University of the Witwatersrand

School of Computer Science and Applied Mathematics
Faculty of Science
Masters Research Report

High-Speed Obstacle Avoidance in Unstructured Environments

Ashton Naidoo
2519631

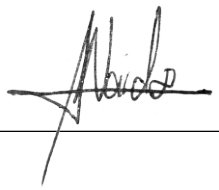
Supervisors:
Dr. Pravesh Ranchod
Prof. Benjamin Rosman

Abstract

Micro aerial vehicles (MAVs) have many applications in various environments, some of which include site surveying, mapping, search and rescue, inspection, delivery, and photography. Even though MAVs have an incredible amount of application space, most industry applications do not make use of autonomous flight. Instead, a human pilot flies the MAV, especially where navigation in complex environments is required. While autonomous navigation systems do exist in industry, the complexity of the environment can inhibit the system's ability to traverse the environment. High-speed autonomous navigation of MAVs, in unstructured environments has received more attention in recent years, which has resulted in new more robust algorithms which can navigate complex unstructured environments quickly and successfully. In this research report, a simulation environment is created in a photo-realistic simulation engine in which high-speed obstacle avoidance algorithms can be evaluated. This is done so that an assessment can be made regarding how well these algorithms generalise as the complexity of the environment increases. Obstacle avoidance research often overlooks the degree of environmental complexity when presenting performance metrics. As a result, the performance metrics for obstacle avoidance algorithms can be vague due to the lack of detailed metrics that adequately capture environmental complexity. To this end, an environmental complexity scoring framework for unstructured environments is proposed. Environments are divided into ten levels which increase in environmental complexity score. The experiment allows the MAV to move through the simulation environment at various target speeds. As the MAV moves through each level, evaluation metrics such as average speed and collisions are logged. Evaluation of these metrics indicates that if the environmental complexity is constant, speed increases cause collision avoidance performance to degrade. Conversely, as the environmental complexity increases, lower speeds are required for successful collision avoidance. It is further shown that object complexity has an impact on environmental complexity and subsequently successful obstacle avoidance. The greater the object complexity the more obstacle avoidance performance is reduced. The project repository can be found here. Results indicate that obstacle avoidance algorithms have maximum speed and environmental complexity threshold scores, below which successful obstacle avoidance can occur.

Declaration

I, Ashton Naidoo (Student Number: 2519631), hereby declare the contents of this research report to be my own work. This proposal is submitted for the degree of Masters of Science in Computer Science at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

October 8, 2024 Signature: 

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Micro Arial Vehicles (MAVs) have many applications in various environments, some of which include site surveying, mapping, search and rescue, inspection, delivery, and photography. MAVs are found in a few different forms which tend to classify the way flight is achieved. Multi-rotor type designs generally use the vertical thrust generated by propellers to achieve flight [Theys et al. 2016]. Fixed-wing MAVs use the horizontal thrust and drag created by air passing the wings to achieve flight. Ornithopters use wings that flap like birds to achieve flight [Thipyopas and Intaratep 2011]. Entomopters use wings that flap like insects to achieve flight [Ellington et al. 1996]. The first two categories are generally used in industry while the latter two are more experimental.

Even though MAVs have an incredible amount of application space, most industry applications do not make use of autonomous flight, instead, a human pilot flies the MAV, especially where navigation in complex environments is required[Floreano and Wood 2015]. Commercial-grade MAVs are expensive, and mistakes made by pilots during navigation can be costly in scenarios where MAVs get lost or damaged. So, care is taken to avoid navigation through dense environments, such as forest canopies, because this entails navigating through complex and unstructured environments [Ljungblad et al. 2021]. There are application spaces that make human piloting infeasible, for example, autonomous delivery MAVs. If many deliveries need to be made, many pilots would be needed, which would be costly due to the skill level required by pilots [Aurambout et al. 2019]. Autonomous navigation can help make autonomous delivery more feasible.

Autonomous navigation algorithms tend to fail or perform extremely slowly in these types of environments, which diminishes their industry utility and applicability. Autonomous navigation failures can in general be attributed to the lack of ability to adequately perceive, generate obstacle-free trajectories, and track these trajectories quickly enough to avoid collisions [Albaker and Rahim 2009]. Generating obstacle-free trajectories is a time-sensitive and computationally expensive task for MAV systems due to the speed at which MAVs move and the constrained computation capabilities. By reducing MAV speed less strain is placed on this sense and avoidance process which improves obstacle avoidance performance [J. Zhang, Chadha, et al. 2018].

Obstacle avoidance performance can be hindered in multiple ways such as complex environments, sensor suite capability, onboard compute availability, and the quality of autonomous navigation algorithm being implemented in the navigation pipeline [Kumar and Michael 2012]. Regardless, autonomous navigation in unstructured environments has been achieved, and the speed at which flight is achieved by these algorithms, in general, is below 5 m/s e.g.: [Aguilar et al. 2017]. This is slow in comparison to the maximum speed that MAVs can move at. Given the limited flight time of some MAVs, autonomous navigation in unstructured environments can become impractical, especially when return flight time must be considered. Take for example a search and rescue application inside a dense forest where heat signatures of a certain size need to be tagged to verify if the target being searched for, has been found or not. If an MAV could reliably traverse the forest faster, more area can be searched in a shorter duration, and the target can be found quicker. This is even more important if there is some form of urgency involved, e.g.: the target which needs to be found requires medical care.

While MAVs are incredibly versatile pieces of hardware, autonomous navigation in unstructured environments at greater speeds would widen and improve their application space to better handle edge case environments when required to do so. A robust high-speed autonomous navigation system would reduce application costs by removing the need for a skilled pilot and reducing the chances of lost or damaged MAVs while adding versatility to the MAV application space.

High-speed aerial autonomous navigation in unstructured environments has received more attention in recent years, which has resulted in new more robust algorithms which can navigate complex unstructured environments. These algorithms are created using different approaches, each attempting to address bottlenecks in the autonomous navigation pipeline. For autonomous navigation to occur, there are a few building blocks that are required. First, the environment in which the MAV is in needs to be sensed, and then the incoming data needs to be processed such that obstacle-free trajectories can be generated. Finally, motor thrusts must be controlled in order to closely track these trajectories.

Each of these elements in the autonomous navigation pipeline affects the speed at which the MAV can move in complex unstructured environments. Any attempts to optimise these elements need to take into consideration onboard MAV constraints. Only after addressing the obstacle avoidance problem can the actual MAV speed be increased. But to what extent?

In this report, a high-speed obstacle avoidance algorithm will be evaluated in a photo-realistic simulation engine, to assess how well it copes in unstructured environments, as the complexity of the environment varies. In order to stress test this algorithm, performance needs to be assessed in unstructured environments, where the complexity of the environment is made incrementally more complex. In stress testing the algorithm we attempt to find the extent to which high-speed obstacle avoidance and unstructured environmental complexity are correlated by examining specific components of the environment's complexity. By testing the algorithm in this way, the source of bottlenecks in obstacle avoidance can be found.

This report makes the following contributions:

- An extensible framework for defining environmental complexity relative to MAV navigation is provided.
- An obstacle avoidance algorithm called FALCO is stress-tested to validate this framework.
- Provides empirical evidence that enforces the assumptions that obstacle avoidance becomes more difficult as:
  - the number of objects in an environment increases.
  - MAV speed is increased.
- Highlights that the root cause of environmental complexity relative to MAV obstacle avoidance stems from object complexity.

Chapter 2 of this report gives background to MAVs and various aspects involved in high-speed autonomous navigation. Chapter 2.3 highlights some of the solutions to the various problems in MAV high-speed obstacle avoidance. Chapter 3 outlines the research methodology, discussing the aim of this report as well as how the evaluation of high-speed obstacle avoidance algorithms shall be conducted. Chapter 4 discusses the implementation of the simulation environment and obstacle avoidance algorithm. Chapter 5 explains the results of the experiments and describes the generated data in graphs. Chapter 6 discusses the results relative to the research questions. Chapter 7 describes possible future work and finally, Chapter 8 concludes with an overview of the research and the findings.

Chapter 2

Background and Related Work

2.1 Introduction

In this section, a few key concepts and problems relating to micro aerial vehicles (MAVs) are discussed to give background to the problem space. Thereafter aspects of the simulation environment used to test high-speed obstacle avoidance are discussed. Finally, the related works section discusses various approaches that address the high-speed obstacle avoidance problem.

2.2 Background

This section discusses some of the key challenges with autonomous navigation in MAVs. Some of these problems include sensor suite limitations and the computational cost involved in generating and processing depth estimation data crucial for autonomous navigation. Simulator requirements are discussed in relation to bridging the simulation to reality gap. Finally, environment classifications are discussed in relation to object and navigational complexity.

2.2.1 MAV Constraints

MAVs are small and are generally under 1m in length, as a result, they have limited payload capacity. Autonomous MAVs need multiple sensors to estimate the state of their environment, which in turn utilises more power and further reduces the payload capacity. Motors, sensors, and processors consume large amounts of power which depletes batteries quickly. Onboard processors running compute-heavy algorithms for planning and obstacle avoidance, further increase power consumption. These constraints are referred to as the size, weight, and power (SWaP) constraints and are proportional to one another in terms of flight time [Kumar and Michael 2012]. MAV size affects payload or weight-bearing capabilities, but an increase in size can mean a reduction in battery life, both of which can result in a reduction in application space. This means that there are limitations on battery size, sensor suite, compute, and overall power requirements. The payload must be a balance between battery size, sensor suite requirements and computational requirements. These constraints mean that MAVs generally have a short flight time, in the region of 20-30 minutes,

with frequent battery changes being a norm. This poses two problems, episodic type learning or online machine learning which may need millions of iterations to achieve usable outcomes become infeasible, and the more computation that is required the faster power is depleted which further reduces flight time.

2.2.2 How fast is "High Speed"?

High-speed obstacle avoidance is in part dependent on whether the actual MAV being used is capable of high-speed flight. Most research conducted on high-speed obstacle avoidance algorithms use either quad-rotors or fixed-wing crafts as seen in A. Barry et al. 2018 and P. Florence et al. 2020. While there are no defined ranges of high speed, A. Barry et al. 2018 classifies it as anything in the region of 10m/s or more. However it should be noted that this distinction is arbitrary if the MAV does not avoid all obstacles successfully during navigation. Therefore collision avoidance success rate is an important metric to consider.

2.2.3 GPS Denied

A common scenario for autonomous MAVs, is when they fly into environments where the propagation of GPS signal is hindered by dense vegetation or large overhead structures. The result is that the MAV cannot position itself accurately or at all in global space. The general approach for solving this issue is to build a map and use a global planner in combination with a local planner for localisation. The global planner takes the final goal state into consideration and the local planner is used during local path planning and obstacle avoidance, such an implementation can be seen in P. Florence et al. 2020.

2.2.4 MAV Sensor Suite

The MAV sensor suite can also limit high-speed autonomous navigation. Slow data acquisition times or complicated post-processing can increase the reaction time of a MAV, by slowing down the perception-action loop. On the other hand, without the correct sensor suite, high-speed obstacle avoidance may not be achievable. A large portion of autonomous navigation systems use depth information to detect and avoid obstacles in their path. While there are many different types of depth sensors, due to their specifications, some are better suited to the application of autonomous navigation onboard MAVs than others. MAVs which navigate autonomously, require depth information of multiple points in their environment so that a traversable path can be found. This depth information is processed into a point cloud, which is a vital input into the system. Other important sensors include accelerometers (for sensing acceleration), gyroscopes (for measuring orientation), barometers (for measuring altitude and wind speeds) and magnetometers (for sensing the earth's magnetic field). These sensors are usually bundled into an onboard circuit called the inertial measurement unit (IMU). These sensors are used to determine the state of the MAV during flight. A common problem with all sensors is noise, to address this issue, sensor fusion is sometimes performed using Kalman filters to help reduce noise and produce better approximations of state estimates [Beard and McLain 2012]. Kalman filters are not as computationally complex as other filters (e.g. particle filters), as

such, they are better suited for onboard MAV implementations. The IMU and depth sensors are core to autonomous navigation, but depth sensors require more complex computational needs to calculate the depth map, which can slow down the perception-action loop. For this reason, the next section will discuss depth estimation techniques used in these sensors.

2.2.5 Depth Sensors

There are three main types of depth estimation sensing techniques that are predominately used onboard MAVs; active stereoscopy, time of flight, and structured light sensors.

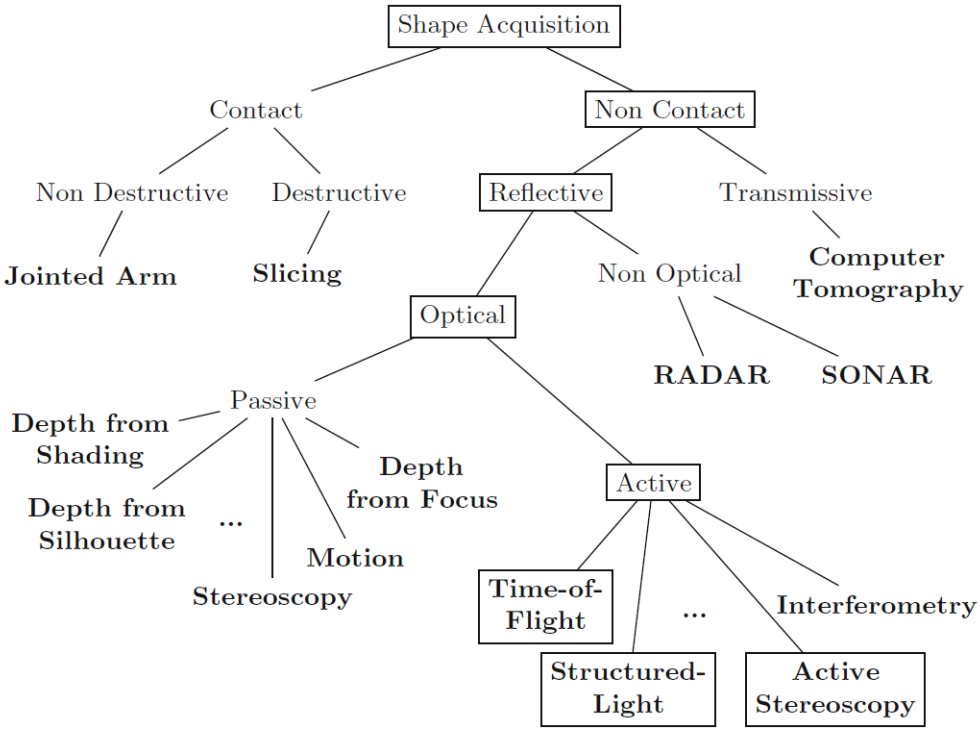


Figure 2.1: Taxonomy of 3D reconstruction techniques used for depth sensing [Giancola et al. 2018].

Stereoscopy

This technique mimics how humans and many other animals view the world. Stereo vision makes use of two cameras that are positioned on the same ground plane, at a set distance apart, and slightly rotated towards each other. The two captured images from the cameras are essentially the same, but the viewing angles differ relative to each other by a slight amount. A specific point in the left image is located in the right image during a process called correspondence matching. The location of this point is slightly different in both images due to the orientation of the cameras, this is called the parallax error. The difference between the location of the matched point in both images is called the disparity and is used to estimate depth [Prince 2012].

Accurate correspondence matching in practice is complex. Most depth from stereo vision algorithms incur a large computational cost when compared to the onboard

computing abilities of the average MAV. Methods of correspondence matching can be categorised as feature-based or correlation-based. Feature-based matching uses geometric shapes such as edges, corners, and lines, present in the scene and assesses the similarity of these shapes between frames. Feature-based matching is less computationally expensive and can handle contrast variations well but tends to produce a sparse disparity map which leads to a sparse depth map. On the other hand, correlation-based matching uses a neighborhood of pixels around a selected pixel in the left image, and the same neighbourhood of pixels in the right image is then assessed using a correlation function to find the corresponding pixel selected in the first image. This category is more computationally expensive, sensitive to intensity variations, and does not work well on uniform intensity regions (textureless regions) of images (walls or regions of excessive saturation), but a dense disparity map can be derived [Giancola et al. 2018]. Regardless of the approach used, the density of a depth map is dependent on the number of features and textured regions in the scene.

Some stereo vision cameras use projected light patterns and others use ambient light to determine correspondence points. Ambient light systems are referred to as **passive stereoscopy** and use the natural features and textures that exist in the scene for correspondence matching. In **active stereoscopy**, a random pattern is projected onto the scene, this pattern is used to apply additional features and textures to the scene which helps improve correspondence matching. At large distances, structured light stereo vision does not work well, this in part, is because the projector can only project the pattern reliably to a certain distance, and ambient light may distort or interfere with the projected pattern. Stereoscopy incurs large computational costs as a result of computing as many correspondence points as possible, in search of a dense disparity map. Field programmable gate arrays (FPGAs) are integrated chips (ICs) generally optimised for one task only but are much faster than traditional CPUs. FPGAs are used to speed up depth map computations onboard depth sensors which reduce the computational load of the MAV but there is still a significant power consumption from these devices [A. J. Barry et al. 2015].

Structured Light Sensors

Much like stereoscopy-type sensors, structured light sensors use the concept of correspondence matching and triangulation to determine depth, but instead of using two cameras, one camera is used, and the second camera is replaced by a projector. The projector projects a known structure onto the scene. Since the projector's position and direction are known, points in the projected pattern can be used to match points in the camera image and triangulation can be used to generate a disparity map. The projected pattern is also encoded in different ways (time, colour, or spatially) which helps to further assist correspondence matching and produce a dense depth map [Giancola et al. 2018].

Time of Flight (ToF) Sensors

Depth estimates can also be obtained from measuring the time taken for a signal sent out at a specific point in time, to return. These types of sensors are referred to as time-of-flight sensors and include sound navigation and ranging (SONAR), radio

detection and ranging (RADAR) and light detection and ranging (LiDAR). The basic idea is that a signal in the form of sound waves, radio waves or light pulses is encoded and sent out into the world, the signals are reflected off of objects and returned to the sensor. Since the speed of light is constant and the wavelength of these signals is known, the overall time that it takes for the signal to return can be used to calculate the distance of the object. This process is then carried out for each point of interest, such that a dense depth map of the environment can be built.

Device	Technology	Range (m)	Resolution	Frame rate (fps)	Field of view
PMD CamCube 2.0™	Time-of-Flight	0–13	200 × 200	80	40° × 40°
PMD CamBoard™	Time-of-Flight	0.1–4.0	224 × 171	45	62° × 45°
MESA SR 4000™	Time-of-Flight	0.8–8.0	176 × 144	30	69° × 56°
MESA SR 4500™	Time-of-Flight	0.8–9.0	176 × 144	30	69° × 55°
ASUS Xtion™	Structured-light	0.8–4.0	640 × 480	30	57° × 43°
Occipital™	Structured-light	0.8–4.0	640 × 480	30	57° × 43°
Sense 3D scanner™	Structured-light	0.8–4.0	640 × 480	30	57° × 43°
Kinect V1™	Structured-light	0.8–4.0	640 × 480	30	57° × 43°
Kinect V2™	Time-of-Flight	0.5–4.5	512 × 424	30	70° × 60°
Creative Senz 3D™	Time-of-Flight	0.15–1.0	320 × 240	60	74° × 58°
SoftKinetic DS325™	Time-of-Flight	0.15–1.0	320 × 240	60	74° × 58°
Google Tango™ Phone	Time-of-Flight	–	–	–	–
Google Tango™ Tablet	Structured-light	0.5–4.0	160 × 120	10	–
Orbbec Astra S™	Structured-light	0.4–2.0	640 × 480	30	60° × 49.5°
Intel SR300™	Structured-light	0.2–1.5	640 × 480	90	71.5° × 55°
Intel R200™	Active stereoscopy	0.5–6.0	640 × 480	90	59° × 46°
Intel Euclid™	Active stereoscopy	0.5–6.0	640 × 480	90	59° × 46°
Intel D415™	Active stereoscopy	0.16–10	1280 × 720	90	63.4° × 40.4°
Intel D435™	Active stereoscopy	0.2–4.5	1280 × 720	90	85.2° × 58°
StereoLabs ZED™	Passive stereoscopy	0.5–20	4416 × 1242	100	110°(*diag.*)

Figure 2.2: Specifications of some depth sensors available [Giancola et al. 2018].

Event Cameras

As a result of the depth sensor latency of frame-type cameras, event cameras are being used where high-speed MAV motion is required to avoid obstacles, especially where dynamic obstacles are taken into consideration. Event cameras have a significant reduction in latency when compared to image frame type sensors, but their output data is output in a different format. Event cameras output a stream of asynchronous data per pixel, where the output stream contains a timestamp, pixel location and intensity information for each pixel. Depending on the type of event camera, output intensity information can be indicated either by instantaneous pixel intensity, or the direction in which the pixel intensity change occurred. Another attribute of event cameras is that they offer a much higher dynamic range, with regards to pixel intensity, and as a result, achieve enhanced utility in low light conditions. Falanga et al. 2020 has demonstrated that event cameras used onboard MAVs can be used for dynamic obstacle avoidance. While event cameras are not depth cameras, they do help cope with the issue of motion blur incurred by cameras which are imaging objects moving at high speed. In contrast to frame-type cameras, event cameras do not suffer from motion blur, have low data acquisition latency, have a much larger dynamic range and consume much less power [Gallego

et al. 2017].

Onboard MAV computing, depth sensor limitations, and power supply/consumption are core problem areas that hinder autonomous navigation and as a consequence, high-speed autonomous navigation is also affected. Although the adoption of event cameras for use onboard MAVs for autonomous navigation is still in its infancy, these cameras have a considerable impact on each of the bottleneck areas mentioned above [Falanga et al. 2020].

2.2.6 Simulation Platform

To test each algorithm's ability to avoid obstacles at high speed in the real world, we will use a high-fidelity simulator. We opt to use a simulator because the physical systems used to test the algorithms are expensive and are not easily available. Since we will not be testing in a real-world scenario, the chosen simulator will have to approximate real-world conditions as closely as possible, closing the simulation to real-world gap in the best way possible. While there are many simulation engines to choose from, only a handful stand out in terms of photorealism. The two main competitors are Unity and Unreal Engine, which are graphics engines built predominantly for the gaming industry. Unreal Engine is widely considered the better option if the comparison metric is realistic graphics. MAV-specific simulators used for research include Flightmare [Song et al. 2021] which is built using Unity and AirSim [Shah et al. 2017] which is built using Unreal Engine. We opt to use AirSim, since better photo-realism can be achieved in Unreal Engine. The Unreal Engine marketplace is an online store where environments and asset packs are available for download which makes creating high-quality environments easier.

2.2.7 Environmental Elements Which Deter Obstacle Avoidance

When creating simulation environments, the environment must replicate common environmental elements that cause the environment to become more complex and deter obstacle avoidance. This section outlines a few of the factors that could deter high-speed obstacle avoidance.

Structured vs. Unstructured Environments

In the categorisation of environments, it is important to note that the literature refers to structured and unstructured environments, but makes no clear differentiation between the two. There are implicit indications that structured environments tend to be urban-type environments and unstructured environments tend to be more rural-type environments. Unstructured environments tend to be discussed as the more complex of the two but the metric used to make this distinction is not quantitative or entirely clear. When a forest is referenced as an unstructured environment, the degree to which it is unstructured is unclear.

For this discussion, we take structured environments to mean environments where most objects are man-made and generally resemble simple geometric shapes such as squares, rectangles, circles, and triangles. Unstructured environments are those

where most objects occur naturally and tend to resemble more complex shapes and patterns.

Navigational Complexity of Unstructured Environments

There are many different types of unstructured environments but focus is given to forests. Forests cover 31% of the global land area making it extremely likely that MAVs would encounter them [Zhongming et al. 2020]. There are many types of forests depending on which part of the world is being considered and what classification system is being used. The United Nations Environment Programme - World Conservation Monitoring Centre (UNEP-WCMC) categorise the forests of the world into 26 different types [Groombridge and Jenkins 2002]. This can further be reduced to 6 less detailed categorisations which are in part based on leaf structure and vegetation sparsity. The geometric structure of a forest is defined by its landscape and vegetation. For this reason, forest categorisations matter when defining unstructured environments, as they relate to the variation of structure and spatial density of vegetation inside a forest.

The navigational complexity of a forest is determined by the general structure and spatial density of vegetation inside the forest. This is because the general structure and spatial density of vegetation define the trajectories that MAVs need to track for successful obstacle avoidance. These characteristics begin to provide insights into how quantitative metrics can be assigned when defining the degree to which unstructured environments vary.

Care should be taken when building unstructured environments in simulation because complexity can easily grow rapidly in environments perceived as simple. If an environment contains too many aspects of complexity it becomes difficult to keep track of which aspects of the environment cause obstacle avoidance failures.

Object Occurrences and Characteristics

The main problem in obstacle avoidance is the obstacles. This is intuitively obvious but the complexities of obstacle avoidance can easily be overlooked. Not only can people classify objects with remarkable ease, but they also have higher levels of understanding about these objects. For example, a person taking a jog could see the front of a parked car, know that it is a car, and understand its general shape. As a result, the person could easily plan where to run to avoid the car without much thought or effort.

A MAV on the other hand, does not have the same ability. While the argument can be made that it could be given these abilities through ML models, the computational requirements to run these models on board a MAV in real-time are one of the main prohibiting factors. As a result, more computationally efficient methods of obstacle avoidance need to be explored and tested. There are various paradigms used to develop obstacle avoidance algorithms, some work better than others. However, there are often fewer testing environments to explore how well these algorithms generalise in different categories and complexities of environments.

This is in part because finding an adequately complex testing environment in reality can be difficult. Due to the dynamic nature of environments, these environments

might also change and become more or less complex unexpectedly. This means that the testing environment becomes inconsistent. This aside, the metrics being used to measure complexity are usually unclear or abstract.

When considering how complex an environment is, its components need to be broken down to better differentiate which components give rise to complexity. Complexity also needs to be defined clearly, what quantitative metrics make an environment complex? For the case of this discussion, complexity is defined with regard to MAV navigation. The more complex an environment the more difficult an environment is to navigate.

If open space removes the requirement for obstacle avoidance and the presence of objects provides the need for obstacle avoidance, then at least one source of complexity must occur from the presence of obstacles. From this point onwards, the characteristics of this presence can vary which can make the environment increase or decrease in complexity. As an example, if the spatial frequency of occurrence of an object increases in a fixed area, there is less free space in that area, which would make MAV navigation more difficult.

Objects in unstructured environments take the form of various structures. These structures, for example, trees, can make MAV navigation more difficult by providing navigational spaces that require the MAV to make intricate path traversals to pass the object successfully. Some MAVs can move through the canopy of trees by pushing past leaves and thin branches. However, these MAVs still need to understand where solid branches are to navigate through the canopy successfully [Geckeler et al. 2024]. Some objects have characteristics, such as textureless regions, which make them difficult to be sensed by the sensor suite which can cause the environment to be perceived erroneously. In structured environments, an example would be large walls painted in one colour under uniform lighting. In unstructured environments, examples would be large bodies of fine-grained sand, such as those seen in deserts, large bodies of untouched snow, or even large stone formations.

Non-Lambertian surfaces are a characteristic present in some objects and can cause vision and LiDAR-based depth sensors to give erroneous depth measurements. These highly reflective surfaces can cause other parts of the environment to be reflected by objects causing vision-based systems to perceive traversable paths where none exist[Tao et al. 2014]. These types of surfaces can reflect light away from LiDAR transceivers and cause irregular sensing measurements. In a structured environment, an example of this type of surface would be a large window on the outside of buildings which creates reflections when viewed from the correct angle. In unstructured environments, these types of surfaces can be seen on large bodies of water. In scenarios where textureless regions of an environment are being reflected by non-Lambertian surfaces, the interaction of two separate object characteristics can cause two depth estimation failure modes to occur at the same time. For this reason, care needs to be taken when adding complexities to a simulation environment.

From the above discussion, there are two important characteristics when considering environmental complexity relative to MAV navigational complexity. The first is the characteristics of the object's presence (spatial frequency of occurrence) and the second is the object's characteristics (general shape and optical characteristics).

Dynamic Obstacles

Dynamic obstacles also pose a problem for high-speed autonomous navigation. If a MAV encounters a dynamic object, the object's velocity relative to the MAVs velocity can cause motion blur due to low frame rates of the imaging systems [Falanga et al. 2020]. In structured environments, dynamic obstacles can be seen in the form of humans or animals, to name a few. In unstructured environments, this can be seen mostly in the form of animals and trees. Where trees can become dynamic due to environmental factors such as wind or rain.

2.3 Related Work

As noted, accurate depth estimation with low latency and reduced computational complexity, are core design requirements to achieve high-speed obstacle avoidance using only onboard MAV hardware. There have been many approaches to address the problem of obstacle avoidance in unstructured environments and some of the approaches indirectly achieve results at high speed. The obstacle avoidance performance of these algorithms needs to be evaluated so that reliability and generalisability can be better understood. This would require a benchmarking system that uses the appropriate metrics.

This section explores the existing literature for solutions to high-speed obstacle avoidance, obstacle avoidance evaluation strategies in unstructured environments, and the associated metrics that quantify environmental or object complexity.

Reactive Obstacle Avoidance

A possible approach to obstacle avoidance is to use a reference in the environment to generate trajectories that are obstacle-free. This approach to obstacle avoidance is proposed by Smolyanskiy et al. 2017, where hiking trails, which are assumed to have a reduced chance of obstacles in the path of the MAV, are used as a guide to navigating a forest. In this approach, the MAV estimates its pose in relation to the trails on hiking paths and follows the hiking path. The system uses a single camera to detect and avoid obstacles reactively. This concept is shown to work in unstructured environments where trails are present but will not work where there is no explicit trail. The goal of high-speed obstacle avoidance is to have a control method that enables a MAV to fly quickly through unstructured environments. This means that the MAV will undoubtedly, at some point, be forced into an area of the environment where a trail does not exist. This potential flaw aside, autonomous navigation in unstructured environments is achieved, which indicates progress.

Using a single RGB camera to detect objects, Aguilar et al. 2017 achieve obstacle avoidance. The system uses reference images of objects from a database and compares them to the incoming stream of images from the MAV. The area and center of mass of the object are then calculated and used to estimate the distance of the object from the MAV and the extent of direction change required to avoid the obstacle. Depth is estimated by using the area of the object, the larger the area the closer the object, the smaller the area the further the object. Obstacle avoidance is achieved by increasing the distance between the center of mass of the object and

the center of vision of the MAV. This implementation uses a proportional controller to adjust the MAV direction when an object in the current path is detected. The authors note that there is no explicit trajectory tracking or recovery, and instead use velocity as the only form of a trajectory to limit the testing to obstacle avoidance. While this is a very simplistic approach, the authors demonstrate that the solution does work in very simplistic environments at speeds in the region of 1m/s. This approach however would not work in complex unstructured environments due to the lack of detailed depth information and a more capable position controller. Another inhibitor to high-speed obstacle avoidance is that all compute is performed on a laptop connected to the MAV via Wi-Fi. This limits the MAV to the signal integrity of the Wi-Fi connection to the laptop and the available bandwidth. This signal can easily be attenuated by vegetation, distance, and antenna propagation patterns.

