uncertainty of the localization increases. By introducing on the fly obstacle avoidance, the error in localization estimates from the state estimator is less concerning if the drift in the estimate is not too large. Our solution is extendable in this regard as any sense and avoid methods described in earlier chapters can be used. In our implementation, we use two strategies. The first is to adjust a frontier to a safer position when an obstacle is seen near a node. This step only occurs before traversing the final edge of a trajectory. While positioned at the last vertex, and before reaching the frontier, the drone orients itself so that its heading is aligned with the edge of the graph. From this position, the UAV uses its forward-facing sensor to observe the area around the frontier with a sweeping motion. This maneuver informs the UAV of any obstructions close to the frontier. For example, when observing the frontier, if it is determined that an obstacle is too close on the left, the frontier is moved slightly to the right. This only happens if the observation maneuver reveals free and open space to the right. Moving the frontier in this way also has the effect of removing unnecessary path length. If a frontier is close to a wall, it will be relocated away from it. As we often observed in experimentation, this meant making zigzag trajectories much straighter. This maneuver is not done when travelling to regular nodes. Changing their location could disrupt a future

path to another frontier that depends on that node. An example of this is shown in Figure 19 where the drone is at node 1 and travelling to the frontier node 2. Here, two potential trajectories are shown where the brighter one uses the obstacle avoidance technique, and the lighter one does not. The bright trajectory had a straighter path through the space and is therefore more optimal. This path was created as the original frontier was adjusted and moved away from the wall. In this instance, there was not enough space on the left to create another open space frontier. For this reason, the only frontier created was in the open space ahead of the drone at node 3. On the other hand, the lighter node 2 shows what would happen if the adjustment was not made. Now the UAV has much more open space to the left of the node and so a frontier is generated. The drone would travel to this frontier, without adjusting it, and the same process would occur (this time with node 4). This ultimately leads to a back-and-forth trajectory which increases the overall exploration time.

The second step in obstacle avoidance is to maintain a safety radius around the UAV using all available onboard sensors. If an object is detected within this safety radius, the drone responds by moving in the opposite direction. The sensor readings in all directions are considered when doing this maneuver so a movement away from one obstacle does not risk

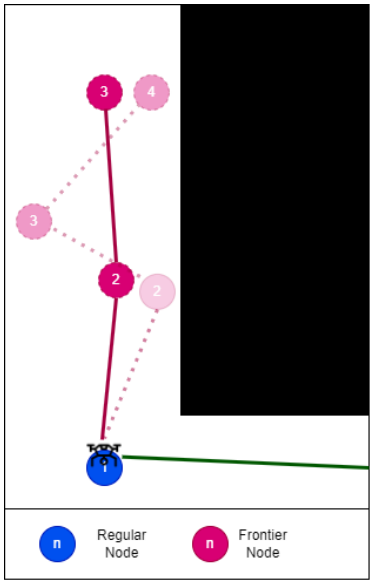

Figure 19 An example of the frontier adjustment technique used to keep the UAV at a safe distance from obstacles when travelling down a narrow corridor. The trajectories are shown when this technique is used vs when it is absent.

a collision with another one. For example, an obstacle is within the radius on the left but moving to the right would cause the drone to get too close to an obstacle on the right. In this case, both obstacles are considered, and the UAV is centered between them. If an object in front of the drone is detected to be within this radius, the drone cannot get any closer to its current goal and so it must stop. The node it was attempting to travel to will be moved to the UAVs current position.

4.5 Exploration Loop

By combining the above components our exploration method runs a continuous loop of frontier detection, mapping, obstacle avoidance, and when needed, path planning. This loop is shown in Figure 20 and is described in this section. The lower levels of the system such as state estimation and control are not discussed further in this section. Instead, the focus remains on our exploration strategy which is platform independent and can be run on various UAV platforms.

At the beginning of each loop, the current sensor information from all available LiDAR sensors is retrieved and temporarily stored for access throughout the rest of the loop. The same applies for state estimation where the current coordinates of the drone are stored alongside the heading. Sensor readings from the previous iteration are also stored and accessible. This position information is in the world frame of reference and converted to the body frame when needed using a rotation matrix. In the first iteration after taking off, the sensor data from the previous iteration does not yet exist but the values are initialized to zero. After retrieving the current data, the frontier detection algorithm is run on four of six sensor readings excluding the upward and downward facing LiDAR sensors. The results of the frontier detection are then passed to the graph/map module which will update the topological map as needed. If no frontiers are present, or the candidate frontiers do not meet the criteria to be inserted into the map, no updates to the map are made. Immediately following this, the short-term obstacle avoidance technique informs of any collision risks. If no collisions are imminent, we skip to graph pruning. If an obstacle is too close to the drone, an obstacle avoidance maneuver is performed to move the drone to a safer position (while not moving so far that it is unable continue its current trajectory). If this occurs, the sensor readings and current location are updated for use during the rest of the loop. Now

that frontiers have been detected, and the drone is in a safe position, the graph is pruned to eliminate any redundant nodes nearby. This is done as per the process described in section 4.3. Next, the current destination node's position information is accessed. The destination node refers to the vertex in the graph at the end of the edge that the drone is presently traversing. In the case that the drone has just taken off, the goal node is created and set to the current position of the drone so that it is necessarily reached on the first iteration. If the drone has not yet reached the destination node, we move directly to the final step of the iteration. If the UAV has reached the location of the destination node, the node is checked to see if it was a frontier node or a regular node. If we have just reached a frontier node, the trajectory is complete, and the node is changed from a frontier node to a regular node to be used for further path planning in later iterations. Path planning is then done to locate a path through the graph to a new frontier. The next edge of the path is then retrieved and the node on the opposite end of the edge is set as the current destination node. As the final step, the sensor readings from this iteration overwrite the sensor readings from the past iteration. These will be needed for frontier detection in the next cycle of the loop. The loop then starts over and will continuously run until no more frontiers are available for exploration.

There are several events that can trigger the system to search for a path back to the starting node rather than a frontier. The first has been mentioned and occurs when all frontiers have been reached and no others exist in the map. In our implementation we also consider current battery capacity to make sure that there will be enough power to come back to the starting point. If the battery is too low, exploration stops, and a path is planned back to the starting node.

