Quadcopter Behaviour Identification

by

Ahmad Traboulsi

A dissertation submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Carleton University

Ottawa, Ontario

Abstract

Commercial drones have become much popular in recent years. Their low cost and high capabilities have rendered drones feasible for different applications. These include weather observations, traffic monitoring, inspection of buildings, fire detection, delivery of goods, agriculture and security. However, the abuse of the advancements of drone capabilities can lead to security and public safety issues. Such abuses include the transfer of illegal or dangerous goods, assaults and terrorizing actions, espionage or spying. In such a context, can the behavior of a quadcopter be determined from observations? Can those observed behaviours be utilized for training a machine learning model for classifying future comportment? In this work, we look at three pieces of information that we can predict about a quadcopter or a group of quadcopters, leveraging behavior observations. First, we try to predict the formation a group of drones intend to make while in transition by training a machine learning model, based on Softmax regression, on navigational data collected for this purpose. Second, we train an Long Short-Term Memory (LSTM) neural network on Mel Frequncy Cepstral Coefficients (MFCCs), which are audio features extracted from an acoustic signal, to predict the weight of the payload of a quadcopter. Furthermore, in our third task, we identify whether a drone pilot is a human or an autopilot using a deep neural network trained on features processed from collected navigational data. In each of the three tasks, we evaluate features extracted from the data, then build and evaluate different models. The best-performing models in each of the three tasks are then compared to three different dummy classifiers. This comparison is made by performing statistical

significance tests demonstrating that the difference in performance is not just a result of a statistical fluke. The three dummy classifiers do not learn any patterns. However, they are based on different classifying strategies, such as always using the most frequent label, generates predictions by respecting the training set's class distribution, or generates predictions uniformly at random.

Contents

Abbreviations

3DVFH 3D Vector Field Histogram.

3DVFH* 3D Vector Field Histogram with A* algorithm.

6-DOF Six Degrees Of Freedom.

ANOVA Analysis Of Variance.

AUC Area Under the ROC Curve.

AUV Autonomous Underwater Vehicle.

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart.

CV Certainty Value.

DCT Discrete Cosine Transform.

DFNN Deep Forward Neural Network.

DFT Discrete Fourier Transform.

DOA Direction of Arrival.

FFT Fast Fourier Transform.

FN False Negative.

FP False Positive.

FPR False Positive Rate.

FPV First-Person View.

LPCC Linear Predictive Cepstral Coefficient.

LPCCs Linear Predictive Cepstral Coefficients.

LSTM Long Short-Term Memory.

MFCC Mel Frequncy Cepstral Coefficient.

MFCCs Mel Frequncy Cepstral Coefficients.

PD Proportional-Derivative.

PID Proportional-Intergal-Derivative.

POI Point-of-Interest.

PRC Precision-Recall Curve.

PRM Probabilistic Roadmap.

PRMs Probabilistic Roadmaps.

RF Radio Frequency.

RNN Recurrent Neural Network.

ROC Receiver Operating Characteristic Curve.

ROS Robot Operating System.

RRT Rapidly Exploring Random Trees.

RSS Received Signal Strength.

SNR Signal-to-Noise Ratio.

SNRs Signal-to-Noise Ratios.

SSB Sum of Squares Between Groups.

SSW Sum of Squares Within Groups.

SVM Support Vector Machine.

TN True Negative.

TP True Positive.

UAV Unmanned Aerial Vehicle.

UAVs Unmanned Aerial Vehicles.

UMVs Unmanned Maritime Vehicles.

VFH Vector Field Histogram.

Chapter 1

Introduction

This chapter defines what drone behaviour is. It states the research questions addressed in the thesis and the approach taken. Moreover, we summarize the contributions, list the publications made during researching this dissertation and outline this work.

We first define the behaviour of a drone as actions it takes as a response to a stimulus. The stimulus can be an input of a path planning algorithm that the drone is executing, an obstacle that causes the drone to take action to maneuver around it or some payload the drone is carrying. We refer to Nehmzow's definition for a more

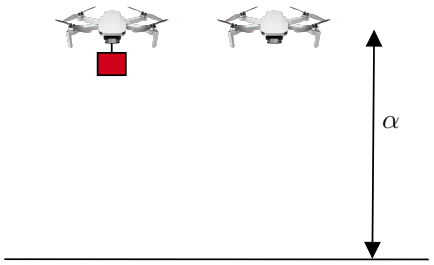


Figure 1.1: Two drones flying at the same altitude α. The left one has a payload while the other has no payload.

general definition of behaviour [1]:

> *... but the question it addresses is that of "analysing behaviour," where behaviour is a very loose concept that could refer to the motion of a mobile robot, the trajectory of a robot arm, a rat negotiating a maze, a carrier pigeon flying home, traffic on a motorway, or traffic on a data network.*

To better illustrate the meaning or the concept of the behaviour of a drone, let us consider two drones that are flying at an altitude α, as shown in Figure 1.1. The drone carrying payload requires its propellers to spin at a higher rate than the drone with no load to maintain the same altitude α. Therefore, the drone carrying payload exhibits a different behaviour than the other one. Another example of behaviour is the set of actions a drone takes when facing an obstacle. For example, a drone might fly over the obstacle or maneuver around it. The drone behaviour, in this case, is the set of actions it takes to avoid the obstacle.

Our intuition for solving classification problems using drone behaviours builds upon the fact that the behaviour changes, as per Nehmzow [1], when a change in the drone's control code, environment, or physical makeup occurs.

Figure 1.2: Behaviour is the result of the interaction between the three components: Task, drone, and environment.

The interaction between the physical makeup, environment and controlling program results in the behaviour as shown in Figure 1.2. As per Nehmzow [1], if we can measure a behaviour quantitatively, then when any of the two components of drone, environment, and task have remained unaltered, the third component can be characterized using those measurements. This also means that a classifier performs better

under the condition of fixing two components. The sensitivity of a classifier performance to the change in the two fixed components is specific to many aspects, including the classification task, application domain and features used.

1.1 Research Questions

In this thesis, we consider the following research questions. Can the behavior of a quadcopter be determined from observations, including telemetry and audio captures? Can those observed behaviours be utilized for training a machine learning model for classifying future drone comportment? We specifically address the following challenges:

1. Identify the formation a group of drones is trying to achieve.

2. Identify whether a drone is carrying payload or not, and if so, what is the estimated weight of the payload.

3. Identify whether a human or an autonomous system is piloting a drone under certain conditions.

1.2 Motivation

Unmanned Aerial Vehicles (UAVs) are becoming increasingly attractive due to being of great utility because of their flexibility, mobility and adaptive altitude. Their applications are in defence, search and rescue operations, wireless networking [2], delivery of goods, area mapping, monitoring and surveillance, building inspections, agriculture and weather forecast. Unfortunately, abuse of Unmanned Aerial Vehicle (UAV) technology has been happening more often lately. For example, terrorists used armed drones to struck oil pumping stations and other facilities in Saudi Arabia [3]. A more concerning incident is the suicide drone, an autonomous drone developed by a Turkish company that targeted a soldier trying to retreat as per [4]. The suicide drone was

autonomous and required no human control or data connectivity between an operator and the munition [4].

Moreover, a rogue drone also led to the closure of controlled airspace at Wellington Airport in New Zealand [5]. Lately, drones have also been used in drug smuggling [6]. According to the article, the border patrol does not have the technology to contain the drones. Another incident of illegal goods transport has been seen in prisons, in which drones are used to smuggle drugs and tobacco into the walls [7]. The drone was exposed by its buzzing sound that got the attention of an officer. The abuse of drones technology is a critical issue that needs to be addressed.

The motivation comes with understanding the threats that come with the advancements of UAVs. One step towards stopping rogue drones is learning more about them by monitoring their behaviour, after which a suitable counter-action might be taken. We identify three pieces of information that might be crucial in determining the correct counter-action. First, by telling what formation a rogue group of drones is trying to achieve beforehand, we can better mitigate the damage the drones might be up to. For instance, if their formation is changing to one that enables a larger lethal radius, the decision to take the counter-action should be handled faster. Another piece of information that matters is whether the drone is carrying a payload or not. Knowing whether it has a payload or not is a piece of important information that might give a better understanding of the risks associated with shooting the drone. For example, what if a rogue drone carries a payload and is over a military base. The payload may be hazardous or may be explosive. A better decision of the right counter action might be taken in this case to minimize the risks. The third piece of information is whether a human or an autopilot is piloting the drone. It might not be enough to jam the signal in the latter case as the drone might be autonomous and need no signal to operate. Therefore a kinetic approach such as shooting it with a net to capture it might be the best option.

1.3 Approach

Our strategy is to use machine learning techniques along with surveillance technologies such as acoustic sensors, radars, RF sensing, and video surveillance to monitor behaviour and find out crucial information about a drone or a group of drones for better and effective decision making processes.

The field of pattern recognition is concerned with discovering regularities in data through the algorithms and then use these regularities to classify data [8]. Machine learning is closely related to pattern recognition. Bishop [8] considered the two approaches to be two facets of the same field. Both approaches learn from data, build a model and come to conclusions about never seen before data. We focus on and use machine learning techniques in this work.

Once behaviour is defined and captured through observations. We train machine learning models to classify those behaviours. We collect navigational and acoustic data and label them for the purpose of capturing behaviours and classifying them. After data collection, we process the data further to extract features relevant to the classification problem. The features are then evaluated with respect to their importance using various strategies. Finally, the data is split into a training and a test set for the purpose of model training and evaluation.

In this work, we look at several classification problems. First, we try the best-case scenario in two of the classification problems. We define the best-case scenario as one where we fix two of the three components shown in Figure 1.2. In the third classification problem, we look at varying two of the three components shown in Figure 1.2 while fixing one. In the latter, we classify the drone's pilot while considering different environments by generating different ones in training and testing datasets. In each classification task, we measure the behaviour and use those measurements to train a model to classify the task or the physical makeup of the drone. Finally, the model performance is evaluated.

Comparison of machine learning algorithms is not the focus of this thesis. However, Softmax regression is suitable for multi-class classification and therefore, it is the model used for classifying different formations of a group of drones. LSTM, on the other hand, is well suited for sequential data such as time-series data, and it is used in classifying payloads using features extracted from audio signals. Finally, a deep feed-forward neural network was chosen to identify the pilot due to the capabilities of a neural network of approximating non-linear functions.

The specific approach for each task is as follow:

- **Recognition of Formation Intentions**. We build a simulation of a group of drones undergoing different formations. We collect navigational data. Data is then processed, and features are extracted related to identifying the formation. This work aims to identify the formation, which is the arrangement of drones positioned in 3D space, the group of drones intend to make. A machine learning model is trained to identify the formation while the flight is in progress. The model is then evaluated, and results are analyzed and presented. The model is also compared to three dummy classifiers that are used as a baseline. Those classifiers are based on three strategies: Predict uniformly at random, always predict the most frequent label in the training data set or predict respecting the class distributions in the training data set. The comparison shows a statistically significant difference between the performance of the model trained on data and each dummy classifiers.

- **Identification of Drone Payload Using Mel-Frequency Cepstral Coefficients and Neural Networks**. We identify the weight of the payload of a drone using features of an audio signal. First, audio recordings of a drone carrying different weights are collected and labelled. Audio features are then extracted and used to train a neural network to identify the weight of a drone's payload. The model is then evaluated using a test set generated with different Signal-to-Noise

Ratios (SNRs) from a set that has not been used in training. The performance is then analyzed and presented. We also compare the performance between the model and three dummy classifiers used as baseline classifiers. We show a statistically significant difference between the performance of the model and each of the three dummy classifiers.

- **A Reverse Turing Like Test for Quad-copters**. For the purpose of building a classifier that would identify the pilot of a drone, whether a human is controlling the drone remotely or an autopilot, we build two types of simulations using Robot Operating System (ROS) and Gazebo. One type of simulation consists of a human controlling the Iris drone to reach a goal position that is set behind an obstacle. The other type of simulation consist of an autopilot controlling the Iris drone with a goal position set behind an obstacle. We vary the environment in each type of simulation by randomizing the position of the obstacle and goal. However, we restrict the goal position to always be behind the obstacle, such that the pilot has to maneuver around the obstacle to reach the goal. Navigational data is collected. Features are then extracted and evaluated. A neural network is then built for the purpose of classification. We present the performance of the classifier. We also perform a comparison to show a statistically significant difference between the performance of the model and each of three dummy classifiers.