Depth information is commonly obtained from stereo cameras because they tend to have a good cost-to-specification ratio when compared to other types of depth sensors. As discussed, a large computational load is incurred as a result of computing the depth map of a scene using stereo cameras. To reduce this computational load A. Barry et al. 2018 compute correspondence points at a set range on the epipolar line. The idea is that a point in the left camera's image plane exists on the epipolar line in the right camera's image plane (line from the right camera's center to the point of interest), so correspondence matching can be constrained to a one-dimensional search through depth. Further reducing the search range on the epipolar line is equivalent to only computing the depth at a set distance away from the MAV. Since the MAV is moving at high speed there exists a certain range of distances ahead which are most important. Distances before a minimum distance in front of the MAV may be too close for the MAV to react in time to avoid obstacles. Distances above a maximum distance ahead of the MAV may recede too far into the horizon, such that their consideration would be meaningless for avoiding current obstacles in the MAV path. The resulting algorithm is called pushbroom stereo vision and demonstrates successful obstacle avoidance at speeds of 10m/s on a fixed-wing MAV. The shortcomings noted in this paper are the lack of virtual simulation platforms in which the algorithm can be further fine-tuned. Tests in this paper were conducted in the real world, onboard a fixed-wing MAV. The authors note that, if the implementation could be implemented in a virtual simulation environment, further refinement could be made to improve the algorithm. The paper provides a solution to local planning for high-speed obstacle avoidance but does not address the issue of a global planner which defines the destination state of the MAV.

Many autonomous navigation systems make use of cameras which use features or objects in the environment to compute depths. The problem with cameras is that when imaging textureless areas in a scene, the depth estimates become sparse or object recognition is adversely affected (camouflaged objects). This means that the tolerance of depth measurements becomes greater or objects cannot be detected. This can make it difficult for the MAV to localise itself within its map and subsequently, the success rate at which the MAV is able to reach its final goal state is affected. Z. Zhang and Scaramuzza 2018 address this problem by explicitly evaluating sensor perception quality and incorporating it into the evaluation of trajectories. The idea is that, if a MAV traversed trajectories with more texture in its environ-

ment, it would be able to more effectively determine features in the environment and more accurately and successfully be able to perceive objects and localise itself. The proposed method uses two maps, the first is a dense depth map which is used for obstacle avoidance, and the second is an active map used for localising the MAV. Taking both maps into consideration, a list of possible trajectories is calculated up to a set distance from the MAV. Each of these trajectories is then evaluated according to the reward received based on progress to the goal state, risk of collision, and quality of texture in the environment. The trajectory with the highest reward is then chosen for execution. It is shown in simulation that the trajectories generally chosen are balanced between collision risk and state estimation uncertainty, which ultimately improves state estimation and obtains a higher success rate in achieving the goal state. This paper addresses an area of autonomous navigation that affects the success rate at which the goal state is achieved. Whilst the speed of the MAV is not addressed directly, this idea is worth considering as MAVs will undoubtedly find themselves in a multitude of environments, of which, some may be void of a useful number of features. In these scenarios the MAV should still be able to perform successful obstacle avoidance and autonomous navigation, therefore selecting trajectories with more texture may be more optimal.

As mentioned before, MAVs have limited onboard computation available. In many systems, part of this computational burden comes from updating an internal map of the environment using sensor data. The map is then used to navigate the environment whilst reactively avoiding obstacles that are not contained in the map.

J. Zhang, Hu, et al. 2020 propose a solution that reduces the computational complexity by moving parts of the online computation, offline. The authors aim to reduce the online computational burden by modeling the environment in two separate ways, objects are deterministically known by sensor data and probabilistically known by using a prior map of the environment. The sensor's view of the environment is used to find voxel groups that contain obstacles. Pre-computed trajectories are defined based on where the MAV can move when there are no obstacles present in the sensor's view of the environment. These trajectories extend to a distance at the edge of the sensor's perception range. Given the starting state of the MAV, the system uses the pre-built map of the environment and the pre-computed trajectory list, to compute the probabilities that a given combination of the pre-computed trajectories will allow the MAV to reach the goal state. The chosen trajectory for immediate execution is decided, based on the trajectory combination with the highest probability of reaching the goal state. When there is an object in the path of the MAV, trajectories that collide with the objects are removed from consideration of traversable paths, which leave trajectories with the highest certainties of reaching the goal state for consideration. It is noted that the system tends to favour more open spaces because the probability of finding obstacle-free paths is increased, which further increases the probability of reaching the goal. This means that the system favors success rate over the shortest time to a goal state. For the scenario where a pre-computed map is not used, the global planner uses a direction as the goal to maintain while implementing collision avoidance. The goal direction can be adjusted by human input (e.g.: via joystick), this is how a human pilot can fly the MAV without the worry of costly collisions. The proposed method is quite robust, because it works with or without a pre-computed map, can take in human input in

the form of a general direction, it generalises to 2D, and 3D environments as well as to aerial and ground-based autonomous vehicles.

Pose Estimation

Another method of obstacle avoidance is to map the environment which the MAV is traversing, when the environment is known, obstacles can be avoided more easily. This is generally done using a version of simultaneous localisation and mapping (SLAM). One of the fundamental problems in SLAM is that pose estimation can be uncertain due to sensor noise. P. R. Florence et al. 2018 focuses on a solution to this problem by taking samples of a recent history of depth measurements and using these measurements to estimate pose whilst accounting for pose uncertainty caused by noisy depth sensor information. It does not map the environment but instead takes a snapshot of recent depth measurements and compares this to noisy relative pose transformations to better estimate pose when deciding whether a trajectory is collision-free or not. In this sense, it improves on existing works by focusing on the problem of accurate pose estimation of the MAV in its environment, and by doing so, it can more accurately decide if a trajectory is collision-free or not. By improving pose estimation, tracking trajectories can be achieved more accurately. This is in essence how it can be used for high-speed obstacle avoidance in unstructured environments with SLAM-based methods for obstacle avoidance. This was tested in a structured environment. NanoMap provides pose uncertainty-aware responses to depth information queries, using the most certain approximation of pose, and can do so much faster than traditional approaches for estimating pose. NanoMap is tested in a Unity simulation environment to verify pose estimation accuracy. There is a scenario where the MAV flies at a target speed of 15 m/s in a straight line to a goal positioned 200m away, but at 100m from the MAV starting point, there is a solid wall. A second environment is also used to test the concept of NanoMap, where the MAV needs to navigate through a forest simulated in Unity, 50m wide and 160m in length, with 53 randomly placed trees at speeds of 2, 5, 8, and 12 m/s [P. Florence et al. 2020]. In this scenario, as the speed increases, the number of successful flights is reduced. NanoMap was also tested extensively in both indoor and outdoor environments during the 2017 DARPA FLA (Fast Lightweight Autonomy) event by the MIT-Draper team and showed promising results. The idea is to test pose estimation during aggressive maneuvers, as this is when the greatest number of inaccuracies in pose estimation is seen.

2.3.1 Reinforcement Learning Based Techniques

The field of Reinforcement Learning (RL) has shown promising results in piloting MAVs [Abbeel et al. 2006], [Loquercio et al. 2021]. RL requires a problem to be broken down into episodic iterations of the process attempting to be learned so that an execution policy can be derived. This can be done in simulation or in the real world. Real-world episodic iterations on actual MAVs are expensive both in human resource time and in MAV repair costs and it would take an extraordinary amount of time to derive a usable policy. These are some reasons why simulators are used over real-world trials. RL algorithms need to understand the transition dynamics of an environment. This means that the new state or pose of the MAV needs to be known when it moves from an initial state or pose after a combination of thrusts is applied

through its motors. The bottleneck, however, is that simulations of many real-world tasks or processes, such as transition dynamics in various environments, do not exist or the simulators do not do a perfect job of recreating the real-world process. Modeling accurate transition dynamics in simulation is difficult. For example, how would heavy rain, fast winds, and cold temperatures affect MAV hardware and what would the downstream effect on a given trajectory be? The inability to accurately recreate the complexities of the real world is known as the simulation to real-world gap. Obstacle avoidance is one such task, where the gaps between simulators and the real world are due to the inability to accurately recreate the complexities of the real world. These complexities come in the form of textures, shading, and movements of real-world objects and scenes inside simulators, especially because many of these elements move in unpredictable and complex ways. A question then arises, can an obstacle avoidance policy be trained in a simulator and then used successfully in the real world?

This question is answered by Sadeghi and Levine 2016, where an RL obstacle avoidance policy is trained entirely in simulation and then tested in a real-world setting. The system uses a single RGB camera to process images into control commands with the goal of avoiding obstacles. During training, the simulator is adjusted such that rendering settings are randomized so that the algorithm can be trained on varying degrees of photo-realism relative to the real world. A reward function that rewards policies that stay far away from obstacles is used for policy evaluation. Successful obstacle avoidance in indoor environments demonstrates that RL policies for MAVs can be effectively trained in simulations and then successfully transferred to real-world operations. This is achievable despite the significant gap between simulation and reality, where the simulation does not perfectly approximate the real world. While this method allows the MAV to avoid obstacles, the authors do note that there are times when the MAV collides with obstacles during testing, which indicates further need for refinement. The method was tested in a structured environment, so for obstacle avoidance in unstructured environments, this method would need to be trained and tested in unstructured environments. The key point to note is that the collision avoidance policy was trained using a reward function that rewards policies that stay far away from obstacles and were trained entirely in a simulation environment.

How well would this method generalise if the simulation environment was an even better approximation of the real world and noise was added to a more high-fidelity simulation? Would this improve the success rate of the collision avoidance policy? This paper uses Gazebo as a simulation engine, but there are other simulation engines to date, that are quickly reducing the simulation to real-world gap. Though improving the simulation engine is not required for successful MAV obstacle avoidance, would doing so improve the collision avoidance success rate?

Loquercio et al. 2021 address these questions in their research on high-speed flight in unstructured environments. The authors propose an end-to-end solution where noisy depth and inertial sensor measurements are directly mapped to navigation commands. Training is done entirely in simulation using stereo-depth images as inputs to a deep-learning navigation policy. The policy is trained by demonstrations using a privileged-based expert, which is essentially a navigation algorithm that has access to the complete point cloud of the training environment, as well as the

complete state information of the quad-rotor, and is essentially computationally unconstrained. The policy takes in noisy depth and inertial sensor measurements and produces a set of short-term trajectories as high-order polynomials, each with individual cost estimates. These trajectories are then compared and mapped to the best trajectories generated by the privileged-based expert. The trajectory selected at run time is then chosen based on the estimated cost. It is shown that a policy trained entirely in simulation can be implemented in real-world environments, at high speed in a multitude of environments ranging from structured to unstructured. Furthermore, the simulation engine used is Unity, which is generally used for gaming and provides much more realistic environments when compared to Gazebo.

2.3.2 Dynamic Obstacle Avoidance

Another issue that further complicates the obstacle avoidance problem is that of dynamic obstacles. Many solutions do not take dynamic obstacles into account explicitly. Dynamic obstacles have velocities that are outside of the control of MAVs, as such, encounters with these types of obstacles would require very fast execution of the perception-action loop, especially when considering MAVs in the high-speed autonomous navigation setting. Falanga et al. 2020 addresses a fundamental issue with using cameras for high-speed obstacle avoidance; MAVs require extremely fast perception sensors to react quickly in rapidly changing environments. The limitation of standard cameras is that image acquisition can take between 1-100ms before any processing is done, this in part limits high-speed obstacle avoidance by slowing down the perception-action loop. This paper addresses this issue by using event-based cameras, which output a stream of asynchronous events, and have a much lower latency than standard cameras. It is shown that a MAV can avoid dynamic obstacles that are moving at relative speeds of 10 m/s in both structured and unstructured environments. The algorithm introduced in this paper has a latency of 3.5ms, which is favorable when looking at a solution to the high-speed obstacle avoidance problem. This paper highlights the capabilities of event cameras over frame-type cameras in MAV applications.

Another paper that investigates the problems associated with autonomous navigation and dynamic obstacle avoidance is "Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots" [Kong et al. 2021]. In this paper a point cloud is generated directly from a solid-state lidar scanner and time-accumulated KD trees are used to partition the point cloud. This partitioned point cloud is then fed to the kinodynamic A* algorithm (referenced in Liu et al. 2017) which generates possible trajectories. Collision checks are then performed to ensure that the flight controller tracks a collision-free path. The results indicate that the proposed method works for both structured and unstructured environments where both static and dynamic objects are present. However, the authors note that their method has discontinuous acceleration trajectories, which means that smooth flight is a problem, as well as limited control options which limit flying at higher speeds.

2.3.3 Evaluating Obstacle Avoidance in Unstructured Environments

Nous et al. 2016 define a relatively comprehensive framework for evaluating obstacle avoidance in 2D space. The evaluation is broken into two separate areas, the ability of the MAV to detect obstacles and the ability of the MAV to avoid obstacles. For object detection, attention is given to environmental characteristics that hinder the ability of the MAV sensor suite such as distances, reflectivity, illumination, texture, infrared illumination, inclination, transparency, and radar cross-section. These characteristics are known to affect specific classes of sensors, which is why they are explored. Five main obstacle avoidance metrics are defined: Traversability, collision state percentage, average avoidance length, average orientation of obstacles, and the percentage of dead-end present. The traversability of an environment is calculated as the average distance a MAV can fly in a straight line before it collides with an object when placed randomly in the environment. Traversability is a measure of the free space of an environment taking the size of the MAV into account. After many iterations of the traversability calculation, traversability approximates the average radius of free space around a MAV given any point in the environment. Collision state percentage is defined as the percentage of the space in which the MAV is unable to avoid a collision given its speed. Average avoidance length is the average lateral distance that the MAV is required to move such that an obstacle can be avoided. The average orientation of obstacles and the percentage of dead-end present are not discussed but it is indicated that they are known shortfalls of navigation planning strategies. The evaluation framework is designed in 2D space to reduce complexity and experiments are performed in controlled 3D space. Environmental characteristics are introduced artificially so that their variation can be controlled.

The experiments test the distance, illumination, and texture characteristics that deter object detection but they may not be entirely representative of the way these characteristics occur in nature. For example, an MAV may be flying below the canopy in a forest on a sunny day illuminating some parts of the forest floor whilst other parts are in the shade of the trees. This creates a scenario where illumination fluctuates, this would cause the auto exposure setting on RGB stereo cameras to require constant adjustment, potentially at a rapid rate, which could impair depth estimation.

Texture variation is achieved through the variation of contrast but this is not entirely the component of texture which makes depth estimation error-prone in stereo vision cameras. As discussed previously, textureless regions make depth estimation difficult by introducing a narrow band of pixel intensities causing correspondence matching to become inaccurate. Contrast variation would make the pixel intensities of a textureless region vary in the same way. Recreating these characteristics in the way they would occur in nature might be more beneficial but it is unclear to what extent. This aside, recreating a testing environment with this requirement can be very difficult due to the way these characteristics occur and interact with one another in nature. For example, characteristics such as reflectivity can depend on the approach angle and direction of the MAV to the object. This determines what the reflective surface reflects, which could be another traversable path if the reflection is large enough or just foliage from the environment which needs to be disregarded

anyway. Nevertheless, the definitions and methodologies help to better define the characteristics of the metrics used to measure obstacle avoidance performance and environmental complexity.

Mishkin et al. 2019 evaluate the performance of obstacle avoidance strategies in indoor 3D environments using the MINOS simulator. While the evaluation is not unstructured, some assessment metrics are defined which generalise to unstructured environments and may be of interest. These include success rate, and success rate weighted by path length and pace. Pace is defined as the average time to complete navigation divided by the budgeted time, which does not assess the MAVs speed but is rather an indicator of how quickly a course can be navigated. The success rate is defined as the average number of times the MAV can complete the course in the allowed time. Defining the success rate in this way does not indicate its relation to collisions. The success rate weighted by path length is an interesting consideration because the path length in a complex unstructured environment might change depending on speed. A MAV may be able to navigate an environment but at a much slower speed and a lower chance of collisions but how that would affect path length is unclear. This is because speed affects the ability of the MAV to track required trajectories.

Another approach to defining navigational complexity uses the mean shortest path as a quantitative metric [Ermacora et al. 2020]. The mean shortest path is the average distance of the shortest paths from a starting position to a goal position in an environment. This is an implicit indicator of speed, path traceability, and environmental complexity.

2.3.4 Discussion

Section 2.3 discusses some of the approaches used to achieve high-speed obstacle avoidance. Many of these approaches incorporate machine learning, sensor fusion and many other techniques. These techniques are used to help build maps, improve collision-free trajectory tracking, improve pose estimation, reduce depth estimation computations and accuracy. Various bottlenecks in the learning pipeline are addressed with varying degrees of success, each with their own set of trade-offs.

We can also see that there are no standardised methods of testing obstacle avoidance. The broad catagories of structured or unstructured testing environments are stated and the degree to which clutter is present is given in an abstract way without any descriptive metrics. Mishkin et al. 2019 and Nous et al. 2016 describe more rigorous approaches to testing and establishing environments which make use of quantitative metrics which stress the perception action loop of MAVs.

It is clear from the literature that there is ambiguity in the description of the extent to which obstacle avoidance can be achieved. If an obstacle avoidance algorithm is tested in an environment and achieves consistent and successful outcomes but complexity metrics of the environment are not discussed then reproducibility becomes difficult. Obstacle avoidance research should be reproducible and as such a standardised procedure to recreate testing environments is required. Recreating an environment which mimics that of another becomes difficult when there are no quantitative metrics in which the complexity between environments can be mea-

sured. This becomes a barrier to reproducibility in obstacle avoidance research because results cannot be readily verified.

2.4 Chapter Summary

MAVs and their application space are discussed with a focus on system constraints as well as deterrents when running high-speed obstacle avoidance algorithms onboard MAVs. The MAV sensor suite was then discussed where the IMU was outlined and a detailed discussion of depth sensors was given. The importance of high-quality depth estimation is discussed and a description of the pros and cons of current techniques are shown. SWaP constraints and the need for balance between these constraints are highlighted. It is also shown, how complex computational loads resulting from planning and obstacle avoidance algorithms, further burden the already constrained MAV platform and extend the perception-action loop time. Simulation environments for testing high-speed obstacle avoidance algorithms were discussed in brief, with the most important requirement being high-fidelity graphics which create the most realistic simulation of the real-world environment for which the MAV can be evaluated. Environmental aspects of various environments are discussed, and the nuances of unstructured environment definitions are expanded on. Finally, current solutions to the problem of obstacle avoidance are given with speed being the general focus. Subsections of the problem domain that are being addressed by these solutions are shown, and the pros and cons are noted in brief. There are many sub-components of the high-speed obstacle avoidance processing pipeline which have been addressed with various solutions. Pose estimation improvement, faster data acquisition times, reactive control methods, RL, and various heuristics are employed with varying degrees of success, each with its own set of trade-offs. But which solutions yield the best results when addressing the high-speed obstacle avoidance problem? For this evaluation, a system for testing obstacle avoidance in the presence of environmental complexity is needed. While there are not many frameworks that evaluate navigational complexity in this way, the current literature gives a good direction for helping define evaluation metrics for this task.

Chapter 3

Research Methodology

3.1 Introduction

As noted in Chapter 2, existing high-speed autonomous navigation systems for MAVs have been shown to work in real-world environments. However, the degree to which they have been evaluated varies based on the research questions relevant to their particular focus area. In this section we clearly define the aim of this experiment, the research questions we aim to answer based on our hypothesis, and the methodology used to decide on quantitative metrics used to evaluate an autonomous navigation algorithm.

3.2 Aim

These experiments aim to evaluate the success rate of obstacle avoidance algorithms onboard a MAV in unstructured environments. Collisions can occur at multiple failure points in the obstacle avoidance pipeline due to the complex and time-sensitive nature of the task. As a MAV moves faster, its reaction time needs to be reduced or its perception range needs to be extended such that trajectories can be tracked. MAV reaction time is characterized by the complexity of the perception-action loop which is hardware (sensor suite as well as compute) and control algorithm dependant. Perception range is defined by MAV hardware, particularly the sensor suite. As a MAV is forced to navigate incrementally complex environments there should be a proportionate reduction in speed due to the increase in trajectory complexity.

Likewise, when the environment is simple and clear, trajectories are simple and easily defined, and the MAV should have a proportionate increase in speed. This of course assumes that there are no limitations in the obstacle avoidance algorithm design. Shortcomings in the design can cause speed reductions, collisions, and other undesirable outcomes. The simulation environment can then help locate the sources of these shortcomings which can then be improved upon.

The Evaluation of the success rate of an obstacle avoidance algorithm is achieved through measures of speed, collision count, altitude, and navigation time, as the MAV moves through levels of a simulation environment which increases in complexity. Success at navigating a level is defined as navigating an environment from start

to finish autonomously without collision. Two scenarios are taken into consideration.

The first is that the obstacle avoidance algorithm can navigate environments perfectly, in which case we should see low collision counts across all levels. The second more realistic scenario is that the obstacle avoidance algorithm is not perfect. In this case, the simulation environment can help debug the obstacle avoidance algorithm and determine whether high collision counts are due to the design of the obstacle avoidance algorithm or due to the complexity of the simulation environment. High collision counts can help identify specific complexities in the environment that cause obstacle avoidance to fail.

Autonomous navigation research tends to define the environmental complexity of MAV testing sites in a very implicit way, as it is usually not the focus of research. Part of the aim of this research is to help better define the extent of environmental complexity quantitatively. By doing so, the limits for obstacle avoidance algorithms in terms of environmental complexities and control limits within the algorithm can be found. By assigning environments quantitative properties that define their complexity, performance bottlenecks of obstacle avoidance can be found. For example, we could note that when the environmental complexity is X, at speed Y, the algorithm begins to consistently encounter collisions. This type of information provides the autonomous navigation field with a focus area to improve obstacle avoidance performance by identifying specific properties of environmental complexity.

3.3 Research Questions

1. How does speed affect the success rate of obstacle avoidance?
2. How does the degree to which the environment varies in complexity, affect the speed at which a MAV can successfully navigate that environment?
3. Does the complexity of an object affect the obstacle avoidance algorithm's ability to navigate around that obstacle?

3.4 Methodology

The following section describes the simulation environment requirements and how each requirement is achieved by carefully defining evaluation metrics for object and course complexity.

3.4.1 Simulation Environment

The simulation environment must:

1. Increase in navigational complexity as the MAV navigates through it
2. Have separate courses to test between different object complexities

The first requirement is satisfied by assigning a complexity score to each obstacle. These objects are then placed in successive levels of the simulation environment. The spatial frequency of occurrence of these objects is successively increased, as such, a simulation environment is created which increases in navigational complexity as the MAV navigates through it. The second requirement is that the simulation environment should be capable of testing the impact of objects with different complexities. To accomplish this two courses are created where each uses a specific category of objects.

The entire simulation environment is divided into 2 courses. Each course is further divided into 10 levels which describe their complexity, where 1 is the least complex and 10 is the most complex. The stages are also divided into height volumes called Low, Middle, and High. These volumes are present in all levels of both stages and vary in complexity vertically as shown in figure 3.1. The Low height volume is the least complex and the High height volume is the most complex. This is to cater for the distinct advantage aerial vehicles have over ground vehicles, which is that they can fly over objects. The MAV will need to fly from the start position to the end position of each stage, maximising a target speed whilst avoiding obstacles. As the MAV navigates to the finishing point of each stage it moves through each level of complexity within each stage. The navigation algorithm can take advantage of the height volumes of each level if it manages to figure out that level complexity reduces as altitude increases. Each level is made incrementally more complex than the last by increasing the number of objects and placing the objects in a manner that minimizes straight-line paths from the start to the finish of each level. Height volume complexity is achieved passively because all objects start on the ground and extend into the height volumes. Obstacles are chosen in part based on their maximum heights to provide blocking volumes in each height volume. Low blockers extend into the Low height volume only. Middle blockers extend into the Low and Middle height volumes and high blockers extend into the Middle and High height volumes. This means that the Low height volumes are the most complicated to navigate and the high-height volumes are the least. Building each level like this means that each level can increase in complexity to fulfill navigational complexity requirements.

The courses are not meant to incentivise algorithms to fly above obstacles but to merely provide the option of flying over obstacles. For this reason, there should be an overall height restriction that stops the MAV from entirely avoiding the simula-

tion environment whilst allowing the option to fly over obstacles into spaces where reduced complexity is perceived.

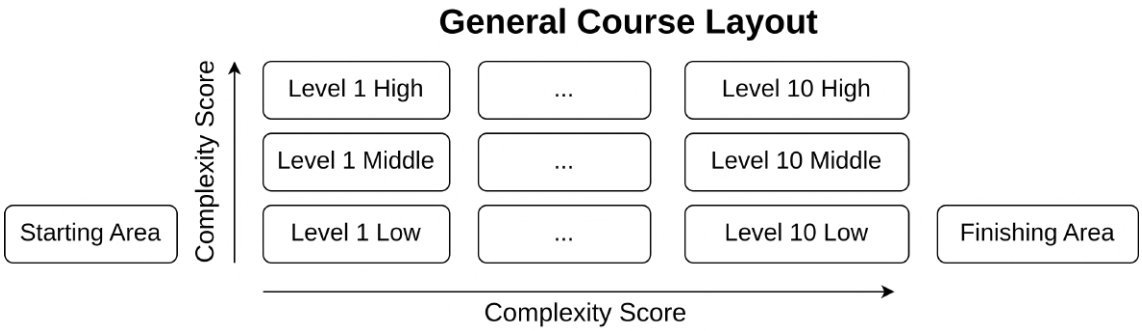


Figure 3.1: The general course layout is divided into three height volumes per level. Complexity increases from the Low to High height volumes. There are 10 levels which increase in complexity from level 1 to 10.

Simulation Environment Level Complexity

Our focus is on unstructured environments, these environments vary in their degree of complexity. In environments absent of adverse weather conditions such as wind and rain, obstacles are the first aspect of an environment that causes navigational difficulty, especially for MAVs.

For this reason, we evaluate environmental complexity using the following quantitative metrics:

- Object Complexity
- Frequency of Object Occurrence

The following section explains how these metrics are defined and used to assign quantitative measures of complexity for the simulation environment requirements.

Object Complexity

Object complexity needs to be defined quantitatively with a complexity score. This provides a mechanism in which to vary the overall complexity of each level in the simulation environment in an incremental way.

Two types of object categories are used to create the simulation environments, with the requirement that one needs to be more complex than the other. If there are too many different types of objects per course, the sources of performance degradation become more difficult to find.

Suppose there are too many types of object categories in a course. In this case, the number of ways the MAV approaches these objects is reduced, and a less rigorous assessment methodology of the impact of object complexity results. Reducing the number of objects per course should allow for enough repetition in the way the MAV approaches obstacles, such that limitations in the obstacle avoidance algorithm are easier to uncover.

Various properties can be used when considering object complexity. We define object complexity using the following properties relative to the navigational trajectories available to the MAV:

- Path Groups
- MAV Gaps
- False Depthmap Noise
- Percentage of Total Height Volume

Path Groups
Path Groups are an indicator of the presence of groups of trajectories that can be used to avoid the obstacle. Path groups define whether groups of traversable trajectories can be generated to the left, right, above, under, or through the object to get past it(see figure 3.2). The "through" categorisation of path groups takes into consideration unoccluded trajectories which can be defined through the object but can also cause false depthmap noise. The "under" categorisation indicates traversable trajectory groups which go underneath the entire object or an overhanging section of the object.

MAV Gaps
MAV Gaps take into consideration the size of the MAV and cater to the scenario where an appropriately sized MAV would be able to navigate through an object. This metric quantifies solid volumes which can pass through the obstacle and validates the "through" categorisation of Path Groups. This means that the environment places a requirement on MAV size.

False Depthmap Noise
False Depthmap Noise indicates that the object contains a structural element that causes false depthmap noise. Uneven surfaces from the point of view of the depth sensor that has large normal variations can cause vast distance fluctuations in clusters on a depthmap. This can be confused with sensor noise and could be filtered out or even make the outputs of thresholded depthmaps look inconsistent. Take the depthmap of a large bushy tree as an example, the structure of the tree means that there are visible gaps between leaves and branches. These gaps allow light from the area behind the tree to pass through to the depth sensor. The depthmap of the tree will then have distance variations in the cluster of pixels that represent the tree. This can be interpreted as noise depending on the sensor's specifications and the design of its noise-filtering algorithm. This makes depthmap data more complicated to handle because this object property introduces noise which is not actually noise but valid measurements. Depending on how the obstacle avoidance algorithm processes the depthmap, this can indicate that trajectories exist which allow the MAV to fly through an object. For this reason, the obstacle avoidance algorithm needs to be able to filter trajectories whilst considering the MAV's size.

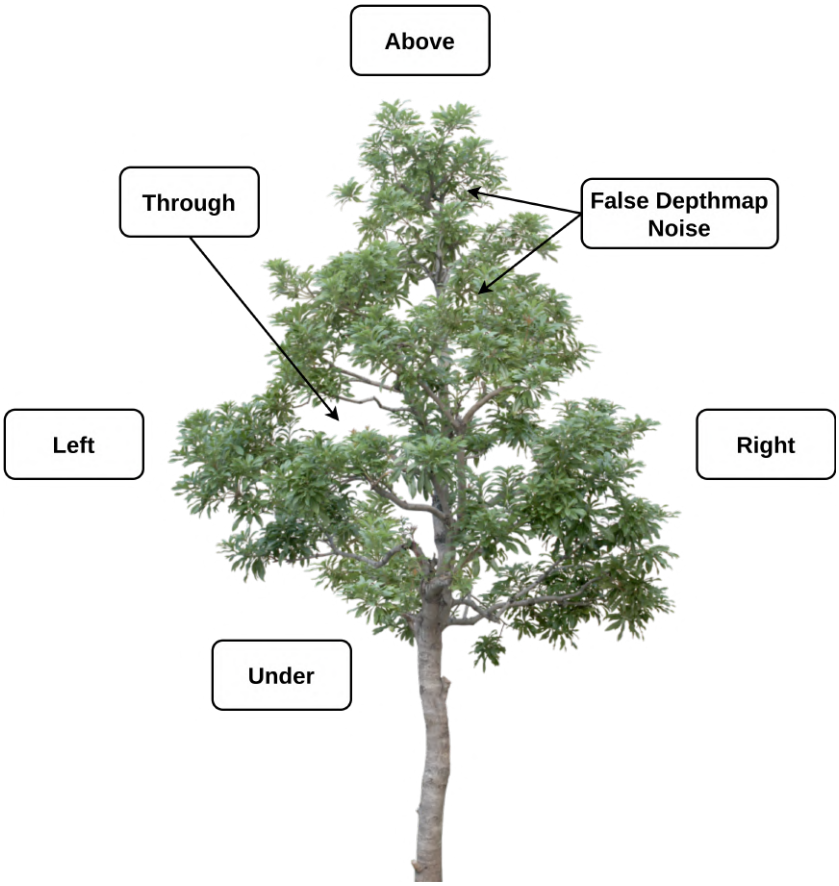


Figure 3.2: The Path Groups and False Depthmap Noise properties of an object relative to MAV navigation.

The difference between False Depthmap Noise and MAV Gaps is that MAV Gaps are an indicator that a MAV is physically able to fly through an object without collision and False Depthmap Noise allows a trajectory through the object with a high likelihood of collision.

Percentage of Total Height Volume

Each object occupies space in one of the vertical height volumes reducing the free space available for the MAV to move in. The exact volume of this space can become difficult to calculate due to the complex structures of objects found in nature. For this reason, the object's volume is calculated using variations of the formulas of simple volumetric shapes allowing for the approximation of object volume. This also takes into consideration that while free space may exist in an environment it may not always be traversable by the MAV.

Trees are a good example, when viewing the outer sides of the canopy of a tree, the leaves obscure the true structure of the branches behind them. The branch structure can create traversable paths through the tree but these paths will not be considered by MAVs that do not consider the general structure of trees. The branch structure may likely block all paths for the MAV, in which case the area is not traversable by the MAV. The actual volume of a tree will be smaller than its approximated volume but provides a more reasonable metric relative to MAV navigation because most of the free space between branches and leaves cannot be traversed by MAVs.

Admittedly there is a size component at play here, if the obstacle avoidance of smaller MAVs is to be evaluated then the MAV Gaps property can be used to test if the MAV can pass through sections of the tree and further refinements of how the tree volume and object complexity score is calculated can be made.