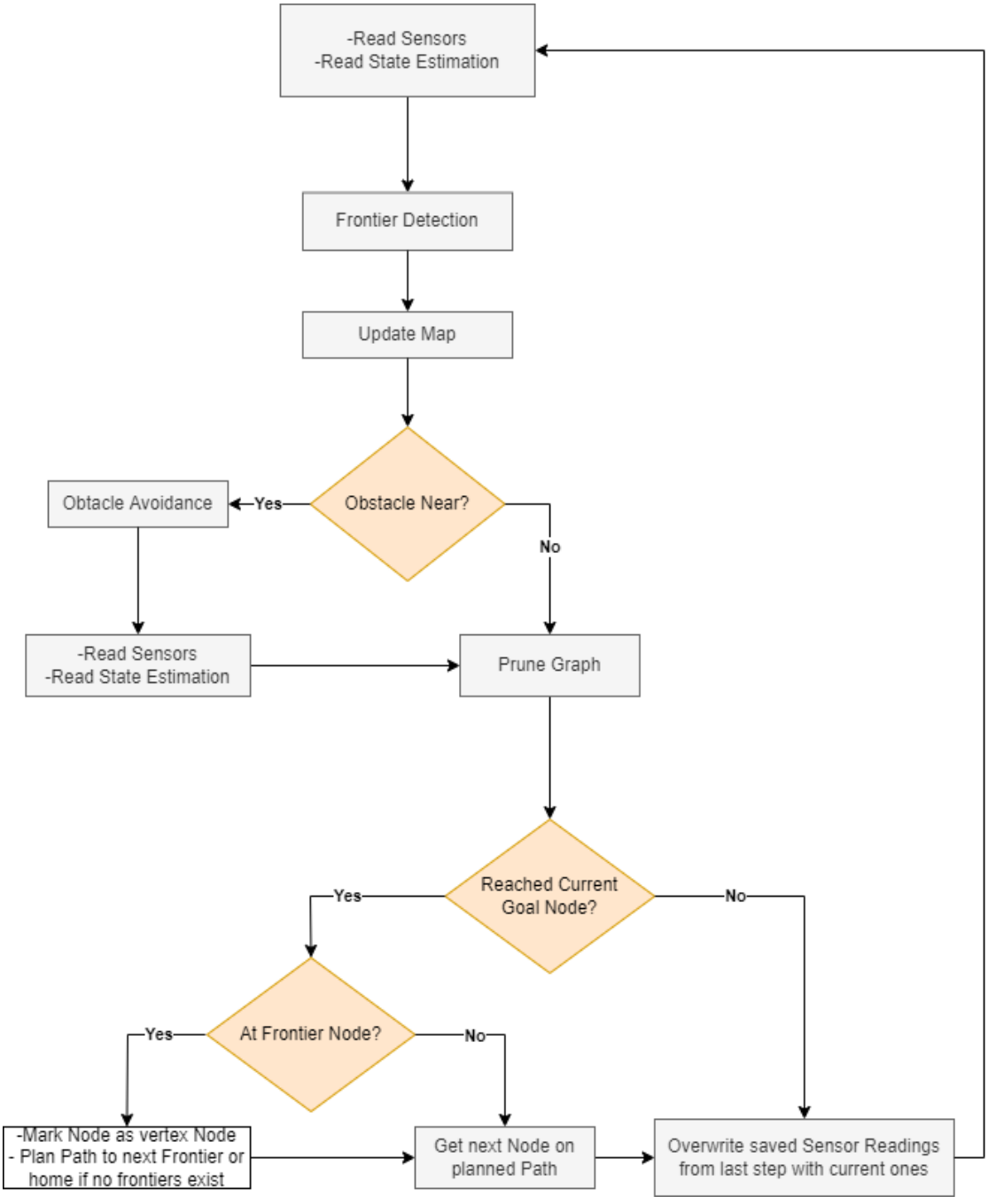

Figure 20 Outer loop of the exploration technique showing the timing and use of the various components of the solution

Chapter 5

5 Experimental Results and Analysis

To test our exploration strategy, we ran it as an application onboard a commercially available NAV equipped with a downward facing optical flow sensor and six LiDAR range sensors. Testing was performed on multiple real-world beds each set up in the Western Information and Networking Group (WING) lab. Three unique environments were assembled with various features and different obstacle types. In each case, the accessible space was explored as each region fell into a sensor's FOV at some point during exploration. In this section we present the platform, testing environments, and results of our experiments.

5.1 UAV Platform

Our selection of UAV had to meet certain criteria to properly deploy a prototype suitable for exploration using our novel planner. In selecting the platform on which to build, we considered the class of drone as well as availability for rapid prototyping. As our frontier detection method requires multi-direction perception, we required a solution that provided an array of sensors. As we are primarily concerned with the functionality of our exploration planner, we focused on finding a platform with established underlying components that were ready to use and exposed for interaction. This includes state estimation alongside a

Figure 21 Image of Crazyflie 2.1 with 6 mounted LIDAR sensors and a downward facing optical flow sensor

controller, commander, and a power distribution module. For these reasons, the CrazyFlie 2.1 nano quadcopter [62] was selected as the open-source platform meeting all our needs.

5.1.1 Hardware Environment

The CrazyFlie 2.1 is a 27-gram 92x92x29mm UAV with two onboard Micro Controller Units (MCUs), a STM32F405 for main applications and a nRF51822 for radio and power management. The STM32405 is 4 by 4.2 mm which provides suitable performance in a small formfactor. This MCU features a cortex-M processor with a built-in floating-point unit (FPU), integrated Digital Signal Processing (DSP) and Single Instruction/Multiple Data as well as multiply-accumulate instructions. At 168Mhz this processor achieves 210 Dhrystone Million Instructions per Second (DMIPS) and 566 CoreMark running from 1Mb of available flash memory and 192kb of Static RAM (SRAM). The nRF51822 MCU features a cortex-M0, at 32Mhz with 16kb of SRAM and 128kb of flash memory. Use of the 2.4Ghz radio was limited to firmware uploads prior to testing and not used to offload any navigational operations during flights.

Many sensors are available to use with the crazyflie platform, but due to the nature of our exploration task offboard sensors were not suitable for this work. Onboard, we mounted the BitCraze MultiRanger deck, providing 5 LiDAR sensors pointing to the front, back, left, right, and upwards. We also mounted the BitCraze Flowdeck v2, a downward facing optical flow sensor, with a LiDAR sensor. All LiDAR range sensors have can measure obstacles up to 4 meters with an accuracy of 4mm. The crazyflie also carries a three-axis accelerometer and gyroscope coupled with a pressure sensor in the IMU. The specifications of the drone are outlined in table 5 and an image of the drone can be seen in Figure 21.

5.1.2 Software Environment

The Crazyflie 2.1 platform provides open-source firmware with many of the necessary software modules to support flight. At the core of this software is the stabilizer loop featuring the following submodules: sensor data acquisition, state estimator, state controller, a commander, and power distribution. These, and all other processes, are managed through the real-time operating system (RTOS) FreeRTOS. Using this

Feature	Details
Weight	27grams
MCU 1	STM32F405 Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash
MCU 2	nRF51822 (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash)
IMU	-Three axis accelerometer / gyroscope (BMI088) -High precision pressure sensor (BMP388)
Flight time	7 Minutes per charge
Payload	15grams
Downward Optical Sensor	PMW3901 optical flow sensor
Range sensors x6	VL53L1x TOF sensor, 4000mm range, 4mm accuracy

Table 5 Properties of the Crazyflie 2.1 [62]

operating system, the software components are assigned as tasks with priority values in place to determine hardware resource control. The firmware exposes the data from these modules to an application layer, running as another tasks, by subscribing to logging data sent out from the underlying components. In this App layer, our exploration planner was implemented by sending/receiving data to/from the commander and state estimator, as well as consuming data provided by the tasks for the Multiranger and Flowdeck V2.