1.4 Contributions

Three different behaviours are sought in this work. First, drones making a specific formation take specific actions cooperatively together. These cooperative actions are the behaviour we seek to identify and classify the formation. The second behaviour is the acoustic signal generated by the propellers as a drone carries different weights of payload. Finally, the third behaviour is the set of actions a drone takes when facing an obstacle.

The contributions of this work include proofs that each of the behaviours we seek can be characterized and captured using methods we present. Furthermore, we show that a model trained on features extracted from the data that captures the behaviour sought, relevant to the classification task, outperforms three baseline dummy classifiers.

The specifics of each chapter contribution are as follow. For the task of formation classification, we consider the drone cooperative actions as behaviours captured in navigational data. We show that it is possible to use features extracted from navigational data to classify the formation a group of drones intends to make. We also show that a model trained on those features outperforms three baseline classifiers.

In the task of payload weight identification, we use an acoustic sensor to collect data from a drone flown indoor with different payloads. The different behaviours of a drone carrying different payloads are captured in the audio signal. We show and analyze the performance of the classifier trained on those features. We also show that it outperforms three baseline classifiers.

Finally, we capture the different behaviours of a drone as it faces an obstacle when on autopilot and when being piloted by a human. The different behaviours are captured through navigational data. We show that the performance of the classifier trained on the navigational data outperforms all three dummy classifiers.

1.5 Outline

Chapter 2 looks at related work where a behaviour is analyzed for a classification task. Based on the task, we summarize and categorize the related work in a table at the end of Chapter 2. Chapter 3 provides a background on the knowledge, tools and libraries required to identify the features for machine learning models, build simulations, build ROS modules to collect data, and build machine learning models with specifics to neural networks. In Chapter 4, we simulate a group of three drones making different formations. Then, we collect data and build a machine learning model to predict the

formation the drones intend to make while in transition, and then analyze the results and present our conclusion for Chapter 4. Next, in Chapter 5, a Parrot Mambo drone is flown indoor. Audio signals are collected and analyzed for feature extraction. A neural network is then built to classify the weight of the drone payload. Finally, in Chapter 6, we take on the task of identifying the drone pilot, whether that is a human or an autonomous system. We first build simulations with specific conditions requiring the drone pilot to maneuver around an obstacle to reach their goal position. A neural network is then built and trained using the features extracted from data collected in the simulations. We conclude our study in Chapter 7 by summarizing the results and stating some future work.

1.6 Publications

Publications that were done during the writing of this dissertation.

- A. Traboulsi and M. Barbeau. Recognition of Drone Formation Intentions Using Supervised Machine Learning. In 2019 International Conference on Computational Science and Computational Intelligence (CSCI), pages 408–411. IEEE Computer Society, (2019) [9].

- A. Traboulsi and M. Barbeau. Identification of Drone Payload Using Mel-Frequency Cepstral Coefficients and LSTM Neural Networks. In Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1, pages 402–412, (2021). Springer International Publishing [10].

- A. Traboulsi and M. Barbeau. A Reverse Turing Like Test for Quad-Copters. In Proceedings of the 2021 17th Annual International Conference on Distributed Computing in Sensor Systems (DCOSS) [11].

Chapter 2

Related Work

2.1 Introduction

Commercial drones' wide availability and their enhanced capabilities have made many drones applications feasible. Soon we will start seeing more drones in our skies. Those include drones with applications in building inspections, delivery of goods, traffic monitoring, agriculture and security. However, as much as this advancement in drones capabilities will facilitate our daily lives, it might also lead to security and public safety issues. Technology in the wrong hands could be disastrous. In order to avoid damages that might be caused due to assaults, transfer of illegal goods or any other abuses of drones, we need to have systems that can monitor and identify possible threats and take actions.

Drones are difficult to detect using conventional radars. It is challenging because micro-drones have low Radar Cross-Section and usually fly at lower altitudes and slower speeds than other aircraft, which means conventional radar systems might not detect and identify micro-drones.

Furthermore, if a rogue drone was detected, the question comes on how to deal with the situation as it might be carrying explosives and shooting it might not be ideal. On the other hand, the drone might be fully autonomous, requiring no signal to navigate

and reach its target, in which a jamming signal will not be helpful. It might also be the case that more than one drone is engaging in an attack, and detecting a transition in the formation of the swarm/group might be helpful info on how to proceed with a counteraction.

This research aims to address behaviours related to these issues and develop machine learning models for classification tasks utilizing these behaviours. The following section presents a literature review on related matters, specifically detection, identification, classification and localization of drones, drone payload, and drone activities, using several methods. At the end of the chapter, We summarize the related work in Table 2.1.

2.2 Literature Review

Detection of a drone and classification of its model and flight mode based on RF signal. Al-Sa'd et al. [12] developed an open-source database that contains the RF signals of drones with their respective drone type and flight mode. The authors then use the database to develop three deep neural networks to detect the presence of a drone, identify its type and flight mode. Specifically, they build three neural networks, the first one is for the sole purpose of detecting the presence of a drone, the second network is to detect the presence of a drone and its type, and the last one for detecting the presence of a drone, its type and its flight mode. Unfortunately, the performance of the third neural network decreased significantly. The authors explained the decrease in performance as an effect of having around ten classes and the similarity of the RF spectra between two of the drones that the same manufacturer makes.

Detection, Localization and Jamming of a detected drone. Shi et al. [13] have reviewed multiple anti-drone systems with multiple surveillance technologies. They categorized the surveillance technologies into radars, audio, video and RF. The paper further discusses the drawback of each technology, and precisely high-power

radar cannot be used in crowded urban areas, which is inappropriate and forbidden due to the harm it may cause. On the other hand, audio surveillance is sensitive to ambient noise and is limited in range detection. While in video surveillance, it is an object detection problem in computer vision, has difficulty distinguishing drones from other small objects like birds in cluttered backgrounds. Finally, for RF to identify the source as a drone, a MAC address database is required, which is not feasible due to the increasing variety of drones. The work done by Shi et al. presents an anti-drone system based on audio, video and RF surveillance technologies. The work done extracted the features of the drone using all surveillance technologies. For the acoustic feature, the distribution of the harmonics strengths is used. Histogram of orient gradient is used as the image feature. Finally, the normalized received strength signal is used as the RF feature vector. Three Support Vector Machine (SVM) classifiers are used for drone detection, in which first a moving object is detected using the inter-frame difference method and then SVM is used to identify if the moving object is a drone or not. A logical OR between the three classifiers are used to determine if a drone is detected or not. Once a drone is detected, localization of the drone is then performed via the estimation of Direction of Arrival (DOA) measurements relative to the L-shape acoustic array using the MUSIC algorithm. In addition to the acoustic array, the DOA and Received Signal Strength (RSS) relative to the RF sensor and images from the camera are used for localization purposes. Finally, after localization, jamming of the RF signal is done to defend against drones. System performance evaluation shows that audio surveillance performance in detection decreases as the distance of the drone from the sensor increases. Like audio surveillance, the performance of the video surveillance detection decreases as the sensor-drone distance increases. Finally, the fusion of all surveillance technologies achieves better performance but increases false alarm detection.

Detection and localization. An anti-drone system based on the cognitive internet of things has been implemented with the code name Dragnet in [14]. The authors classify the studies related to such systems into four categories: warning techniques,

spoofing techniques, jamming techniques, and mitigation techniques. The latter includes destroying or capturing the drone. Data analytics is the core of cognitive tasks for detection, localization and tracking in Dragnet. Heterogeneous data fusion from different detection approaches is done. Two techniques are mentioned for tracking the drone, and the first is context-aware tracking via trajectory filtering for drones. The other is drone behaviour prediction by mining massive historical data on amateur drones activities.

Drone detection using Mel Frequncy Cepstral Coefficient (MFCC) and hidden markov model. Detecting a drone by the sound emitted by its propellers has been studied by Shi, Ahmad, He and Chang [15]. First, the authors extracted MFCC as features of the audio signal. The features are then used in a hidden Markov model classifier. The authors also tested two schemes of MFCC; one uses 24 features while the other uses thirty-six features. The experiment results show that the model has high recognition rates even in noisy environments.

Drone detection using MFCC, Linear Predictive Cepstral Coefficient (LPCC) and Support Vector Machine (SVM). Anwar, Kaleem and Jamalipour [16] developed a machine learning sound-based amateur drone detection for public safety that uses audio features. Both MFCC and LPCC are considered as features for a SVM model to classify whether the sounds are that of a drone, a bird, a plane or a thunderstorm. The model that used MFCC outperformed the model using LPCC features.

Using micro-Doppler signatures to classify the type of a rotary-wing UAV in terms of the number of rotors. An important step is to understand the capabilities of the UAV which is being analyzed for intention identification. Equipped with the knowledge about the specific UAV a more appropriate countermeasure can be taken. A piece of vital information that would help in understanding those capabilities is finding the type of rotary-wing the UAV belongs to, whether that is a helicopter, quadcopter, hexacopter or octocopter. By doing such classifications of the UAV, we

can understand more about the maneuvering capabilities of the UAV. The more we know about the capabilities of the UAV the better we can plan the countermeasures to stop it. A step toward this kind of classification as to which type of rotary-wing does the UAV belongs to and the size of its rotor diameter has been investigated in [17] by Wit, Harmanny, and Premel-Cabic. The study shows the potential of using a micro-Doppler signature for rotary-wing type identification.

Drone payload detection using micro-Doppler signature and multistatic radar. Another piece of information that would help us in identifying the drone's intention is whether it is carrying a load or not. A drone carrying a load may be an indication of possible hostile activity. Fioranelli, Ritchie, Griffiths and Borrion [18] attempts to classify micro-drones to be loaded or not loaded. The approach is similar to what has been done in [17] in the sense of using a micro-Doppler signature and a multi-static radar. The features used are based on the micro-doppler signature, and those features are the Doppler centroid and Doppler bandwidth. Naïve Bayes and the diagonal-linear variant of the discriminant analysis classifier are used for classification. Two approaches were tested, one in which the classifier was fed the three samples, one from each radar and the centralized classifier made the classification. In another approach, three separate classifiers were used along with a voting procedure for the final decision.

Drone activity/maneuvers recognition. Machine learning has been used in past research for drone activity recognition. Drone actions being performed could be hovering, flying forward (pitch forward) or flying backward (pitch backward). The goal that has been pursued in the literature is figuring out the action that a single drone is performing using data obtained from sensors. Research on drone activity recognition has been conducted by Barták and Vemlelová [19]. The authors study both supervised and unsupervised learning to identify the activities of a flying drone. In unsupervised learning, the authors use and compare several approaches, including $k-$mean clustering, mixture of Gaussians, hierarchical clustering and hidden Markov

model. However, the authors found that unsupervised techniques fail to discover true control activities satisfactorily. Unlike unsupervised learning techniques, decision tree learning was found to be much more accurate.

Recognizing the activity of an aquatic drone has been done by Castellini et al. [20]. The authors used unsupervised learning techniques to analyze unlabeled data. The unlabelled data are sensor readings and commands to propellers. The approach was validated using annotated dataset. The performance evaluation of the framework employed two measures, purity and silhouette, 0.94 and 0.20, respectively, both of which were considered encouraging.

Robot formations. Research has been done on recognition of robots formations and strategies in RoboCup Baez [21] and Trevisan and Veloso [22]. In [22] actual data from RoboCup tournaments have been processed to find strategies and compare strategies of two teams. The comparison of strategies relies on the Frobenius norm of matrices representing the duration of what the authors define as an episode. An episode is defined as the time interval when the ball is in the opponent's field, and the team is performing an attack strategy. The authors then use the comparison of strategies technique to classify the teams by their defence strategy. Baez [21] predicts opponent team activity using a Markov model. Each opponent state constitutes the position and the velocity of the robot. The centroid is defined to be the team's average position. The possible state transitions are stored, and the probabilities are learned. The prediction is reduced to finding the most probable state that the robot will transition to.

Drone Point-of-Interest (POI). Commercial drones provide First-Person View (FPV)) over Wi-Fi channels. Such streams sent over Wi-Fi can be intercepted using an RF scanner and a Network Interface Controller (NIC) in promiscuous mode. Moreover work has been done by Nassi, Ben-Netanel, Shamir and Elovici [23] to detect streamed POI from FPV channel, in which the target point of interest is extracted from the FPV channel.