Once the approximate volume of an object is known then the percentage volume it occupies in the height volume can be calculated.

Object complexity Score Each object complexity property (Path Groups, MAV Gaps, False Depthmap Noise) is either present or not. If present, then a 1 is assigned if not present then a 0 is assigned. The scores are then averaged to give an object an overall complexity score.

The object complexity score is an indicator of the amount of valid traversable trajectory options available to the MAV to avoid an obstacle. The higher the complexity score the more traversable trajectory groups are allowed by the object's structure. Intuitively, it is easy to assume that the more trajectories available the greater the likelihood that an object is easy to navigate past.

This is in part true when evaluating the "left", "right" and "above" categorisations of Path Groups. These are the first and most straightforward trajectory groups to consider when attempting to navigate past an obstacle.

When considering navigating under or through an object like a tree, the trajectory introduces complexities that can push the sensor suite performance to its limits. Imagine a MAV moving through a tree, there are very thin branches that may be below the sensing threshold of a depth sensor. This is further exacerbated by the speed at which the MAV is moving and the distance and angle at which the MAV is viewing these thin branches. This means that the MAV might have a very tight window in which to sense and avoid the obstacle. So while a valid path may exist, it may lead the MAV into an area where it encounters more complex trajectories that are more difficult to track.

Defining object properties relative to MAV navigation allows for more properties to be added in the future and highlights the interdependencies of these object properties. Additional object properties would further help refine the object complexity score whilst retaining the same general environmental complexity classification framework.

Frequency of Occurrence

To create an environment that varies in complexity, objects with a defined complexity need to be placed in designated areas incrementally. This is how incremental level complexity is achieved in successive levels. Since object complexity is quantitatively defined per object, increasing the number of objects increases the overall level of complexity.

As the number of objects increases in a level the minimum distance between obstacles is reduced. The minimum distance between objects affects the maximum speed the MAV can achieve. This is because when the MAV is flying straight from the start of a level to the end of the level, the distances between objects in this direction can slow the MAV down. If these distances are small and objects are offset horizontally

from the point of view of the MAV, the MAV is forced to encounter these objects in successive rows. The MAV then needs to make tight turning circles or rapid ascents or descents to avoid the obstacles. To make trajectories more trackable the MAV would need to reduce its speed.

3.4.2 Selecting Algorithms to Evaluate

The ideal state of autonomous navigation of MAVs, is such that a user can use a MAV in any application and know that the MAV will reach its destination. The MAV should be able to traverse any environment and only be limited by its specifications. For this to be realised, obstacle avoidance algorithms must be able to cope with complex environments and navigate them as required.

To this end, if a user wants to deliver a package to a destination in a dense forest, the MAV should be able to accomplish this as far as its hardware allows. Applications involving time-sensitive tasks would require the MAV to move at high speed and in this scenario, successfully achieving the goal state is important.

As such, we evaluate a high-speed autonomous navigation algorithm in environments with varying degrees of complexity and assess the system's collision count whilst navigating to a desired destination. The algorithm chosen for evaluation is called FALCO [J. Zhang, Hu, et al. 2020]. This algorithm is chosen because of its performance in real-world experiments onboard constrained MAV systems and its ability to autonomously navigate environments at speeds of up to 10 m/s. FALCO generalises to ground and aerial-based autonomous navigation and systems with reduced computational complexity.

3.4.3 Experiment Steps

To evaluate the algorithm, the MAV will be required to navigate the simulation environment at a predefined target speed, this process will be repeated 10 times for each speed. The target speed will increase from 1 m/s to 10 m/s in increments of 1 m/s. This way, the obstacle avoidance algorithm can be evaluated in an unstructured environment with incrementally varying complexity, at various target speeds.

The evaluation procedure will be conducted using the following steps:

1. Select a predefined target speed.
2. Position the MAV at the starting point.
3. Start logging the evaluation metrics.
4. Allow the MAV to fly through the environment towards the finish area.
5. Repeat steps 2-4, 10 times.
6. Repeat steps 1-5, using the next target speed in step 1.

3.4.4 Evaluation Metrics

This evaluation focuses on a MAVs ability to sense and avoid obstacles in unstructured environments, whilst maximising speed and achieving a high success rate in

reaching a desired location. As such the following metrics are recorded:

- **Navigational Autonomy Success Rate**
  The navigational autonomy success rate is calculated by taking the number of times the MAV completed a level without collisions or manual corrections due to stalls and dividing by the number of times the MAV was tested on that level.
- **Collision Avoidance Failure Rate**
  Collision avoidance failure rate is measured by summing the number of collisions per level and dividing by the total number of experiments performed on that level. This provides a failure rate, where the higher the number the more collisions are encountered, and the lower the number the fewer collisions are encountered. Note that this is not a percentage because there could be more than 1 collision in every level which would cause the average to be higher than 1. The success rate is not measured because that would mean counting the number of times the MAV avoided an obstacle successfully. This indicates when collision avoidance worked correctly but we are interested when collision avoidance does not work correctly.
- **Time**
  Time taken to complete each level is measured and used as a rough approximation of the algorithm's ability to perform obstacle avoidance efficiently at the target speed. Time is measured from where the MAV crosses into a level to the point that it moves out of the level.
- **Speed**
  Speed is the average speed that the MAV moved at whilst navigating a level. If collision avoidance is performing well then the average MAV speed should be very close to the target speed set.
- **Collision Count**
  Collision count is measured to determine the collision avoidance failure rate and help find states that cause collisions. If the collision count is high at a specific region of the simulation environment then it is also likely that a specific characteristic of the environment, at that point in a level, is a limiting condition for a given algorithm. Collision count may also be correlated with speed accounting for another reason for failure.
- **Altitude**
  Altitude is measured so that the MAV can be tracked in scenarios where it traverses height volumes. It is very likely that in the more complex levels of the simulation environment, the MAV does not traverse the level in a straight line but in a path influenced by open spaces in its environment. For this reason, altitudes shall be measured at every third interval of the MAVs path taken in a level.
- **Manual Corrections or Stalls**
  Some systems allow for user intervention when navigation systems reach states where further navigation is impossible. This is in effect a stall in the navigational system. This is a deviation from autonomous navigation but MAV hardware is undamaged and the MAV is not forced into places it cannot navigate. However, no collisions have occurred meaning obstacle avoidance is still achieved. For this reason, manual corrections are logged. Manual corrections are not penalised in the same way that collisions are. Manual collisions indi-

cate that the user took control of the MAV to correct its position after a stall and then control was passed back to the navigational system so that it can complete the course. The time taken to correct the MAV position can vary based on the MAV position and user skill, both incurring a time penalty on their own, which is already accounted for when the level time is recorded. It is important to note that navigational stalls are preferred over collisions.

- **Percentage Target Speed Achieved**
  Since high-speed obstacle avoidance is being evaluated the MAV is required to achieve obstacle avoidance at high speeds. To force navigation at high speed a target speed is set. The MAV must try to attain the target speed but this may not always be possible. The MAV may need to reduce its speed so that Due to complex trajectories track to avoid obstacles. For this reason, the average MAV speed is divided by the target speed to provide a percentage score of how close to the target speed the MAV can achieve as environmental complexity increases.

Two indicators of navigational success are calculated. The first is the collision avoidance failure rate and the second is the navigational autonomy success rate. Differentiating between the two is important because a successful collision avoidance system may encounter states that cause navigational autonomy to stall. This stall state is a deviation from complete autonomy but not a failure of collision avoidance. These states are indicators that there could be limitations in the navigation system and help understand the origin and cause of these limitations. They provide focus areas for the improvement of autonomous MAV navigation as a whole.

3.5 Chapter Summary

In this section, we have indicated how the experimental procedure will be conducted and which evaluation metrics will be used. The simulation environment and experimental procedure, allow for the answers to the research questions to be found and may help adjust or refine our hypothesis. The overall structure of the simulation environment is described. Building the Environment in this way allows for incremental complexity as the MAV flies through each level of the courses. The course design caters for incremental complexity in altitudes by including height volumes. Incremental complexity in each level is achieved by assigning a complexity score to each object in a level and then increasing the number of those obstacles in successive levels. For height volumes, complexity is reduced as the MAV's altitude increases. This is because objects start from the ground and move upwards into the environment, as such incremental complexity is achieved naturally. The simulation environment tests two categories of object by providing two courses with distinct objects of different complexity. This will help narrow down the sources of potential bottlenecks of FALCO and evaluate obstacle avoidance in the presence of objects with differing complexities.

Chapter 4

Implementation

4.1 Introduction

To conduct the experiment described in the research methodology, components of the experiment need to be set up or created and then configured. These components include the simulation environment, AirSim, and the obstacle avoidance algorithm. This section details how the simulation environment was built and how AirSim and FALCO are configured.

4.2 Simulation Environment Design

The simulation environment is built in Unreal Engine 5.2. This allows for a high-fidelity environment to be built which can replicate real-world environments more realistically. This means that results from simulation tests that indicate obstacle avoidance problems are more likely to be mirrored in reality. Realistic simulations also help reduce the requirement to cope with the simulation to reality gap.

4.2.1 General Environment Layout

The entire simulation environment contains two courses. Each course is designed to loosely resemble environments found in reality. The Trees course is designed to replicate a forest inside a quarry, while the Pillars course is designed to replicate the Lena Pillars. An effort was made to ensure that these environments resemble reality because obstacle avoidance needs to be tested realistically to minimize the sim-to-real gap.

There is a caveat however; the addition of too many obstacles would allow environmental complexity to grow too rapidly, making it difficult to find performance bottlenecks. As such, care is taken to reduce the number of object categories so that environmental complexity can be more clearly tracked. The nature of stress testing one aspect of an environment is where the deviation from true environments begins. Environments in reality often have multiple types of vegetation, whereas the simulation environments created aim to have fewer obstacles. This enables environmental complexity to be established more clearly.

The Pillars course aims to loosely replicate an environmental occurrence that is not common. This provides the opportunity to stress test the obstacle avoidance algorithm against pillars as an object category with a reduced complexity score compared to trees. Multiple occurrences of the same object help expose the MAV to as many possible ways it would encounter obstacles while flying in reality. Object occurrence may not happen naturally in this way, but this state can be forced in simulation. This improves the stress testing ability of the simulation environment and could enhance obstacle avoidance algorithms by indicating bottlenecks to performance.

Ground foliage is not present because the goal is to make the environment contain fewer object categories. Instead a Ground Material Instance (shown in figure 4.1) is used to give the ground a relatively realistic look but is actually a flat surface. The introduction of ground foliage may introduce complexities that mask the source of bottlenecks in the obstacle avoidance algorithm. MAV control algorithms usually implement height restrictions which would not allow the MAV to fly into the ground.

Unreal Engine uses centimeters as the standard unit of measurement so care needs to be taken to ensure that levels are correctly sized and that objects are appropriately spaced. In terms of distances, the structure of each level is straight from the starting to the finishing position, so courses do not bend to the left or right. Each level is 100m in depth into the environment and 85m in width. Ideally, the MAV would fly in the straightest path possible to achieve the fastest level time possible. The MAV would likely be unable to find straight-line paths through all levels so a wide enough area is provided so the MAV can choose where it wants to fly.

The general layout of each course from start to finish goes straight for 1009m meters. Each level is divided by white beams which are situated on the ground of the simulation environment. Each beam is 1m thick to provide adequate visibility and timing points for intervals between levels. Each level is further divided into vertical height volumes increasing in complexity from the lowest to highest height volumes.

Boundaries, Starting and Finishing Areas and Level Divisions

The boundaries of each course are created with a quarry cliff face asset which is duplicated and reoriented to create long stretches of cliff faces as shown in figure 4.2. These cliff faces provide boundaries that help confine the simulation space to an area that resembles a quarry. These boundaries prevent the MAV from deviating too far from the goal position. Each course is built within this quarry cliff face. The general layout of the environment has starting and finishing areas for each course. Starting and finishing areas are simple 3D rectangles with a mossy grass Material Instance applied to them which give the effect of a patch of neatly trimmed grass with clovers in them.

Figure 4.1: Unreal Engine Material Instance used to give the ground a relatively realistic look.

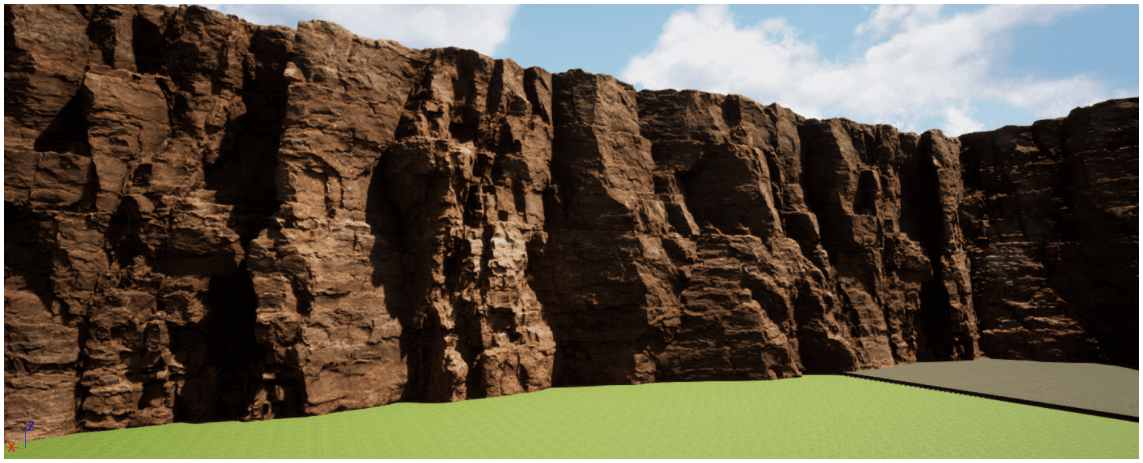


Figure 4.2: The image shows the course boundaries and start area used for each course.

Each course is further divided into ten levels by white beams. White beams in the simulation environment indicate level transitions to a more complex area within the environment. A white beam not only marks the end of one level and the beginning of another but also marks course timing points. These areas additionally help make it easier to discern where the MAV is at each level.

Unreal Engine Marketplace and Quixel Megascans

In Unreal Engine, objects are referred to as actors. Creating detailed actors that resemble reality is time-consuming and can require a very specialised skill set. To circumvent this, assets can be procured from the Unreal Engine Marketplace. The Marketplace is an online store where users can buy and sell asset packs and environments. There are many free assets and environments to choose from, but for this experiment, realistic assets are required. For this, Quixel Megascans are used, which are available in the Unreal Engine Marketplace at no cost.

Quixel Megascans are libraries of photo-realistic 3D assets, textures, and materials that have been created by scanning real-world items such as trees, plants, rocks, and much more. All the assets used in the simulation environment are from Quixel Megascan collections.

4.2.2 Building the Tree Stage

The Tree course is designed to replicate a forest inside a quarry. The idea behind the Tree course in the environment is to test how well the obstacle avoidance algorithm can avoid trees, given their complex geometry.

Trees are common in many unstructured environments and have various structures. How an obstacle avoidance algorithm decides to avoid trees can vary based on this structure. It is also important to understand how these trees are placed relative to one another. If the density of trees is too high then there will be no object-free paths to traverse and if the density is too low then the simulation environment will fail to adequately test high-speed obstacle avoidance algorithms.

Object Selection

The Tree environment uses three trees of two distinct species from the Quixel Megascan asset collection shown in figure 4.3. The first is short and very leafy, offering no traversable paths through it, and is used as a Low height volume blocker. The second is used twice but at different growth stages. The younger tree is used as a middle-height volume blocker and the older tree is used as a high-height volume blocker.

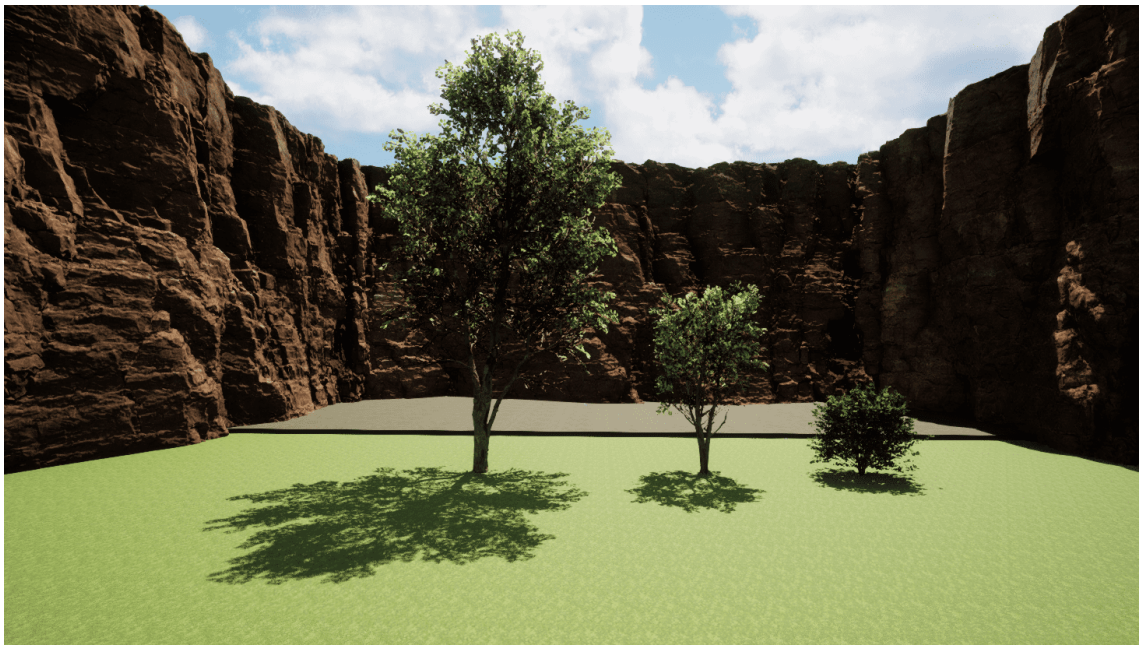


Figure 4.3: The three trees used as Low, Middle and High height volume blockers.

The tree type or species is not important because commonly found trees have very similar structures. Trees are a common occurrence in many environments and though we are accustomed to what they are and have a general idea of how they

work and what they do, the complexities of their geometric structure can easily be overlooked. In winter trees can be with or without leaves. They can occur in clusters that grow at various distances from one another based on the amount of sunlight, nutrients, and soil space available. Each tree in a cluster can be at various stages of growth so the spaces between branches are not the same despite the same type and structure of tree. The age of a tree is a determining factor in the quantity and size of its branches. This indicates that trees have very complex structures which vary relative to their environments. The complex structure of trees can provide interesting obstacle-free paths that a MAV can choose to navigate depending on the overall nature of their immediate environment.

Leafy trees are used which also have sections of leafless branches. Thin leafless branches are difficult to perceive which creates additional requirements for obstacle avoidance systems to cope with.

Object Complexity

In order to calculate the environmental complexity for each level, it is essential to establish a complexity score for each object. This requires a thorough assessment of the properties of each object and the assignment of an appropriate complexity score. Table 4.1indicates the average size of the trees used to create obstacles in the Trees course.

Blocker Type	Tree Type	Megascan Name	Height (Z)	Width (Y)	Depth (X)
Low Blockers	Hazlewood	`SM_CommonHazel_Field_03`	8m	6.8m	6.8m
Mid Blockers	Black Alder	`SM_BlackAlder_Field_03`	15.5m	6.4m	8.3m
High Blockers	Black Alder	`SM_BlackAlder_Forest_05`	34m	16m	18.3m

Table 4.1: Simulation environment tree types and their average sizes

The properties of each object are assessed and scores are then assigned. The scores are either 1 indicating that the property is present or 0 indicating that the property is not. Fraction scores are also assigned between 0 and 1 in the case of continuous property characteristics such as percentages. The following table indicates the outcome of this process:

Object Complexity Scores for Tree Course			
Height Vol. Obstacle	**Low Tree**	**Middle Tree**	**High Tree**
Path Group Categories			
Left	1	1	1
Right	1	1	1
Above	1	1	1
Under	0	1	1
Through	0	1	1
MAV Gaps (Sphere Diameter)			
0.5m	0	1	1
1.0m	0	0	1
2.0m	0	0	1
False Depthmap Noise	1	1	1
% of Total Height Vol. Est.	0.002	0.007	0.033
Complexity Score	0.400	0.701	0.903
Avg. Score	**0.668**		

Table 4.2: Complexity properties present for each tree in the Trees course

Path groups are assessed by checking if there are traversable trajectories to the left, right, above, under, and through the object. MAV Gaps are assessed by creating the relevant sphere size and checking whether those spheres can pass through sections of the tree. This verifies trajectory constraints relative to MAV size. False depthmap noise is assessed by checking to see if straight-line gaps can pass through the object. The "through" categorisation of Path Groups is validated by MAV Gaps. The size of the MAV model used in these experiments is 0.5m by 0.5m.

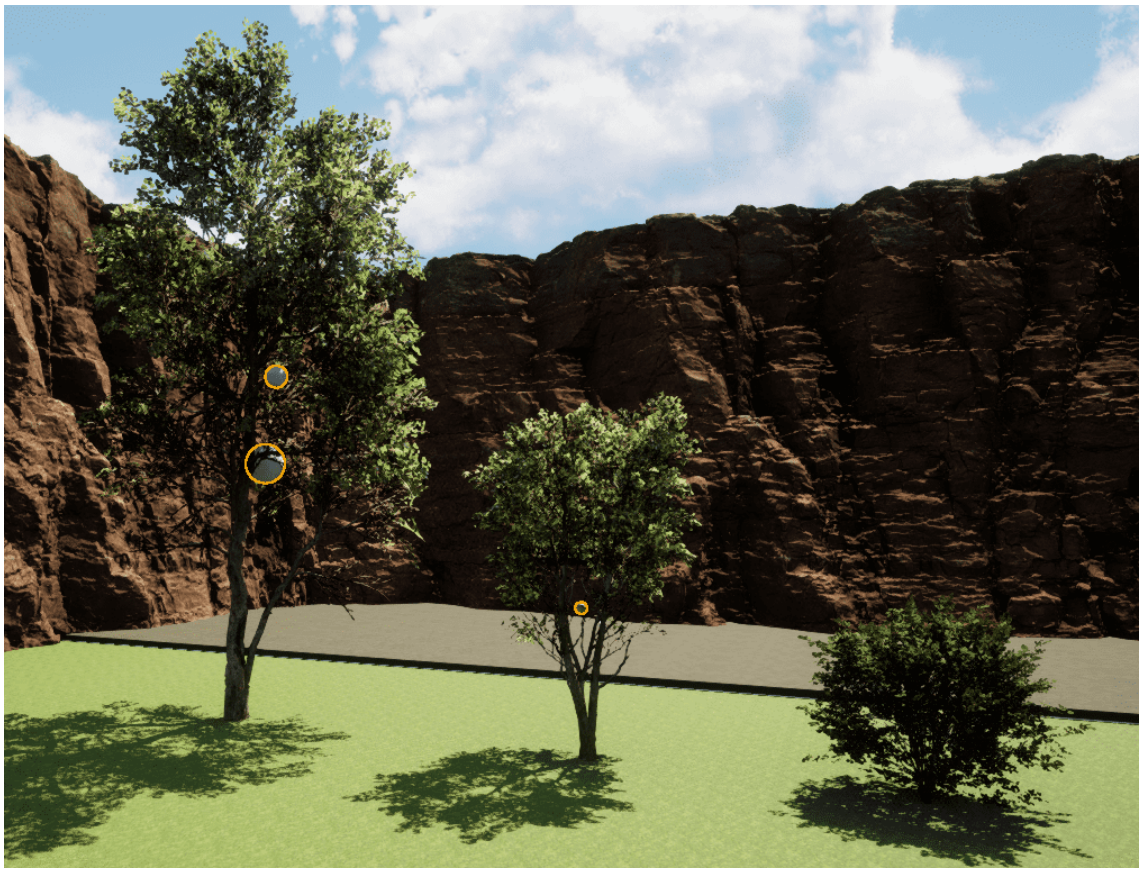


Figure 4.4: The three spheres are positioned at various positions inside the high and middle blocker trees. Each sphere is moved from the outside of the tree into the tree whilst making sure that the sphere does not touch any leaves or branches during the process. Each sphere is circled in orange in the image with the top-most to the bottom-most spheres having diameters 1, 2, and 0.5m

The Percentage of Total Height Volume property indicates the total volume that the tree occupies in each level's height volume. Tree volume is calculated using the following shape approximation formulae based on the tree's approximate shape:

Spherical: $\frac{3}{4}\Pi r^2$

Conic: $\frac{h}{6}\Pi r^2$

Note that for the calculation of the conic volumes, we divide the tree height by 2 to get a closer approximation of volume. This helps to account for free space between branches and leaves some of which are physically traversable by the MAV. The following two tables indicate the dimensions of each height volume and the approximate dimensions of each object in the Trees course:

Dimensions: Trees Course			
Height Volumes	**Low**	**Middle**	**High**
Lowest Point	0	8	15.5
Highest Point	8	15.5	34
Vertical Height Z (m)	8	7.5	18.5
Width Y (m)	85	85	85
Depth X (m)	100	100	100
Area (m^2)	8500	8500	8500
Volume (m^3)	68000	63750	157250

Table 4.3: The dimensions, areas, and volumes of each height volume in the Trees course

Obstacle Dimensions: Trees Course			
Height Vol. Obstacle	**Low Tree**	**Mid Tree**	**High Tree**
Height (m)	8	15.5	34
Width (m)	6.8	6.4	16
Depth (m)	6.8	8.3	18.3
General Shape	Spherical	Conic	Conic
Object Vol.	164.636	438.435	5236.074
% of Total Height Vol.	0.24%	0.69%	3.33%

Table 4.4: The dimension information of each tree used in the Trees course

The object complexity score is calculated by finding the average of the object property score of all object properties. This gives a score between 0 and 1, where numbers towards 0 indicate a lower complexity and numbers towards 1 indicate a higher complexity.

Level Complexity

There are 10 levels that vary in complexity where level complexity is a measure of object frequency of occurrence and the average complexity score of trees in that level.

The level complexity score is measured from zero to a maximum largely dependent on the number of objects per hectare. This imitates reality because the maximum number of objects per hectare is dependent on the size of the object. This allows the maximum level complexity score to be limited by what is permissible in reality at the intersection of object size and overlapping objects. The maximum level complexity score grows to limits defined by reality which can vary widely. This then generalises to any object.

Each level is 100m in depth and 85m in width, which means all levels and height volumes have an area of 0.85 hectares. The following table indicates the number of

trees present per hectare in each level and height volume as a quantifiable metric of object frequency of occurrence:

Frequency of Occurrence of Trees						
	No. Objects in Height Vol.			Objects per Cubic ha		
Levels	**Low**	**Mid**	**High**	**Low**	**Mid**	**High**
1	0	0	0	0.000	0.000	0.000
2	3	2	1	44.118	31.373	6.359
3	6	4	2	88.235	62.745	12.719
4	12	8	4	176.471	125.490	25.437
5	18	12	6	264.706	188.235	38.156
6	27	18	9	397.059	282.353	57.234
7	36	24	12	529.412	376.471	76.312
8	48	32	16	705.882	501.961	101.749
9	60	40	20	882.353	627.451	127.186
10	75	50	25	1102.941	784.314	158.983

Table 4.5: The number of obstacles present in each level and height volume per hectare.

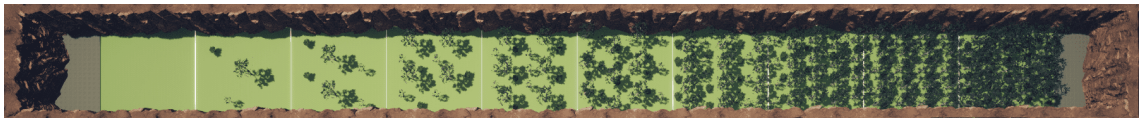


Figure 4.5: Top view of the Trees course showing how the frequency of occurrence of trees increases from level 1 to level 10.

The average object complexity is calculated and then multiplied by the frequency of object occurrence per hectare. This essentially weights the number of objects per hectare by the average object complexity. The complexity scores for each level and height volume are shown in the following table:

Overall Level Complexity for the Trees Course									
	Avg. Object Complexity			**Frequency of Occurrence**			**Level Complexity**		
Level	**Low**	**Mid**	**High**	**Low**	**Mid**	**High**	**Low**	**Mid**	**High**
1	0	0	0	0.000	0.000	0.000	0.000	0.000	0.000
2	0.668	0.802	0.903	44.118	31.373	6.359	29.474	25.161	5.745
3	0.668	0.802	0.903	88.235	62.745	12.719	58.949	50.322	11.489
4	0.668	0.802	0.903	176.471	125.490	25.437	117.898	100.644	22.978
5	0.668	0.802	0.903	264.706	188.235	38.156	176.846	150.966	34.467
6	0.668	0.802	0.903	397.059	282.353	57.234	265.270	226.450	51.701
7	0.668	0.802	0.903	529.412	376.471	76.312	353.693	301.933	68.935
8	0.668	0.802	0.903	705.882	501.961	101.749	471.591	402.577	91.913
9	0.668	0.802	0.903	882.353	627.451	127.186	589.488	503.221	114.891
10	0.668	0.802	0.903	1102.941	784.314	158.983	736.860	629.026	143.614
Average Height Volume Complexity							**280.007**	**239.030**	**54.573**
Course Complexity							**191.203**		

Table 4.6: Overall level complexity for different levels in the course of the tree. The Low height volume's complexity scores are the average of all the tree complexity scores multiplied by the Low height volumes frequency of occurrence. The Middle height volume's complexity scores are the average of the Middle and High tree's complexity scores multiplied by the Middle height volumes frequency of occurrence. The High tree is the only tree in the High height volume. The complexity score for this region is calculated by multiplying the high tree complexity score by the frequency of occurrence of trees in the relevant High height volume.

Notice that the complexity score increases in successive levels and from lowest to highest height volumes. Building the environment in this way means that the spatial frequency of occurrence of trees in each stage can be adjusted resulting in a quantifiable adjustment in complexity. This results in the ability to vary the complexity of each level as needed. The first level serves as a baseline level to validate that the simulation is working correctly. This is done by verifying that given an obstacle-free environment and a target speed, the MAV can achieve the target speed in the correct time over the 100m level distance. E.g.: A MAV with a target speed of 10 m/s should move at 10m/s and take 10s to complete the baseline level which is 100m from start to finish.

Object Placement

A MAV is not restricted physically in how high it can fly, as such it can choose to fly low or high to avoid obstacles. For this reason, obstacles are chosen to block each of the three height volumes and are simply categorised as Low, Middle, and High blockers. This is done to stress test the obstacle avoidance algorithm vertically. To stress-test obstacle avoidance algorithms horizontally, objects are spaced at specific intervals between one another. These rows of obstacles are then offset in the depth direction of each level to reduce the number of straight-line paths to the end of a level.

To simplify the repetitive task of tree placement required to achieve various frequencies of occurrence and offsets, an Unreal Engine Blueprint was created. Blueprints are essentially graphical coding environments that can be considered a function but can also be as complex as a class. The blueprint created for the clutter environment allows for copies of the selected object to be placed in a grid and for the adjust-

ment of the distances between objects in the X (depth) and Y (width) directions in the Unreal Engine coordinate system. Unreal Engine uses a left-handed coordinate system, which means the following:

- X: is the direction that moves you forward/backward in the environment.
- Y: the direction that moves you left/right.
- Z: the direction that moves you up/down.

The above is defined relative to the starting area of each course. The blueprint provides options to offset the odd and even rows which pushes the grid out of alignment. This helps reduce the perceived repetitive nature of the grid structure in the simulation environment helping the environment seem more realistic. Most importantly though, it reduces straight-line paths to the finish area. There is also the option to scale objects as needed, this helps adjust the height and perceived age of trees.