In the stabilizer loop, the data is first acquired from all available sensors discussed in the hardware environment section. The data collected is sent to the state estimation module where and extended Kalman filter is implemented and provides an estimate for the attitude (roll, pitch, yaw), position (x, y, z), and velocity (x, y, z). These values are available for reading in the app layer. The controller was implemented as a proportional integral derivative (PID) controller taking instructions from the commander coming in the form of either velocity or position setpoints. In the case of our prototype these setpoints originate in the app layer based on the decisions made by our implemented planner. Finally, the power distribution module takes the information from the controller and determines how each motor should respond to achieve the desired orientation [63].

5.1.3 Extra Modifications

In our implementation, we made some small changes to the firmware to cater to our specific application. For example, the battery is constantly monitored and once it falls below a certain capacity, the drone pre-empts any trajectory and creates a new one to come back to the starting position. A small change was made to the PID controller to force the drone to travel in a straight line when moving from one setpoint to another. This was done to ensure the drone stays along the straight edges connecting nodes in the graph. Since these edges represent a free space path the drone cannot stray from them too far without risking a collision. In our experiments we varied the frontier detection parameters to observe the impact on exploration and collision avoidance. As seen in the presented results, these parameters can be tuned to respond differently to different environmental circumstances or to compensate for uncertainty in localization. Parameters for graph pruning were also changed to observe the consequence of increasing or decreasing the number of nodes pruned.

5.2 Testing In a Combination of Environment Types

The first testing environment features a mixture of obstacle types constructed in a 700cm by 350cm section of the lab. It is largely centered around a cylindrical pipe structure with an opening at either end. This pipe is made from a large and flexible ducting hose which is traditionally used for construction ventilation. This product is both lightweight and collapsible making it suitable for testing inside a lab. The pipe is 100cms in diameter, 800cms in length, and features a 90-degree turn. A picture from inside the pipe is see in Figure 22. At either end of the pipe, boxes are used to challenge the drone’s navigation capabilities with obstacles and narrow passages. These sections of the test bed are seen in Figures 24 and 25. With these boxes, passages that are 70cms wide are created to force the drone to find paths in confined spaces. At the exit of the pipe a more open area exists to test frontier detection and exploration when nearby obstacles are not present. This also challenges the system to transition between multiple types of environments and not just a pipe or maze of boxes.

Figure 22 Pipe testing environment. The entrance to the pipe as seen from inside and outside of the pipe. The diameter of the pipe is 100cm and the length is 700cms. A 90-degree bend occurs 600cms into the Pipe

Using the combined environment test bed, three experiments were run to test our exploration planner. While testing the viability in terms of collision-free exploration of small spaces, we also wanted to gather some data on the response to changing parameters of the frontier detection, pruning, and obstacle avoidance. The details of the specific parameters used, their description, and their impact on exploration are summarized in Table 6. Several figures (25, 26, 27, 29, 30) within this chapter show the path and map of an experiment in further detail. These images are created using the position information, combined with the LiDAR readings of the drone. The LiDAR readings are converted to the world frame and then represented with a red dot in the left part of each figure. The green lines represent free space between the drone and the detected obstacle. Each node that is created by the frontier detection and graphing modules are numbered. Nodes that were later pruned are still included in the image but do not exist in the graph and are never travelled to. Pruned nodes can be recognized because the blue path line in these figures does not come close to them. Sudden breaks in the blue path line are due to obstacle avoidance maneuvers. The path continues after the maneuver has been completed.

Figure 24 Exit of Pipe featuring open space to test frontier detection and exploration in the absence of close obstacles.

Figure 23 The Beginning section of the combined test bed. On the left the space where the drone is deployed. The right image shows the entrance into the pipe.

In our first experiment on this testing environment, we use a wide obstacle avoidance radius with modest graph pruning and frontier detection thresholds. Using these parameters, the drone found a safe and efficient path through the entire course. With zero knowledge about the space before deployment, Figure 25 shows the path taken and frontier nodes used to explore the space. Notably, with the graph pruning parameter set to prune visible nodes that fall within 30cms of the drone location, we avoid back and forth movements within a region of the space. These images also show the scalability of our system as the entire testbed was explored using only 25 nodes. The map adapts to the shape of the environment and can extend in any direction while preserving the distance between nodes. The path

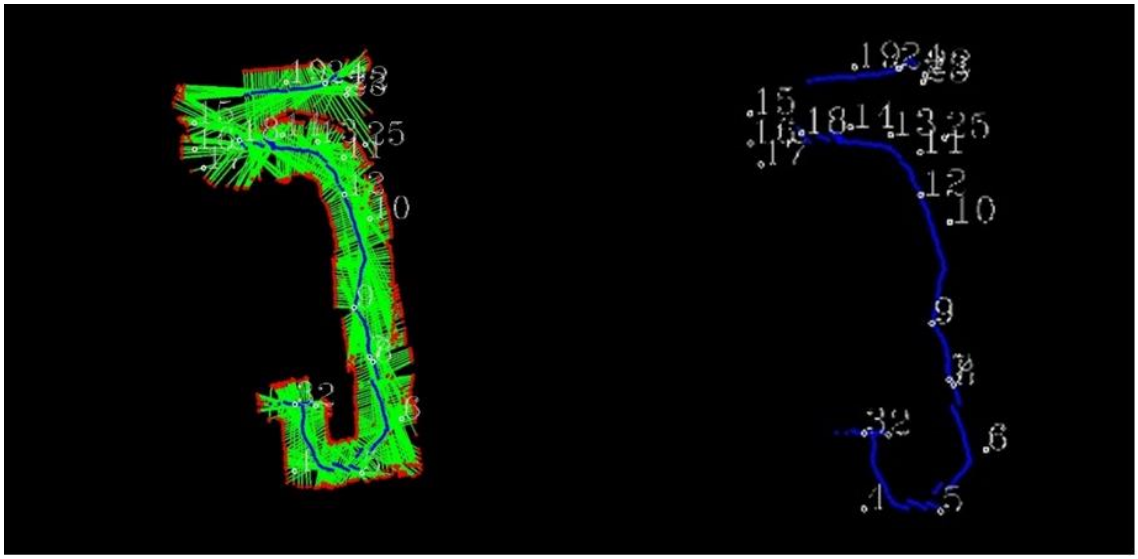

Figure 25 LiDAR readings (left) and path generated (right) for the first experiment on the combined

generated through this environment allowed the UAV to fully explore the region in under 1.5 minutes. The trajectories taken were optimal in terms of moving the UAV to previously unexplored space. In some cases, collision avoidance caused the UAV to repeatedly move away from a nearby obstacle. Overall, most trajectories were executed without these extra maneuvers.

In our second experiment, we generated more frontiers to explore the space. This was done by decreasing the frontier detection threshold and the number of nodes created in open space. Nodes could also be closer together and still be accepted as frontiers by the graphing module. To eliminate redundancy, we also increased the graph pruning in each iteration by pruning any redundant node within 500mm of the UAV instead of the regular 300mm. As a result, the number of nodes generated increased from 25 to 32. During this experiment, some unnecessary nodes were generated, and pruning was not able to prevent the system from travelling back and forth in a region that was already explored. This is seen in Figure 26 where the drone backtracked from node 21 to 20 near the turn in the pipe. This also occurred before entering the pipe at nodes 8 and 9. Overall exploration was incomplete in this image due to a large state estimation variance triggering the underlying state estimation system to reset. This exposes an area for improvement which is addressed in chapter 6. In any case, this test also shows the cost of backtracking in terms of overall exploration time.