Turing test for pilot identification. The application of the Turing test to UAVs

has been studied in the literature. The work by Young in [24] provides four protocols to evaluate autonomous systems flying UAVs. The author suggests that human observers, a group of subject matter expert reviewers, monitor the UAVs performing multiple mission tasks. Moreover, the paper mentions some cues used to differentiate between whether an aerial vehicle is piloted or autonomous. Note that a piloted aircraft, as defined by the author, is one where a pilot or operator is either physically on-board or remotely controls it. In both cases, the operator is directly providing real-time flight control inputs and thus operating the aircraft.

Artificial Intelligence and autonomous systems evaluation. Hernandez-Orallo [25] has divided the system evaluation types in artificial intelligence to be either task-oriented or ability-oriented. Autonomous navigation systems in drones are considered specialized systems that require task-oriented evaluation. Furthermore, the author has defined three types of behavioural evaluation for task-oriented evaluation: human discrimination, problem benchmarks and peer confrontation. In human discrimination, the assessment is made by observation. The Turing test falls under this type of evaluation. The idea is to evaluate the system by human judges or by comparing those systems to humans.

Reverse Turing test. The reverse Turing test is related to the Turing test. An example of a reverse Turing test is the Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) [26]. It is a test to distinguish a bot from a human. In this test, a program tells whether the user is a bot or a human. Many variations of this test have been proposed, such as Activity Recognition CAPTCHA [27].

Turing test in the automotive field. In [28], Kalik and Prokhorov present an implementation of the Turing test in the automotive field. Rather than assessing the intelligence of a computer program, the authors extend the scope of the test to the domain of intelligent automotive vehicles, consider different formats for such a test, and study various measures produced by these tests. Both real-world environments

and simulated environments have been considered in their research. An interesting part of the research is where the authors suggest using a passive interrogation method, which departs from the original Turing test by removing the interaction between the interrogator and subject, and converting the interrogator's role to an observer role. The authors also mention the possibility of implementing a computerized classifier system, placing humans out of the testing loop. It is the latter approach we adopt in this work.

A CAPTCHA variation using distorted speech. Another variation of the CAPTCHA using speech is studied in [29] by Kochanski, Lopresti and Shih. The paper presents a test that depends on the fact that humans can better recognize distorted speech far more robust than automatic speech recognition techniques.

A reverse Turing test for machine-made texts.A reverse Turing test for detecting machine-made texts is proposed in [30] by Shao, Uchendu and Lee. The authors study the classification task of distinguishing between human-made vs. machine-made texts. Their study utilizes financial earning reports, research articles, and chatbot dialogues. Their model achieved an 84% accuracy.

Unmanned Maritime Vehicles (UMVs) autonomous assessment. Underwater vehicle metric assessment concerning autonomous capabilities has been studied by Insuarralde et Lane [31]. The authors review approaches to assess autonomous behaviours in UMVs. They provide a baseline study for methodologies to assess maritime autonomy so that metrics can be defined for UMVs. Also, the authors present a statistical comparison of existing assessment frameworks for autonomy in different domains. The authors in [32] also review the current approaches to assess autonomous behaviour in self-governed vehicles. The paper also proposes metrics for undersea autonomy to identify the degree of autonomy in an Autonomous Underwater Vehicle (AUV).

Table 2.1: Related Work Summary

Drone Detection	**Drone Activity Detection**	**Payload Detection**	**Drone Type**	**Turing Tests and Autonomous Systems Evaluation**
RF signal for drone detection [12]	Figuring out an action a single drone is performing (hovering, pitching forward, backward etc.) [19, 33] and for an aquatic drone [20]	Detecting whether a drone is carrying any payload or not using micro-Doppler signature [18]	Using micro-Doppler signature for rotary wing type identification [17]	Turing tests for human identification in aerial vehicles and automotive [24, 28]
Fusion of multi-surveillance technologies (radar, video, audio and RF) [13]	Robot formations in robocup [21, 22, 34]	Detecting explosives, drugs and chemical compounds [35]		Artificial intelligence and autonomous system evaluation [25, 31, 32]
Using cognitive internet of things for drone detection [14]	Drone group formation detection while in transition [9]			Reverse Turing tests [26, 27, 29, 30]
Audio for drone detection [15, 16]	Detecting POI of a drone's FPV channel [23]			

Chapter 3

Background

Feature engineering is the process of selecting relevant features for the task. The background in the domain of the task is required to extract better features. For this purpose, we define our drone model and then the controllers used in the simulations done in this work. Next, we look at specific machine learning algorithms used in this work. Finally, a brief introduction to each of the tools and libraries used to develop the simulations, algorithms and results in this work.

3.1 Quadcopter Concept, Dynamics Modelling, and Controllers

To be able to produce data for machine learning models, we use a simulator of some drones. By doing so, much more data can be generated with much less time by reducing field testing times and avoiding crashes of real quad-copters. The components found on real quad-copters are simulated by Gazebo plugins and by the Gazebo physics engine. The quadcopter simulator model and controller are explained in the following subsection.

3.1.1 Modelling Quad-copter Dynamics

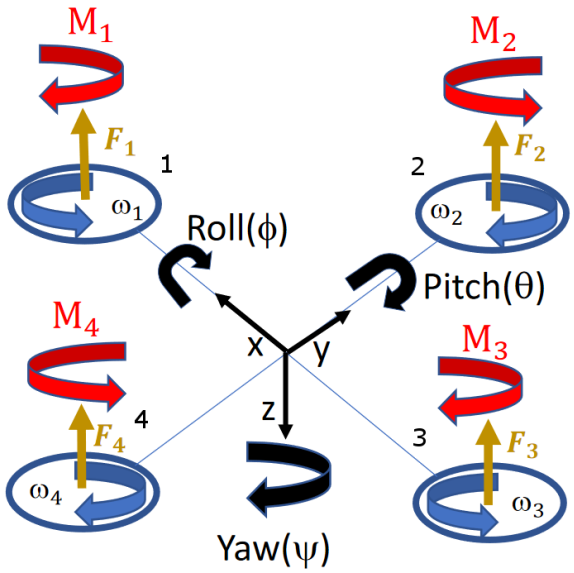


Figure 3.1: Caption

The quadcopter includes four rotors, each with its propeller laid out in an X configuration as shown in Figure 3.1. Quad-copters may also include a variety of sensors. Opposite motors spin in the same direction but opposing the other pair to balance the total system torque.

The importance of modelling a quadcopter is that it allows the development of control strategies for it. The quadcopter's motion can be divided into two systems, the rotational motion subsystem and the translational motion subsystem. The rotational motion subsystem is fully actuated. However, the translational subsystem is underactuated.

The quad-copter is a rigid body having Six Degrees Of Freedom (6-DOF), therefore, six variables are used to express its position in 3D space (x, y, z, ϕ, θ and ψ). Each of $x, y,$ *and* z represent the distance of the quad-copter center of mass along the x, y *and* $z - axis$ respectively from the inertial frame (earth frame). The three Euler angles ϕ, θ *and* ψ represent the orientation of the quad-copter about each of the x, y *and* $z - axis$ respectively.

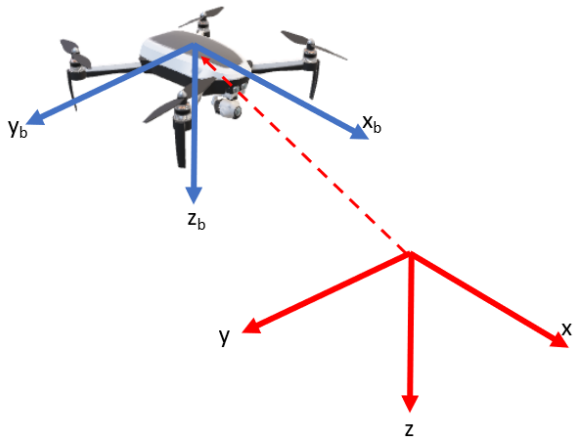


Figure 3.2: Caption

Before discussing the modelling of the quadcopter, we first need to define the body coordinate frame and the inertial frame. The earth frame, defined as the inertial frame, is a fixed frame of reference. The body frame, as shown in Figure 3.2 is centered at the quadcopter body. In order to transform coordinates from the inertial frame to the body frame, we use a rotation matrix R.

A rotation matrix performs a rotation in the Euclidean space. Two intermediate coordinate systems Fr_1 and Fr_2 are used to transform from the inertial frame to the body frame. The notation R_x^y indicates a rotation from frame x to frame y. $R_i^{Fr_1}$ is a rotation from the inertial frame to the intermediate frame Fr_1:

$$R_i^{Fr_1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix}$$

$R_{Fr_1}^{Fr_2}$ indicates a rotation from the intermediate frame Fr_1 to the second intermediate frame Fr_2 by rotating Fr_1 around its $y-axis$ by a pitch angle θ:

$$R_{Fr_1}^{Fr_2} = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

Finally, we rotate frame Fr_2 around the $z-axis$ by the yaw angle ψ, which results in the body frame.

$$R^b_{Fr_2} = \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R^b_i = R^b_{Fr_2} R^{Fr_2}_{Fr_1} R^{Fr_1}_i$$

$$R^b_i = \begin{pmatrix} \cos(\psi)\cos(\theta) & \cos(\phi)\sin(\psi)+\cos(\psi)\sin(\phi)\sin(\theta) & \sin(\phi)\sin(\psi)-\cos(\phi)\cos(\psi)\sin(\theta) \\ -\cos(\theta)\sin(\psi) & \cos(\phi)\cos(\psi)-\sin(\phi)\sin(\psi)\sin(\theta) & \cos(\psi)\sin(\phi)+\cos(\phi)\sin(\psi)\sin(\theta) \\ \sin(\theta) & -\cos(\theta)\sin(\phi) & \cos(\phi)\cos(\theta) \end{pmatrix}$$

It is important to note that R^b_i performs roll first, pitch second, and yaw third. The order of rotations defines the rotation matrix. The order of rotations is opposite to that of the order of multiplication of the rotation matrices.

To get from the body frame to the inertial frame, the inverse of R^b_i is used, which is also its transpose:

$$R = (R^b_i)^T = R^i_b$$

The quad-copter's motion can be divided into two subsystems:

1. Rotational motion (roll, pitch and yaw)
2. Translational motion

The rotational motion is fully actuated while the translational subsystem is underactuated. Newton's second law of motion states that the vector sum of all forces acting on an object equals the mass m of the object times its acceleration.

$$F = ma$$

Besides gravitational force, there are the forces F_B acting on the body frame. We use the rotation matrix R^i_b that transforms a vector from a body frame to the inertial frame

to translate those forces to the inertial frame.

$$ma = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R_b^i F_B \tag{3.1}$$

The Forces F_1,F_2, F_3 and F_4 in Figure 3.1 are forces produced by the rotors and known as lift force, and is defined as the aerodynamic force constant K_f times the square of angular velocity of the rotor.

$$F_i = K_f \omega_i^2$$

When the quad-copter is not rolling or pitching, the only non-gravitational forces acting on it is thrust which is produced by the rotation of the propellers.

$$F_B = \begin{bmatrix} 0 \\ 0 \\ -(F_1 + F_2 + F_3 + F_4) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -K_f(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{bmatrix}$$

Let $U_1 = (F_1+F_2+F_3+F_4)$ be the input. Substituting the input U_1 in Equation (3.1) We have acceleration in the X, Y and Z directions in terms of U_1:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + m^{-1} R_b^i \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} = \begin{pmatrix} -\frac{U_1 \sin(\theta)}{m} \\ \frac{U_1 \cos(\theta) \sin(\phi)}{m} \\ g - \frac{U_1 \cos(\phi) \cos(\theta)}{m} \end{pmatrix} \tag{3.2}$$

We derive the rotational equation of motion using Euler's rotation equation for rigid body dynamics:

$$M_B = I \cdot \alpha + \omega \times (I \cdot \omega) \tag{3.3}$$

Where α is the angular acceleration, I is the inertia matrix and ω is the angular velocity.