Once objects have been placed, some overlap occurs. Trees are then manually placed to reduce overlap which also helps break pattern repetition in the environment. Overlaps are not removed entirely but some minor overlaps in leaves and branches occur in the higher complexity levels which creates a more realistic simulation of reality.

4.2.3 Building the Pillars Course

When considering objects with simple structures as opposed to the complex structures of a tree, boulders, pillars, and rocks come to mind. Pillars work well because they naturally extend vertically from the ground and can reach various heights. This allows for Low, Middle, and High height volume blockers. The Pillars course in the simulation environment is built to loosely emulate the Lena Pillars.

Figure 4.6: (Top) Lena's stone forest. (Bottom) The Pillars course.

While this type of environment is less frequently encountered in nature, it allows for the use of a simpler object structure. This enables the comparison between two object classes, pillars and trees, of different complexities. The quarry cliff faces used in the Trees course are again used in the Pillars course as boundaries.

Object Selection

In Unreal Engine actors are made up of a mesh and 2D materials. Meshes are the 3D geometry that defines the shape of an actor. Materials define what a mesh looks like. For example, if you had a mesh of a tree trunk, a material would define the color, texture, and where artifacts unique to this tree trunk are positioned on the actual mesh.

Materials contain data such as UV maps and texture maps. These define where and how materials should be stretched over a static mesh so that realistic properties such as texture can be achieved. UV maps define how a material should be stretched over a mesh so that the material corresponds to the correct position on the mesh. For example, if you had a material that defined the skin of a person, you would not want their feet skin to be on their face geometry, correct UV maps would ensure that this would not be the case.

This course uses a limestone rock (`S_Small_Limestone_Rock`) shown in figure 4.7 from the Megascans library as the main obstacle.

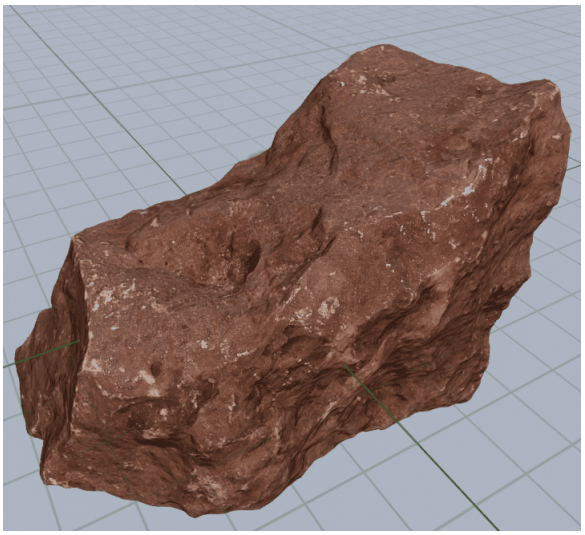


Figure 4.7: The base limestone rock unchanged, used to create pillars in the Pillar course.

By elongating the rock mesh and keeping the material properties the same, we essentially distort the geometry of the actor and the material instance. If the rock is distorted too much then the objects begin to look unrealistic as the material instance is stretched. To avoid this each rock group is distorted to an extent which still makes it plausible that they could exist in that state in reality. This rock is re-scaled in combinations of X, Y, and Z directions to form obstacles which create the complexity components for each height volume as shown in figure 4.8.

Figure 4.8: The three pillars used as obstacles for the Low, Middle and High height volumes.

Object Complexity

Pillars are relatively simple in structure in comparison to trees. They tend to be made of solid stone and are not as structurally complex as tree branching patterns. They tend to take on simple shapes which makes the approximation of their volume simpler. The following two tables show the dimensions of the height volumes in the pillars course as well as the dimensions for each pillar:

Dimensions: Pillars Course			
Height Volume	**Low**	**Middle**	**High**
Lowest Point	0	10	20
Highest Point	10	20	30
Vertical Height Z (m)	10	10	10
Width, Y (m)	85	85	85
Depth X, (m)	100	100	100
Area (m^2)	8500	8500	8500
Volume (m^3)	85000	85000	85000

Table 4.7: Dimensions of the Pillars Course

Object Dimensions: Pillar Course			
Height Vol. Obstacle	**Low Pillar**	**Mid Pillar**	**High Pillar**
Height (m)	10	20	30
Width (m)	5	6	7.3
Depth (m)	3.3	5.8	8
General Shape	Rect. Cube	Rect. Cube	Rect. Cube
Object Vol.	165	696	1752
% of Total Height Vol.	0.19%	0.82%	2.06%

Table 4.8: The dimension information of each pillar used in the Pillars course

The properties of each pillar are assessed and scores are then assigned. The scores are either 1 indicating that the property is present or 0 indicating that the property is not. Fraction scores are also assigned between 0 and 1 in the case of continuous property characteristics such as percentages. The following table indicates the outcome of this process:

Object Complexity Scores for Pillars Course			
Height Vol. Obstacles	**Low Pillar**	**Mid Pillar**	**High Pillar**
Path Group Categories			
Left	1	1	1
Right	1	1	1
Above	1	1	1
Under	0	0	0
Through	0	0	0
MAV Gaps (Sphere Diameter)			
0.5m	0	0	0
1.0m	0	0	0
2.0m	0	0	0
False Depthmap Noise	0	0	0
% of Total Height Vol. Est.	0.002	0.008	0.021
Complexity Score	**0.300**	**0.301**	**0.302**
Avg. Score	0.301		

Table 4.9: Object complexity scores for pillars in the Pillars course

Level Complexity

Just like in the Trees environment, the quantity of pillars per height volume is assigned such that successive levels contain greater quantities of pillars. This is done so that levels in the Pillars course can increase in complexity further into the course the MAV travels. The following table indicates the number of trees present per hectare in each level and height volume as a quantifiable metric of object frequency of occurrence:

Frequency of Occurrence of Pillars						
	No. Objects in Height Volumes			**Objects per Cubic ha**		
Levels	**Low**	**Mid**	**High**	**Low**	**Mid**	**High**
1	0	0	0	0.000	0.000	0.000
2	3	2	1	35.294	23.529	11.765
3	6	4	2	70.588	47.059	23.529
4	12	8	4	141.176	94.118	47.059
5	18	12	6	211.765	141.176	70.588
6	27	18	9	317.647	211.765	105.882
7	36	24	12	423.529	282.353	141.176
8	48	32	16	564.706	376.471	188.235
9	60	40	20	705.882	470.588	235.294
10	75	50	25	882.353	588.235	294.118

Table 4.10: The number of obstacles present in each level and height volume per Cubic hectare.

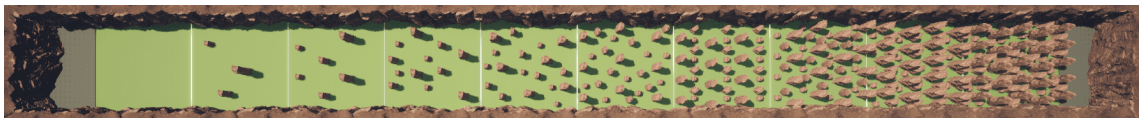


Figure 4.9: Top view of the Pillars course showing how the frequency of occurrence of pillars increases from level 1 to level 10.

The average object complexity is calculated and then multiplied by the frequency of object occurrence per hectare. This essentially weights the number of objects per hectare by the average object complexity. The complexity scores for each level and height volume are shown in table 4.11.

Overall Level Complexity for the Pillars Course									
	Avg. Object Complexity			**Frequency of Occurrence**			**Level Complexity**		
Level	**Low**	**Mid**	**High**	**Low**	**Mid**	**High**	**Low**	**Mid**	**High**
1	0	0	0	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3010	0.3014	0.3021	35.294	23.529	11.765	10.624	7.093	3.554
3	0.3010	0.3014	0.3021	70.588	47.059	23.529	21.249	14.185	7.107
4	0.3010	0.3014	0.3021	141.176	94.118	47.059	42.498	28.371	14.215
5	0.3010	0.3014	0.3021	211.765	141.176	70.588	63.746	42.556	21.322
6	0.3010	0.3014	0.3021	317.647	211.765	105.882	95.620	63.834	31.983
7	0.3010	0.3014	0.3021	423.529	282.353	141.176	127.493	85.112	42.644
8	0.3010	0.3014	0.3021	564.706	376.471	188.235	169.990	113.483	56.859
9	0.3010	0.3014	0.3021	705.882	470.588	235.294	212.488	141.854	71.073
10	0.3010	0.3014	0.3021	882.353	588.235	294.118	265.610	177.318	88.842
Average Height Volume Complexity							**100.932**	**67.381**	**33.760**
Course Complexity							**67.357**		

Table 4.11: Overall level complexity for different levels in the Pillars course.

Notice that the complexity score increases in successive levels and from lowest to highest height volumes. Building the environment in this way means that the spatial frequency of occurrence of pillars in each stage can be adjusted resulting in a quantifiable change in complexity. This results in the ability to vary the complexity of each level as needed. The first level serves as a baseline level to validate that the simulation is working correctly. This is done by verifying that given an obstacle-free environment and a target speed, the MAV can achieve the target speed in the correct time over the 100m level distance. E.g.: A MAV with a target speed of 10 m/s should move at 10m/s and take 10s to complete the baseline level which is 100m from start to finish.

Object Placement

The objects in the Pillars course are placed similarly as in the Trees environment. The same Blueprint is used to place the Pillars in a grid with offsets in the X and Y directions. Once objects have been placed, some overlap occurs. Pillars are then manually placed to reduce overlap which also helps break pattern repetition in the environment. In the Pillars environment pillars do not overlap at all.

4.2.4 Simulation Environment Design Summary

This section details how the simulation environment was built and discusses its overall layout. There are two courses in the simulation environment, Trees and Pillars, and both have 10 levels. Each level consists of three height volumes used to assess how the MAV makes ascents and descents to avoid obstacles. Each level is separated by white dividers and successive levels increase in complexity. Level complexity is assigned by the number of objects in a level, where each object has an assigned complexity score. A UE Blueprint was created to help with the placement of these objects but some manual adjustments were made to reduce overlapping objects. Building each course like this allows for the control of incremental complexity in succeeding levels which test obstacle avoidance as the MAV moves through each course. The two courses differ in overall complexity which helps verify whether collisions occur due to object complexity and how MAV speed is affected.

4.3 AirSim

The following section gives a more comprehensive description of what AirSim is, what sensor types are supported and how it is configured to capture data during experimentation.

4.3.1 Introduction

AirSim is a simulation plugin for Unreal Engine that takes advantage of Unreal Engine's high-fidelity graphic capabilities. This allows control strategies for autonomous vehicles in realistic simulations to be tested. Autonomous navigation research often relies on open-source libraries and frameworks. The Robotic Operating System (ROS) is a commonly used middleware framework for building and testing MAV systems.

To achieve computational performance, many developers and researchers use C++ for its memory efficiency or Python to quickly test out concepts, especially in the machine learning space. This means that AirSim needs to be compatible with as many of these tools as possible for obstacle avoidance algorithms to be used as they were built. This is especially true if the latest research is to be tested by community members without access to specific hardware used in the new research.

AirSim supports a variety of tools for autonomous navigation, most importantly ROS and ROS2. Airsim provides a ROS and Python API which is wrapped around the AirSim C++ API.

AirSim works by interacting with the UE environment and exposes endpoints that allow you to query the state of the environment, vehicle, or sensor data. This allows for environments to be built in UE and during gameplay, ROS nodes can interact with the UE environment to query MAV state, sensor data, and environment state. AirSim coupled with ROS and UE provides an ecosystem for photorealistic simulation environments to be built and autonomous navigation algorithms to be evaluated.

4.3.2 AirSim Sensor Suite

Autonomous navigation requires the use of sensors so that vehicles can perceive their environments and react accordingly. AirSim's API includes various sensor functions that can be used to obtain information about how the MAV perceives its environment. The following simulated sensors are provided by AirSim:

- Cameras
- Barometers
- IMUs
- GPSs
- Magnetometers
- Distance Sensors
- LiDARs

Each sensor has multiple properties that can be configured. Some properties are required so that the sensor can function properly and others are optional. Each of these properties is set in the settings.json file which is stored in the home directory inside the "AirSim" folder by default. Multiple sensors can be simulated as they would occur in reality by allowing users to configure their sensor suite.

A sensor uses the physical properties of the environment to take measurements of the environment. This means that sensors need to emulate the way that they work in the real world inside the simulation. Physical artifacts that occur in reality affect sensor measurements causing noise and sometimes erroneous sensor readings. Some of AirSim's sensor suites do not cater to these artifacts, the LiDAR sensor in particular. AirSim's Lidar sensor emulates a LiDAR sensor that works perfectly in simulation. This is a limitation because if the LiDAR sensor is exposed to large reflective surfaces then the simulation would indicate that the sensor perceives this surface as just another surface. In reality, however, this surface would cause the sensor to give erroneous depth measurements which are not present in LiDAR sensor outputs in AirSim.

4.3.3 AirSim Configuration

After building the environment, the default MAV frame used in AirSim was rescaled to 0.5m by 0.5m for a more realistically sized MAV. The settings.json file was then configured for use with Falco. Minor adjustments were made to the configuration file specified by FALCO to enable SI units when making API calls and turning off camera streams when rendering the environment during simulation.

Data Logging Node

The ROS integration was used to create a data logging node to log durations, calculate average speeds, and altitudes, and count MAV collisions during each level.

Whilst building the simulation environment, white beams were used to divide stages into levels. The beginning and ending edges of the side of the beams crossed by the MAV are used as timing points between levels. Each level in the simulation environment is timed providing an indicator of how fast the level is being traversed.

Collisions are an indicator of success and need to be minimised, as such they are counted. Collisions can be potentially catastrophic which is largely dependent on speed. The faster the MAV moves the more momentum it gains and the more likely it is that it would break if it collided with an object. For this reason, each collision is given a time penalty. This time penalty is calculated as follows:

$$\text{collision count} \times \text{target speed} \times \text{penalty time} = \text{collision time penalty}$$

where:

- **collision count** indicates the number of collisions that occurred.
- **target speed** indicates the speed that the MAV should try to maintain.
- **penalty time** indicates the time penalty in seconds is used to scale collision time penalties by speed.
- **collision time penalty** indicates the time penalty incurred by collisions.

The speed penalty time used is 5 seconds, this means that per collision at 2m/s, the MAV incurs a 10-second time penalty. Whether this time penalty is an accurate reflection of real-world recovery times is unclear because of the wide variance that collision recovery times can have. This can further be extended due to the complex nature of the environment in which collisions occur. Collisions at higher speeds cause greater impacts and can even cause the MAV to require repairs before the next flight can occur. In this experiment, catastrophic collisions are not taken into account and it is assumed that after a collision the MAV will recover. Collisions are failures of obstacle avoidance systems, as such they incur time penalties which are scaled by MAV speed.

MAV speed is logged to verify the percentage target speed the MAV can achieve whilst avoiding obstacles. The average speed per level is determined by logging instantaneous MAV speed as the MAV navigates a level. These speeds are then used to find the average MAV speed the MAV moved at whilst navigating the course. The MAV can traverse height volumes to avoid obstacles, as such, altitudes are logged to determine when this happens and what effect it has on obstacle avoidance success rate.

4.3.4 AirSim Summary

AirSim provides a photo-realistic simulation platform available for testing a variety of concepts in the MAV application space. AirSim provides the requisite sensor suites, integrations, and programmatic control over MAVs, enabling obstacle avoidance algorithm evaluation. After AirSim was configured and tested to verify that it was working, a data logging node in ROS was created. This node is used to capture information about the MAV as it navigates through the simulation environment.

4.4 Obstacle Avoidance Algorithm

The following section describes how FALCO works, why is was chosen and how it was setup to be used in this experiment.

4.4.1 Introduction

Collision avoidance strategies are constantly being improved upon because it is one of the major problems in autonomous navigation. However, there are not many widely used benchmarking simulation environments that are used to compare various obstacle avoidance strategies. If there were it would provide a baseline platform for comparison of the various strategies. This experiment can be seen as an exploration of this idea. A simulation environment has been created that defines reasonable aspects of obstacle avoidance algorithms to test. AirSim provides an interface between the simulation environment and obstacle avoidance algorithms via ROS. The next step is to choose an obstacle avoidance algorithm to test. This section of the implementation goes through the implementation of FALCO, the chosen obstacle avoidance algorithm for evaluation.

It must be noted though, that there are many different types of strategies autonomous navigation algorithms make use of. Some papers attempt to improve only parts of general autonomous navigation algorithms. FALCO is one such algorithm, with the primary aim of improving obstacle avoidance. So care should be taken when making comparisons between complete autonomous navigation systems.

4.4.2 FALCO

FALCO: **FA**st **L**ikelihood-based **CO**llision avoidance with extension to human-guided navigation, uses precomputed trajectories to reduce the real-time computational load on an autonomous vehicle. Based on the MAVs state, position, and a given goal position, Falco chooses trajectories based on their likelihood of reaching goal positions successfully.

Pre-computed trajectories are defined based on where the MAV can move when there are no obstacles present in the sensor's view of the environment. These spaces are determined from the depthmap generated by the depth sensor. Depth estimation is one of the most important inputs into the MAV control algorithm, which means that the type of depth sensor used is important. Depth sensors use various strategies to estimate depth and some of these strategies have known shortcomings. The depth sensor used by FALCO is a LiDAR sensor but can be changed as long as a depthmap can be generated.

Precomputed trajectories span vertical and horizontal directions relative to the depth sensor. When an object is in front of the MAV, some of the precomputed trajectories are occluded by obstacles. These trajectories are removed from the selection pool because they would cause collisions. A trajectory from the remaining pool is then chosen based on its likelihood of reaching the goal position successfully. This is how collision avoidance is achieved.

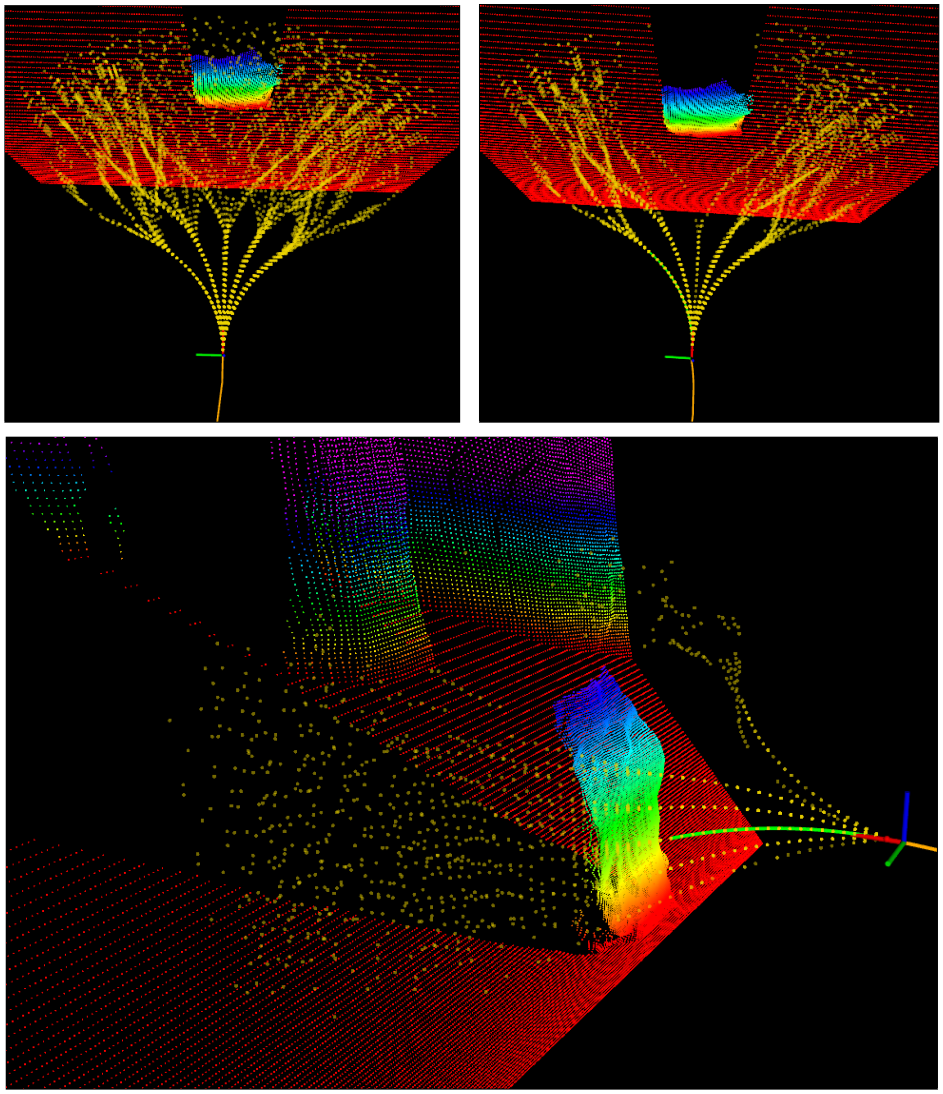


Figure 4.10: The precomputed trajectories visualised through Rviz. (Top-Left) The MAV encounters an obstacle and begins filtering out trajectories from the selection pool. (Top-Right) As the MAV gets closer to the object more trajectories are filtered out. (Bottom) Perspective view of the precomputed trajectories showing vertical trajectories being removed from the selection pool.

The system operates in the following modes of flight:

- Manual Flight
- Smart Joystick Flight
- Waypoint Flight

Manual flight allows for the control of the MAV via an X-Box controller and does not implement obstacle avoidance. This mode is for repositioning the MAV manually to a desired location in the simulation environment. Smart joystick flight allows for the control of the MAV with collision avoidance. In this mode, the user can control the MAV as long as the MAV is directed into an obstacle-free path. If the path is not obstacle-free then the system takes control of the MAV and makes path corrections which put the MAV on an obstacle-free path. Waypoint flight sets a series of goal

positions for the MAV to follow autonomously whilst employing collision avoidance.

The existing FALCO implementation was built using ROS and is simulated in Gazebo. It is common practice for code implementation of research ideas to be shared on version control platforms such as GitHub. FALCO's git repository can be found here, along with instructions on how to set FALCO up.

Hard-coded waypoints are set in the simulation environment to allow the MAV to fly through the courses using the waypoint flight mode. There are three starting positions and ending positions which vary in the Y direction. One waypoint for the starting position and one for the ending positions are randomly selected. This forces the obstacle avoidance system to use different paths through the levels in the simulation environment. The idea behind doing this is to vary the entry and exit points into and out of the environments.

FALCO takes a statistical approach to trajectory selection, assigning higher values to open spaces to improve the chances of reaching goal positions. This could result in the same paths continuously being selected. This is not a problem if these paths are optimal and the use of waypoints allows for the exploration into whether there is convergence to an optimal path or not.

If a path cannot be found through the environment then FALCO stops the MAV and leaves it hovering in position. This usually occurs when the MAV ends up in a position where it has stopped in front of a large obstacle. The obstacle blocks open spaces which FALCO uses to define trajectories. In these scenarios, the manual flight mode is used to reposition and switch the MAV back into the waypoint flight mode so that the MAV can continue the navigation process. This counts as a manual intervention and due to the nature of human error, incurs a time penalty.

4.5 Chapter Summary

In this Chapter, the design of the simulation environment was discussed. To create an environment that varies in complexity incrementally, a framework is defined which is used to create quantitative measures of complexity. Level complexity is scored based on object complexity and the frequency of occurrence of objects per level. Object selection is discussed and each object used to create complexity at each level is analysed to determine an object complexity score. The UE Blueprint design used to achieve the appropriate object placement is discussed. Once object complexity scores and frequency of object occurrences are defined, the final level complexity score is calculated. It is shown through the level complexity score that each course contains levels and height volumes that increase in complexity from the starting to finishing areas of both simulation courses. Object complexity scores and frequency of object occurrences are used to design both the Trees and Pillars courses. The general layout of both courses is explained and the overall course complexity difference is shown to be 16.808 for the Trees course and 6.729 for the Pillars course. Finally, the setup and configuration of AirSim and FALCO are discussed.

Chapter 5

Results

5.1 Introduction

The experiment requires the MAV to navigate 10 levels in each course in the simulation environment. The Trees course has a higher complexity score than the Pillars course showing that it is the more complex of the two. As the MAV navigates through the courses its speed, altitude, collision count, stalls, and navigation times per level are logged. Each experiment requires the MAV to maintain a target speed while moving through the course. Each successive level in both courses increases in complexity with level 10 being the last and having the highest complexity. The Setup section discusses how the experimental setup was achieved. The results section shows the processed data and discusses what the data indicates. Finally, artefacts discovered in FALCO are discussed.

5.2 Setup

This section discusses how each aspect of the experimental setup was achieved before running the experiment.

5.2.1 Simulation Environment Setup

The simulation environment contains two courses, both with starting and finishing areas. Both courses are next to each other in the simulation environment, such that the starting areas are offset horizontally. Opening the environment using the unreal engine editor allows you to start the simulation with AirSim enabled. When the simulation is started, AirSim spawns a quadrotor in the position where the PlayerStart instance is. The PlayerStart instance is an object in UE that sets the spawn point for the quadrotor when the simulation is started and is positioned whilst building the simulation environment. The PlayerStart Position is moved a set distance in the Y axis to move it between courses. All measurements are taken using the PlayerStart location as a reference. First, the simulation in UE is started, then the AirSim ROS node is run, and then the FALCO node. When the FALCO node starts, the MAV moves to the first waypoint in the starting area before the level begins. It then begins to fly to the next waypoint in the finishing area of the

course. This gives the MAV a running start so it can reach its target speed in an obstacle-free area before the baseline level begins.

The baseline level is the first level of both courses and is used to validate the physics of the MAV. This is done by ensuring that the MAV can approximate the correct distance in the correct time given the target speed.

5.2.2 AirSim Node Setup

The AirSim node is the ROS node that controls the MAV in the UE environment. FALCO passes control commands to AirSim topics which control the motion of the MAV. Without this node, FALCO would not be able to control the MAV or retrieve any data about the environment or MAV from the simulation. This node is essentially a bridge between ROS and UE via the AirSim API.

5.2.3 FALCO Setup

FALCO is built in ROS and so a launch file is used to start all the nodes to run the system. Once each node is started, the FALCO system begins to move the MAV towards the first waypoint. One of the nodes in the launch file starts up Rviz, which enables FALCO's pre-computed trajectories to be visualised. By being able to see the trajectories as the MAV moves through the course a better understanding of how FALCO performs can be gained.

5.2.4 Data Logging Node Setup

After the FALCO nodes start, the data logging node is started. When the MAV crosses the start position of level 1, the timing for that level begins. There are 10 experiments carried out per target speed from 1 m/s to 10 m/s in 1 m/s intervals. The data logging node records the time taken to navigate each level, the average MAV speed, collisions, stalls, and the MAV altitude throughout the course. The raw data for both courses can be found in the projects git repository.

5.3 Results

In this section, the data gained through experimentation is processed and shown along with graphs generated using the processed data. A brief explanation of each column in the processed data tables is given. The graphs are then analysed to better understand what the data means.

5.3.1 Results: Trees Course

The hypothesis is that the faster the MAV travels the faster its reaction time needs to be to avoid obstacles. As such, we should observe an increase in collisions as the target speed is increased and the MAV is required to navigate past a threshold complexity level. The average number of collisions per level at a target speed is the collision avoidance failure rate. For the following discussion, it is more intuitive to understand the actual collision counts per level. For this reason, figure 5.1 is used when discussing the results, which is essentially a scaled version of the average number of collisions per level vs target speed.

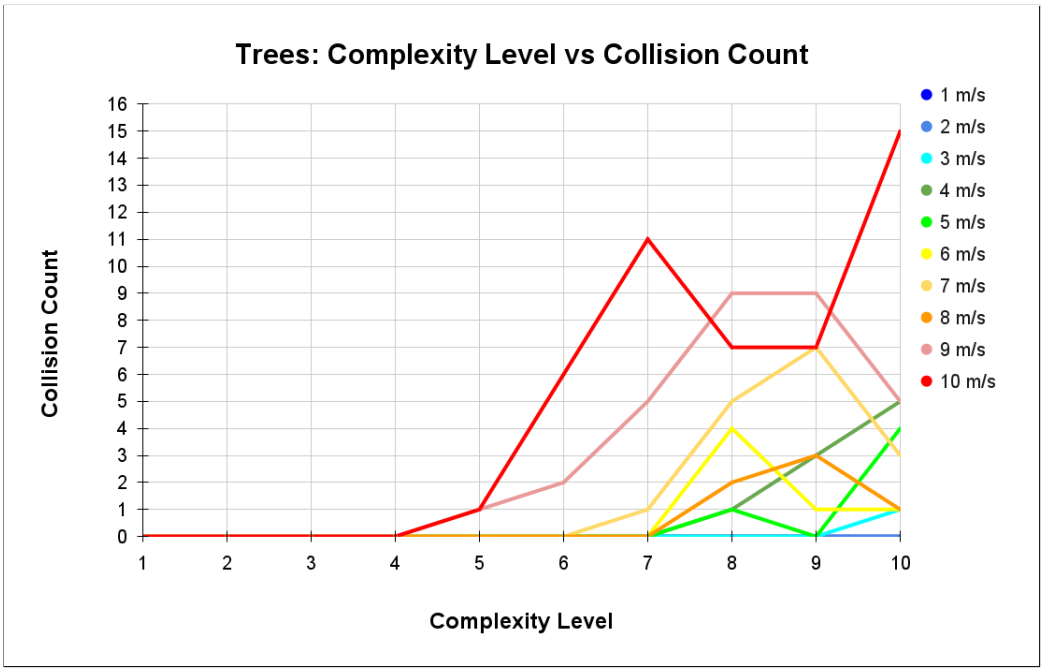


Figure 5.1: The average number of collisions increases as speed and level complexity increase in the Trees course. Note that at target speeds of 1 and 2 m/s, the MAV does not incur any collisions.

Figure 5.1 is examined to find out when the first collisions occur relative to MAV speed. It illustrates that there are no collisions for all the levels when the target speed is 1 or 2 m/s. This means that the bottleneck complexity level for target speeds 1 and 2 m/s is not reached. Recall that the target speed is the speed that the MAV aims to achieve whilst prioritising collision avoidance. Collisions begin to occur at level 10 for a target speed of 3 m/s. This indicates that the threshold complexity score for 3 m/s begins at complexity level 10. For target speeds 4, 5, 6, and 8 m/s, collisions begin to occur at level 8 indicating that the threshold complexity score for these speeds is being reached. Collisions begin to occur at level 7 for a target speed of 7 m/s and at level 5 for target speeds of 9 and 10 m/s. This shows that in general, as the target speed increases, collisions are encountered sooner in the Trees course. This means that increasing the target speed reduces the obstacle avoidance success rate. It also shows that lower target speeds have higher level complexity thresholds allowing the MAV to navigate more complex environments.

The table below summarises the information given above and shows the environmental complexity threshold scores per target speed.

Target Speed Complexity Thresholds		
Target Speed (m/s)	**Env. Complexity Threshold**	**Level**
1	-	undefined
2	-	undefined
3	736.860	10
4	471.591	8
5	471.591	8
6	471.591	8
7	353.693	7
8	471.591	8
9	176.846	5
10	176.846	5

Table 5.1: The table indicates the complexity threshold for each target speed.

This shows that as the target speed is increased the environmental complexity level at which the MAV can navigate is reduced. This results in a higher collision count as MAV speed and environmental complexity are increased. This also indicates that at lower speeds the MAV is able to navigate more complex environments successfully.

There are a few deviations from this trend though. Figure 5.1 indicates that at a target speed of 10m/s, the collision count at complexity levels 8 and 9 is lower than complexity level 7. A similar occurrence is present between complexity levels 9 and 10 at target speeds of 7, 8, and 9 m/s.