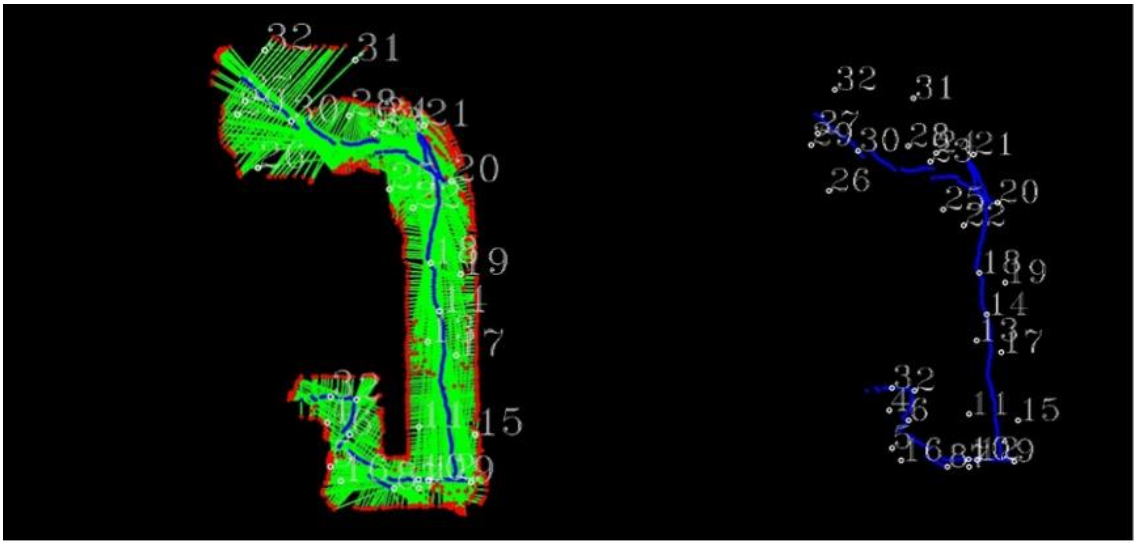

Figure 26 Lidar readings(right) and path generated (left for the second experiment on the combined

The extra backtracking resulted in nearly 20 seconds of unnecessary planning and path execution. This highlighted the need for proper parameters to be set when deploying this system such that unnecessary nodes are not created due to a small Max Range parameter setting.

In our third experiment on testbed, we tuned the parameters to use less obstacle avoidance and enabled the drone to come back to its starting position after exploration was complete. Figure 27 shows the exploration path going through the environment and back again. While the drone did avoid major collisions, it came too close to obstacles and the propellers brushed up against obstacles in several places around the exit of the pipe. These can be seen circled in Figure 27. Looking at the path generated in this figure, we see straighter paths through the long sections of the tunnel without extra maneuvers being used to stay further away from obstacles. While the overall exploration was equivalent to that of the first experiment, by reducing the safety radius the UAV was able to get closer to obstacles and could potentially enter smaller spaces. In its current state, reducing the safety radius this far cannot guarantee collision free flight and so it is safer to keep this parameter to a higher value. We believe that with a more robust state estimation that has less uncertainty, the safety radius parameter could be reduced further.

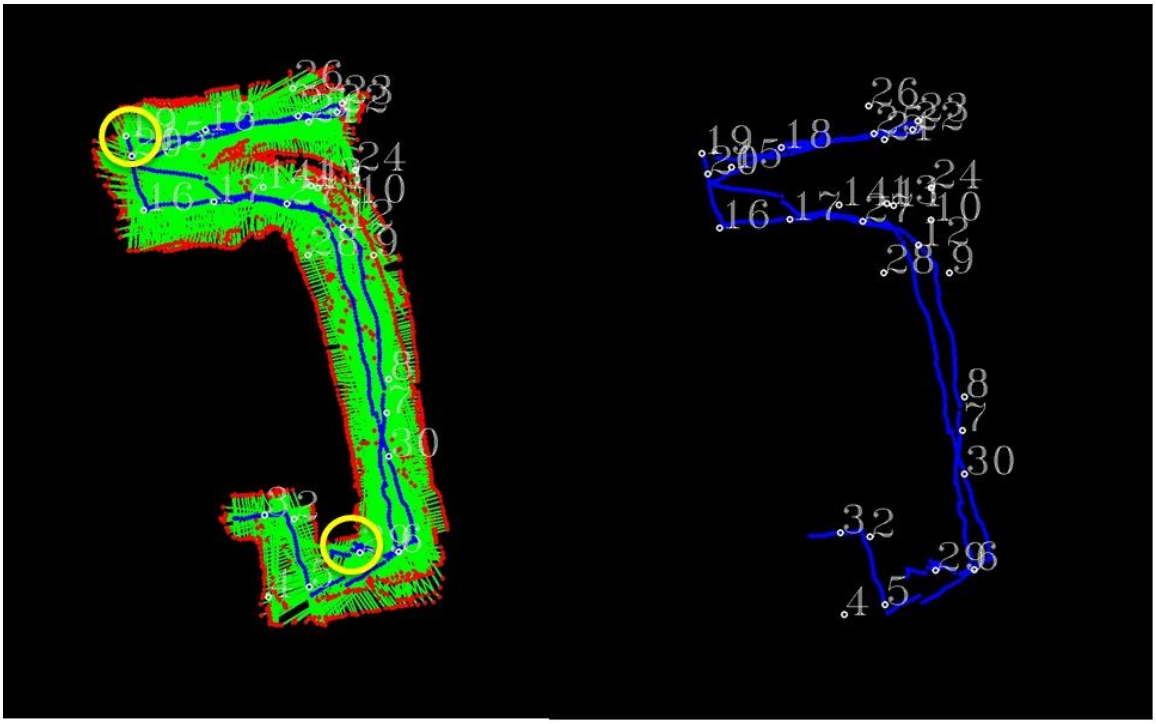

Figure 27 The path and lidar readings from the third experiment on the combined testing environment. In this experiment the drone travelled back to its starting position.

5.3 Looping Testbed with Narrow Passages

In our next test we arranged dozens of boxes into a large loop structure. Aside from the challenge of completing a loop, narrow passages with an opening as small as 50cms are included along with multiple corners and obstacles coming away from the wall. The entire testbed is 400cm by 300cm. As with the combined environment tests, our prototype successfully explored all accessible spaces while avoiding collisions. This test proves the ability of our exploration technique to adapt to very small, cluttered spaces where it can find a safe trajectory and return with a map of the free space in the environment. Figure 28 shows two views from within the testing environment.