M_B is the vector of moments acting on each of $X - axis$, $Y - axis$ and $Z - axis$.

$$M_B = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} \tag{3.4}$$

The moments and forces generated by the propellers can be seen in Figure 3.1. The rotating propeller generates the forces F_1, F_2, F_3 and F_4, which pushes air down, causing the upward force (thrust). In addition, rotating propellers generate the moments M_1, M_2, M_3 and M_4, and each of those moments is equal to the constant K_M times the square of the angular velocity of that propeller. The thrust forces also generate moments about the body frame's axes.

$$M_i = K_m \omega_i^2 \tag{3.5}$$

A negative moment about the $X - axis$ is generated by the force F_2, and this moment is equal to F_2 times the moment arm L. A positive moment about the $X - axis$ is generated by the thrust force F_4 and is equal to F_4 times the moment arm L. The total moment about the $X - axis$ is equal to the sum of those two moments.

$$M_x = -F_2 L + F_4 L \tag{3.6}$$

A negative moment about the $Y - axis$ is generated by the force F_3, and this moment is equal to F_3 times the moment arm L. A positive moment about the $Y - axis$ is generated by the thrust force F_1 and is equal to F_1 times the moment arm L. The total moment about the $Y - axis$ is equal to the sum of those two moments.

$$M_y = -F_3 L + F_1 L \tag{3.7}$$

The thrust force generated by the rotors, do not cause any moments about the

$Z - axis$. However, the rotating propellers generates the moments M_1, M_2, M_3 and M_4 about the $z - axis$.

$$M_z = M_1 - M_2 + M_3 - M_4 \tag{3.8}$$

Equation (3.3) becomes:

$$\begin{bmatrix} L(F_4 - F_2) \\ L(F_1 - F_3) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} = I \cdot \alpha + \omega \times (I \cdot \omega) \tag{3.9}$$

Furthermore the matrix on the left hand side of the equation can be rewritten as follow:

$$\begin{bmatrix} 0 & -L & 0 & L \\ L & 0 & -L & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = I \cdot \alpha + \omega \times (I \cdot \omega) \tag{3.10}$$

Where

$$\gamma = \frac{K_m}{K_f}$$

Let the left side of the Equation 3.10 be our inputs U_2:

$$U_2 = \begin{bmatrix} 0 & -L & 0 & L \\ L & 0 & -L & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{3.11}$$

The inputs U_{2_1}, U_{2_2}, and U_{2_3} are as follow:

$$U_{2_1} = L(F_2 - F_4) \tag{3.12}$$

$$U_{2_2} = L(F_1 - F_3) \tag{3.13}$$

$$U_{2_3} = \gamma F_1 - \gamma F_2 + \gamma F_3 - \gamma F_4 \tag{3.14}$$

The angular velocity and angular acceleration along each of the $X-axis$, $Y-axis$, $Z-axis$ are as follow:

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \alpha = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{3.15}$$

Substituting each of 3.11 and 3.15 in equation 3.10:

$$U_2 = I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.16}$$

3.1.2 Controller

Types of controllers

The are four main types of controlling techniques, linear, non linear, hybrid and learning based flight controllers [36]. Most common control techniques used are Proportional-Intergal-Derivative (PID) or Proportional-Derivative (PD) controllers [36].

We focus on the controllers used by the simulators we used in the upcoming chapters. The first controller used in Chapter 4, is based on PID controllers. The second controller used in Chapter 6 is a non-linear geometric controller. In the following, we introduce each briefly.

PID Controller

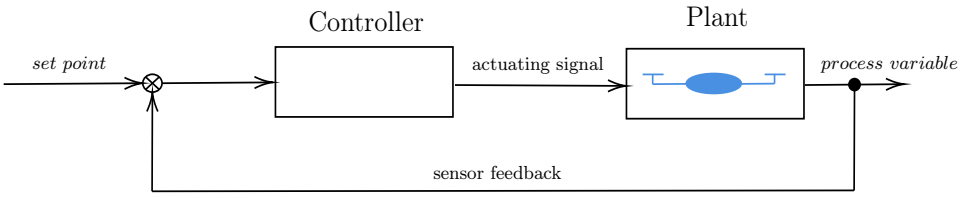


Figure 3.3: Feedback loop controller

Before we proceed with the PID specifics, we will define what a process variable and a set point are. A process variable is the system parameter that needs to be controlled. A sensor is used to measure the process variable and provide feedback to the control system. A setpoint is the desired value of the process variable. The main idea of PID controller is reading sensor data then compute the desired actuator signal that would drive the measured process variable to the desired set point. The control system algorithm uses the difference between the process variable and the setpoint, the PID, to determine the desired actuating signal. This type of control system is called a closed-loop system because of the feedback component as shown in Figure3.3.

The performance of such a system is often measured by applying a step function as the setpoint command variable and then measuring the response of the process variable.

The response is then quantified by measuring the defined waveform characteristics:

1. Rise Time, is the amount the system takes to go from 10 to 90% of the steady-state, or final, value.

2. Percent overshoot is the amount the process variable overshoot the final value, expressed as a percentage of the final value.

3. Settling time is the time required for the process variable to settle to within a certain percentage (5% commonly) of the final value.

4. Steady-state error is the final difference between the process variable and set point.

Disturbance rejection of the control system measures how well the control system can overcome the effect of disturbances.

A nonlinear system is one in which control parameters that produce the desired response at one operating point might not produce a satisfactory response at another operating point. Robustness is defined as the measure of how well the control system tolerates disturbances and nonlinearities. The three components of a PID are as follow:

- The proportional component is the difference between the setpoint and the process variable and is referred to as the error term. The proportional gain determines the ratio of output response to the error signal. Large gains may cause the system to have large oscillations in the process variable, which would cause the system to become unstable.

- The integral component sums the error term over time. Even a small error will drive the integral component to increase slowly and continue until the error reaches zero.

- The derivative component has a dampening effect that causes the output to decrease if the process variable increases rapidly. This component is susceptible to noise in the process variable. A noisy sensor feedback signal might cause the control system to become unstable.

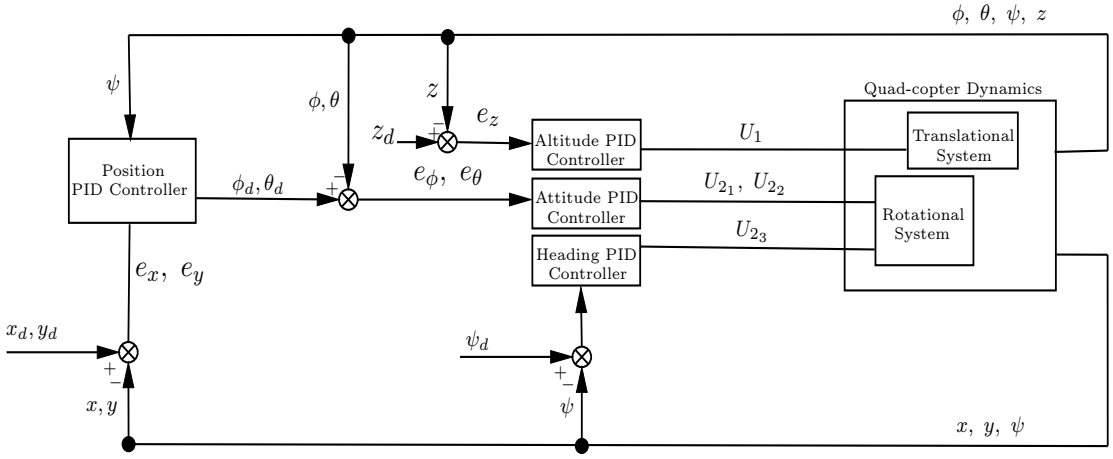


Figure 3.4: Drone controller using multiple PID controllers.

In a PID controller, the output consists of summing each of the proportional, integral and derivative terms. The controller we use in Chapter 4, consists of multiple PID controllers as shown in Figure 3.4. A position controller takes as an input the error position along the $x - axis$ and $y - axis$, that is, the difference between the desired position and the estimated current position. In addition, it takes as an input the estimated current yaw ψ and generates as output the desired roll ϕ and pitch θ. The difference between the desired roll and pitch with the estimated roll and pitch is the input of the attitude PID. The attitude PID generates the inputs U_{2_1} and U_{2_2} (equations (3.12) and (3.13)) to the rotational subsystem. The heading PID controller takes as an input the difference between the desired yaw and the estimated yaw and generates the input U_{2_3} (Equation (3.14)). Finally, the attitude controller takes as an input the difference between the desired altitude and the currently estimated altitude and generates the input U_1.

Geometric Flight Controller

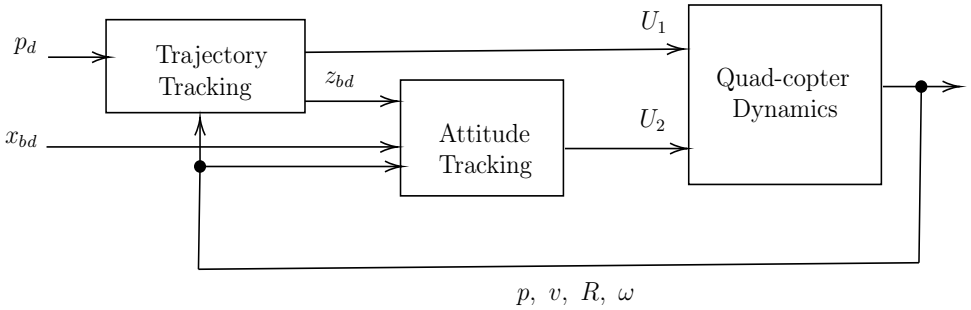


Figure 3.5: Geometric flight controller.

A non-linear geometric tracking controller developed in [37] and used in the simulator [38] is briefly introduced in this section. Lee, Leok and McClamroch [37] developed a geometric tracking controller that follows a prescribed trajectory of the location of the center of gravity of the quad-copter and the desired direction of the heading of the quadcopter.

The controller is a cascade controller as shown in Figure 3.5. The trajectory tracking component takes as an input the desired position $p_d \in \mathbb{R}^3$ and produces z_{bd} the desired direction of the z-axis of the fixed body frame. Once z_{bd} is chosen, x_{bd} is selected. x_{bd} corresponds to the desired heading direction of the quad-copter in the plane normal to z_{bd}. This is done by projecting x_{bd} onto the plane normal to z_{bd} as shown in Figure 3.6. The deisred attitude R_d becomes as follow:

$$R_d = \Big[y_d \times z_{bd}, y_{bd}, z_{bd} \Big]$$

Where y_{bd} is:

$$y_{bd} = \frac{z_{bd} \times x_{bd}}{||z_{bd} \times x_{bd}||}$$

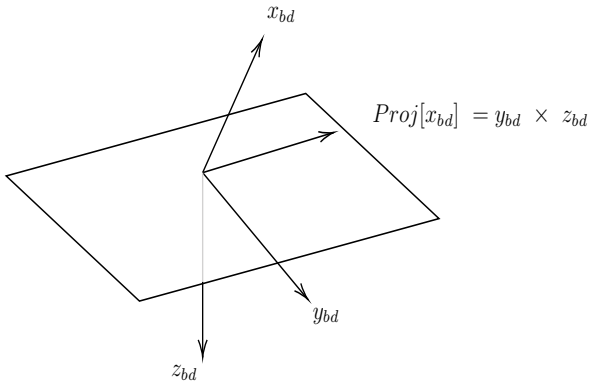


Figure 3.6: Projecting x_{bd} onto the plane normal to z_{bd}.