In the altitude range from 5-8m, the MAV encounters various positional configurations of the Mid, Low, and High Trees. The Mid trees are more easily avoided but the combination of the Low and High Trees creates a complexity concentration at a specific altitude. When the MAV encounters the High tree at this altitude, it can choose to navigate around the left or right side of the tree as well as through the tree. This decision is based on obstacle-free trajectories but also depends on the MAVs angle of approach.

Given these three choices, the paths taken that result in the least collisions are the left and right path groups completely around the High tree. When the MAV flies through the MAV gaps of the High tree, it can choose to navigate the four MAV gaps shown in Figure 5.2 by white spheres.

In complexity level 9, target speeds 7, 8 and 9 m/s have average altitudes of 5.714, 5.890, and 7.120m. In this altitude range in complexity level 9, there is an overhanging branch (starting at 7m) on the left-hand side of the High tree. This branch contains a MAV gap (between 5.5 and 6.6m) along with a group of fine branches at the same position. This MAV gap is illustrated by sphere B in 5.2. The smaller branches create false depthmap noise which strains the perceptron action loop. The MAV gap also occurs at an angle, this can further complicate navigation depending on the approach angle of the MAV. The combination of the MAV gap and the overhanging branch causes a complexity concentration within the Low Height volume

and causes the collision count to increase.

In complexity level 10 some of the target speeds have better collision avoidance than in complexity level 9, particularly target speeds 7, 8 and 9 m/s. When we examine the average altitudes of these target speeds in level 10 we see that they are 7.443m, 6.937m, and 7.858m. This altitude range in complexity level 10 allows for a good obstacle-free trajectory balance between the top of the Low tree and the bottom of the overhanging branch on the right side of the high tree. This particular overhanging branch is located at ±8m of the High tree. A MAV gap is created between the Low tree and the overhanging branch. The MAV gap is between 6.9m (6.4) (above the top of the Low tree) and 7.9m (below the branch) and is shown as sphere D in figure 5.2. This MAV gap allows for collision-free paths to be found more easily resulting in a reduced collision count.

This indicates that target speeds 7, 8, and 9 m/s, favored navigating the right side of the High tree in complexity level 10 but favored the left side of the High tree in complexity level 9. This results in fewer collisions at complexity level 10 than at complexity level 9 explaining anomalies at target speeds 7, 8, and 9 m/s.

Note that these conditions are not as difficult to navigate when there are fewer objects in a level, creating wide open spaces around each object. In the higher complexity levels, there are more obstacles, and the navigation space around each obstacle is reduced making obstacle-free navigation more difficult. This is in part why there are fewer collisions at lower levels.

Figure 5.1 indicates that at a target speed of 10m/s, there are a few anomalies as well. In level 7 there are a high number of collisions but in levels 8 and 9 the collision count is reduced. The average altitudes in these levels are 6.796, 6.501, and 6.484m. The high collision count in level 7 can then be attributed to the MAV favoring navigation on the left side of the High tree and suffering a higher collision count due to the overhanging branch above MAV gap B. In levels 8 and 9, the MAV favors the right side of the tree where it encounters fewer collisions.

The average altitude of the MAV at complexity level 10 and a target speed of 10 m/s is 7.526m, which would seem to indicate that it should encounter fewer collisions but in fact, encounters the most collisions. If at this altitude the MAV chose to navigate the left side of the tree then it would have encountered two additional MAV gaps (see Figure 5.2). MAV gap A requires navigating even more fine branches than MAV gap B. If MAV gap C is chosen to be navigated then the MAV would need to make an extremely aggressive turn because it would be moving at high speed towards a thicker part of the main trunk of the tree. These two MAV gaps are significantly more difficult to navigate and cause an increase in collision count.

It is noted that as the target speed increased throughout the experiments the MAVs ability to slow down when an obstacle was encountered, was reduced. This reduction in the MAVs ability to reduce its speed meant that it could not avoid obstacles as well as lower target speeds.

The combination of the MAVs reduced ability to slow down at high speeds, the complexity concentrations, and the increased object frequency of occurrence creates a complex navigational scenario. At the intersection of these complexities, it can be difficult to isolate the exact source of obstacle avoidance failure.

For example, at target speeds of 5 and 6 m/s there are more Collisions at level 8 than there are at level 9. The altitudes at these target speeds at level 8 are 5.367 and 5.457m and the altitudes at level 9 are 5.233 and 5.509m. At these altitudes, the MAV can navigate through MAV gap B or collide with the small branches seen around MAV gap B. The reduced speed increases the chances of the MAV successfully navigating this MAV gap but encountering the fine branches still strains perception. At these intermediate target speeds, the target speed environmental complexity threshold is also reached.

When the MAV chooses to navigate the left side of the High tree then it incurs more collisions because the path is more complex. On the other hand, if the MAV chooses to navigate the right side of the High tree then it incurs fewer collisions. These conditions are then exacerbated as the MAV speed and number of objects in a level increase.

The reason why the MAV decides to navigate the left side of the tree over the right side, is based on the approach angle and the time the MAV is given to select a precomputed trajectory. The variation of the MAVs approach angle changes the perspective the MAV has of the High and Low tree configuration. This change in perspective allows for some paths to be perceived as less complex than they are. Collisions occur because once the MAV chooses these more complex paths, it positions itself such that aggressive maneuvers or intricate traversals are required to avoid collisions. This means that the perception-action loop needs to have an appropriate latency to cope with these requirements. The time optimality in which the perception-action loop can sense obstacles, select the best trajectory and timeously track the selected trajectory is an important indicator of obstacle avoidance success in these complex environments.

As we can see, the environmental complexity has a direct effect on the perception-action loop latency. If a quantitative relationship between the environmental complexity score and the perception-action loop latency can be established, then perception-action loop latency could be a direct indicator of a MAVs ability to navigate a complex environment. This would provide an indicator of obstacle avoidance performance before physical testing and could reduce design time and costs. This of course necessitates the need for a comprehensive environmental complexity scoring framework.

Figure 5.2: The High tree and white spheres are highlighted and separated from the Trees course for clarity. **Sphere D** on the right side of the tree shows the MAV gap where the reduced complexity can be found. Here the MAV flies between the top of the Low tree and underneath the branch. **Sphere B** to the left of the High tree shows the MAV gap located between 5.5 and 6.6m and the surrounding fine branches. **Spheres A and C** indicate the MAV gaps encountered when the target speed was 10 m/s and anomalies occurred. **Sphere A** indicates a MAV gap that is surrounded by fine branches whilst **Sphere C** indicates a MAV gap that requires aggressive manurers to navigate.

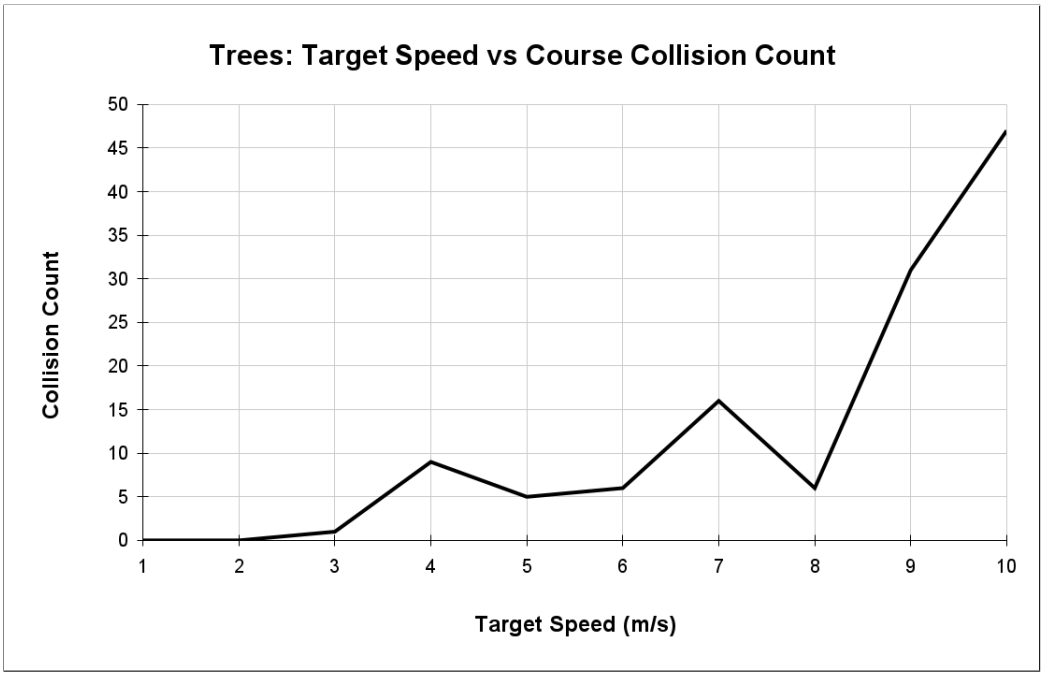


Figure 5.3: The number of collisions that occur over the entire course given a specific target speed.

Figure 5.3 shows the total collisions which occur over the entire Trees course, at a given target speed. The same trend is observed where collisions increase as the target speed increases but there is something more happening. There are more collisions at target speeds 4 and 7 m/s than at target speeds 5, 6, and 8 m/s. This seems to go against the current argument but can be explained by analysing the altitude data.

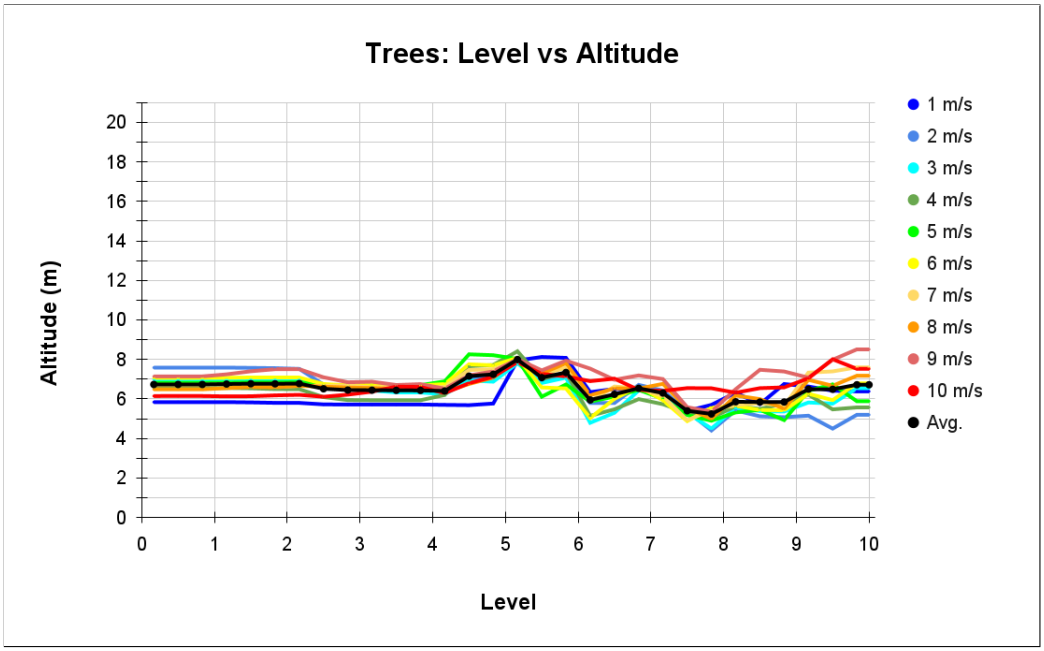


Figure 5.4: This figure shows how the MAV changes altitude as it moves through the course levels at a set target speed. Note that each number on the x-axis indicates the end of the corresponding level and the beginning of the next level. So the start of level 1 is indicated by 0 on the horizontal axis and 1 indicates the end of level 1 and the beginning of level 2. The vertical lines divide the course up into levels accordingly.

Figure 5.4 shows the average altitude of the MAV in all ten experiments per target speed. The black line shows the average altitudes at all speeds and for clarity is plotted in isolation in figure 5.5. Examining the average altitude of the MAV in figure 5.5 reveals that the MAV begins to ascend to a higher altitude throughout level 5 until the beginning of level 6 where it reaches an altitude of ±8m. From this point onwards it starts to descend until the beginning of level 7 to an altitude of around 6m. From here it maintains a relatively constant altitude between 6 and 6.5m until the beginning of level 8 at which point it begins to descend to ±5.2m towards the end of level 8. At this point, the MAV ascends to and maintains an altitude of between 5.8m and 6.8m until the end of the Trees course. The average altitudes in levels 4 and 7 are **6.438m** and **6.242m**. The objects in these levels and altitudes are further examined to find an explanation for the higher collision counts.

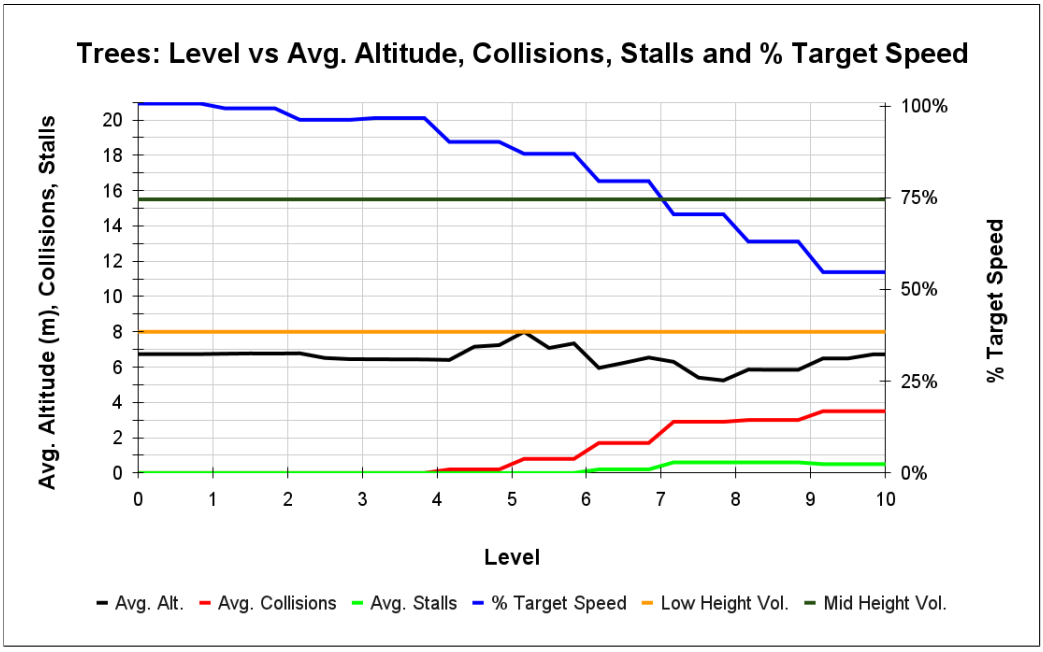


Figure 5.5: The left axis indicates the average number of collisions and stalls as well as the average altitude that the MAV flew at during each level of the Trees course. The right axis indicates the average percentage target speed the MAV attained for each level.

The High trees have an overhanging branch on their left side which begins at around 7m (was 6.8m) in altitude. The overhanging branches of the High trees form MAV gaps between the branches that the MAV moves through during simulation. This section on the left side of the High tree creates MAV gaps, false depthmap noise, and a "U" shaped overhang below the 7m altitude mark. There are also very fine branches in this overhang which further strains the perception capabilities of the LiDAR sensor. The mid-point of the MAV gap created by these branches is at **±6.1m** in altitude. The MAV is observed navigating through levels 4 and 7 at an average altitude of **6.438m** and **6.324m** which is very close to the midpoint of the MAV gap of the overhanging branch. Whereas, the average altitude on levels 5, 6, and 8 are 6.934m, 7.472m, and 5.648m. This indicates that the MAV is flying to the right of the High tree in levels 5 and 6 and below the MAV gap on the left side of the tree. When navigating through the complexity concentration on the left side of the High trees, flying lower is safer because MAV gaps quickly get wider and turn into obstacle-free areas. Flying above the **±6.1m** but below the 8m altitude mark indicates times when the MAV usually flew to the right of the High trees. The MAV favored lower altitudes with more open spaces in the Trees course. It barely ever leaves the Low height volume and on average only briefly moves into the Middle height volume at the beginning of level 6. The higher collision counts occur because FALCO chooses to navigate the more complex altitude at ±6.1m on the left side of the High tree in levels 4 and 7.

The following table shows the level complexity score based on the average altitude of the MAV whilst navigating the Trees course:

Level Complexity Based on MAV Altitude			
Level	Avg. Alt. (m)	Height Volume	Complexity Score
Level 1	6.735	Low	0.000
Level 2	6.764	Low	29.474
Level 3	6.582	Low	58.949
Level 4	6.438	Low	117.898
Level 5	6.934	Low	176.846
Level 6	7.472	Low	265.270
Level 7	6.242	Low	353.693
Level 8	5.648	Low	471.591
Level 9	5.850	Low	589.488
Level 10	6.567	Low	736.860

Table 5.2: This table shows the level complexity at the average height that the MAV moved at while navigating the Trees course.

Table 5.2 indicates the complexity scores given the MAV position. In the Trees course, many collisions can be attributed to the concentrated altitude complexity that occurs at the upper part of the Low height volume. If the Low height volume was further divided into smaller intermediate volumes a better approximation of how environmental complexity varies in altitude can be defined. This would capture the concentrated altitude complexity discussed previously. The concentrated altitude complexity results from a combination of tree properties and the spatial frequency of occurrence of these trees. This indicates that more complex objects can cause complexity concentrations at altitudes where object properties intersect. This shows that height volumes should be more granular and account for object properties that affect the complexity scores of these granular height volumes. When comparing complexity scores to collision counts in the Trees course, altitude and the concentrated altitude complexity volume should be taken into account.

Levels 3 and 10 are interesting because the MAV flies at an altitude of $\pm$6.6m which is close to the complexity concentration on the left side of the High trees discussed which causes collisions. Yet fewer collisions are observed in level 3 than in level 10. This is because level 10 has twelve times more occurrences of the High tree than level 3 does. These observations help differentiate between collisions caused by the overhanging branch of High trees and the frequency of occurrence of objects in a level. The general trend of an increase in collision count as target speed is increased is observed. The results indicate that the obstacle avoidance failure rate is a function of the artefacts present in the obstacle avoidance algorithm and environmental complexity. It additionally indicates that optimal MAV speed is a function of environmental complexity.

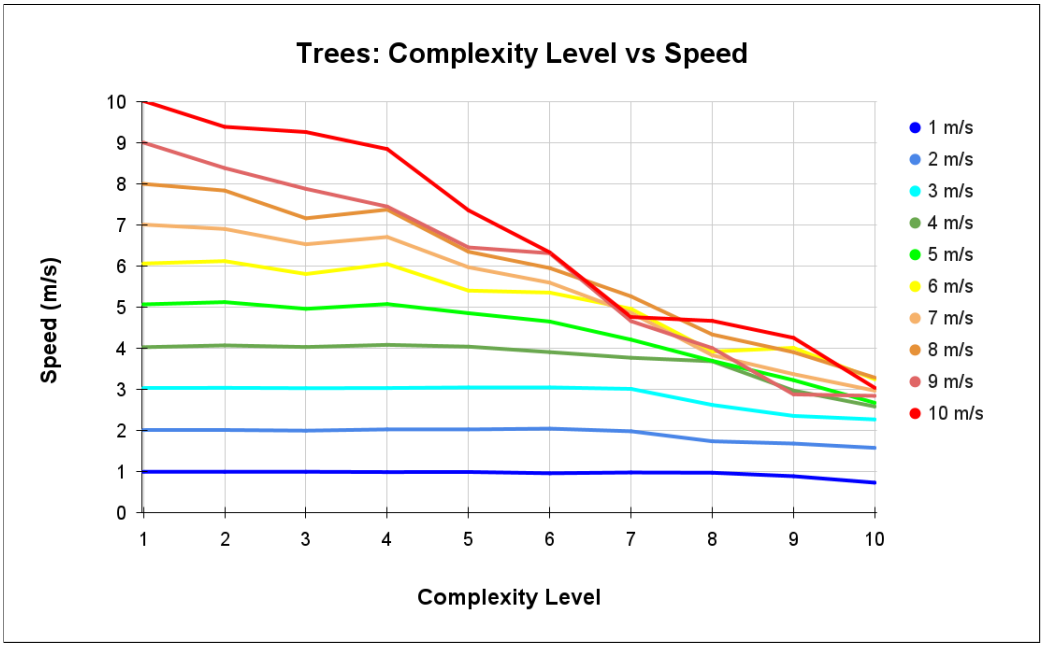


Figure 5.6: The graph shows the average speed the MAV traveled at whilst navigating the level and attempting to maintain the target speed without collision.

Figure 5.6 shows that as the MAV navigates through the course and encounters obstacles, its speed begins to decrease. Level 1 is a baseline level with no obstacles and is used to verify that the MAV can traverse the 100m distance very close to the target speed as indicated in the figure. There are some slight deviations from this target speed in the baseline level. This can be caused by the fight controller oscillating around the target speed as it adjusts MAV speed to try to reach the target speed. The MAV speed is calculated by taking the average of the instantaneous speeds of the MAV as it navigates the course. This means that MAV speed is an approximation and some minor deviations from the exact speed are expected.

If the MAV begins to encounter obstacles from level 2 onwards, it then deviates from a straight-line path to avoid these obstacles, and speed reductions are observed. This is not the case for all target speeds, the lower target speeds can track collision avoidance trajectories and maintain the target speed until a bottleneck level complexity is encountered. Recall that the level complexity is derived from a combination of object complexity and frequency of object occurrence. At a target speed of 1 m/s, the MAV begins to slow down from level 9 to level 10. This indicates that the bottleneck level complexity for the target speed of 1 m/s occurs at level 9. Beyond this level complexity, the MAV will not be able to maintain the target speed. A similar observation is made with target speeds of 2 to 6 m/s with the only difference being that the bottleneck level complexities occur sooner, as the target speed increases. For target speeds 7 to 10 there is an immediate reduction in speed from level 2. This is because the trajectories required to avoid obstacles become more difficult to track and the MAV reduces speed to track the trajectories more accurately.

Another interesting observation is that at target speed 8, the average speed on complexity level 3 is lower than complexity level 4. This can sometimes occur because the MAV might position itself such that if it flies straight it does not encounter obstacles until the end of the level. Another possibility is that the MAV moves to a less

complex altitude with fewer obstacles or the complexity of the shape of the obstacle is reduced creating more navigational space. Examples of these two scenarios can be accomplished by moving to higher-height volumes where complexity is reduced by reducing the object frequency of occurrence. The MAV can also move to lower altitudes in the Low height volume where only the trunks of the Mid and High trees are encountered. Figure 5.5 indicates that the MAV maintains an average altitude of ±6.5m, oscillating around this point, attempting to find trajectories with reduced complexity. At this altitude, the MAV can fly above the Low trees, below the first overhanging branch of the High trees, and avoid the section of the Mid trees which are not traversable. This shows that the MAV attempts to maintain altitudes that reduce complexity, which results in the MAV being able to maintain a more optimal balance between speed and complexity.

While figure 5.6 is insightful it does not account for speed reductions due to collisions.

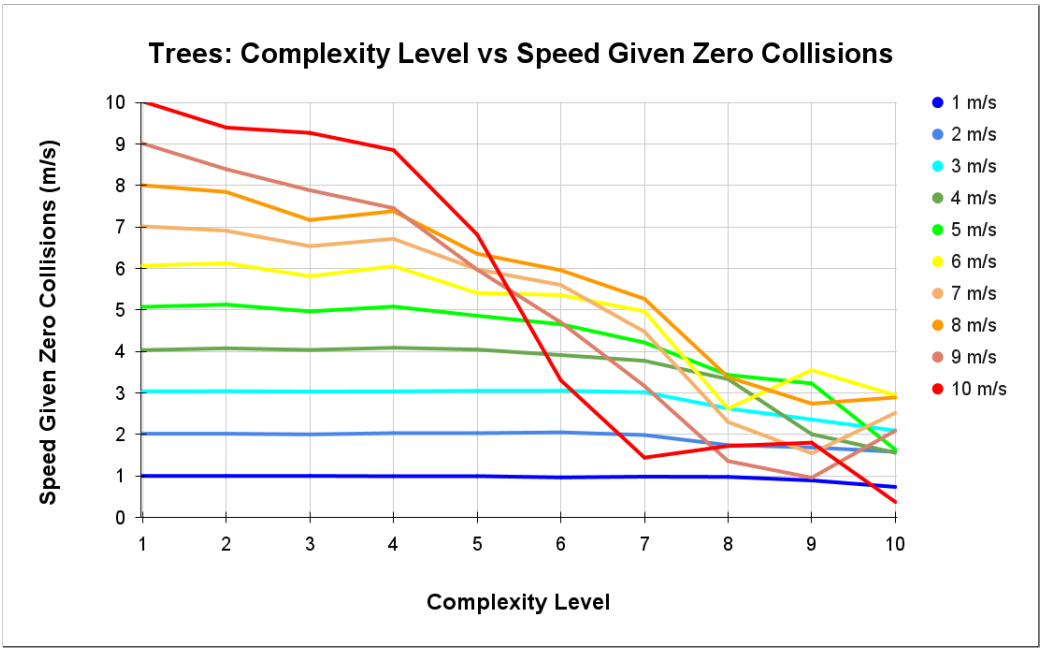


Figure 5.7: This graph indicates the time the MAV took to navigate each level of the course at a set target speed when no collision occurred.

Figure 5.7 is plotted using collision-free experiment data and eliminates the need to consider collisions so that MAV speed can be evaluated in isolation. In general, the average gradient of each target speed curve becomes more steep as the level of complexity increases, indicating that a lower speed is required to navigate more complex levels successfully using FALCO for collision avoidance.

If the actual MAV speed was not limited by the target speed, it would be limited by complexity levels or MAV hardware. MAV hardware can easily allow for speeds above 10 m/s so level complexity is evaluated. MAV speed reductions are seen in figure 5.7 as level complexity increases. The recommended MAV target speed ranges given level complexity are shown in the following table:

Recommended Speed Ranges Per Level			
Level	**Speed Range (m/s)**	**Height Volume**	**Level Complexity Threshold**
Level 1	10+	Low	0.000
Level 2	6-7	Low	29.474
Level 3	5-6	Low	58.949
Level 4	6-7	Low	117.898
Level 5	4-5	Low	176.846
Level 6	3-4	Low	265.270
Level 7	1-1.5	Low	353.693
Level 8	1-1.4	Low	471.591
Level 9	0-1	Low	589.488
Level 10	0-0.4	Low	736.860

Table 5.3: The table indicates the recommended speed ranges for successful collision avoidance given an environmental complexity threshold score.

The speed ranges defined in table 5.3 are obtained by taking note of the speed at which the first speed reduction is observed and the associated level in figure 5.7. This provides an upper limit that confines the optimal speed. The target speed before the upper limit becomes the lower limit above which the optimal speed can be found.

This is not to say that collisions would not occur at all in these levels at these speeds but that the chances of collisions are significantly reduced. This is because the experimentation sample space was limited to ten experiments per target speed. Note that level 1 is the baseline level which does not contain any obstacles, this means that the maximum MAV speed is limited by MAV hardware.

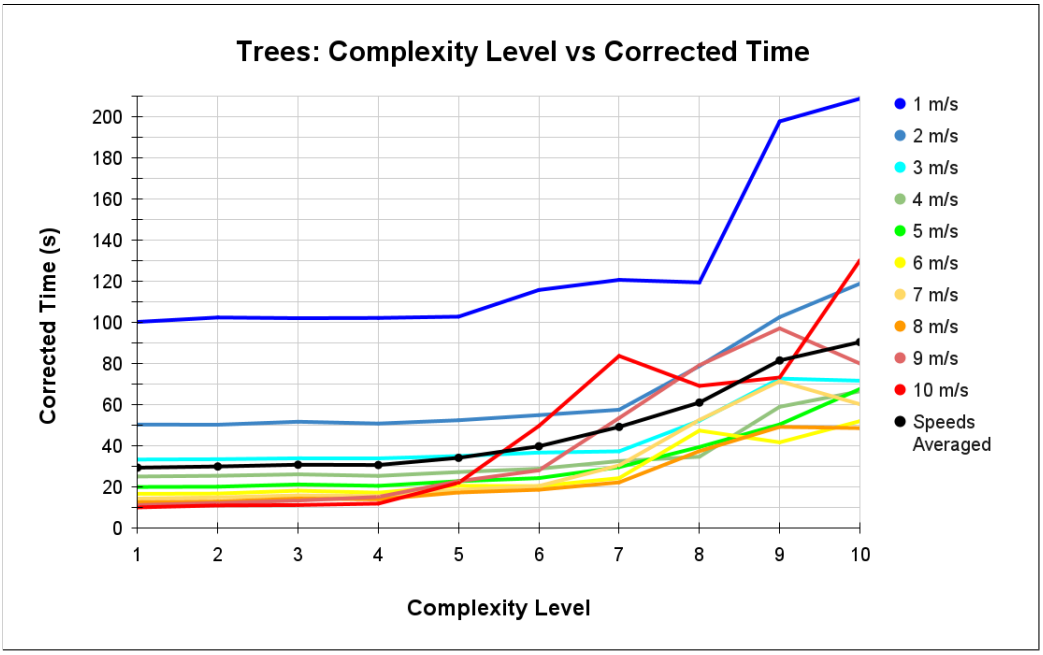


Figure 5.8: This graph indicates the time the MAV took to navigate each level of the course at a set target speed accounting for delays caused by collisions or level complexity. Note that for speeds 1 and 2 m/s there are no collisions which means that the increase in time is due to level complexity alone. This highlights that more complex levels take longer to navigate.

When collisions occur in reality, there is a recovery time required for the MAV to resume flight. The exact duration of these times can vary based on the severity of the collisions. When collisions occur in simulation we apply a time penalty to replicate this challenge as an indicator of the effects these collisions can have on the duration the MAV takes to navigate a level. It is assumed that collisions are not catastrophic and that the MAV does not require repair after collisions. The time penalty is scaled with target speed to provide longer time penalties at higher target speeds because at higher speeds collisions are likely to be more severe. Figure 5.8 accounts for collision penalties and shows how the time taken to navigate each level increases as level complexity increases. Figure 5.5 shows that the average percentage target speed achieved reduces as the complexity of each level increases. These two insights show that as level complexity increases, MAV speed decreases resulting in longer navigational durations to complete a given level.

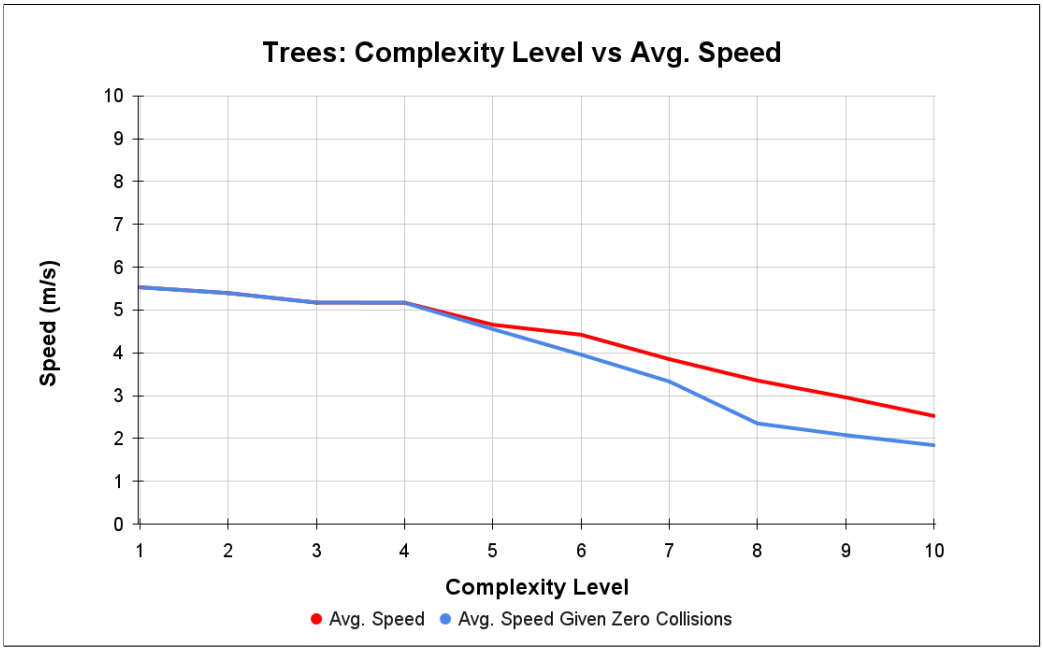


Figure 5.9: The graph shows the average MAV speed as well as the average MAV speed when no collisions occurred as it traversed all complexity levels. It shows that by reducing speed the chance of collisions is reduced and provides an estimate of how much this speed reduction needs to be.