Parameters for testing in this environment were set to allow the drone to move through the tight spaces. To do this we adjusted the obstacle avoidance radius to 25cms while setting graph pruning and frontier detection parameters to values that were most effective in the

Figure 28 view from inside loop test bed including narrow 50cm corridor

previous experiments. The pruning and frontier detection thresholds were set to 30cms. The minimum distance that a new frontier could be from an existing one was set to 60cms. If a LiDAR reading was greater than 100cms, a free space frontier was detected. The result of this experiment can be seen in Figure 29. Some backtracking was present around nodes 16 to 19 as the path overlapped itself. In the narrowest portion of the environment, between nodes 16 and 9, the obstacle avoidance caused the drone to continuously recenter itself between obstacles resulting in a back-and-forth saw-like pattern. While costly in terms of time, this technique prevented the drone from colliding or brushing up against any

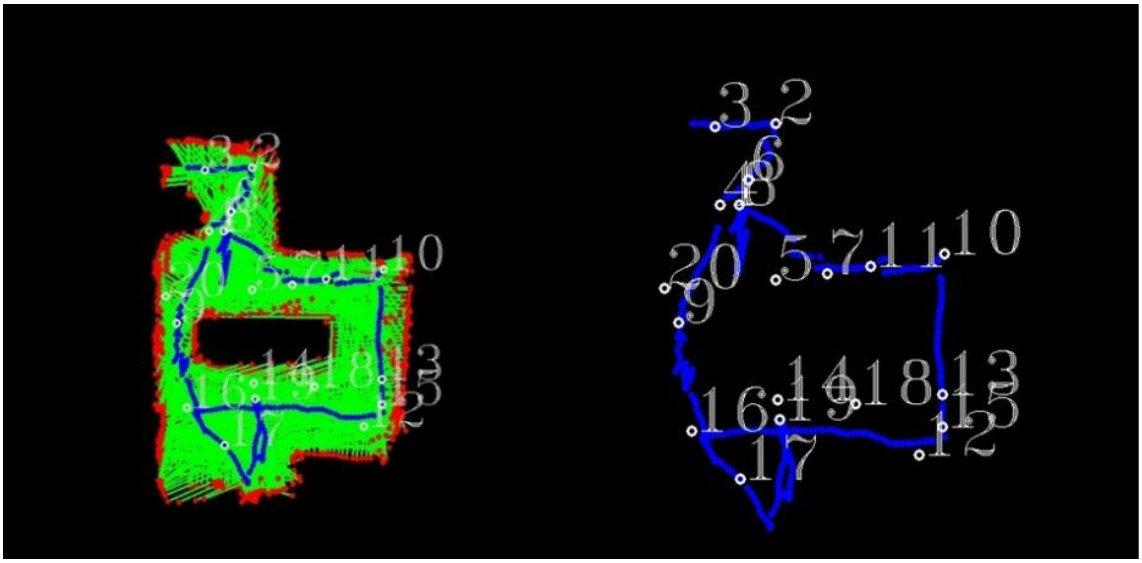

Figure 29 The path and lidar readings from the second testing environment. This testbed features very narrow passages and a large loop.

obstacles. With the smaller safety radius being used, the system did not have to reject traveling to any frontier for safety reasons. Much of the space is explored with straight, near optimal paths resulting in an overall exploration time of 176 seconds.

5.4 Indoor Hallway Testing

In our final experiment we tested the performance of our prototype in a hallway environment. This test was done in a 650cm by 300cm space featuring a long central hall with a smaller, 75cm wide hallway branching off. At the end of the main hallway there is a doorway to a small room which is also reachable. Additional obstacles include garbage and recycling bins, and areas where the hallway narrows. The exploration of this space is seen in Figure 30. A path from the starting position to each section of the environment was found before the drone returned to the starting position. Some redundancy is seen but overall, the entire area was explored using just 18 nodes while all obstacles were successfully avoided. In Figure 30 there is some evidence of state estimation drift causing some obstacles to appear as though they have a double wall in the figure. This occurs because the state estimation is becoming more inaccurate over time. When the drone passes an obstacle for the first time it uses the state estimation combined with the LiDAR sensor information to determine where the obstacle is. The drone continues exploring but

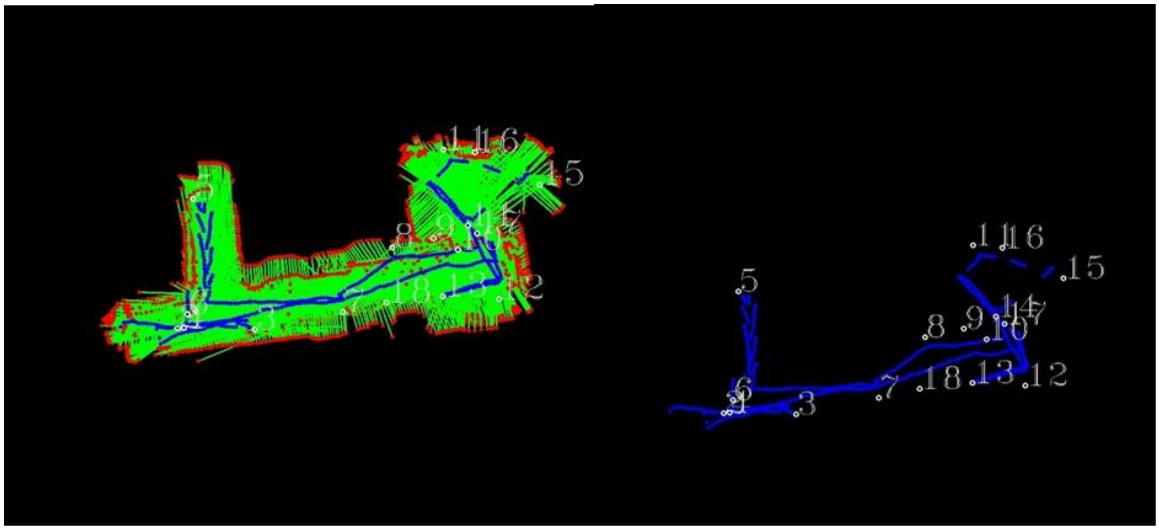

Figure 30 The lidar and trajectory record of the third test space. Green shows open space, red shows a lidar reading, and the blue line shows the trajectory through the course

eventually returns to the same point. This time the state estimation has accumulated error and so the calculation of the obstacles position is affected.

5.5 Summary of Results

The following table shows a summary of all tests presented in this section. Along the top of the table are parameter titles for the experiment. Obstacle Avoidance Radius (OAR) refers to the required open space around the drone in every direction. If an obstacle is closer than this distance, an avoidance maneuver is performed. For frontier detection parameters, the threshold represents the amount of change needed between two subsequent LiDAR readings to be considered an edge of an obstacle. The distance from frontiers is the value used to determine if a newly found frontier is too close to an already existing frontier. If this is the case, the frontier is not included in the map. The Max Range parameter is used to find open space frontiers. It determines how much open space can exist before having to add a new frontier. If a LiDAR reading is greater than this value, a frontier is created at the max range value in the direction of the LiDAR's heading. The pruning parameter is used to determine how close a node must be to the drone's current location for it to be pruned from the map. The node will have to fall into the sensors FOV and be within this range to be pruned. Further details for these parameters can be found in Chapter 4. Highlights for each test are included in the summary column. These testbeds support the viability of our exploration planner by running it in real world environments. With this design, the planner can be ported to any system which can provide data form the underlying modules and the ability to mount LiDAR sensors.