Tracking Errors

We define the tracking errors as in [37] of each of:

$p \in \mathbb{R}^3$, the location of center of mass in the inertial frame:

$$e_p = p - p_d$$

$v \in \mathbb{R}^3$, the velocity of center of mass in the inertial frame:

$$e_v = v - v_d$$

Where v is:

$$v = \dot{p}$$

$R \in SO(3)$, the rotation matrix from the body-fixed frame to the inertial frame. The attitude tracking error is chosen to be:

$$e_R = \frac{1}{2}(R_d^T R - R^T R_d)^\vee$$

The vee map $^\vee : SO(3) \longrightarrow \mathbb{R}^3$ is the inverse of the hat map. The hat operator $\hat{}$ takes a vector and transforms it into its equivalent matrix. That is the hat

map $\hat{}\ : \mathbb{R}^3 \longrightarrow SO(3)$.

$$\mathbf{a} \times \mathbf{b} = \hat{\mathbf{a}}\mathbf{b}$$

example of the hat map, in three dimensions,

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \times \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \hat{\mathbf{a}}\mathbf{b}$$

Finally, the tracking error for the angular velocity vector Ω is chosen to be as follow:

$$e_\Omega = \Omega - R^T R_d \Omega_d$$

Tracking Controller

For a given p_d, x_{bd}, and some positive constants k_p, k_v, k_R, k_Ω, then U_1, U_2 becomes as follow:

$$U_1 = -(-k_p e_p - k_v e_v - mge_3 + m\ddot{p}_d) \cdot R_{e_3},$$

$$U_2 = -k_R e_R - k_\Omega e_\Omega + \Omega \times I\Omega - I(\hat{\Omega} R^T R_d \Omega_d - R^T R_d \dot{\Omega}_d)$$

3.2 Motion/Trajectory Planning

Definition and Problem Description

Motion planning is about determining where to go based on a set of goals and objectives. The problem description can be formulated as follow, given obstacles, a drone and its maneuvering capabilities compute a collision-free motion from the start to goal.

There are three competing paradigms in path planning, those are classical robots; reactive paradigms; and hybrid approach. The classical robots are known as the exact

models and require no sensing at all. On the other hand, the reactive paradigms relies heavily on good sensing and does not use a model. Finally the hybrid approach utilizes both techniques, a model-based at higher levels and reactive at lower levels.

Path Planning - Navigation

In navigation path planning, the planner requires as an input the geometry of the environment, the geometry and kinematics of the drone in addition to the initial and goal configurations. The output of the planner is a collision free path from the initial configuration to the goal configurations as illustrated in Figure 3.7.

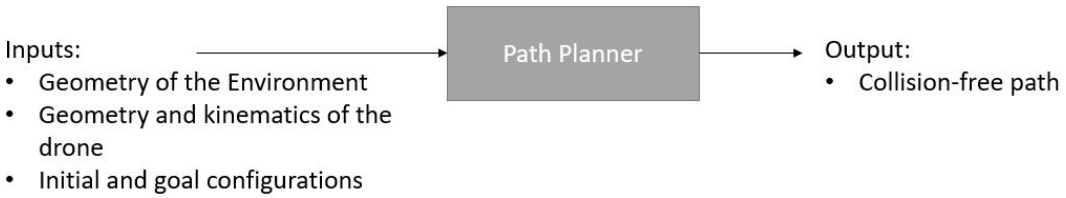


Figure 3.7: Path planner - Navigation

Configuration Space

The set of all possible transformations that could be applied to a drone is known as the configuration space. One of the main concepts for motion planning is a configuration, which is a specification of the position and orientation of an object in the system. The space of all such configurations is called a configuration space.

Formally, we define the configuration space $C = \mathbb{R}^3 \times SO(3)$, it is the Cartesian product between $\mathbb{R}^3$, which represents the coordinates of the body frame relative to the inertial frame, and $SO(3)$, which represents the rotation matrices that define the orientation of this frame relative to the inertial frame. Therefore, the configuration space is six-dimensional. Three of the coordinates describe the center of mass of the drone and three are Euler angles that describe the orientation of the drone.

Consider a world $W = \mathbb{R}^3$ that comprises an obstacle region, $O \subset W$, and a

drone, $D \subset W$. Let $q \in C$ denote a configuration of D, where $q = (x, y, z, \phi, \theta, \psi)$. The configuration space is further divided into two disjoint sets, such that, $C = C_{obs} \cup C_{free}$.

The obstacle region, C_{obs}, is defined as follow:

$$C_{obs} = \{q \in C | D(q) \cap O \neq \emptyset\}$$

It is the set of all configurations such that the transformed drone $D(q)$, using configuration q, intersect with the obstacle region O. C_{free} is what left from configurations in C:

$$C_{free} = C \setminus C_{obs}$$

C-Space Discretization

There are two competing paradigms in C-space discretization. The first is combinatorial planning which is also known as exact planning. The second is sampling-based planning which is a probabilistic/randomized kind of planning. However, due to that fact that combinatorial methods are inefficient when used in high-dimensional problems this chapter will only focus on sampling-based methods.

Sampling-Based Methods

The main idea is to avoid the explicit construction of C_{obs} and instead conduct a search that probes the C-space with a sampling scheme [39]. This probing is done by a collision detection module, which the motion planning algorithm considers as a black-box [39]. Sampling-based methods include Probabilistic Roadmaps (PRMs) and Rapidly Exploring Random Trees (RRT).

Probabilistic Roadmaps

PRMs do not represent the entire free configuration space but rather a roadmap through it. The free space is approximated by random sampling. This sampling creates a probabilistic roadmap. These algorithms consist generally of two phases, namely, the Generic Preprocessing Phase as shown in Algorithm 1 and the Query Phase [39].

Algorithm 1: Generic Preprocessing Phase

1 Take a random sample.
2 Check whether the sample is in C_{free} or C_{obs}
3 If not in C_{free} continue until a collision free sample is obtained
4 The sample is then inserted as a vertex in G the roadmap.
5 Try to connect the newly inserted vertex to some nearby vertices, q of G, considering only those that have a collision free line-of-sight. (However before checking if path is collision-free between the newly inserted vertex and its neighbour we test to make sure they are in different connected components.)

Method for selecting neighbouring samples include:

- Nearest K: The K closest points to the sample (newly inserted vertex)
- Radius: All points within a ball of radius r centered at the newly inserted vertex. An upper limit K may be set to prevent too many connections from being attempted.
- Component K: Try to obtain up to K nearest samples from each connected component of G. A reasonable value of K=1 otherwise too many connections would be tried.

Query Phase In the query phase, pretend q_I and q_G to be chosen samples and run the algorithm two more times. After which we check if q_I and q_G are successfully connected to other vertices in G then a search is performed for a path that connects

q_I to q_G. A path in the graph corresponds to a path in C_{free} which is a solution to the query [39]. This method cannot determine conclusively whether a solution exists or not in case it failed [39]. Probabilistic roadmaps are probabilistic complete and scale well to higher dimensional C-space but they're not optimal, not complete and do not work well for some problems (e.g. narrow passages Figure 3.8).

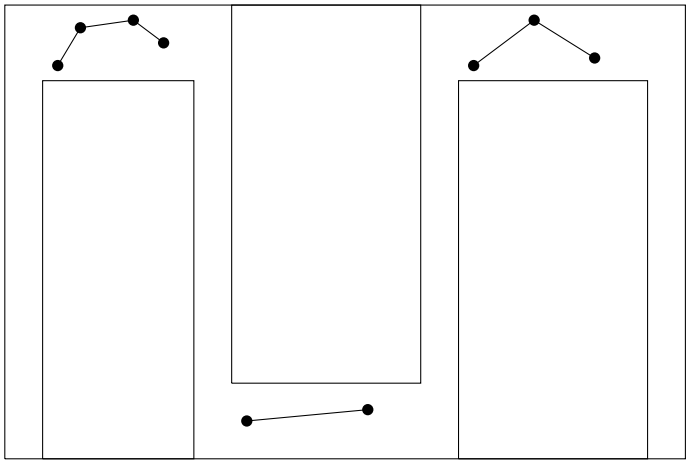


Figure 3.8: Narrow Passage where PRMs do not perform well.

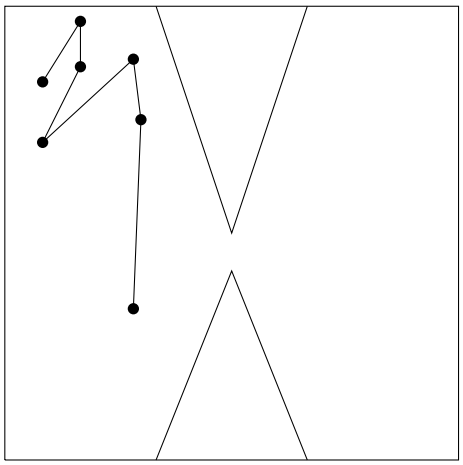


Figure 3.9: Bad coverage.

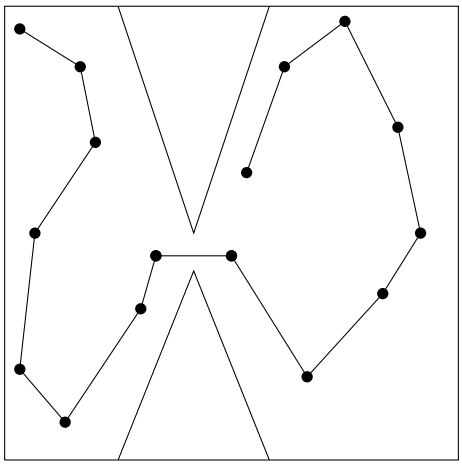


Figure 3.10: Good coverage.

Visibility Roadmap is a variation of Probabilistic Roadmap (PRM). It's a sampling approach that is not random. This approach works very hard to ensure that the roadmap representation is small but covers C_{free} well. In good coverage almost any point in C-space can connect as in Figure 3.10, unlike good coverage, in a bad coverage

it might not be possible to reach certain components as in Figure 3.9.

The running time for visibility graph is often greater than PRM but the extra expense is worthwhile. The idea is to define two different kinds of vertices in G (the roadmap):

1. Guards: A vertex, q, must not be able to see other guards. The visibility region $v(q)$ as shown in Figure 3.11 is empty of other guards.

2. Connectors: A connector vertex, q, must see at least two guards as shown in Figure 3.12. Let q_1 and q_2 be guards, then $q \in v(q_1) \cap v(q_2)$.

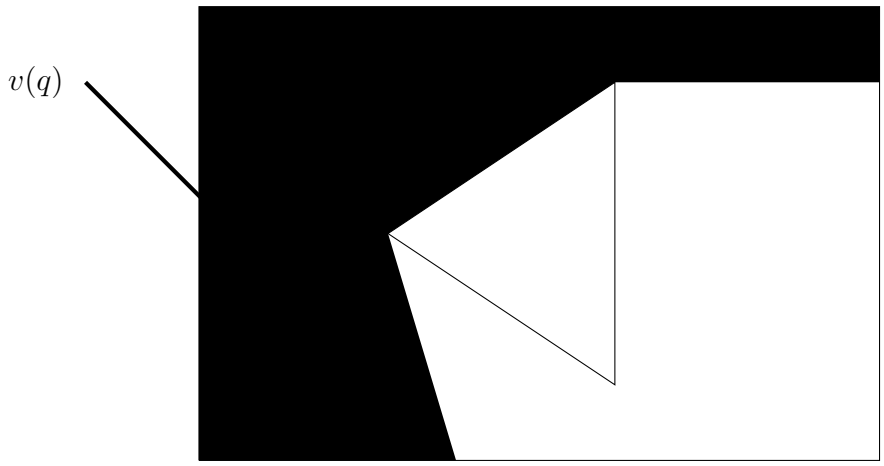


Figure 3.11: Visibility region shown in the gray shaded area.

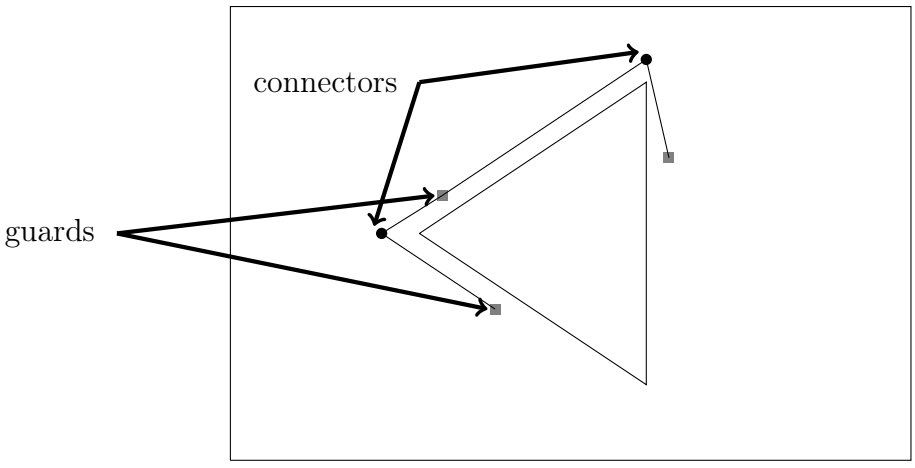


Figure 3.12: Visibility roadmap showing guards and connectors.