Figure 5.9 shows the average MAV speed for all levels with and without collisions and indicates that for collision-free navigation slower speeds are required as level complexity increases. It also gives an indicator as to just how much of a speed reduction is required for successful collision avoidance. These results indicate that MAV speed, obstacle avoidance performance, and navigational success are inversely proportionate to level complexity.

5.3.2 Results: Pillars Course

The hypothesis is that the faster the MAV travels the faster its reaction time needs to be to avoid obstacles. As such, we should observe an increase in collisions as the MAV target speed is increased and it is required to navigate past a threshold complexity level. The results from the Trees course indicate that increases in level complexity cause reductions in speed but an improvement in obstacle avoidance performance. The results of the Pillars course provide additional evidence of this.

In the Pillars course, the complexity threshold is obscured by Artefact 1 found in FALCO. This artefact is discussed in greater detail along with the others in the Artefacts section later in this chapter. In brief, during experimentation, this artefact is observed to cause indecision in trajectory selection whilst the MAV directly approaches an object. This causes stalls and collisions making it difficult to completely isolate collisions that occur due to complexity level.

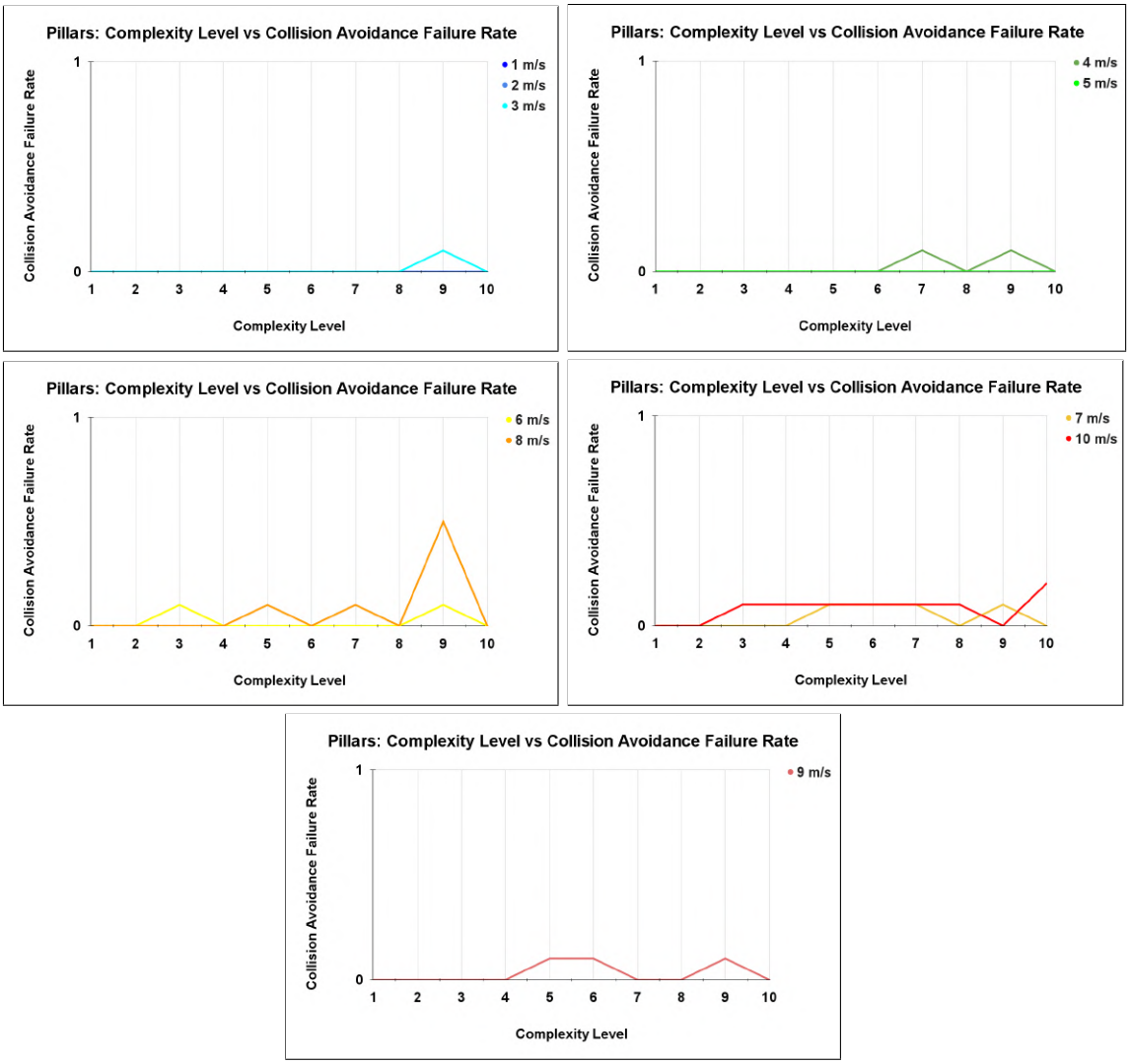


Figure 5.10: The failure rate of collision avoidance per level at the specified target speed.

Figure 5.10 shows the collision avoidance failure rate at each target speed per level. The graphs are split up for clarity as some of the results overlap and obscure one another.

At target speeds 1, 2, and 5 m/s there are no collisions. Target speed 3 m/s incurs 1 collision, target speeds 4 and 6 m/s incur 2 collisions, target speed 9 m/s incurs 3 collisions, and target speed 7 m/s incurs 4 collisions. Most collisions occur at target speeds of 8 and 10 m/s with 7 and 8 collisions. At a target speed of 8 m/s in complexity level 9, there are 7 collisions but at a target speed of 9 m/s, there is only one collision at complexity level 9. This does not make sense because, at higher speeds, more collisions should occur. This deviation from the assumptions made can be further explained by figures 5.12 and 5.13. These results are echoed in figure 5.11 which shows the target speed versus the course collision counts but note that this figure does not distinguish at which complexity level collisions occur.

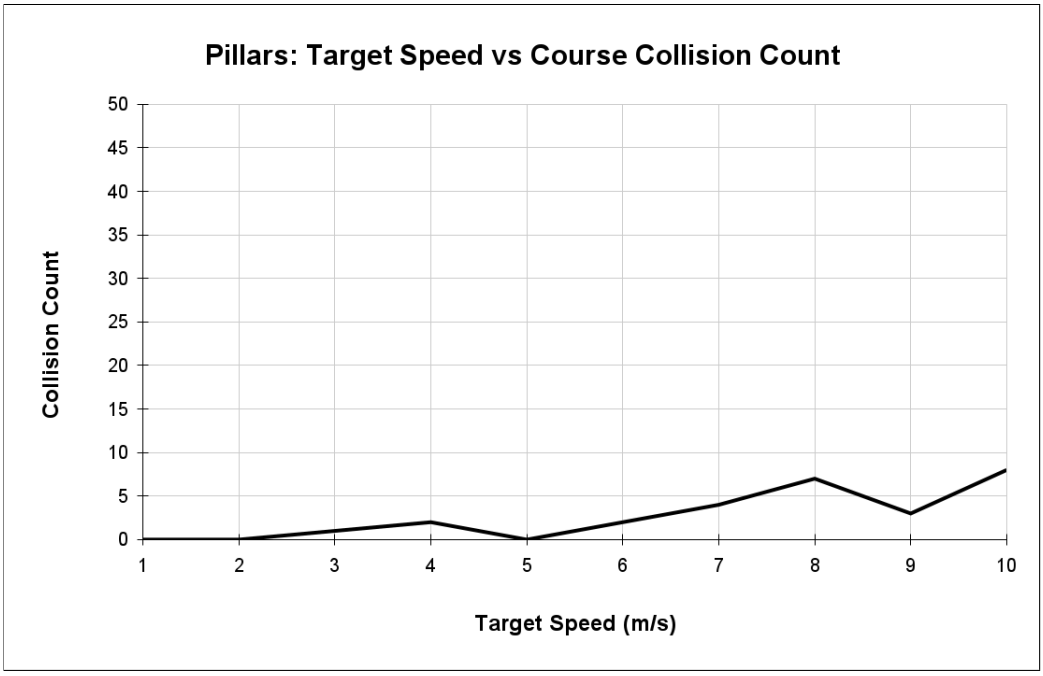


Figure 5.11: The collision count per target speed over the entire course. Note that collisions are averaged over the entire course so there are no indicators of which level collisions occurred.

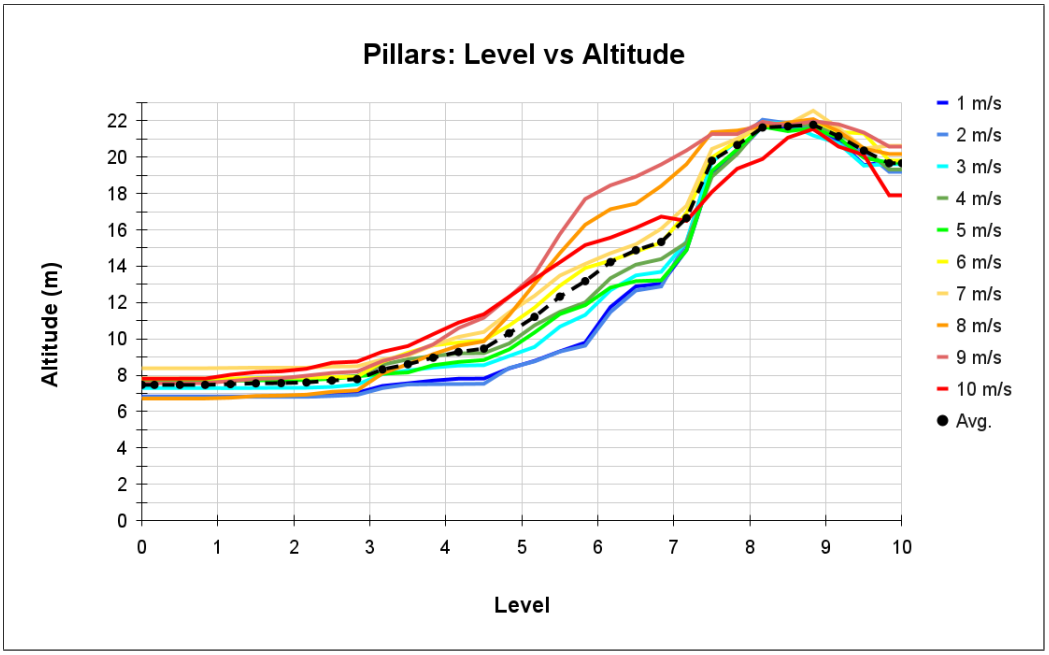


Figure 5.12: The altitude of the MAV as it moves through the course. Note that numbers on the horizontal axis show the end of the corresponding level. Vertical lines on the graph divide the course into 10 levels.

Figure 5.12 shows the altitude that the MAV flew at in each level at specific target speeds. It shows that at lower target speeds the MAV tends to begin its ascent at later levels in the course at a more gradual rate of ascent as opposed to the higher target speeds. Levels 8 and 9 are interesting because the MAV maintains a rapid

rate of ascent in level 8 and a relatively consistent altitude in level 9 on average. This indicates that there is a level complexity threshold encountered at level 8 which the MAV struggles to cope with at all target speeds. This results in the MAV moving to a lower level of complexity in level 9, which it can cope with more easily.

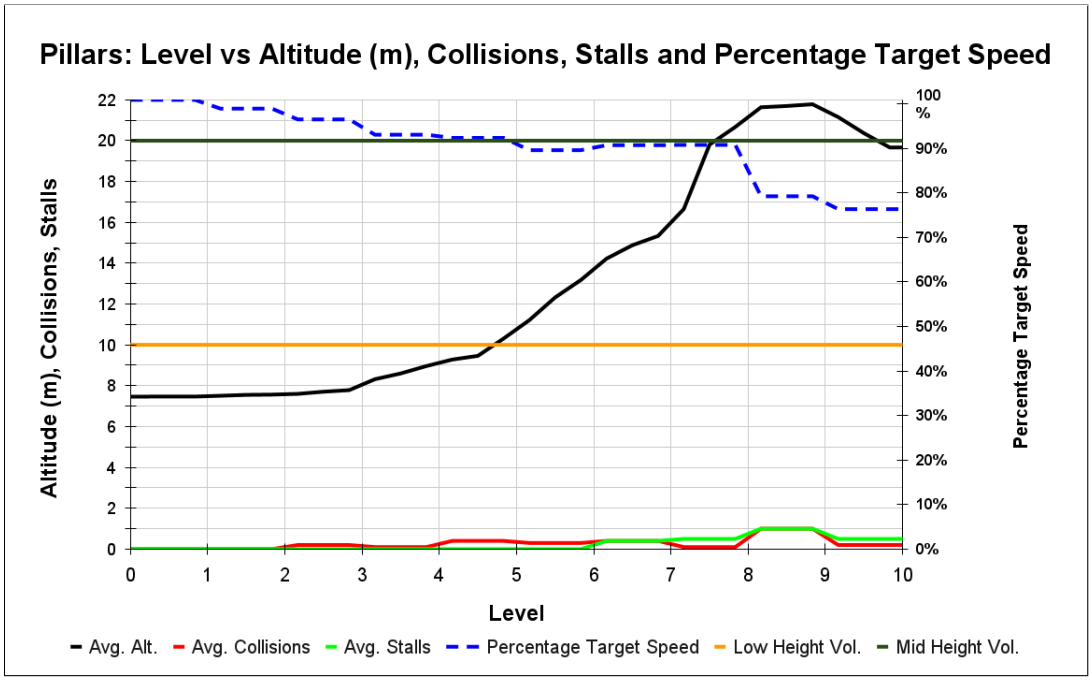


Figure 5.13: The average altitude of the MAV is plotted in a solid black line showing how the MAV ascends to find areas of reduced complexity. The graph shows that as level complexity increases, collisions and stalls begin to increase. The average percentage target speed that the MAV can achieve at each level is shown to decrease as level complexity increases.

In figure 5.12 altitudes follow the same general trend for all speeds and complexity levels and are indicated by the black dotted average line. Figure 5.13 plots this average in isolation along with other metrics which helps provide clarity.

The following table shows the level complexity score based on the average altitude of the MAV whilst navigating the Pillars course:

Level Complexity Based on MAV Altitude			
Level	Avg. Alt.	Height Volume	Complexity Score
Level 1	7.468	Low	0.000
Level 2	7.541	Low	10.624
Level 3	7.697	Low	21.249
Level 4	8.627	Low	42.498
Level 5	9.684	Low + Mid	56.683
Level 6	12.237	Mid	63.834
Level 7	14.812	Mid	85.112
Level 8	19.042	Mid + High	85.171
Level 9	21.704	High	71.073
Level 10	20.394	High + Mid	94.667

Table 5.4: The table shows the level complexity at the average height that the MAV moved at while navigating the Pillars course. Highlighted rows indicate when the MAV moved between height volumes. In these rows, the complexity score is calculated by adding the level complexity scores weighted by the approximate duration the MAV spent in those levels.

Figure 5.13 shows that the MAV begins to increase its altitude as levels become more complex. It then crosses between height volumes in levels 5 and 8. In level 10 the MAV reduces its altitude, moving from the High height volume to the Middle height volume because the goal position in the finishing area is at an altitude below 15m.

In level 5 the MAV moves from the Low height volume to the Middle height volume reducing the level complexity score from 63.746 to 42.556. Note that the average complexity score becomes 56.683 because the MAV spent two-thirds of its time in the Low height volume and a third of its time in the Middle height volume.

The top of the Low pillars in the Low height volume begins to reduce in width and depth at ±7.5m in altitude, essentially creating a chisel head shape shown in figure 5.14. This in part explains why the altitude begins to increase from complexity level 2 onwards. This reduction in width and depth causes a slope on the face of the upper part of the pillar. FALCO's precomputed trajectories begin to become occluded by the lower part of the slope first, causing the lower altitude trajectories to be filtered out of the trajectory selection process. This results in the MAV incrementally increasing its altitude every time it encounters this sloped face.

The MAV begins level 5 at an altitude of ±9m which is the midpoint of the chisel head shape at the top of the pillars.

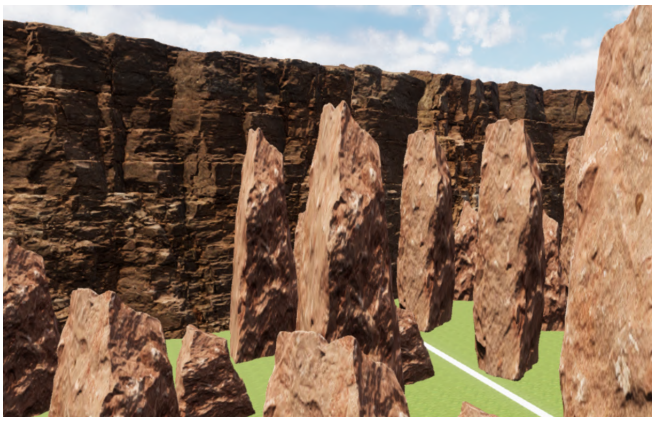


Figure 5.14: The image shows the chisel head shape at the top of each pillar in the Pillars course.

In level 8 a similar occurrence is observed. The MAV crosses from the Middle height volume to the High height volume in the middle of level 8, reducing the complexity score from 113.483 to 56.859. The MAV spent an even amount of time in both height volumes and thus has an average complexity score of 85.171. A rapid ascent is observed at the beginning of level 8 caused by the increased level of complexity. At this point, the MAV encounters a higher frequency of obstacle occurrence as well as the chisel head shape at the top section of the Middle pillars. The chisel head shape of the Middle pillars begins at $\pm$15m in altitude and is encountered towards the end of level 7 and throughout level 8. These two factors cause the MAV to ascend from an altitude of 14.812m to 19.042m.

The MAV keeps a constant altitude whilst navigating the High height volume of level 9 where it can cope with a reduced complexity score of 71.073. At this altitude, the MAV can more easily pass the pillar at the higher target speeds.

The MAV finds lower complexity as altitude increases which results in more collision-free trajectories which it can make use of whilst maintaining the target speed. This trend continues because, unlike the Mid and High trees in the Trees course, the pillars do not have overhangs. This means that there are more open spaces higher in altitude increasing the chances of the MAV reaching its goal position and avoiding collisions.

There is a similar environmental complexity in the Middle height volume of level 5 compared to the Low height volume of level 4 which explains why there are fewer collisions in level 5 than in level 4.

From level 6, collisions begin to occur as the level complexity in the Middle height volume increases. The MAV maintains its ascent and clears the Middle height volume in the middle of level 8. During level 8 there is a much steeper ascent because the complexity level is high and collision-free spaces exist higher in altitude. As the MAV clears the Middle height volume its ascent slows because the complexity level is drastically reduced. In the middle of level 9, the MAV altitude is constant but towards the end of the level it begins to descend until the end of the course so that the goal position can be reached. Level 9 encounters fewer collisions because the level complexity is reduced from 113.483 in level 8 (85.171 is the average) to 71.073

in level 9.

The above indicates that as speed increases and the MAV begins to encounter more complex environments, it begins to move to higher altitudes with less complexity. The MAV can navigate the course at lower speeds without collisions but as the speed increases the MAV is required to navigate through more complex levels which cause collisions. As the levels get more complex, the MAV is forced to slow down so that trajectories can be more accurately tracked. This again, indicates that collision avoidance failure rate is a function of speed and speed is a function of environmental complexity.

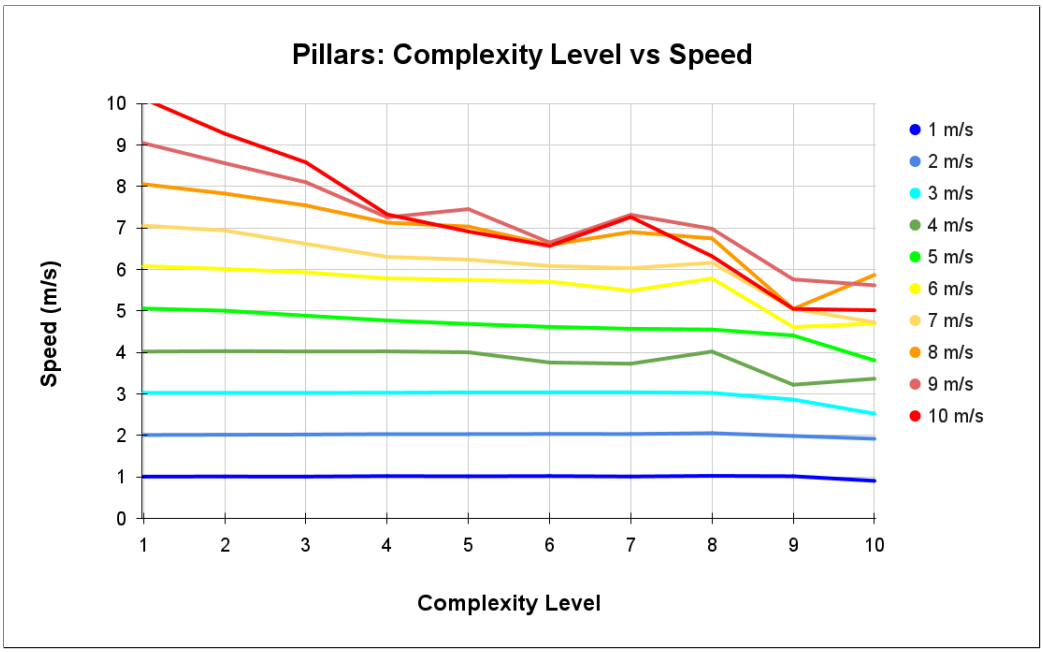


Figure 5.15: The average speed the MAV moves at whilst navigating through each level attempting to reach a target speed.

Figure 5.15 indicates that as the complexity level increases the speed at which the MAV can navigate the level is reduced. The higher the target speed the lower the level at which a reduction in average MAV speed occurs. The graph indicates that as target speed and level complexity increase, the actual MAV speed decreases. Note that 5.15 neglects the consideration of how collisions affect speed.

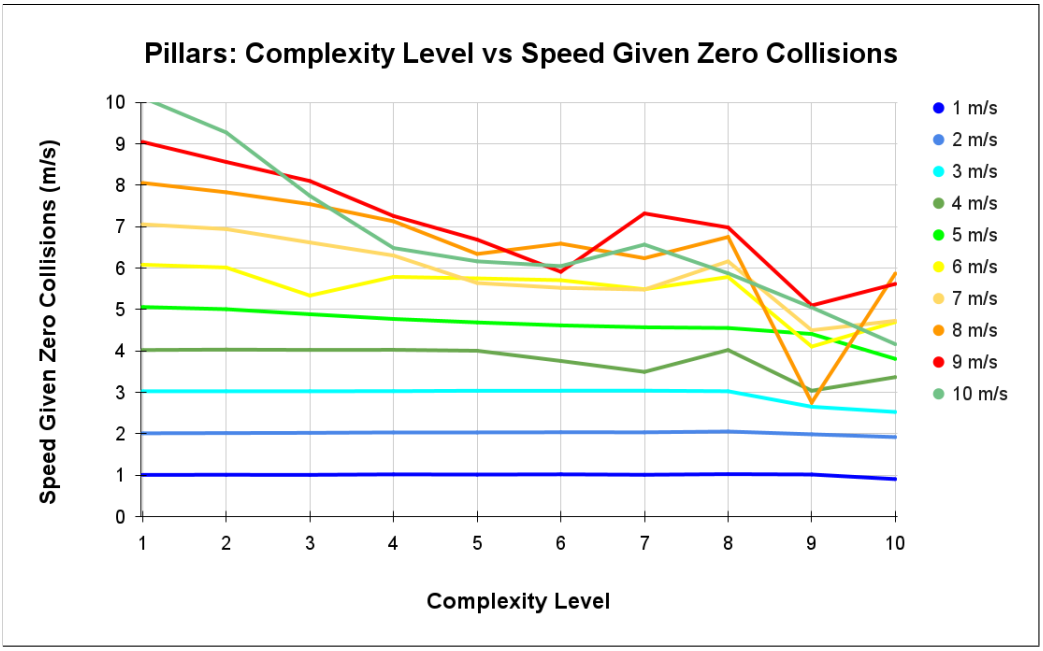


Figure 5.16: This graph indicates the average MAV speed whilst traversing each level without collision.

Figure 5.16 shows how fast the MAV moved through each level without collisions. This gives an indicator of MAV speeds required for successful navigation of each level but also shows that lower speeds increase the chance for successful collision avoidance. This indicates that an obstacle avoidance algorithm may work with 100% collision avoidance in an environment at low speed but as soon as speed is increased past a threshold collisions begin to occur. This of course is contingent on the complexity level remaining constant.

If the actual MAV speed was not limited by the target speed, it would be limited by complexity levels or MAV hardware. MAV hardware can easily allow for speeds above 10 m/s so level complexity is further evaluated. MAV speed reductions observed in figure 5.16 can give indicators of which speeds are appropriate for a given environmental complexity score. The recommended speed ranges for the MAV given level complexity threshold are shown in the following table:

Recommended Speed Ranges Per Level			
Level	**Speed Range (m/s)**	**Height Volume**	**Level Complexity Threshold**
Level 1	10+	Low	0.000
Level 2	6-7	Low	10.624
Level 3	4-5	Low	21.249
Level 4	4-5	Low	42.498
Level 5	4-5	Low + Mid	56.683
Level 6	3-4	Mid	63.834
Level 7	3-4	Mid	85.112
Level 8	3-4	Mid + High	85.171
Level 9	1-2	High	71.073
Level 10	0-1	High + Mid	94.667

Table 5.5: The table indicates the recommended speed ranges for successful collision avoidance given an environment complexity score.

The speed ranges defined in table 5.5 are obtained by taking note of the target speed at which the first speed reduction is observed and the associated level. This provides an upper limit in which to confine the optimal speed search. The target speed before the upper limit becomes the lower limit in which the optimal speed can be found.

This is not to say that collisions would not occur at all at these environmental complexity scores but that the chances of collisions are significantly reduced. This is because the experimentation sample space was limited to ten experiments per target speed.

This shows that when performance characteristics of obstacle avoidance algorithms are stated, they should be stated with an environmental complexity level and MAV speed limit. If these characteristics are not stated then it may lead the reader to believe that it is possible to fly a MAV in any environment such that speed is only limited by MAV hardware.

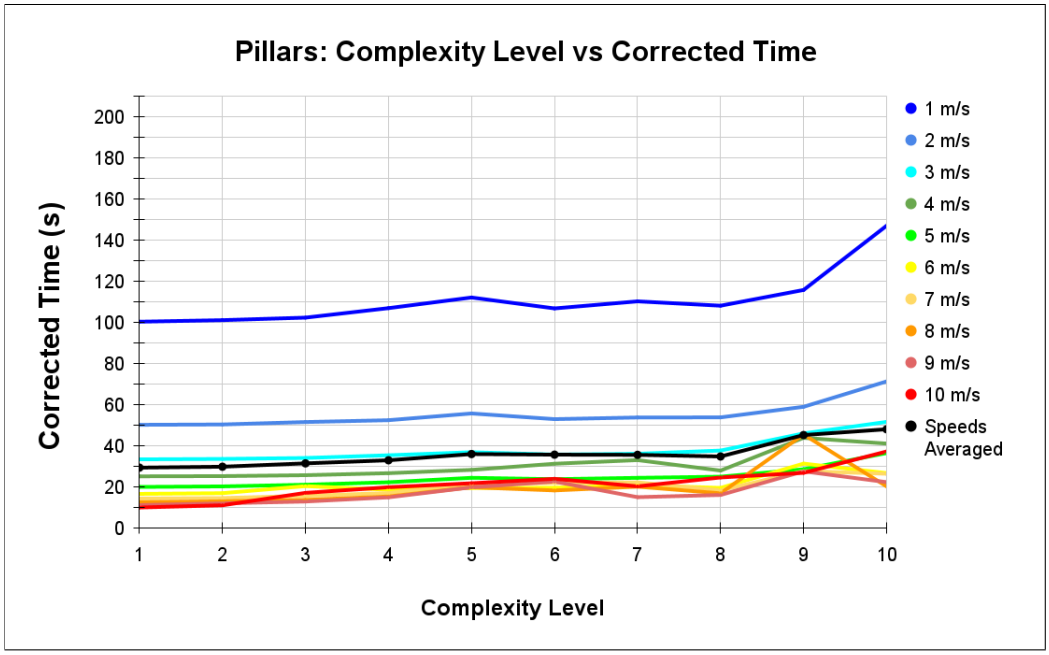


Figure 5.17: This graph shows that with increases in target speed, there is a proportionate reduction in the time taken to navigate a level. This is dependent on complexity and shows that some levels have reduced navigation times regardless of target speed.

Figure 5.17 shows how long it took for the MAV to navigate each level whilst accounting for time incurred by collisions. It shows that increasing the target speed reduces the time taken to navigate each complexity level. However, there are bottlenecks on some levels showing that as complexity levels increase and more collisions occur it takes the MAV longer to navigate those levels regardless of target speed.

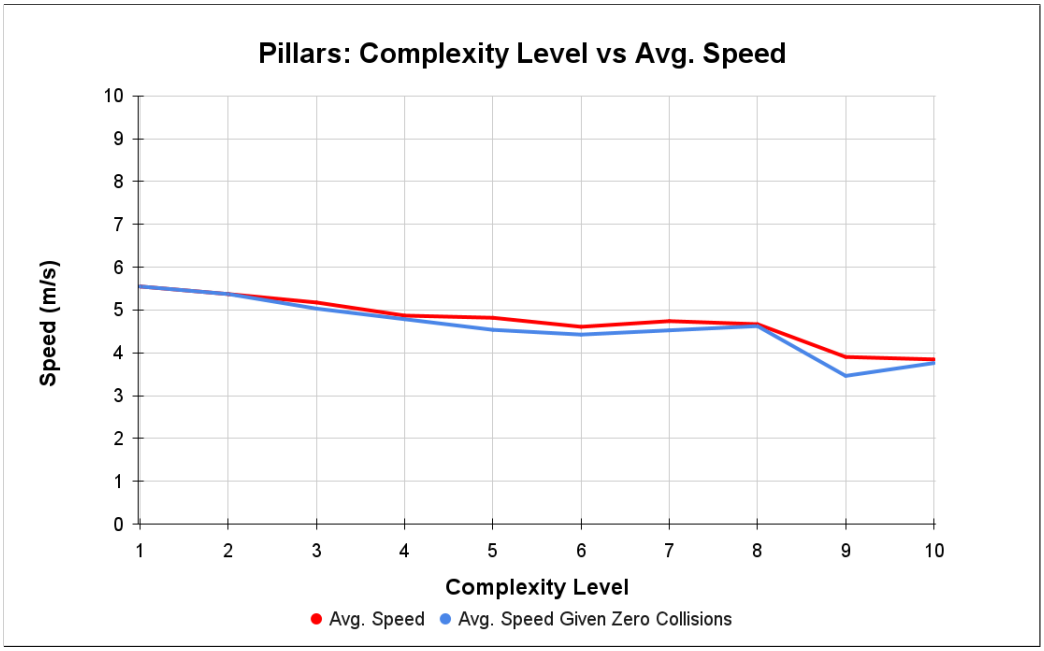


Figure 5.18: The average speed at which the MAV navigated each level, with and without collisions.