Test Bed	OAR	Frontier detection and Mapping:			Pruning	Time (seconds)	Summary
		Threshold	Distance from Frontiers	Max Range			
Combined	300	300	600	1000	300	129	-safe traversal with full exploration of the testing environment. - all redundant nodes pruned creating an efficient flight path through the environment
Combined	300	200	400	500	500	147	-more nodes created compared to other parameter settings - increased pruning alleviated much of the redundancy -some back tracking still exists
Combined	200	300	600	1000	300	284	- lower safety radius - enable feature to bring drone back to deployment location - unsafe flight in several places where drone brushed an obstacle
Box Loop	250	300	600	1000	300	176	- very tight corridor (50cms wide) was safely explored - full loop explored with no collisions
Hallway	250	300	600	1000	300	324	- hallway fully explored and drone came back to starting position - tight 50cm die corridor explored - extra obstacles included garbage\recycling bins and doorways

Table 6 Summary of experiments in various testing environments using the specified parameters for the presented exploration prototype

Chapter 6

6 Conclusion

Remote data collection and exploration provide economic opportunities, health and safety benefits, and is being used to further research across many fields. To extend all these uses, UAVs will need to be used in more restrictive environments helping to gather information in areas that are not accessible at present. Solutions to help break through the current barriers facing autonomous exploration will allow us to reach these areas. In this thesis, we have presented our design and testing of a novel approach to help drones reach these inaccessible spaces.

6.1 Limitations and Future Contributions

We have demonstrated that this solution has many applications today, but we feel that future work will be able to extend these possibilities. First, with a focus on specific data collection applications which can be run onboard this system, the sensors required to do other types of data collection could be optimized for use in frontier detection. While our system does assure that the heading of the drone will be in the direction of exploration, we have not explored the use of sensors other than LiDAR ranger finders and small cameras. Collecting data from other sensors may require unique compression strategies to store this data such that it can fit in onboard storage but also such that processing does not impact other system modules.

One of the most desirable future contributions is to improve and extend the state estimation used in the prototype. There are several reasons for this. First, as outlined in chapter 2, NAVs operating in confined spaces are impacted by the turbulence from propellors reflected of nearby structures. If this can be counteracted through a more robust flight controller and state estimator, the collision safety radius can be reduced. As seen in the first experiment this will increase performance of the system allowing it to reach even smaller spaces. Second, the state estimation used in our prototype has a large dependency on the optical flow sensor which requires light as well as detectable variations on the ground to determine the UAVs direction of motion. What this means is that dark areas, or those where

the floor is all the same colour, cannot provide high quality motion data. While on board lighting is available, we feel that another supplementary motion estimate such as the one presented in [57] would help to explore dark and smooth floored environments. We also see an opportunity to reduce any drift in the state estimation, by leveraging a loop closure technique. This technique could operate by associating features in environment with a node. This would allow the robot to check for the features (and their relative position) again on subsequent visits to the node. Position could be corrected by using the drone's position relative to the feature and fused with the other state estimation data. With better localization the collision avoidance, radius can be decreased further as there is less uncertainty as to how close the drone is to an obstacle.

The presented solution has been tested indoors operating in mostly 2D testbeds. However, this work is intended to scale to more complex 3D environments and welcomes use in environments where external positioning information is available. In future work both areas should be tested, specifically the exploration of indoor 3D environments can be tested by perceiving the environment in more directions to detect frontiers above and below the UAV. Fortunately, the addition of these sensors will be easily integrated into the frontier detection algorithm which has no directional requirement. To bring this solution outdoors, the use of GPS or other external positioning could be added in place of the optical flow sensor to provide a better state estimate even in the presence of environmental conditions like limited light or high winds.

6.2 Closing Remarks

In this thesis, we investigated autonomous drones for the purpose of exploration in constraining environments. The behaviors of different drone configurations, environmental restrictions, localization, mapping, path planning, and exploration were studied as the building blocks of developing a fully autonomous system. This background allowed us to identify the areas of autonomous flight that must be improved to push drones into smaller spaces without support from external devices. After recognizing the constraints of such a system, we developed a highly scalable exploration planner with no dependence on powerful hardware or outside communication. This solution leverages a lightweight topological mapping technique and discovers exploration frontiers on the fly. Our system

can navigate through a previously unexplored space with an extra layer of sense-and-avoid obstacle avoidance. We tested this solution by prototyping our algorithms onboard a NAV in several real-world environments. Our results have proven that exploration can be accomplished in very confined spaces while still having the resources to run extra tasks on board. We are excited to continue designing, developing, and testing our work to help open the door to even more use cases.

Bibliography

[1] “Unmanned Aircraft Systems - Federal Aviation Administration,” FAA Aerospace Forecast Fiscal Years 2021–2041. [Online]. Available: https://www.faa.gov/DATA_RESEARCH/AVIATION/AEROSPACE_FORECASTS/MEDIA/UNMANNED_AIRCRAFT_SYSTEMS.PDF. [Accessed: 10-Jun-2021].

[2] H. Shakhatreh et al., "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," in IEEE Access, vol. 7, pp. 48572-48634, 2019, doi: 10.1109/ACCESS.2019.2909530.

[3] J. Wilke, “A drone program taking flight,” The Amazon blog: dayone [Online]. Available: https://blog.aboutamazon.com/transportation/a-drone-program-taking-flight

[4] Drones Going Postal a Summary of Postal Service Delivery Drone Trials, Dec. 2017, [online] Available: http://unmannedcargo.org/drones-going-postal-summary-postal-service-delivery-drone-trials/.

[5] N. Elmeseiry, N. Alshaer, and T. Ismail, “A detailed survey and future directions of Unmanned Aerial Vehicles (uavs) with potential applications,” Aerospace, vol. 8, no. 12, p. 363, 2021.

[6] M. Arjomandi, S. Agostino, M. Mammone, M. Nelson, και T. Zhou, ‘Classification of unmanned aerial vehicles’, Report for Mechanical Engineering class, University of Adelaide, Adelaide, Australia, σσ. 1–48, 2006.

[7] “Can drones make construction safer?,” Centers for Disease Control and Prevention. [Online]. Available: https://blogs.cdc.gov/niosh-science-blog/2017/10/23/drones-construction/. [Accessed: 14-Mar-2022].

[8] S. Choi and E. Kim, “Design and implementation of vision-based structural safety inspection system using small unmanned aircraft,” 2015 17th International Conference on Advanced Communication Technology (ICACT), 2015.

[9] “How UAV and drone technology is influencing mining operation,” Drone companies in India. [Online]. Available: https://www.equinoxsdrones.com/blog/how-uav-and-drone-technology-is-influencing-mining-operation. [Accessed: 27-Feb-2022].