When constructing the roadmap in visibility graph, there are three possible cases for a new sample. The first case is when the new sample is not able to connect to any guard which implies the new sample becomes a guard and is inserted into G. The second case is when the new sample can connect to guards from at least two different connected components, therefore the new sample becomes a connector that is inserted into G along with the associated edges. Finally, if neither of the previous cases are satisfied then the sample is discarded.

After the construction of a roadmap a search algorithms such as dijkistra, breadth first search, depth first search, A-star or iterative deepening can be used to find a path between the initial position and goal position.

Randomly Exploring Random Trees

Is an incremental sampling and search approach. A search tree is constructed incrementally in a way that it improves the resolution (Lavalle [39]). A dense sequence of samples is used in building the tree.

Exploration Algorithms

Algorithm 2: Basic Algorithm for the Construction of RRT

for *i=1 to k* **do**

1. add the sample $\alpha(i)$ to G
2. find the nearest point to $\alpha(i)$ in current tree S
3. add an edge between the nearest point and $\alpha(i)$

end

The basic algorithm shown in Algorithm 2 starts with graph G having only one node that is q_i and a sequence of samples. At each iteration a sample $\alpha(i)$ is chosen to be added to G, such that we find the nearest point in G, whether that is a vertex or a

point on an edge, and we do the following in each case:

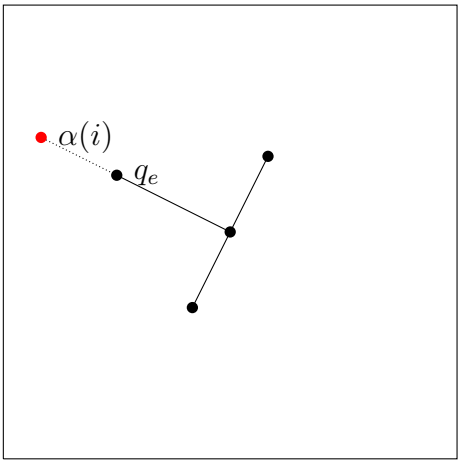


Figure 3.13: Case 1: Nearest point to G is a vertex.

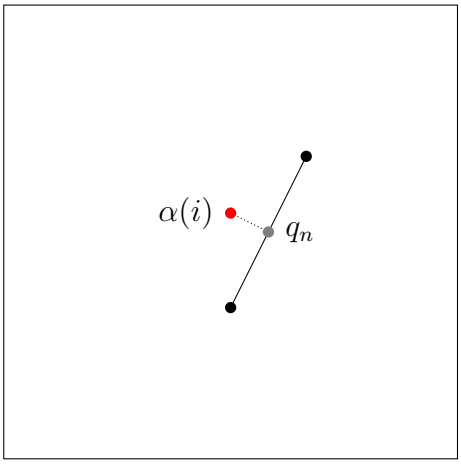


Figure 3.14: Case 2: Nearest point to G lies on an edge.

- Case 1: Nearest point to G is a vertex q_e, we create an edge between the new sample $\alpha(i)$ and q_e. The new edge and $\alpha(i)$ are added to G.

- Case 2: Nearest point to G lies on an edge. The edge is split and a new vertex q_n is created at that point of the edge. The new sample $\alpha(i)$, q_n and the edge between them are added to G.

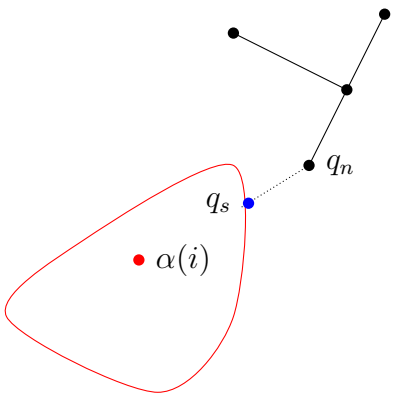


Figure 3.15: Stopping configuration q_s between sample $\alpha(i)$ and q_n.

Algorithm 3: Algorithm for the Construction of RRT taking into consideration C_{obs}

```
1 Initially a vertex is made at q_0
2 for i=1 to k iterations do
3 |  find the nearest point to α(i) in the current tree. Let that point be q_n.
4 |  find the stopping configuration q_s between q_n and α(i).
5 |  if q_s ≠ q_n then
6 |  |  add q_s to tree.
7 |  |  add edge between q_n and q_s
  |  else
8 |  |  add edge between q_n and q_s
  |  end
  end
```

Taking into account C_{obs} the algorithm for constructing the tree becomes as shown in Algorithm 3.

Using Trees For Planning

Single search tree:

Use the algorithm for growing the tree starting with q_I as the initial location of the robot/drone and periodically check whether its possible to connect the tree to the goal q_G.

Balanced, bidirectional search (much better performance):

Grow two trees one from q_I and another from q_G as shown in Algorithm 4. The algorithm works by growing two trees in a way that keeps them balanced in the terms of number of nodes in each tree.

Algorithm 4: Balanced, bidirectional search algorithm

```
1  Let T_a and T_b be two trees starting from q_I and q_G respectively
2  for i=1 to k iterations do
3      find the nearest point to α(i) in T_a. Let that point be q_n.
4      find the stopping configuration q_s between q_n and α(i).
5      if q_s ≠ q_n then
6          add q_s to T_a.
7          add an edge between q_n and q_s to T_a
8          find the nearest point to q_s in T_b. Let that point be q'_n.
9          find the stopping configuration q'_s between q'_n and q_s.
10         if q'_s ≠ q'_n then
11             add q'_s to T_b
12             add an edge between q'_n and q'_s to T_b
           end
13         if q_s = q'_s then
14             return solution
           end
       end
15     if |T_b| > |T_a| then
           SWAP Ta and Tb
       end
       return Failure
   end
```

Collision Avoidance Algorithm

Global vs Local Obstacle Avoidance Algorithms

Global obstacle avoidance algorithms search a map of the environment for the optimal path to a goal. The map is either available to the algorithm or is built using sensor data. The data used for building the map can be a point-cloud of the environment captured using a stereo camera, and then the data can be saved in Octomap [40]. Unlike the global approach, the local or reactive approach are algorithms that do not build a map of the environment but rather calculate the best action based on the latest sensor data. The advantage of local algorithms is the low computational cost, which is critical in drones.

Obstacle Avoidance Using Vector Field Histogram (VFH) Algorithms

3D Vector Field Histogram with A* algorithm (3DVFH*) [41] is a local obstacle avoidance algorithm that is used in Chapter 6. 3DVFH* algorithm is based on one of the most common strategies for local obstacle avoidance called VFH introduced by Borenstein and Koren [42].
The following introduces VFH algorithm briefly, after which we elaborate on the 3DVFH*.

Introduction to VFH

In the original VFH [42] a Cartesian histogram grid is built for the environment. Each cell in the grid has a Certainty Value (CV) which is a measure of confidence that indicates an obstacle exists within the cell area. Using the range sensor readings, and for every reading, if an obstacle exists within the conical field of view of the sensor, then the CV of one cell is incremented. That one cell would be the one that lies on the acoustic axis of the sensor and would be at a distance d from the sensor. The distance d is calculated from the sensor reading.

As the vehicle moves, a window of $w_s \times w_s$ cells move with it. This window makes up a square region called the active region, denoted as C^*. Active cells are cells that momentarily belong to the active region.

A polar histogram H is then generated from the 2D occupancy grid. The polar histogram has a resolution α and consists of $\frac{2\pi}{\alpha}$ sectors. Each sector k holds a value h_k that represents the polar obstacle density in the direction of sector k. The content of each cell is then transformed to an obstacle vector as follow:

- The direction of the obstacle vector β of a cell is defined as the direction from the active cell to the vehicle center point.

$$\beta_{i,j} = \arctan\left(\frac{y_i - y_0}{x_i - x_0}\right)$$

where

x_i, y_i : Coordinates of the active cell(i,j)

x_0, y_0 : Coordinates of the vehicle center point

$\beta_{i,j}$: Direction from the active cell(i,j) to the vehicle center point

- The magnitude of the obstacle vector of a cell is defined as follow:

$$m_{i,j} = (c_{i,j})^2(a - bd_{i,j})$$

where

$c_{i,j}$: Certainty value of the active cell(i,j)

$m_{i,j}$: Magnitude of the obstacle vector of the cell(i,j)

a, b : Constants

$d_{i,j}$: Distance between active cell(i,j) and the vehicle center point

Finally, the obstacle density for each sector is calculated as follow:

$$h_k = \sum_{i,j} m_{i,j}$$

Every cell is related to the sector using the following:

$$k = \lfloor B_{i,j}/\alpha \rfloor$$

The second stage in the VFH algorithm calculates the steering direction θ_d. Sectors with polar obstacle density below a certain threshold are considered candidates. The candidates are further grouped into valleys. All neighbour candidates (consecutive sectors) are grouped to form a valley. A valley that is closer to the direction of the target is chosen.

Once a valley is chosen, a sector within the valley is chosen, thus finding the steering direction. However, before choosing a sector, valleys are classified into two classes: wide and narrow. The classification is based on a threshold s_{max}, such that if a valley consists of more than s_{max} sectors, then it is considered wide, otherwise narrow. In a wide valley the sector nearest to the target is denoted as k_f and the sector furthest from k_f by s_{max} sectors is denoted as k_n. $k_n = s_{max} + k_f$. The steering direction is defined as:

$$\theta_d = \frac{k_n + k_f}{2} \tag{3.17}$$

In a narrow valley the sector farthest from the target k_n is less than s_{max} sectors away from k_n. However, the direction θ_d is still calculated as Equation (3.17).

Enhancements to the Original VFH

Furthermore, VFH+ [43] is an enhancement of the VFH algorithm that takes into consideration the maximum turn radius of the robot and safety margins. VFH* [44] is another enhancement that combines VFH+ and the A* planning algorithm. The VFH* build a look-ahead tree which is searched for the best path using the A* algorithm.

The first to consider a VFH algorithm in a 3D environment is Vanneste et al. [45]. Their 3D Vector Field Histogram (3DVFH)+ algorithm builds a map in the form of Octomap. The Octomap is used to build a 3D occupancy map by considering voxels in the bounding sphere around the vehicle. A 2D polar histogram is built and used to find the best direction to move.

The 3DVFH* Algorithm

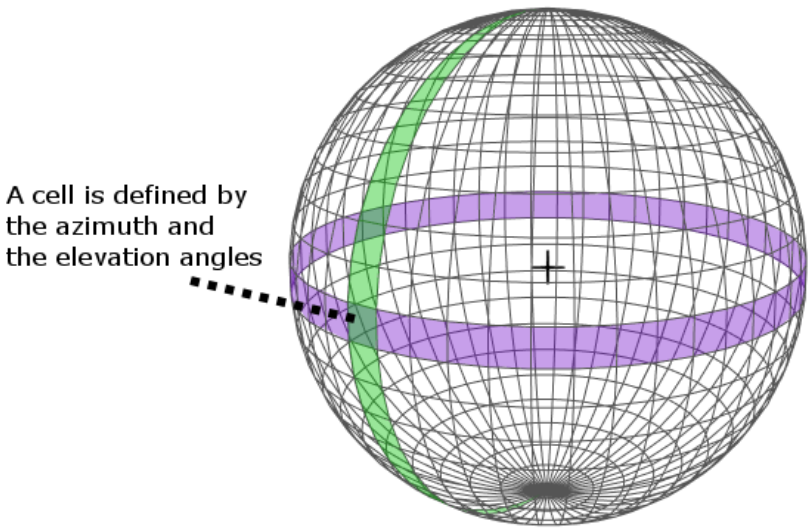


Figure 3.16: A cell is defined by the azimuth and elevation angles of its four corners.

The 3DVFH* [41] used in Chapter 6 is a combination between the VFH* [44] and 3DVFH+ [45]. Unlike the 3DVFH+, the 3DVFH* does not use a global map, nor the concept of Octomap. However, it uses a 3D point-cloud provided by the stereo camera. A 2D polar histogram is built from the 3D point-cloud. An example of a 2D polar

histogram is shown in Figure 3.16, where a cell is defined by azimuth and an elevation angle of its four corners. The polar histogram is then converted to a binary histogram by using a threshold and comparing it to the point count in each cell. A cost function is used to evaluate each free cell, and the cell with the lowest cost is chosen.