Figure 5.18 shows the average MAV speed for all levels with and without collisions and indicates that for collision-free navigation slower speeds are required as level complexity increases. It also gives an indicator as to just how much of a speed reduction is required for successful collision avoidance. These results indicate that MAV speed, obstacle avoidance performance, and navigational success are inversely proportionate to level complexity.

5.4 FALCO Artefacts

During experimentation with FALCO a few artefacts are found. These artefacts are uncovered through experimentation in the simulation which highlights one of the benefits of testing obstacle avoidance in simulation. This section details these artefacts as observed during experimentation in both simulation courses and through understanding how the FALCO codebase works.

5.4.1 Artefact 1

When a new trajectory is required for collision avoidance FALCO selects a trajectory from a precomputed trajectory set. On occasion, an object comes into view of the depth sensor in such a way that it splits the precomputed trajectories vertically in half as shown in figure 5.19.

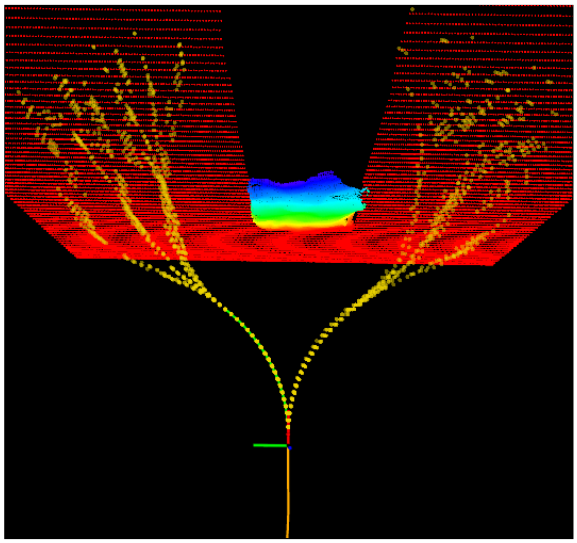


Figure 5.19: The precomputed trajectories are split into two groups by the object. This causes FALCO to switch selection between both groups when the groups are equally likely to reach the goal position successfully.

The algorithm then alternates trajectory selection between left and right trajectory groups until a trajectory is selected. This alternating selection rate begins to increase as the MAV gets closer to the object. At this point one of three things can occur; FALCO settles on a trajectory and the obstacle is avoided, the MAV stalls or it collides with the object. When a stall occurs, the MAV stops just in front of the obstacle because the obstacle is too close to the depth estimation sensor. When this happens all of the precomputed trajectories are occluded by the object and a

collision-free path around the object is not available. Manual intervention is required to reposition the MAV so that autonomous navigation can begin again. Collisions of this nature are made worse as speed increases because the MAV approaches the object quicker which requires the MAV to make a definitive trajectory selection faster with the same perception-action loop latency. The sooner the MAV makes a trajectory selection the further away the object will be, giving the MAV time and space to track the new trajectory and avoid the obstacle.

This artefact causes collisions which can occur at any level making it difficult to differentiate between collisions caused by the artefact and the level complexity. These artefacts occur in the Trees course when the MAV encounters the trunks of the Mid and High trees. The complexity of the object makes a definite difference. In the Trees course, collisions caused by this artefact occur less frequently and usually only at higher speeds. The tree trunks of the Mid and High trees are much smaller in width than the pillars are in width. This allows for more trajectories to pass by the right and left sides of the trunk, giving the MAV more time to settle on a trajectory as it approaches the tree trunk. This means less strain on the perception-action loop which allows for the MAV to make a trajectory selection quicker, reducing the distance between the object, and allowing for more space for the MAV to avoid the object successfully.

In the Pillars course, pillars range from 5 to 7.3m in width, and in the Trees course the tree trunks are between 0.6m and 1.2m in width at the base of the trunk. The tree trunks begin to divide into smaller branches as the tree height increases. This allows the MAV to navigate through the branches and is mostly observed at higher speeds. The pillars are much wider than the tree trunks, so when the artefact occurs, there are fewer precomputed trajectories around the object for the MAV to choose from. This gives FALCO less time to select from these groups because as the MAV approaches the obstacle more trajectories are removed from consideration. Once all trajectories are removed from consideration the MAV begins to slow down and stops. Depending on the speed at which the MAV is moving, a stall or collision can occur.

This artefact is found but not corrected because it is not within the scope of the research to correct the obstacle avoidance algorithm only to assess it.

5.4.2 Artefact 2

FALCO does not consider the size of the MAV when filtering collision-free trajectories. This creates an artefact that causes trajectories to be generated which forces the MAV through very tight gaps. This occurs in both scenarios where the MAV can and cannot fit through these gaps, where the latter case causes collisions. A mechanism needs to be put in place such that the size of the MAV needs to be taken into consideration when trajectories are selected.

5.4.3 Artefact 3

FALCO makes use of trajectories which are assumed to be easily tractable but at high speeds, this may not be the case. At high speeds, MAV motors may already be reaching their maximum capabilities which means any additional acceleration

required to track obstacle-free trajectories may not be available. This is something that all MAVs would encounter as they are pushed to their limits. For this reason, trajectory tractability needs to be assessed relative to MAV state so that precomputed trajectories can be improved and more appropriate trajectory selection can occur at high speeds. This artefact may be a contributing factor to the increased collision counts at high speeds.

5.5 Comparative Analysis: Trees verses Pillars

Using the complexity framework, the Trees course is assigned an average complexity score of 191.203 and the pillars course, an average complexity score of 67.357. The higher complexity in the trees course comes from the the complex structure of the trees. The trees used are assigned an average object complexity of 0.688 which is more than twice the object complexity assigned to the pillars, 0.301. This indicates that Trees course should be more difficult to navigate than the Pillars course and should incur more speed reductions and collisions during simulation.

The results of the simulations correlate with this deduction as can be seen in figure 5.20. The MAV moves slower and encounters more collisions as the levels become more complex in both courses but is more severe in the Trees course.

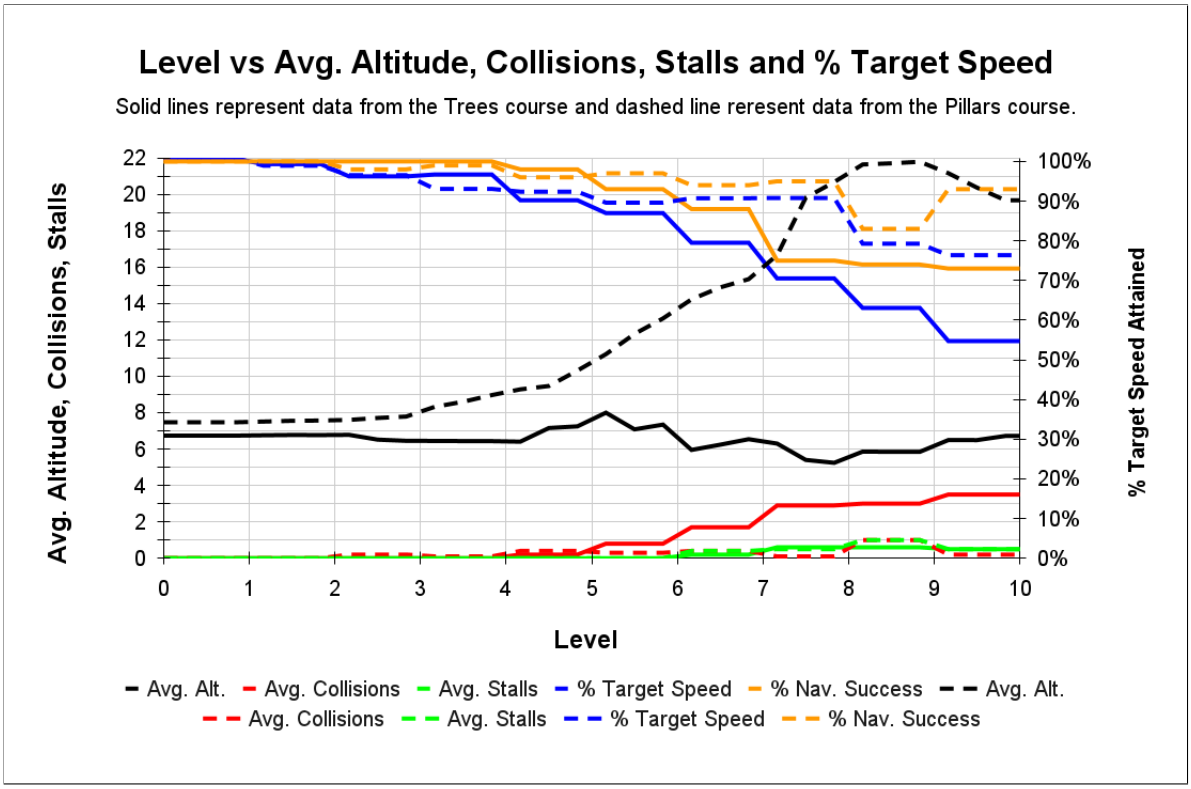


Figure 5.20: A comparison of the average altitudes, collisions, stalls, and percentage target speeds achieved between the Trees and Pillars course. Note that the percentage target speed achieved and percentage navigation success make use of the y-axis to the right.

The MAV tries to maintain a balance between speed and collision avoidance but

the complex structure of the trees make this a difficult task. This is reduced in the pillars course but collisions are further exacerbated by limitations in the obstacle avoidance algorithm. Artefact 1 discussed in section 5.4.1, describes how trajectories are split and explains why Artefact 1 affects the the MAV more in the Pillars course. In short when the MAV approaches pillars, the flat surface of the pillars facing the MAV causes trajectories to be rapidly removed from consideration leaving collision free trajectories on the far left and right of the MAV. The MAV switches rapidly between these trajectories in an attempt to find a collision free path. This occurs whilst the pool of collision free trajectories is actively being reduce because the MAV is still moving towards the face of the pillar. When there are no more collision free trajectories then MAV either stalls or a collision occurs. The large widths and flat surfaces of the pillars as well as high MAV speeds cause Artefact 1 to become a noticeable reason for the occurrence of collisions.

This also indicates that the source of collisions occur not only from object complexity but also from limitations in the obstacle avoidance algorithm. Understanding the source of collisions and why they occur can help improve the design of obstacle avoidance algorithms. Overall the MAV moves faster attaining a higher percentage of the target speed and has fewer collisions in the Pillars course than the Trees course.

5.6 Chapter Summary

In this chapter the experimental setup was discussed, metrics explained and experimental results explored. FALCO was evaluated and during simulations, artefacts were uncovered. Artefacts 1 and 2 were found to affect collision avoidance negatively through observations made whilst running the simulations. The experimental results indicate that MAV speed, obstacle avoidance performance, and navigational success are inversely proportionate to the environmental complexity. The higher the environment complexity score, the slower the MAV needs to move to track collision-free trajectories resulting in improved collision avoidance performance. This is further shown when comparing MAV performance between the two courses. Reduced complexity in the Pillars course allowed the MAV to move faster and to avoid collision more successfully.

Chapter 6

Discussion

6.1 Introduction

In this section, the results will be discussed relative to each research question. The goal of this report is to assess the factors that degrade obstacle avoidance performance. Intuitively, the more obstacles in a set area of space the more difficult it is to navigate that area. More importantly, the actual complexity of the objects is often understated. The two most commonly used categorisations of environments in autonomous navigation are unstructured and structured environments. Object complexity is the differentiating source of the division between these catagories. Unstructured environments generally contain objects that take on shapes that are generally more complex with large deviations from basic geometric shapes. This results in unstructured environments being more difficult to navigate. It then makes sense to further examine what makes these unstructured objects more complex as this is the root cause of complexity in an environment.

The complexity characteristics used to define the complexity score are object complexity and frequency of object occurrence. The object complexity score is defined using the following object properties: path groups, MAV gaps, false depthmap noise, and the percentage of total height volume an object occupies. These properties are chosen because they induce complexities that the MAV needs to cope with to avoid obstacles successfully. Path groups indicate whether the MAV can fly around, over, under, or through objects. MAV gaps verify that there are wide enough spaces in the general volume that the object occupies so that the MAV can fly through the object. This property also caters for MAVs of different sizes. False depthmap noise caters to groups of small gaps that pass through objects which the MAV may perceive as traversable but are not. Finally, the percentage of total height volume quantifies the area taken up by the object which could have otherwise been used for navigation.

When considering MAV hardware relative to obstacle avoidance algorithms, trajectory tracking given MAV speed becomes a deterrent to successful obstacle avoidance. To successfully avoid a collision, MAV control algorithms should be able to select collision-free trajectories that they can track accurately. When trajectories are not tracked accurately collisions begin to occur. With object complexity quantified the assessment of obstacle avoidance algorithms can yield valuable insights.

6.2 Research Question 1

How does speed affect the success rate of obstacle avoidance?
In the experiments, the MAV tries to maintain the target speed whilst navigating the course. This in effect allows for MAV speed to remain constant where possible, whilst prioritising collision avoidance. The MAV is tested at 10 target speeds through each course, where the complexity score of each level increases. Figures 5.3 and 5.11 show that course collision counts for the Tree and the Pillar courses. These graphs indicate that as the target speed increases the collision count increases. The graphs do not always adhere to this trend though, in some levels, a higher collision count is expected but does not occur. This is because the MAV moves to an altitude with a reduced environmental complexity score. For the Tree course, the MAV moves to altitudes that help it avoid the 6.4m mark. At this altitude, a combination of object properties causes an environmental complexity concentration in the upper part of the Low height volume. In the Pillars course, the MAV incrementally moves to higher altitudes because of the chisel head shape of the pillars and the reduction in the frequency of occurrence of objects resulting in a reduction in local environmental complexity.

This also indicates that the level complexity influences speed. Figure 5.20 shows that as level complexity increases the percentage of the target speed obtained decreases. These results show that if level complexity is constant, as speed increases collision avoidance performance degrades.

Each target speed is tested 10 times and it is shown that speed increases reduce collision avoidance performance. The results indicate that target speed can be adjusted based on environmental complexity to ensure a balance between speed and optimal collision avoidance performance.

6.3 Research Question 2

How does the degree to which the environment varies in complexity, affect the speed at which a MAV can successfully navigate that environment?

Figure 5.20 shows that as level complexity increases the percentage of the target speed achieved decreases. This proves that environmental complexity causes speed reductions because the trajectories required to avoid collision become more difficult to track at higher speeds. This forces the MAV to reduce its speed so that trajectories can be more easily tracked and collisions can be avoided. This indicates that increases in environmental complexity cause a proportionate reduction in MAV speed during which collision count increases.

Figure 5.20 also indicates that as collision count increases the percentage of times that navigational success was achieved is reduced. Recall that navigational success is when the MAV reaches the finish area of a course without any collisions or stalls. In both courses, the MAV tried to maintain or move to altitudes with less complexity.

These results indicate that each target speed has an environmental complexity threshold at which obstacle avoidance begins to degrade and target speeds cannot be maintained.

6.4 Research Question 3

Does the complexity of an object affect the obstacle avoidance algorithm's ability to navigate around that obstacle?

The experiments show that environmental complexity is one of the main causes of slow MAV speeds in autonomous navigation in unstructured environments, more so as the spatial frequency of occurrence of objects increases. Environmental complexity as defined in this report is predominantly characterised by object complexity. Object properties create navigational spaces which MAVs need to be able to detect and react to, to avoid colliding with objects. These properties essentially scope design requirements for obstacle avoidance algorithms and most certainly affect obstacle avoidance. The object properties identified are used to generate an object complexity score for each level. The average complexity score for the Pillars course is 67.357 and 191.203 for the Trees course. This shows that the Trees course is 2.8 times more complex than the Pillars course. This difference in complexity is due to the properties of the objects used in each course. The average pillar has an object complexity score of 0.301 but the average complexity score of the trees is 0.668. This shows that course complexity is derived from object complexity and the results of the experiments correlate with the framework used to define object and course complexity in general.

Figure 5.20 shows that there are more collisions in the Trees course than there are in the Pillars course, which indicates that the Pillars course is easier to navigate. This is further correlated by the increased percentage target speed which the MAV was able to attain in the Pillars course as opposed to the Trees course. The navigational success rate is also higher in the Pillars course. This shows that object complexity does affect the MAV's ability to navigate around the obstacle.

6.5 Chapter Summary

In this report, an obstacle avoidance algorithm called FALCO was stress tested to better understand the effect that MAV speed, environmental complexity, and object complexity have on obstacle avoidance performance. It is found that the complexity of objects and the frequency of occurrence of objects in an environment affects the maximum speed at which MAVs can traverse an environment without collision. Object and environmental complexity are quantitatively defined relative to MAV navigation. Object properties that strain obstacle avoidance are used to generate object complexity scores. These object complexity scores are used in conjunction with the number of object occurrences in a level to define environmental complexity. By quantitatively defining environmental complexity, MAV speed relative to environmental complexity allows for a better understanding of why obstacle avoidance can fail.

MAV speed affects obstacle avoidance by making trajectories more difficult to track as speed increases. This is inherently an obstacle avoidance and autonomous navigation problem that needs to be addressed through improved trajectory generation and selection based on MAV speed. This is separate from limitations imposed by environmental complexity. Trajectory tractability as a function of speed needs to be

considered when collision avoidance systems are required to make trajectory selections during collision avoidance. The limitation is then the ability of MAV hardware to track complex trajectories as a function of its speed.

Assuming that high-quality collision avoidance trajectories can be generated as a function of MAV speed and that MAV hardware can accurately track these trajectories, the next bottleneck is environmental complexity. Environmental complexity physically allows for collision-free paths through an environment or not. This affects the complexity of trajectories required to navigate these collision-free paths assuming that the control algorithm can find them. This limitation can be defined in an abstract way through the question, "Is there space for the MAV to move from point A to point B without collision?"

In conclusion, MAV speed affects trajectory tractability which influences collision avoidance. Environmental complexity affects trajectory complexity to the point where trajectories cannot be accurately tracked resulting in collisions. Object complexity drastically influences environmental complexity and can strain perception through the presence of specific object properties. This provides two focus areas to explore, trajectory generation and selection and object properties which strain perception sensors.

Chapter 7

Conclusion

MAVs are extremely versatile and have vast application spaces. Though many industry applications tend to make use of drone pilots to operate MAVs, there is an emergent trend of autonomous flight. Autonomous MAV navigation systems need to sense their environments, plan a route to a goal destination, and follow that route whilst avoiding obstacles.

While autonomous navigation has been achieved most solutions navigate environments very slowly. There are two important aspects of autonomous navigation, collision avoidance and path planning. The focus is placed on collision avoidance as it is a fundamental prerequisite for path planning.

In this report, the reasons for the slow traversal speed of autonomous navigation systems are explored through experimentation in simulation. One of the problems encountered in creating a simulation environment is defining environmental complexity. This definition is important because it is used to create a simulation environment in which to test obstacle avoidance realistically. A framework for defining complexity is proposed which assigns a complexity score to an environment. This environmental complexity is comprised of two variables, the frequency of occurrence of objects and object complexity. The frequency of occurrence of objects accounts for the reduction of free space available which the MAV can use to find collision-free paths. Object complexity describes the complexity of the object relative to MAV navigation and accounts for object properties that make collision-free path selection more difficult.

The simulation environment needs to mimic the characteristics of the real world so that the complexities of the real world can be analysed to find how they impact autonomous navigation. Using this framework for defining environmental complexity, objects are assessed and the frequency of occurrence of objects is defined and used to create two simulation courses that incrementally increase in complexity. Each course is divided into 10 levels where successive levels have incrementally higher complexity scores.

During experimentation, the MAV navigates from the starting area of the course through each of the 10 levels at a target speed, to the finishing area of the course. During this time, data such as MAV speed, collisions, stalls, and level times are recorded. The data is then used to create descriptive graphs to aid interpretation.

One of the main assumptions around collision avoidance is that the more objects there are in an environment the more difficult it is to navigate that environment. Though this assumption is widely accepted there are not many endeavours to empirically prove that this assumption is true.

Through experimentation in the simulation environments, this assumption is proved empirically and it is shown how environmental complexity affects obstacle avoidance.

Two domains need to be considered when collision avoidance is being assessed. The first is contained entirely by the design of the MAV. When a MAV encounters an obstacle, collision-free trajectories need to be generated which move the MAV past the obstacle. These trajectories need to be accurately tracked to ensure that the MAV does not collide with any objects. The ability of the MAV to track these trajectories is dependent on multiple factors including MAV hardware and the control algorithm used. FALCO uses precomputed trajectories, reducing the online computational cost of trajectory generation. This reduces the latency of the perception-action loop allowing for faster trajectory selection and higher navigation speeds to be achieved.

The second domain is environmental complexity, which is contained entirely by the environment. If an environment in a set area contains no objects then collision avoidance is not required. It is the addition of objects to this environment that creates environmental complexity creating the need for collision avoidance. Through this, it is understood that object complexity gives rise to environmental complexity. Certain object properties create sensory inputs which make collision-free trajectory selection more difficult. This in combination with the frequency of occurrence of objects, means that sometimes complex trajectories are required to be generated and tracked to avoid collisions.

Collision avoidance is at the intersection of MAV design and environmental complexity. Environmental complexity defines collision-free trajectories, which MAV control algorithms need to find and accurately track for successful collision avoidance. Through experimentation, the effects of this relationship can be seen.

When the environmental complexity increases in the simulation environment, the collision count increases and MAV speed decreases. Collision count increases because trajectories become more difficult to track causing a speed reduction. Increasing the MAV speed makes trajectory tracking more difficult because at higher speeds the MAV needs to develop appropriate thrusts in addition to current thrusts, to accurately track complex trajectories. This is one of the reasons in general why MAV navigation at slower speeds is more successful.

When the target speed is increased, the collision count increases, given that environmental complexity is held constant. By varying the environmental complexity incrementally, and noting where collisions occur, the approximate environmental complexity score at which collisions occur can be found. Bottleneck environmental complexity scores can then be defined for a specific MAV configuration at a specific target speed.

There are two simulation environments that are tested, the Pillars and Trees environments. The Pillars environment has less complex objects while the Trees environment has more complex objects. It is shown that there are more collisions on the Trees environment than the Pillars environment. The navigational success

rate and the average percentage of target speed attained was higher in the Pillars environment than in the trees course. This shows that the root cause of collisions in the environmental domain stems from object complexity.

In conclusion this report makes the following contributions:

- An extensible framework for defining environmental complexity relative to MAV navigation is provided.
- An obstacle avoidance algorithm called FALCO is stress-tested to validate this framework.
- Provides empirical evidence that enforces the assumptions that obstacle avoidance becomes more difficult as:
  - the number of objects in an environment increases.
  - MAV speed is increased.
- Highlights that the root cause of environmental complexity relative to MAV obstacle avoidance stems from object complexity.

Future Work

This section provides details regarding additions to the study that would further refine the assessment of high-speed MAV obstacle avoidance in unstructured environments by further quantifying the environmental complexity of the real world.

Additional Object Complexity Properties

Two complexity metrics are used to assign a complexity score to an environment, object complexity, and the frequency of occurrence of those objects. Object complexity makes use of four properties to describe complexity, this can easily be expanded on. When adding additional properties to object complexity, navigational complexity must be the guiding requirement. Navigational complexity is drastically affected by MAV constraints such as compute and sensor suite. The computation required for autonomous navigation occurs from the need to process large amounts of sensor data, primarily from images or depthmaps. If sensor data is erroneous then this computation can be in vain because the MAV might decide to move on paths that incur collisions. Thus the ability of the sensor suite to detect objects and the rate at which it does so is important. It is not enough to say that because a sensor can detect one class of objects it can detect most others. Objects occur in nature in a plethora of conditions and these conditions determine how their structures are formed. These structures can become complex making obstacle avoidance difficult because they place stringent requirements on MAV systems.

This is why object complexities are to be assigned relative to navigational complexity. When deciding on object complexity properties to add, figuring out how these complexities make it difficult for the sensor suite to detect objects is considered. Detection is the first limitation that needs to be stressed in a simulation environment. Once a system can detect objects with high degrees of certainty trajectory generation needs to be designed. This is challenging in its own right as trajectories need to be trackable by the MAV and each MAV has unique kinodynamic capabilities. To generalise this requirement to any MAV there needs to be an autonomous way to define the kinodynamic capability of any MAV in general. Once this is defined, trajectories

can be generated which are tractable. These trajectories can then be selected from, such that obstacles can be avoided relative to where they are and the MAV's speed. When attempting to avoid obstacles, tractable trajectories can be generated which lead the MAV into a more complex state making the ability to avoid the obstacle more difficult. This is caused by the complexity of the environment which in turn is caused by the complexity of the objects in that environment. By understanding the failure mechanisms relative to object complexity, object avoidance algorithms can plan to avoid MAV states which induce these failure mechanisms. Object properties need to be better understood for a complete understanding of these failure mechanisms can be achieved.

Assigning object complexities relative to MAV navigation is the initial step that can be used to assign quantitative metrics to environments and more granular specifications for MAV design requirements.

Dispersion Metrics

The frequency of occurrence is one way to define how objects are positioned in an environment. There are however more comprehensive metrics that can be used to place objects in simulation environments. Adding a dispersion metric when defining the spatial frequency of occurrence might yield a more comprehensive calculation of the complexity score.

Optimal Path Convergence

During experimentation, it is noted that the MAV tends to move on very similar paths. Whether these paths are optimal is uncertain. If the optimal paths for a given MAV inside a simulation environment can be defined, then they can be compared to the path that the navigation algorithm converges to. An optimal path would be defined as a path from the start to the end of the environment with the shortest traversal time, the most simple and easy-to-track trajectories, and the least chance of collision. This could potentially indicate characteristics of an environment that MAV systems should look out for or point out a source of reward function characteristics for RL implementations.

Altitude Complexity

The altitude complexity score varies irregularly in altitude due to the complexity properties of each object. For example, the structure of a tree might be such that there are bushy branches at the bottom and top but not in the middle. This would mean that the complexity would go from more complex to less complex and then back to more complex. The height volumes used in these experiments are based on the heights of the Low, Middle, and High blockers, but a more granular division may yield results that are more easily interpreted. A more comprehensive description of altitude complexity could be defined. If the Low, Mid, and High height volumes were redefined to be more granular, more information on how the environment changes in complexity as altitude increases can be understood relative to obstacle avoidance.

Automating Complexity Score Calculation

As the number of object complexity properties increases, the longer it takes to assess the complexity score manually. In addition, objects vary in complexity as altitude increases, defining this complexity manually would be very time-consuming especially if the altitude increments are vastly smaller than the overall height of the object. As such, an automated way of evaluating this complexity would greatly benefit the ease of assessing the environment.

Bibliography

Theys, B., Dimitriadis, G., Hendrick, P., & De Schutter, J. (2016). Influence of propeller configuration on propulsion system efficiency of multi-rotor unmanned aerial vehicles. *2016 international conference on unmanned aircraft systems (ICUAS)*, 195–201.

Thipyopas, C., & Intaratep, N. (2011). Aerodynamics study of fixed-wing mav: Wind tunnel and flight test. *International Micro Air Vehicle conference and competitions 2011 (IMAV 2011),'t Harde, The Netherlands, September 12-15, 2011.*

Ellington, C. P., Van Den Berg, C., Willmott, A. P., & Thomas, A. L. (1996). Leading-edge vortices in insect flight. *Nature*, *384*(6610), 626–630.

Floreano, D., & Wood, R. J. (2015). Science, technology and the future of small autonomous drones. *nature*, *521*(7553), 460–466.

Ljungblad, S., Man, Y., Baytaş, M. A., Gamboa, M., Obaid, M., & Fjeld, M. (2021). What matters in professional drone pilots' practice? an interview study to understand the complexity of their work and inform human-drone interaction research. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–16.

Aurambout, J.-P., Gkoumas, K., & Ciuffo, B. (2019). Last mile delivery by drones: An estimation of viable market potential and access to citizens across european cities. *European Transport Research Review*, *11*(1), 1–21.

Albaker, B., & Rahim, N. (2009). A survey of collision avoidance approaches for unmanned aerial vehicles. *2009 international conference for technical postgraduates (TECHPOS)*, 1–7.

Zhang, J., Chadha, R. G., Velivela, V., & Singh, S. (2018). P-cap: Pre-computed alternative paths to enable aggressive aerial maneuvers in cluttered environments. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8456–8463.

Kumar, V., & Michael, N. (2012). Opportunities and challenges with autonomous micro aerial vehicles. *The International Journal of Robotics Research*, *31*(11), 1279–1291.

Aguilar, W. G., Casaliglla, V. P., & Pólit, J. L. (2017). Obstacle avoidance based-visual navigation for micro aerial vehicles. *Electronics*, *6*(1), 10.

Barry, A., Florence, P., & Tedrake, R. (2018). High-speed autonomous obstacle avoidance with pushbroom stereo. *Journal of Field Robotics*, *35*(1), 52–68.

Florence, P., Carter, J., & Tedrake, R. (2020). Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In *Algorithmic foundations of robotics xii* (pp. 304–319). Springer.

Beard, R. W., & McLain, T. W. (2012). *Small unmanned aircraft: Theory and practice*. Princeton university press.

Giancola, S., Valenti, M., & Sala, R. (2018). *A survey on 3d cameras: Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies*. Springer.

Prince, S. J. (2012). *Computer vision: Models, learning, and inference*. Cambridge University Press.

Barry, A. J., Oleynikova, H., Honegger, D., Pollefeys, M., & Tedrake, R. (2015). Fpga vs. pushbroom stereo vision for mavs. *IROS Workshop on Vision-based Control and Navigation of Small Lightweight UAVs*, *1*.

Falanga, D., Kleber, K., & Scaramuzza, D. (2020). Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, *5*(40), eaaz9712.

Gallego, G., Lund, J. E., Mueggler, E., Rebecq, H., Delbruck, T., & Scaramuzza, D. (2017). Event-based, 6-dof camera tracking from photometric depth maps. *IEEE transactions on pattern analysis and machine intelligence*, *40*(10), 2402–2412.

Song, Y., Naji, S., Kaufmann, E., Loquercio, A., & Scaramuzza, D. (2021). Flightmare: A flexible quadrotor simulator. *Proceedings of the 2020 Conference on Robot Learning*, 1147–1157.

Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2017). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *Field and Service Robotics*. https://arxiv.org/abs/1705.05065

Zhongming, Z., Linong, L., Xiaona, Y., Wangqiang, Z., & Wei, L. (2020). The state of the world's forests: Forests, biodiversity and people. *FAO, Rome, Italy*.

Groombridge, B., & Jenkins, M. (2002). *World atlas of biodiversity: Earth's living resources in the 21st century*. Univ of California Press.

Geckeler, C., Aucone, E., Schnider, Y., Simeon, A., von Bassewitz, J.-P., Zhu, Y., & Mintchev, S. (2024). Learning occluded branch depth maps in forest environments using rgb-d images. *IEEE Robotics and Automation Letters*, *9*(3), 2439–2446. https://doi.org/10.1109/LRA.2024.3355632

Tao, M. W., Wang, T.-C., Malik, J., & Ramamoorthi, R. (2014). Depth estimation for glossy surfaces with light-field cameras. *European Conference on Computer Vision*, 533–547.