[10] A. Abdilla, A. Richards and S. Burrow, "Power and endurance modelling of batterypowered rotorcraft," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 675-680

[11] S. J. Kim, G. J. Lim, and J. Cho, “Drone flight scheduling under uncertainty on battery duration and air temperature,” Computers & Industrial Engineering, vol. 117, pp. 291–302, 2018,

[12] M. Hassanalian and A. Abdelkefi, “Classifications, applications, and design challenges of drones: A Review,” Progress in Aerospace Sciences, vol. 91, pp. 99–131, 2017.

[13_70] S. Norouzi Ghazbi, Y. Aghli, M. Alimohammadi, and A. Akbari, "Quadrotors unmanned aerial vehicles: A review." International Journal on Smart Sensing & Intelligent Systems, vol. 9, no. 1, 2016, pp. 309-333

[14_13] S. Nishanth, S. M. Dash and K. B. Lua, "A Numerical Study on the Influence of the Rear Wing Size on the Thrust Performance of the Two-Dimensional Tandem Flapping Wings," 2020 11th International Conference on Mechanical and Aerospace Engineering (ICMAE), 2020, pp. 77-81, doi: 10.1109/ICMAE50897.2020.9178898.

[15] Fenelon, Michael Angelo Amith and Tomonari Furukawa. “Design of an Active Flapping Wing Mechansim and a Micro Aerial Vehicle using a single Rotary Actuator.” (2007).

[16] N. Haider, A. Shahzad, M. N. Mumtaz Qadri, and S. I. Ali Shah, “Recent progress in flapping wings for Micro Aerial Vehicle Applications,” Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, vol. 235, no. 2, pp. 245–264, 2020.

[17] A. Gibiansky, “[PDF] Quadcopter Dynamics Simulation and control introduction: Semantic scholar,” [PDF] Quadcopter Dynamics Simulation And Control Introduction | Semantic Scholar, 01-Jan-1970. [Online]. Available: https://www.semanticscholar.org/paper/Quadcopter-Dynamics-Simulation-And-Control/cc1e023e4ca9e97822c42dd13d9fb8a162502761. [Accessed: 30-Mar-2022].

[18] A. Gupta and X. Fernando, “Simultaneous localization and mapping (SLAM) and data fusion in unmanned aerial vehicles: Recent advances and challenges,” Drones, vol. 6, no. 4, p. 85, 2022.

[19] S. Dwivedi, J. Nygren, F. Munier, and F. Gunnarsson, “5G positioning: What you need to know,” erricson, 08-Dec-2020. .

[20] Wang Shule, Carmen Martínez Almansa, Jorge Peña Queralta, Zhuo Zou, Tomi Westerlund, UWB-Based Localization for Multi-UAV Systems and Collaborative Heterogeneous Multi-obot Systems, Procedia Computer Science, Volume 175, 2020, Pages 357-364, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.07.051.

[21] B. Lin, Z. Ghassemlooy, C. Lin, X. Tang, Y. Li and S. Zhang, "An indoor visible light positioning system based on optical camera communications", IEEE Photonics Technology Letters, vol. 29, no. 7, pp. 579-582, 2017.

[22] G. Niu, J. Zhang, S. Guo, M. -O. Pun and C. S. Chen, "UAV-Enabled 3D Indoor Positioning and Navigation Based on VLC," ICC 2021 - IEEE International Conference on Communications, 2021, pp. 1-6, doi: 10.1109/ICC42927.2021.9500633.

[23] 6. C. Eling, L. Klingbeil and H. Kuhlmann, "Real-time single-frequency GPS/MEMS-IMU attitude determination of lightweight UAVs", Sensors, vol. 15, no. 10, pp. 26212-26235, 2015.

[24] Y. Lu, Z. Xue, G.-S. Xia, and L. Zhang, “A survey on vision-based UAV navigation,” Geo-spatial Information Science, vol. 21, no. 1, pp. 21–32, 2018.

[25] B. D. Lucas and T. Kanade. An iterative image- registration technique with an application to stereo vision. Proceedings of the 1981 DARPA Imaging Understanding Workshop, pp. 121-130, 1981.

[26] Z. Gosiewski, J. Ciesluk and L. Ambroziak, "Vision-based obstacle avoidance for unmanned aerial vehicles," 2011 4th International Congress on Image and Signal Processing, 2011, pp. 2020-2025, doi: 10.1109/CISP.2011.6100621.

[27] S. Hayat, R. Jung, H. Hellwagner, C. Bettstetter, D. Emini and D. Schnieders, "Edge Computing in 5G for Drone Navigation: What to Offload?," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 2571-2578, April 2021, doi: 10.1109/LRA.2021.3062319.

[28] R. E. Kalman, “A new approach to linear filtering and prediction problems,” Journal of Basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[29] S. J. Julier and J. K. Uhlmann, “New extension of the Kalman filter to Nonlinear Systems,” SPIE Proceedings, 1997.

[30] N. P. Santos, V. Lobo and A. Bernardino, "Unmanned Aerial Vehicle Tracking Using a Particle Filter Based Approach," 2019 IEEE Underwater Technology (UT), 2019, pp. 1-10, doi: 10.1109/UT.2019.8734465.

[31] T. Ozaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, “Autonomous Navigation and mapping for inspection of penstocks and tunnels with mavs,” IEEE Robotics and Automation Letters, vol. 2, no. 3, pp. 1740–1747, 2017.

[32] A. A. Zhilenkov and I. R. Epifantsev, "System of autonomous navigation of the drone in difficult conditions of the forest trails," 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 1036-1039, doi: 10.1109/EIConRus.2018.8317266.

[33] H. Kinjo, M. Morita, S. Sato, T. Suriyon and T. Anezaki, "Infrastructure (transmission line) check autonomous flight drone (1)," 2017 International Conference on

Intelligent Informatics and Biomedical Sciences (ICIIBMS), 2017, pp. 206-209, doi: 10.1109/ICIIBMS.2017.8279694.

[34] A. Fabris, S. Kirchgeorg and S. Mintchev, "A Soft Drone with Multi-modal Mobility for the Exploration of Confined Spaces," 2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2021, pp. 48-54, doi: 10.1109/SSRR53300.2021.9597683.

[35] J. N. Yasin, S. A. S. Mohamed, M. Haghbayan, J. Heikkonen, H. Tenhunen and J. Plosila, "Unmanned Aerial Vehicles (UAVs): Collision Avoidance Systems and Approaches," in IEEE Access, vol. 8, pp. 105139-105155, 2020, doi: 10.1109/ACCESS.2020.3000064.

[36] STMicroElectronics “A new generation, long distance ranging Time-of-Flight sensor based on ST’s FlightSense™ technology” DocID031281 Feb. 2018 [Revised April 2022]

[37] K. Aizawa and S. Tanaka, "A Constant-Time Algorithm for Finding Neighbors in Quadtrees," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 7, pp. 1178-1183, July 2009, doi: 10.1109/TPAMI.2008.145.