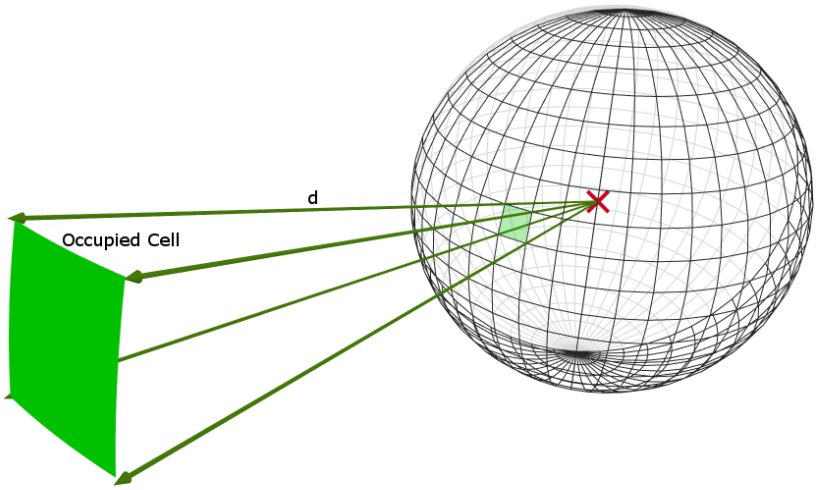


Figure 3.17: Re-projection of of occupied cell at distance d.

The algorithm also has a memory of previously seen objects stored in the form of 2D polar histograms. The strategy is to use old histograms to generate newer ones combined with new histograms generated from the new point-cloud. The memory histograms are generated from old histograms by re-projecting each occupied cell in the old histogram into 3D points as shown in Figure 3.17. The reason for re-projecting the histogram is that the data of the old histogram are bound to the old location at which it was built. The azimuth and elevation angles are used to re-project the cell into four 3D points. A distance layer is also saved for the old histogram. Using the re-projected points a memory histogram is built. The memory histogram is then combined with the new histogram in a certain way discussed further in this section.

Safety margins are taken into consideration such that occupied cells in the histogram are enlarged by setting adjacent cells to blocked and cannot be chosen as directions or

candidates. Two different safety margins are considered depending on the distance of the obstacle.

A histogram saved from the previous time steps is utilized to decide the direction of motion in the current time step, improving navigation. For example, it is sometimes the case that the drone chooses a direction A over direction B, even though B is a better choice, and the reason for choosing the worse direction is due to discarding data of the obstacle seen before and not in the current field of view.

Re-projecting Old Histogram Into 3D Points and Building the Memory Histogram

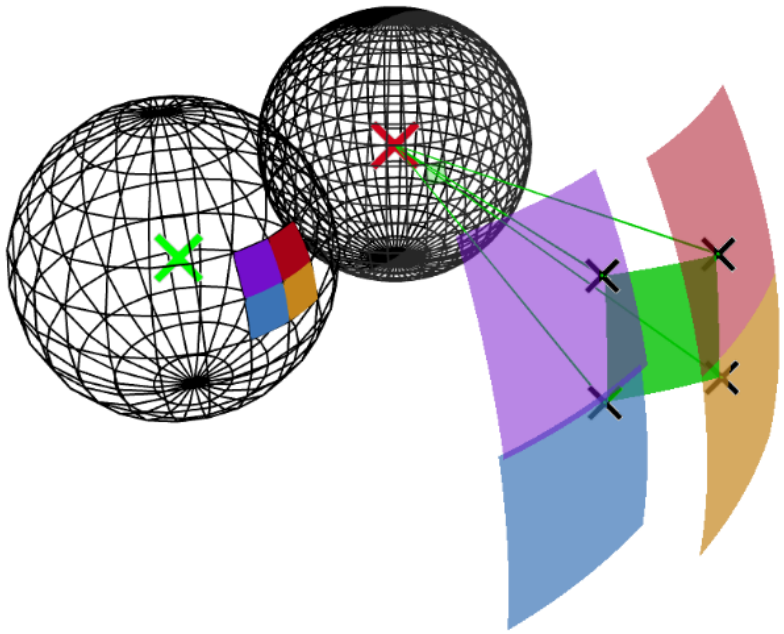


Figure 3.18: Re-projecting old histogram into 3D points and rebuilding it from the current time step. The memory histogram is half the resolution of that of the old histogram and therefore have larger cells.

The old histogram consists of three layers: an age layer, a distance layer, and a binary layer. The old histogram was saved at an older location of the drone, therefore the need of re-projecting and rebuilding the old histogram at the current time step. First, the old histogram is scanned for occupied cells. For each occupied cell, a check is made

to see if the age of the occupied cell does not exceed a defined threshold. Next, the cell is then re-projected into four 3D points corresponding to the corners of the occupied cell.

An occupied cell is defined by an elevation angle ϵ and an azimuth angle ζ. The azimuth and the elevation of each of the four points are calculated by subtracting/adding half the angular resolution. The elevation angle and the azimuth angle for each of the four corners are calculated as follow:

$$\begin{aligned}
\epsilon_1 &= \epsilon + \frac{\alpha}{2},\ \zeta_1 = \zeta + \frac{\alpha}{2} \\
\epsilon_2 &= \epsilon - \frac{\alpha}{2},\ \zeta_2 = \zeta + \frac{\alpha}{2} \\
\epsilon_3 &= \epsilon + \frac{\alpha}{2},\ \zeta_3 = \zeta - \frac{\alpha}{2} \\
\epsilon_4 &= \epsilon - \frac{\alpha}{2},\ \zeta_4 = \zeta - \frac{\alpha}{2}
\end{aligned} \tag{3.18}$$

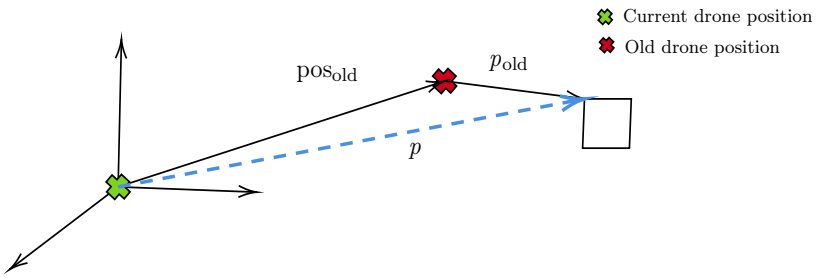


Figure 3.19: Calculating the position of the occupied cell corners from the current time step position.

The four points corresponding to the corners of the cell are then re-projected into 3D space. This is done by first converting the spherical coordinates into Cartesian coordinates. Then, adding the two vectors shown in Figure 3.19 corresponding to the position vector from the current drone position to the old drone position and the position vector from the old drone position to the point corresponding to the cell corner position.

$$
\begin{aligned}
p_{i_x} &= pos_{old_x} + d \cdot \cos(\epsilon_i) \cos(\zeta_i) \\
p_{i_y} &= pos_{old_y} + d \cdot \cos(\epsilon_i) \sin(\zeta_i) \\
p_{i_z} &= pos_{old_z} + d \cdot \sin(\epsilon_i)
\end{aligned}
\tag{3.19}
$$

Where

$pos_{old_x}, pos_{old_y}, pos_{old_z}$: the old coordinates of the drone along the x-axis, y-axis and z-axis

d : The distance between the old drone position and the obstacle

ϵ_i, ζ_i : The elevation and azimuth angle corresponding to the point i with the origin being old drone position

The memory histogram is then built by finding the cell that the re-projected points fall in. We do so by finding the elevation ϵ and the azimuth ζ angles for each of the 3D points (corners of the cell) from the drone's current position. This is just a transformation from Cartesian coordinates to spherical coordinates as follow:

$$
\begin{aligned}
\zeta &= \frac{180}{\pi} \cdot atan2(p_x - pos_x, p_y - pos_y) \\
\epsilon &= \frac{180}{\pi} \cdot atan(\frac{p_z - pos_z}{\sqrt{(p_x - pos_x)^2 + (p_y - pos_y)^2}})
\end{aligned}
$$

The cell or the histogram index to which the point belongs is then found by determining which sector and stack these angles belong to.

The sphere is divided into stacks and sectors. A sector is the section between two constant longitude lines and is shown in green in Figure 3.16. A sector is defined by an azimuth angle ζ.

Given an azimuth angle $\zeta \in [0, 360]$, the sector index S_{index} to which the azimuth

angle belongs to is found as follow:

$$S_{index} = \frac{\zeta + (\alpha - (\zeta \% \alpha))}{\alpha} - 1$$

A Stack is the section between two constant latitude lines and is shown in purple in Figure 3.16. A Stack is defined by the elevation angle ϵ.

Given an elevation angle $\epsilon in [0, 180]$, the stack index T_{index} to which the elevation angle belongs to is found as follow:

$$T_{index} = \frac{\epsilon + (\alpha - (\epsilon \% \alpha))}{\alpha} - 1$$

The intersection between the stack and the sector defines a cell to which the point belongs to as shown in Figure 3.16.

The same procedure is done for the new point-cloud of the current step to produce the new histogram.

Combining Memory and New Histograms

Distance layers of each memory and new histograms are calculated for each bin in each histogram. The distance is calculated by taking the average of the distances of all the points falling into a bin. Since a new histogram is generated from new data, the age layer is set to zero for all the bins. However, for the memory histogram, each bin age is calculated as the average age of all the points in the bin.

Thresholds to calculate binary layer histogram from the 2D polar histograms are chosen for each memory histogram and new histogram. For a new histogram, if at least one 3D point falls into a specific bin in the binary layer, that bin is considered occupied. As for the memory histogram, which is at half the resolution of the other histograms (new and old histograms), the threshold is 6 points. The memory histogram is then up-sampled before being combined with the new histogram.

The current field of view region in the new histogram is given priority in the resulting combined histogram. The current field of view region in the new histogram is copied as-is into the combined histogram, prioritizing it over the memory histogram. A logical OR operation is used for all the other cells in the binary layer. As for the age and distance layers, if a cell is occupied in the new histogram age/distance layer, it is copied to the combined histogram and thus takes priority over the memory histogram. Whereas, if the cell is only occupied in the memory histogram, age and distance values are copied from the memory histogram.

Selecting Direction of Motion

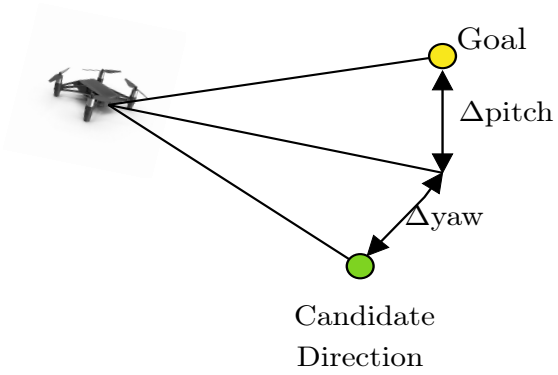


Figure 3.20: Yaw difference and the pitch difference between the candidate direction and the direction of the goal.

A safety margin in the histogram is marked around occupied cells. The size of the safety margin depends on the proximity of the obstacle. If the obstacle is at a short distance, a larger margin is chosen, whereas a smaller margin is chosen if the obstacle is far.

A cost function is used to evaluate every non-occupied and non-blocked cell (due to safety margins). The best candidate direction with the lowest cost is chosen. The cost function consists of two terms. The first is the goal orientation term which compares the evaluated direction to the goal direction. The second term is the smoothing term, which compares the evaluated direction to the last step's chosen direction.

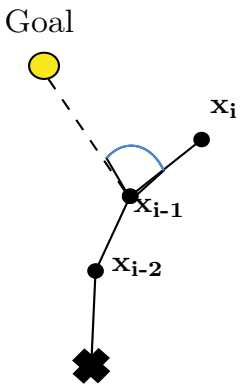
The goal orientation term is divided into three parts, yaw difference, pitch upward difference and pitch downward difference. The candidate direction is projected to a point at the same distance as the goal to calculate the yaws and pitches differences. The distance between the goal and the projected point in the x-y plane is the yaw difference. A similar calculation is made between the direction chosen in the last time step and the candidate direction.