Smolyanskiy, N., Kamenev, A., Smith, J., & Birchfield, S. (2017). Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4241–4247.

Zhang, Z., & Scaramuzza, D. (2018). Perception-aware receding horizon navigation for mavs. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2534–2541.

Zhang, J., Hu, C., Chadha, R. G., & Singh, S. (2020). Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, *37*(8), 1300–1313.

Florence, P. R., Carter, J., Ware, J., & Tedrake, R. (2018). Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 7631–7638.

Abbeel, P., Coates, A., Quigley, M., & Ng, A. (2006). An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems*, *19*.

Loquercio, A., Kaufmann, E., Ranftl, R., Müller, M., Koltun, V., & Scaramuzza, D. (2021). Learning high-speed flight in the wild. *Science Robotics*, *6*(59), eabg5810.

Sadeghi, F., & Levine, S. (2016). Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*.

Kong, F., Xu, W., Cai, Y., & Zhang, F. (2021). Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots. *IEEE Robotics and Automation Letters*, *6*(4), 7869–7876.

Liu, S., Atanasov, N., Mohta, K., & Kumar, V. (2017). Search-based motion planning for quadrotors using linear quadratic minimum time control. *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2872–2879.

Nous, C., Meertens, R., De Wagter, C., & De Croon, G. (2016). Performance evaluation in obstacle avoidance. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3614–3619.

Mishkin, D., Dosovitskiy, A., & Koltun, V. (2019). Benchmarking classic and learned navigation in complex 3d environments. *arXiv preprint arXiv:1901.10915*.

Ermacora, G., Sartori, D., Rovasenda, M., Pei, L., & Yu, W. (2020). An evaluation framework to assess autonomous navigation linked to environment complexity. *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, 1803–1810.

Appendix A

Processed Data

A.1 Processed Data Columns Explained

The processed data is generated by experimenting in both courses ten times per target speed. There are ten target speeds, starting at 1 m/s to 10 m/s in increments of 1 m/s at a time. These ten experiments per target speed are averaged and shown in the tables in the following section. Some of the columns in the processed data tables are intuitive and some are not. This section clarifies what each column means and how the data is interpreted.

Level: The segment in each course that increases in complexity from level 1 to level 10.

Time (s): The average time it takes the MAV to navigate a particular level.

Avg. Speed (m/s): The instantaneous speed is stored as MAV moves through each level. It is then used to calculate the average speed at which the MAV moved while navigating the course.

Avg. Collisions: The average number of collisions that occurred on a particular level. This is effectively the collision avoidance failure rate.

Stalls: The average stalls are the number of times that the MAV required manual intervention. This occurs when the obstacle avoidance system stops the MAV in a position such that it cannot find a collision-free path toward the goal position. This indicates a failure of autonomous navigation but not collision avoidance.

Start Alt. (m): The average height of the MAV during the first 1/3 of the total path length traveled in a particular level.

Mid Alt. (m): The average height of the MAV during the second 1/3 of the total path length traveled in a particular level.

End Alt. (m): The average height the MAV traveled at during the last 1/3 of the total path length traveled in a particular level.

Avg. Alt. (m): The average height the MAV traveled at in a particular level.

Collision Penalty (s): The average time penalty the MAV received based on the number of collisions and MAV target speed.

Corrected Time (s): The actual time it took the MAV to navigate the level plus the collision penalty incurred.

Nav. Success: The average number of times that the MAV was able to complete a level without collisions or stalls. This is effectively the navigational autonomy success rate.

Speed Given 0 Col. (m/s): The average speed at which the MAV navigated a level without collisions including stalls.

Percentage Target Speed Achieved: The average percentage of the target speed that the MAV was able to achieve whilst navigating a particular level.

Total Collisions: The total collision seen over all ten experiments on a particular level.

Total Stalls: The total stalls seen over all ten experiments on a particular level.

A.2 Processed Data: Trees Course

Level	Time (s)	Avg. Speed (m/s)	Avg. Collisions	Stalls	Start Alt. (m)	Mid Alt. (m)	End Alt. (m)	Avg. Alt. (m)	Collision Penalty (s)	Corrected Time (s)	Nav. Success	Speed Given 0 Col. (m/s)	% Target Speed Achieved	Total Collisions	Total Stalls
Averages of the 1 m/s experiments															
1	100.214	1.002	0	0	5.842	5.842	5.842	5.842	0	100.214	1	1.002	100.19%	0	0
2	102.366	1.003	0	0	5.842	5.827	5.804	5.824	0	102.366	1	1.003	100.28%	0	0
3	102.029	1.003	0	0	5.802	5.726	5.720	5.749	0	102.029	1	1.003	100.31%	0	0
4	102.146	0.996	0	0	5.720	5.720	5.720	5.720	0	102.146	1	0.996	99.57%	0	0
5	102.769	0.998	0	0	5.701	5.687	5.770	5.719	0	102.769	1	0.998	99.79%	0	0
6	115.739	0.967	0	0	7.951	8.123	8.080	8.052	0	115.739	1	0.967	96.73%	0	0
7	120.647	0.986	0	0	6.346	6.508	6.451	6.435	0	120.647	1	0.986	98.56%	0	0
8	119.419	0.980	0	0	6.251	5.394	5.702	5.783	0	119.419	1	0.980	97.96%	0	0
9	197.718	0.895	0	0	6.316	5.740	6.757	6.271	0	197.718	1	0.895	89.50%	0	0
10	208.767	0.738	0	0	6.631	6.417	6.384	6.478	0	208.767	1	0.738	73.82%	0	0
Averages of the 2 m/s experiments															
1	50.292	2.019	0	0	7.587	7.587	7.587	7.587	0	50.292	1	2.019	100.94%	0	0
2	50.223	2.020	0	0	7.587	7.574	7.559	7.573	0	50.223	1	2.020	100.99%	0	0
3	51.670	2.004	0	0	7.522	6.727	6.603	6.950	0	51.670	1	2.004	100.19%	0	0
4	50.787	2.034	0	0	6.584	6.537	6.537	6.553	0	50.787	1	2.034	101.70%	0	0
5	52.397	2.033	0	0	6.470	7.040	7.060	6.856	0	52.397	1	2.033	101.66%	0	0
6	54.886	2.052	0	0	6.864	6.908	7.177	7.316	0	54.886	1	2.052	102.62%	0	0
7	57.501	1.989	0	0	5.799	5.797	6.712	6.103	0	57.501	1	1.989	99.43%	0	0
8	78.748	1.745	0	0	6.427	5.378	4.405	5.403	0	78.748	1	1.745	87.25%	0	0
9	102.554	1.689	0	0	5.422	5.123	5.078	5.207	0	102.554	1	1.689	84.47%	0	0
10	118.827	1.585	0	0	5.154	4.500	5.200	4.951	0	118.827	1	1.585	79.23%	0	0
Averages of the 3 m/s experiments															
1	33.308	3.041	0	0	6.987	6.987	6.987	6.987	0	33.308	1	3.041	101.38%	0	0
2	33.451	3.045	0	0	6.987	6.987	6.987	6.987	0	33.451	1	3.045	101.50%	0	0
3	33.894	3.037	0	0	6.983	6.636	6.558	6.726	0	33.894	1	3.037	101.22%	0	0
4	33.857	3.040	0	0	6.429	6.328	6.328	6.362	0	33.857	1	3.040	101.35%	0	0
5	34.984	3.055	0	0	6.238	6.935	6.875	6.683	0	34.984	1	3.055	101.82%	0	0
6	36.670	3.056	0	0	7.808	6.811	7.077	8.232	0	36.670	1	3.056	101.85%	0	0
7	37.284	3.019	0	0	4.794	5.302	6.432	5.509	0	37.284	1	3.019	100.65%	0	0
8	52.169	2.629	0	0	6.298	5.295	4.503	5.365	0	52.169	1	2.629	87.63%	0	0
9	72.656	2.362	0	0	5.557	5.388	5.364	5.437	0	72.656	1	2.362	78.75%	0	0
10	70.114	2.276	0.1	0	5.822	5.766	6.541	6.043	1.5	71.614	0.9	2.097	75.88%	1	0
Averages of the 4 m/s experiments															
1	25.038	4.034	0	0	6.558	6.558	6.558	6.558	0	25.038	1	4.034	100.85%	0	0
2	25.400	4.078	0	0	6.558	6.539	6.501	6.532	0	25.400	1	4.078	101.96%	0	0
3	26.099	4.038	0	0	6.495	6.095	5.945	6.178	0	26.099	1	4.038	100.96%	0	0
4	25.351	4.091	0	0	5.936	5.936	5.936	5.936	0	25.351	1	4.091	102.27%	0	0
5	27.203	4.048	0	0	6.201	7.655	7.721	7.192	0	27.203	1	4.048	101.19%	0	0
6	28.819	3.914	0	0	8.412	7.006	7.337	7.585	0	28.819	1	3.914	97.86%	0	0
7	32.546	3.777	0	0	5.146	5.477	5.993	5.539	0	32.546	1	3.777	94.42%	0	0
8	32.663	3.691	0.1	0	5.735	5.296	5.042	5.358	2	34.663	0.9	3.335	92.29%	1	0
9	52.938	2.978	0.3	0	5.635	5.499	5.765	5.633	6	58.938	0.7	2.008	74.46%	3	0
10	56.474	2.591	0.5	0.1	6.220	5.473	5.573	5.755	10	66.474	0.6	1.562	64.78%	5	1
Averages of the 5 m/s experiments															
1	19.954	5.076	0	0	6.816	6.816	6.816	6.816	0	19.954	1	5.076	101.51%	0	0
2	20.087	5.131	0	0	6.827	6.839	6.839	6.835	0	20.087	1	5.131	102.61%	0	0
3	21.183	4.968	0	0	6.855	6.710	6.666	6.744	0	21.183	1	4.968	99.35%	0	0
4	20.503	5.083	0	0	6.690	6.690	6.698	6.693	0	20.503	1	5.083	101.65%	0	0
5	22.766	4.862	0	0	6.891	8.259	8.214	7.788	0	22.766	1	4.862	97.23%	0	0
6	24.266	4.658	0	0	8.027	6.115	6.733	6.958	0	24.266	1	4.658	93.15%	0	0
7	29.600	4.217	0	0	5.869	6.119	6.462	6.150	0	29.600	1	4.217	84.35%	0	0
8	36.876	3.701	0.1	0.2	6.013	5.188	4.899	5.367	2.5	39.376	0.9	3.436	74.03%	1	2
9	50.347	3.232	0	0	5.319	5.449	4.931	5.233	0	50.347	1	3.232	64.63%	0	0
10	57.585	2.681	0.4	0	6.416	6.738	5.885	6.346	10	67.585	0.6	1.638	53.61%	4	0

Table A.1: Processed data for the Trees course for target speeds 1-5m/s.

Averages of the 6 m/s experiments															
Level	Time (s)	Avg. Speed (m/s)	Avg. Collisions	Stalls	Start Alt. (m)	Mid Alt. (m)	End Alt. (m)	Avg. Alt. (m)	Collision Penalty (s)	Corrected Time (s)	Nav. Success	Speed Given 0 Col. (m/s)	% Target Speed Achieved	Total Collisions	Total Stalls
1	16.617	6.070	0	0	7.067	7.067	7.067	7.067	0	16.617	1	6.070	101.16%	0	0
2	16.744	6.128	0	0	7.072	7.088	7.088	7.083	0	16.744	1	6.128	102.14%	0	0
3	18.151	5.815	0	0	7.086	6.756	6.689	6.844	0	18.151	1	5.815	96.91%	0	0
4	17.177	6.057	0	0	6.699	6.689	6.689	6.692	0	17.177	1	6.057	100.96%	0	0
5	20.557	5.410	0	0	6.756	7.757	7.701	7.405	0	20.557	1	5.410	90.17%	0	0
6	20.353	5.362	0	0	8.083	6.584	6.533	7.067	0	20.353	1	5.362	89.37%	0	0
7	24.251	4.967	0	0	5.039	6.005	6.639	5.894	0	24.251	1	4.967	82.78%	0	0
8	35.369	3.928	0.4	0.2	5.936	4.879	5.556	5.457	12	47.369	0.6	2.627	65.46%	4	2
9	38.656	4.018	0.1	0	5.657	5.442	5.429	5.509	3	41.656	0.9	3.552	66.97%	1	0
10	48.951	3.253	0.1	0.2	6.264	5.941	6.792	6.332	3	51.951	0.9	2.947	54.21%	1	2
Averages of the 7 m/s experiments															
1	14.309	7.017	0	0	6.717	6.717	6.717	6.717	0	14.309	1	7.017	100.24%	0	0
2	14.877	6.911	0	0	6.757	6.714	6.606	6.692	0	14.877	1	6.911	98.73%	0	0
3	15.865	6.538	0	0	6.664	6.673	6.682	6.673	0	15.865	1	6.538	93.40%	0	0
4	15.554	6.715	0	0	6.585	6.628	6.543	6.585	0	15.554	1	6.715	95.93%	0	0
5	18.540	5.979	0	0	6.614	7.465	7.508	7.195	0	18.540	1	5.979	85.42%	0	0
6	20.432	5.604	0	0	7.935	7.249	7.581	7.588	0	20.432	1	5.604	80.05%	0	0
7	26.921	4.868	0.1	0	5.888	6.629	6.536	6.351	3.5	30.421	0.9	4.478	69.55%	1	0
8	35.047	3.838	0.5	0.1	6.143	4.921	5.426	5.497	17.5	52.547	0.6	2.303	54.84%	5	1
9	46.872	3.378	0.7	0.2	5.695	5.793	5.654	5.714	24.5	71.372	0.4	1.548	48.26%	7	2
10	49.693	2.977	0.3	0	7.333	7.397	7.599	7.443	10.5	60.193	0.8	2.522	42.52%	3	0
Averages of the 8 m/s experiments															
1	12.571	8.007	0	0	6.488	6.488	6.488	6.488	0	12.571	1	8.007	100.08%	0	0
2	12.888	7.845	0	0	6.549	6.606	6.588	6.581	0	12.888	1	7.845	98.06%	0	0
3	14.310	7.170	0	0	6.664	6.623	6.577	6.622	0	14.310	1	7.170	89.62%	0	0
4	13.958	7.381	0	0	6.555	6.513	6.516	6.528	0	13.958	1	7.381	92.26%	0	0
5	17.263	6.357	0	0	6.353	6.760	7.078	6.730	0	17.263	1	6.357	79.46%	0	0
6	18.615	5.959	0	0	7.999	7.294	7.807	7.700	0	18.615	1	5.959	74.49%	0	0
7	22.182	5.270	0	0	6.188	6.536	6.525	6.416	0	22.182	1	5.270	65.88%	0	0
8	29.378	4.342	0.2	0	6.764	5.536	4.992	5.764	8	37.378	0.8	3.384	54.28%	2	0
9	37.151	3.912	0.3	0.1	6.174	5.992	5.504	5.890	12	49.151	0.7	2.745	48.90%	3	1
10	44.581	3.293	0.1	0	6.966	6.669	7.176	6.937	4	48.581	0.9	2.893	41.17%	1	0
Averages of the 9 m/s experiments															
1	11.155	9.012	0	0	7.143	7.143	7.143	7.143	0	11.155	1	9.012	100.13%	0	0
2	12.174	8.392	0	0	7.247	7.402	7.506	7.385	0	12.174	1	8.392	93.25%	0	0
3	13.359	7.886	0	0	7.507	7.097	6.845	7.150	0	13.359	1	7.886	87.62%	0	0
4	15.110	7.455	0	0	6.874	6.707	6.752	6.778	0	15.110	1	7.455	82.83%	0	0
5	18.274	6.461	0.1	0	6.524	7.191	7.396	7.037	4.5	22.774	0.9	5.973	71.79%	1	0
6	19.029	6.320	0.2	0	8.018	7.447	7.940	7.802	9	28.029	0.8	4.703	70.22%	2	0
7	30.969	4.668	0.5	0.1	7.534	6.971	7.191	7.232	22.5	53.469	0.6	3.170	51.87%	5	1
8	38.657	4.011	0.9	0.1	7.007	5.583	5.358	5.983	40.5	79.157	0.3	1.361	44.57%	9	1
9	56.572	2.887	0.9	0.3	6.494	7.477	7.391	7.120	40.5	97.072	0.3	0.961	32.07%	9	3
10	57.483	2.850	0.5	0.1	6.079	6.986	8.510	7.858	22.5	79.983	0.7	2.095	31.67%	5	1
Averages of the 10 m/s experiments															
1	10.029	10.026	0	0	6.150	6.150	6.150	6.150	0	10.029	1	10.026	100.26%	0	0
2	10.978	9.395	0	0	6.132	6.139	6.182	6.151	0	10.978	1	9.395	93.95%	0	0
3	11.171	9.268	0	0	6.209	6.112	6.218	6.180	0	11.171	1	9.268	92.68%	0	0
4	11.865	8.854	0	0	6.371	6.622	6.608	6.534	0	11.865	1	8.854	88.54%	0	0
5	16.987	7.364	0.1	0	6.292	6.798	7.127	6.739	5	21.987	0.9	6.818	73.64%	1	0
6	19.599	6.340	0.6	0	7.862	7.270	7.140	7.424	30	49.599	0.5	3.308	63.40%	6	0
7	28.666	4.768	1.1	0.1	6.903	7.027	6.457	6.796	55	83.666	0.3	1.444	47.68%	11	1
8	34.098	4.674	0.7	0	6.412	6.549	6.542	6.501	35	69.098	0.4	1.724	46.74%	7	0
9	38.175	4.261	0.7	0	6.318	6.546	6.587	6.484	35	73.175	0.4	1.805	42.61%	7	0
10	55.186	3.044	1.5	0.1	7.044	8.008	7.525	7.526	75	130.186	0.1	0.376	30.44%	15	1

Table A.2: Processed data for the Trees course for target speeds 6-10m/s.

A.3 Processed Data: Pillars Course

Level	Time (s)	Avg. Speed (m/s)	Avg. Collisions	Stalls	Start Alt. (m)	Mid Alt. (m)	End Alt. (m)	Avg. Alt. (m)	Collision Penalty (s)	Corrected Time (s)	Nav. Success	Speed Given 0 Col. (m/s)	% Target Speed Achieved	Total Collisions	Total Stalls
Averages of the 1 m/s experiments															
1	100.316	1.011	0	0	6.822	6.822	6.822	6.822	0	100.316	1	1.011	101.13%	0	0
2	101.067	1.012	0	0	6.822	6.822	6.822	6.822	0	101.067	1	1.012	101.25%	0	0
3	102.299	1.011	0	0	6.825	6.903	7.032	6.920	0	102.299	1	1.011	101.09%	0	0
4	106.909	1.023	0	0	7.408	7.551	7.700	7.553	0	106.909	1	1.023	102.29%	0	0
5	112.082	1.018	0	0	7.807	7.814	8.375	7.999	0	112.082	1	1.018	101.76%	0	0
6	106.778	1.024	0	0	6.783	9.311	9.793	9.296	0	106.778	1	1.024	102.36%	0	0
7	110.215	1.013	0	0	9.752	12.887	13.034	12.558	0	110.215	1	1.013	101.30%	0	0
8	108.100	1.029	0	0	12.901	19.181	20.262	18.115	0	108.100	1	1.029	102.87%	0	0
9	115.750	1.020	0	0	19.694	21.689	21.555	21.646	0	115.750	1	1.020	101.96%	0	0
10	146.984	0.909	0	0	20.890	19.569	19.734	20.064	0	146.984	1	0.909	90.91%	0	0
Averages of the 2 m/s experiments															
1	50.180	2.014	0	0	6.808	6.808	6.808	6.808	0	50.180	1	2.014	100.71%	0	0
2	50.345	2.019	0	0	6.808	6.808	6.808	6.808	0	50.345	1	2.019	100.97%	0	0
3	51.528	2.025	0	0	6.809	6.855	6.914	6.859	0	51.528	1	2.025	101.23%	0	0
4	52.423	2.035	0	0	7.290	7.488	7.507	7.428	0	52.423	1	2.035	101.75%	0	0
5	55.697	2.035	0	0	7.513	7.527	8.395	7.811	0	55.697	1	2.035	101.75%	0	0
6	52.941	2.039	0	0	8.756	9.299	9.627	9.227	0	52.941	1	2.039	101.93%	0	0
7	53.744	2.038	0	0	11.475	12.660	12.885	12.340	0	53.744	1	2.038	101.88%	0	0
8	53.796	2.057	0	0	15.325	19.828	20.897	18.683	0	53.796	1	2.057	102.83%	0	0
9	58.942	1.990	0	0	22.063	21.869	21.534	21.822	0	58.942	1	1.990	99.48%	0	0
10	71.277	1.923	0	0	20.707	20.358	19.195	20.086	0	71.277	1	1.923	96.13%	0	0
Averages of the 3 m/s experiments															
1	33.392	3.030	0	0	7.292	7.292	7.292	7.292	0	33.392	1	3.030	101.00%	0	0
2	33.589	3.029	0	0	7.289	7.298	7.309	7.299	0	33.589	1	3.029	100.97%	0	0
3	34.053	3.030	0	0	7.310	7.357	7.484	7.384	0	34.053	1	3.030	100.98%	0	0
4	35.373	3.030	0	0	8.056	8.258	8.420	8.245	0	35.373	1	3.030	101.01%	0	0
5	36.853	3.041	0	0	8.530	8.551	9.057	8.713	0	36.853	1	3.041	101.37%	0	0
6	35.695	3.041	0	0	9.554	10.670	11.316	10.513	0	35.695	1	3.041	101.37%	0	0
7	36.106	3.043	0	0	12.678	13.492	13.690	13.287	0	36.106	1	3.043	101.42%	0	0
8	37.662	3.029	0	0	15.363	19.713	20.933	18.670	0	37.662	1	3.029	100.96%	0	0
9	44.502	2.867	0.1	0	21.934	21.834	21.195	21.654	1.5	46.002	0.9	2.654	95.55%	1	0
10	51.601	2.528	0	0.1	20.761	19.537	19.618	19.972	0	51.601	1	2.528	84.27%	0	1
Averages of the 4 m/s experiments															
1	25.128	4.023	0	0	7.789	7.789	7.789	7.789	0	25.128	1	4.023	100.58%	0	0
2	25.236	4.033	0	0	7.789	7.789	7.789	7.789	0	25.236	1	4.033	100.84%	0	0
3	25.687	4.025	0	0	7.797	7.885	7.957	7.880	0	25.687	1	4.025	100.63%	0	0
4	26.665	4.028	0	0	8.583	8.872	9.062	8.839	0	26.665	1	4.028	100.70%	0	0
5	28.266	4.006	0	0	9.198	9.212	9.745	9.385	0	28.266	1	4.006	100.14%	0	0
6	31.250	3.760	0	0	10.741	11.486	12.004	11.410	0	31.250	1	3.760	93.99%	0	0
7	31.016	3.731	0.1	0.2	13.339	14.084	14.391	13.938	2	33.016	0.9	3.496	93.28%	1	2
8	27.929	4.023	0	0	15.297	18.927	20.170	18.131	0	27.929	1	4.023	100.59%	0	0
9	41.844	3.224	0.1	0.3	21.803	21.834	21.757	21.798	2	43.844	0.9	3.042	80.59%	1	3
10	41.053	3.369	0	0.1	21.296	20.461	19.320	20.359	0	41.053	1	3.369	84.23%	0	1
Averages of the 5 m/s experiments															
1	19.977	5.061	0	0	7.685	7.686	7.686	7.685	0	19.977	1	5.061	101.21%	0	0
2	20.237	5.008	0	0	7.695	7.719	7.723	7.712	0	20.237	1	5.008	100.17%	0	0
3	21.053	4.887	0	0	7.732	7.801	7.933	7.822	0	21.053	1	4.887	97.74%	0	0
4	22.241	4.773	0	0	8.081	8.150	8.551	8.261	0	22.241	1	4.773	95.45%	0	0
5	24.415	4.689	0	0	8.727	8.837	9.418	8.994	0	24.415	1	4.689	93.78%	0	0
6	23.742	4.618	0	0	10.348	11.362	11.853	11.188	0	23.742	1	4.618	92.36%	0	0
7	24.401	4.572	0	0	12.824	13.177	13.221	13.074	0	24.401	1	4.572	91.44%	0	0
8	24.964	4.555	0	0.2	14.878	19.238	20.435	18.184	0	24.964	1	4.555	91.09%	0	2
9	28.566	4.410	0	0	21.677	21.433	21.588	21.566	0	28.566	1	4.410	88.21%	0	0
10	36.433	3.810	0	0	21.071	19.967	19.698	20.245	0	36.433	1	3.810	76.19%	0	0

Table A.3: Processed data for the Pillars course for target speeds 1-5m/s.

Averages of the 6 m/s experiments															
Level	Time (s)	Avg. Speed (m/s)	Avg. Collisions	Stalls	Start Alt. (m)	Mid Alt. (m)	End Alt. (m)	Avg. Alt. (m)	Collision Penalty (s)	Corrected Time (s)	Nav. Success	Speed Given Success (m/s)	% Target Speed Achieved	Total Collisions	Total Stalls
1	16.600	6.081	0	0	7.810	7.811	7.811	7.811	0	16.600	1	6.081	101.35%	0	0
2	16.935	6.014	0	0	7.817	7.843	7.858	7.839	0	16.935	1	6.014	100.23%	0	0
3	17.366	5.933	0.1	0	7.862	7.899	7.912	7.891	3	20.366	0.9	5.338	98.89%	1	0
4	18.634	5.788	0	0	8.732	9.239	9.641	9.204	0	18.634	1	5.788	96.47%	0	0
5	19.452	5.751	0	0	9.799	9.921	10.734	10.151	0	19.452	1	5.751	95.86%	0	0
6	19.610	5.708	0	0	11.712	12.925	13.898	12.845	0	19.610	1	5.708	95.14%	0	0
7	20.769	5.489	0	0	14.306	14.795	15.295	14.799	0	20.769	1	5.489	91.49%	0	0
8	19.502	5.786	0	0.1	16.928	19.986	20.894	19.269	0	19.502	1	5.786	96.43%	0	1
9	28.322	4.614	0.1	0.1	21.803	21.774	22.013	21.863	3	31.322	0.9	4.106	76.91%	1	1
10	26.707	4.699	0	0.1	21.387	21.305	19.813	20.835	0	26.707	1	4.699	78.31%	0	1
Averages of the 7 m/s experiments															
1	14.344	7.056	0	0	8.380	8.382	8.382	8.382	0	14.344	1	7.056	100.81%	0	0
2	14.573	6.941	0	0	8.398	8.415	8.415	8.409	0	14.573	1	6.941	99.16%	0	0
3	15.662	6.618	0	0	8.415	8.475	8.490	8.460	0	15.662	1	6.618	94.55%	0	0
4	17.010	6.303	0	0	8.865	9.089	9.643	9.199	0	17.010	1	6.303	90.05%	0	0
5	17.934	6.239	0.1	0	10.105	10.380	11.455	10.647	3.5	21.434	0.9	5.641	89.13%	1	0
6	18.449	6.086	0.1	0	12.366	13.482	14.123	13.324	3.5	21.949	0.9	5.524	86.94%	1	0
7	18.694	6.034	0.1	0	14.714	15.203	16.060	15.326	3.5	22.194	0.9	5.482	86.19%	1	0
8	18.343	6.162	0	0.1	17.326	20.454	21.015	19.598	0	18.343	1	6.162	88.03%	0	1
9	24.193	5.052	0.1	0.2	21.816	21.814	22.557	22.062	3.5	27.693	0.9	4.500	72.17%	1	2
10	26.373	4.729	0	0	21.602	20.483	20.629	20.905	0	26.373	1	4.729	67.55%	0	0
Averages of the 8 m/s experiments															
1	12.526	8.055	0	0	6.716	6.717	6.717	6.717	0	12.526	1	8.055	100.69%	0	0
2	13.007	7.830	0	0	6.758	6.855	6.877	6.830	0	13.007	1	7.830	97.87%	0	0
3	13.687	7.544	0	0	6.927	7.084	7.163	7.058	0	13.687	1	7.544	94.29%	0	0
4	15.339	7.130	0	0	8.097	8.603	9.156	8.619	0	15.339	1	7.130	89.12%	0	0
5	15.816	7.035	0.1	0	9.622	9.865	11.305	10.264	4	19.816	0.9	6.341	87.94%	1	0
6	18.272	6.591	0	0	13.016	14.719	16.281	14.672	0	18.272	1	6.591	82.39%	0	0
7	16.102	6.903	0.1	0	17.137	17.445	18.416	17.666	4	20.102	0.9	6.239	86.29%	1	0
8	16.877	6.750	0	0	19.615	21.375	21.464	20.818	0	16.877	1	6.750	84.38%	0	0
9	25.474	5.050	0.5	0.1	21.705	21.902	22.095	21.901	20	45.474	0.6	2.757	63.13%	5	1
10	20.237	5.870	0	0	21.426	20.474	20.176	20.692	0	20.237	1	5.870	73.38%	0	0
Averages of the 9 m/s experiments															
1	11.143	9.044	0	0	7.564	7.566	7.566	7.565	0	11.143	1	9.044	100.49%	0	0
2	11.984	8.561	0	0	7.676	7.798	7.831	7.768	0	11.984	1	8.561	95.12%	0	0
3	12.883	8.101	0	0	7.973	8.121	8.201	8.098	0	12.883	1	8.101	90.02%	0	0
4	14.915	7.257	0	0	8.777	9.147	9.694	9.206	0	14.915	1	7.257	80.64%	0	0
5	15.283	7.455	0.1	0	10.591	11.165	12.305	11.354	4.5	19.783	0.9	6.687	82.83%	1	0
6	18.060	6.650	0.1	0	13.546	15.765	17.698	15.670	4.5	22.560	0.9	5.912	73.88%	1	0
7	14.998	7.319	0	0.1	18.451	18.929	19.599	18.993	0	14.998	1	7.319	81.32%	0	1
8	16.073	6.981	0	0.1	20.372	21.274	21.270	20.972	0	16.073	1	6.981	77.56%	0	1
9	23.055	5.762	0.1	0.3	21.949	21.746	21.971	21.888	4.5	27.555	0.9	5.098	64.02%	1	3
10	22.292	5.619	0	0.1	21.810	21.354	20.583	21.249	0	22.292	1	5.619	62.43%	0	1
Averages of the 10 m/s experiments															
1	9.971	10.116	0	0	7.806	7.811	7.812	7.810	0	9.971	1	10.116	101.16%	0	0
2	11.053	9.272	0	0	8.025	8.166	8.214	8.135	0	11.053	1	9.272	92.72%	0	0
3	12.090	8.581	0.1	0	8.353	8.683	8.750	8.595	5	17.090	0.9	7.744	85.81%	1	0
4	14.831	7.327	0.1	0	9.291	9.603	10.256	9.716	5	19.831	0.9	6.484	73.27%	1	0
5	16.758	6.916	0.1	0	10.893	11.357	12.318	11.523	5	21.758	0.9	6.163	69.16%	1	0
6	18.919	6.570	0.1	0	13.290	14.215	15.162	14.223	5	23.919	0.9	6.048	65.70%	1	0
7	15.257	7.264	0.1	0.1	15.570	16.113	16.730	16.137	5	20.257	0.9	6.568	72.64%	1	1
8	19.575	6.319	0.1	0	16.491	18.088	19.365	17.982	5	24.575	0.9	5.874	63.19%	1	0
9	26.837	5.051	0	0	19.901	21.074	21.553	20.843	0	26.837	1	5.051	50.51%	0	0
10	27.214	5.019	0.2	0.1	20.586	20.118	17.898	19.534	10	37.214	0.8	4.164	50.19%	2	1

Table A.4: Processed data for the Pillars course for target speeds 6-10m/s.