[38] K. Jiang, L.D. Seneviratne and S.W.E. Earles, “A Shortest Path Based Path Planning Algorithm for Nonholonomic Mobile Robots,” Journal of Intelligent and Robotic Systems, vol. 24, 1999, pp. 347–366

[39] Z. Ahmad, F. Ullah, C. Tran, and S. Lee, “Efficient Energy Flight Path Planning Algorithm Using 3-D Visibility Roadmap for Small Unmanned Aerial Vehicle,” International Journal of Aerospace Engineering, vol. 2017, 2017, pp. 1-13

[40] G. Frontera, D.J. Martín, J.A. Besada, . et al., “Approximate 3D Euclidean Shortest Paths for Unmanned Aircraft in Urban Environments,” Journal of Intelligent & Robotic Systems, vol. 85, 2017, pp. 353–368

[41] Z. Lin, L. Castano and H. Xu, "A Fast Obstacle Collision Avoidance Algorithm for Fixed Wing UAS," 2018 International Conference on Unmanned Aircraft Systems (ICUAS), 2018, pp. 559-568, doi: 10.1109/ICUAS.2018.8453307.

[42] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” Computer Science Department, Iowa State University, Ames, IA, TR 98-11, Tech. Rep., 1998

[43] L. E. Kavraki, P. Svestka, J. -. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566-580, Aug. 1996, doi: 10.1109/70.508439.

[44] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning", Int J Rob Res, vol. 30, pp. 846-894, 2011.

[45] Chen, Yong-Bo, et al. “UAV Path Planning Using Artificial Potential Field Method Updated by Optimal Control Theory,” International Journal of Systems Science, vol. 47, no. 6, 2014, pp. 1407–1420

[46] A. Ma’Arif, W. Rahmaniar, M. A. M. Vera, A. A. Nuryono, R. Majdoubi and A. Çakan, "Artificial Potential Field Algorithm for Obstacle Avoidance in UAV Quadrotor for Dynamic Environment," 2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT), 2021, pp. 184-189, doi: 10.1109/COMNETSAT53002.2021.9530803.

[47] M. Zohaib, S. M. Pasha, N. Javaid and J. Iqbal, "IBA: Intelligent bug algorithm—A novel strategy to navigate mobile robots autonomously", Proc. Int. Multi Topic Conf., vol. 414, pp. 291-299, 2013.

[48] B. Yamauchi, "A frontier-based approach for autonomous exploration," Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation', 1997, pp. 146-151, doi: 10.1109/CIRA.1997.613851.

[49] L. Zhao, L. Yan, X. Hu, J. Yuan, and Z. Liu, “Efficient and high path quality autonomous exploration and trajectory planning of UAV in an unknown environment,” ISPRS International Journal of Geo-Information, vol. 10, no. 10, p. 631, 2021.

[50] C. Connolly et al., "The determination of next best views", Robotics and Automation. Proceedings. 1985 IEEE International Conference on, vol. 2, pp. 432-435, 1985.

[51] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova and R. Siegwart, "Receding Horizon "Next-Best-View" Planner for 3D Exploration," 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1462-1468, doi: 10.1109/ICRA.2016.7487281.

[52] L. Heng et al., "Autonomous visual mapping and exploration with a micro aerial vehicle", J. Field Robot., vol. 31, no. 4, pp. 654-675, 2014.

[53] 17.B. Charrow et al., "Information-theoretic planning with trajectory optimization for dense 3D mapping", Robot. Sci. Syst., vol. 11, pp. 3-12, 2015.

[54] Z. Xu, D. Deng and K. Shimada, "Autonomous UAV Exploration of Dynamic Environments Via Incremental Sampling and Probabilistic Roadmap," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 2729-2736, April 2021, doi: 10.1109/LRA.2021.3062008.

[55] M. Selin, M. Tiger, D. Duberg, F. Heintz and P. Jensfelt, "Efficient Autonomous Exploration Planning of Large-Scale 3-D Environments," in IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1699-1706, April 2019, doi: 10.1109/LRA.2019.2897343.

[56] H. H. González-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments", The International Journal of Robotics Research, vol. 21, no. 10–11, pp. 829-848, 2002.

[57] M. Petrlík, T. Krajník and M. Saska, "LiDAR-based Stabilization, Navigation and Localization for UAVs Operating in Dark Indoor Environments," 2021 International Conference on Unmanned Aircraft Systems (ICUAS), 2021, pp. 243-251, doi: 10.1109/ICUAS51884.2021.9476837.

[58] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova and R. Siegwart, "History-Aware Autonomous Exploration in Confined Environments Using MAVs," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1-9, doi: 10.1109/IROS.2018.8594502.

[59] “Inspection tunnel [GPS-denied] - ACSL: ACSL Ltd..,” ACSL. [Online]. Available: https://www.acsl.co.jp/en/solutions/inspection-tunnel-gps-denied/. [Accessed: 26-Apr-2022].

[60] F. SA, “Elios 2 - indoor drone for confined space inspections,” Elios 2 - Indoor drone for confined space inspections. [Online]. Available: https://www.flyability.com/elios-2. [Accessed: 26-Apr-2022].

[61] E. Technologies, “Exynaero Aerial Mapping Drone: Exyn Technologies,” Industrial Drone Technology. [Online]. Available: https://www.exyn.com/products/exyn-aero-aerial-mapping-drone. [Accessed: 26-Apr-2022].

[62] “Crazyflie 2.1,” Bitcraze. [Online]. Available: https://www.bitcraze.io/products/crazyflie-2-1/. [Accessed: 02-Feb-2022].

[63] “Stabilizer module,” Bitcraze. [Online]. Available: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/. [Accessed: 12-Jun-2021].

[64] “Scaneagle series - fixed-wing UAV by Insitu, Inc.: Directindustry,” The B2B marketplace for industrial equipment. [Online]. Available: https://www.directindustry.com/prod/insitu-inc/product-101665-939529.html. [Accessed: 26-Aug-2022].

[65] M. Waghorn, “Tiny drones that fly like insects have been built by scientists,” Metro, 03-Feb-2022. [Online]. Available: https://metro.co.uk/2022/02/03/tiny-drones-that-fly-like-insects-have-been-built-by-scientists-16040057/. [Accessed: 26-Aug-2022].

[66] “RRT path planning algorithm,” RRT path planning algorithm (MATLAB implementation) - programmer sought. [Online]. Available: https://www.programmersought.com/article/74233831887/. [Accessed: 26-Aug-2022].

[67] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger and P. Feyzeau, "Path planning: A 2013 survey," International Conference on Industrial Engineering and Systems Management (IESM), Rabat, 2013, pp. 1-8.

[69] “Transit map,” Data Viz Project, 30-Aug-2017. [Online]. Available: https://datavizproject.com/data-type/transit-map/. [Accessed: 12-Aug-2022].

Curriculum Vitae

Name:	Kirk Vander Ploeg
Post-secondary Education and Degrees:	The University of Western Ontario London, Ontario, Canada 2015-2019, B.Sc.
Honours and Awards:	Deans Honor List 2016, 2017, 2018, 2019
Related Work Experience	Programmer Analyst Social Science Technology Department**,** The University of Western Ontario 2022-Present Research Assistant The University of Western Ontario 2019-2022 Teaching Assistant The University of Western Ontario 2019-2022