Look Ahead

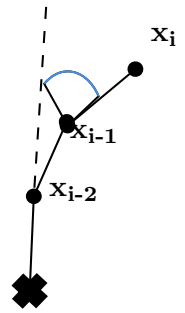
The idea of a look ahead is to consider and evaluate possible subsequent movements in the future. The current position of the drone is set to be the root node of the tree. A node is expanded by cropping the point-cloud around the current node position. Next, the field-of-view of the drone at its position is calculated, the histogram is built using a combination of the cropped point-cloud and the histogram of re-projected points. Candidate directions are evaluated according to the cost function previously discussed when selecting the direction of motion. Next, the best K candidate directions are added. A candidate direction is only added if no close nodes to that direction were added from the same origin. All the newly added candidate directions are then evaluated using the tree cost function and the heuristic function. Finally, the total cost or the sum of the heuristic and the tree cost is assigned to every node.

The candidate node corresponds to the candidate direction at a distance L from the current position. Next, the node with the lowest total cost is expanded. Only up to N nodes are expanded. Once N nodes are expanded, the node with the least total cost corresponds to the end of the tree path. The best direction of movement is chosen based on the back-traced tree from the end node.

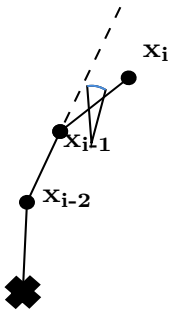
The total cost function comprises a cost function and a heuristic function as in any A* algorithm. However, directions rather than distances are considered in these functions. Specifically, angular differences are used. The cost function has four terms, target cost term, yaw cost term, path smoothing cost term and tree smoothing cost



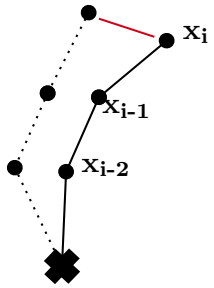
(a) The target cost term is the angle between the edge $\{x_{i-1}, x_i\}$ and the edge $\{x_{i-1}, Goal\}$.



(b) Yaw cost term is the angle between the current drone yaw and the edge $\{x_{i-1}, x_i\}$.



(c) The path smoothing term is equal to the angle between the edge $\{x_{i-2}, x_{i-1}\}$ and the edge $\{x_{i-1}, x_i\}$.



(d) Finding the hidden area sample space (shown in orange) and placing the goal randomly in it.

term.

The target cost term is calculated using the angle shown in Figure 3.21(a) between the edge $\{x_{i-1}, x_i\}$ and the edge $\{x_{i-1}, Goal\}$. The yaw cost term is calculated using the angle in Figure 3.21(b) between the current drone yaw and the edge $\{x_{i-1}, x_i\}$. The third term which is the smoothing term is calculated using the angle between the edges $\{x_{i-2}, x_{i-1}\}$ and $\{x_{i-1}, x_i\}$ as shown in Figure 3.21(c). Finally, the tree smoothing cost term is not an angle difference but rather considers the distance between the node with the same depth as the current node x_i in the last path chosen with the same depth. For the heuristic term, the target cost of the current node is used.

3.3 Machine Learning and Pattern Recognition

Pattern recognition and machine learning can be viewed as two facets of the same field [8]. A definition of pattern recognition as given by Bishop [8]:

> *Pattern Recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories.*

A machine learning approach is adopted in many modern pattern recognition techniques [8], given a training set of N vectors $\{x_1, ..., x_N\}$ along with their corresponding labels $\{y_1, ..., y_N\}$, the training set is used to tune the parameters of an adaptive model.

The original input data is usually preprocessed in the hope that the pattern recognition problem will be easier to solve [8]. The preprocessing stage is also known as feature extraction. The data is converted into a representation that the model can utilize to produce an output in machine learning. This representation is known as the feature vector. A feature vector has two attributes: dimension and a type.

Parametric vs Non-parametric Machine Learning Algorithms

Parametric algorithms make use of parametric distributions. Parametric distributions are governed by a number of adaptive parameters, such as the mean and variance in the case of a Gaussian distribution. Parametric algorithms assume the functional form for the distribution and use the data to learn the distribution parameters. This assumption is a limitation of the parametric approach, as it is inappropriate to make such assumptions for some applications. Parametric algorithms are simple, fast to learn from data and need much fewer data in comparison to the non-parametric approach [46].

An alternative approach to the parametric algorithms is the non-parametric approach which uses non-parametric density estimation methods. Non-parametric algorithms still contain parameters that control the model complexity rather than the form of the distribution [8].

Learning Types

Machine learning is divided into three types: supervised learning, unsupervised learning and reinforcement learning [47].

Supervised Learning

Supervised learning is about tuning the parameters of an adaptive model by using a training data set. It requires labelled data in the training dataset. The result is a machine learning algorithm that can be expressed as a function $f(x)$ that takes a new input data and generate a label for that data. The form of $f(x)$ is determined during the training phase. The ability to correctly categorize the new inputs that differ from those in the training set is called generalization. The model developed is usually tested and evaluated using 25% of the data set while trained using 75% of the data set. If the goal is to assign the input vector to one of the finite numbers of discrete categories, the task is called classification. On the other hand, regression is the case where the desired output consists of one or more continuous variables.

Unsupervised Learning

When there are no labels in the dataset, the goal of the learning problem might be to discover groups of similar examples, determine the distribution of data, or project the data from a high-dimensional space to a lower-dimensional space. The process of discovering groups within the dataset is called clustering, while the projection of data from a high-dimensional space to a lower one is called dimensionality reduction.

Reinforcement Learning

Reinforcement learning is about learning from feedback where the environment is the teacher. The system receives feedback on its actions with no assurance that it is progressing correctly. It is a type of machine learning that interacts with the environment to discover the combination of actions that leads to the best result. The more data, the better the performance gets. In this case, the data is the experience gained through history.

Pattern Recognition and Machine Learning Algorithms

It is our focus to build an adaptive model for pattern recognition by adopting a machine learning approach. The following sections introduce algorithms used in this research: Linear models for classification and neural networks.

Linear Models for Classification

Three approaches to classification:

1. Using a discriminant function to directly assigns each input vector x to a specific class. The discriminant function converts the input into one or more real values and using a threshold, we determine the class. In this approach, probability plays no role.

2. Model the posterior probability $p(C_k|x)$ directly, where C_k is the class, as a parametric model and optimize the model's parameters using the training set.

3. Using the generative approach, the class-conditional densities $p(x|C_k)$ are modelled along with the prior probabilities $p(C_k)$ and then we compute the required posterior using the Bayes' theorem.

Using approach number 2 to classification, we introduce Softmax regression for multi-class classification problems.

Linear Models For Classification

Softmax regression [48] is a multi-class classifier. To better understand Softmax regression, we first introduce linear regression and logistic regression.

Linear Regression. The goal in linear regression is to predict a target value y, given an input vector $x \in \mathbb{R}^n$. For instance, we want to predict the price y of a house given the elements of x, representing the house's features such as its size, number of bedrooms.

The goal is to find a function $h(x)$, a hypothesis function, that predicts the price of any house, given x the features vector of that house. Let $h_\theta(x)$ be a linear function of the parameters $\theta = (\theta_0, ..., \theta_j)$, such that $h_\theta(x) = \theta^T x$.

The task becomes finding θ such that $h_\theta(x) \approx y$. This is done by defining a cost function that measures the error incurred in predicting y_i for a chosen θ. In this example, we will use the least squares:

$$J(\theta) = \frac{1}{2} \sum_i \Big(h_\theta(x_i) - y_i \Big)^2$$

After which, we find the choice of θ that minimizes $J(\theta)$. Thus, the least-square estimate can be used to estimate θ. However, the least-square estimate method is an analytical solution that is not scalable. One algorithm that is scalable and can be used for minimizing the cost function is gradient descent. It uses an iterative method for finding a numerical approximation of the solution to the problem of minimizing the cost function.

Logistic regression. Logistic regression is similar to linear regression but uses a different cost function and transforms the model response using the *logistic sigmoid* function. The logistic regression is a binary class classifier. The posterior probability of class C_k, where $k \in \{0, 1\}$, can be written using the logistic sigmoid as follow:

$$p(C_k|x) = \sigma\big(\theta^T \phi(x)\big) = \frac{1}{1 + e^{-\big(\theta^T \phi(x)\big)}}$$

Where,

x : Feature vector

ϕ : A vector of basis functions of the input variables

$\sigma(\cdot)$: Sigmoid function

θ : Parameters of the model

The reason nonlinear basis functions are used is to make it possible for the model to adapt to nonlinear relationships of x. The cost function used for the logistic regression is as follow:

$$J(\theta) = -\sum_i \Big\{ y_i \log\big(h_\theta(x_i)\big) + \big(1 - y_i\big) \log\big(1 - h_\theta(x)\big) \Big\}$$

Again this becomes a minimization problem where we need to find θ that minimizes the cost function $J(\theta)$. Also, *gradient descent* can be used to find the vector of parameters θ.

Softmax Regression. Softmax regression [48] is a generalization of logistic regression and is a multiclass classifier. The labels in Softmax regression can take on K different values. The main difference is the transformation function applied to the model, in which in the logistic regression, the sigmoid function was used. In the Softmax regression, the Softmax function, also known as the normalized exponential, is used as the transformation function. The posterior probability becomes as follow:

$$p(C_k|x) = \frac{e^{(a_k)}}{\sum_j^K e^{(a_j)}}$$

Where,

$$a_k = \theta_k^T \phi(x)$$

The cost function used for the Softmax regression:

$$J(\theta) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln p(C_k|x)$$

Where,

N : the number of training data points

K : the number of classes

t_{nk} : is an element of an $N \times K$ matrix of labels

Again the problem is narrowed to finding θ that minimizes the cost function $J(\theta)$.

Gradient descent

Gradient descent numerically estimates where a function outputs its lowest values. The steps of the gradient descent algorithm are as follow:

1. Pick random values for the parameters
2. Take the partial derivative of the loss function with respect to each parameter.
3. Substitute the parameter values into the derivatives
4. Calculate the Step Sizes as follow:

$$StepSize = Slope \cdot LearningRate$$

5. Calculate the new parameters as follow:

$$NewParameter = OldParameter - StepSize$$

6. Repeat from Step 2 until either the number of max iterations is reached or the cost function is close to zero.

Gradient descent limitation is that it only finds local minima rather than a global minimum. One way to avoid this is by using stochastic gradient descent, which introduces randomness or noise to allow escaping local minima.

Neural Networks

Neural networks are used in chapters 5 and 6. In Chapter 5 we use an LSTM neural network and in Chapter 6 we use a Deep Forward Neural Network (DFNN). A DFNN consists of more than one hidden layer as shown in Figure 3.21.

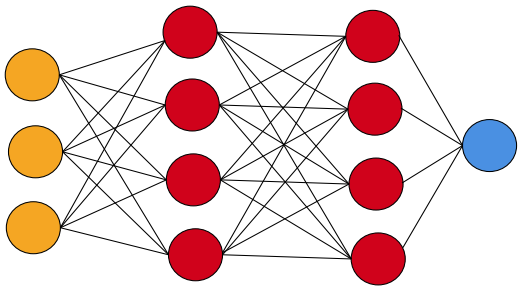


Figure 3.21: Deep feed forward neural network. The network shown consists of two hidden layers with nodes shown in red, one input layer with nodes shown in orange and an output layer with one node shown in blue.

The building block of any neural network is a single artificial neuron which is represented as follow:

$$y = f\left(b + \sum_i w_i x_i\right)$$

Where f is a non-linear function that is referred to as the activation function. Three popular non-linear functions are used: sigmoid, tanh and rectified linear ReLU. One of the simplest kinds of neural networks is the feedforward network. The feedforward network has no cycles, and outputs from units in each layer are passed to those in the

next layer. Such a neural network can be trained using gradient descent with back-propagation. First, a loss function is defined to model the distance between the system and the goal outputs. Then, gradient descent is used as an optimization algorithm to find the parameters (weights and biases) that minimize the loss function. The gradient descent algorithm requires the gradients of the loss function, which is the vector that contains the partial derivative of the loss function with respect to each parameter. In neural networks, one might have millions of parameters. To efficiently partial out the loss over all the layers, we use an algorithm called backpropagation. For a complete introduction on training a neural network and specifically the backpropagation algorithm, we refer to Bishop [8].

Long Short-Term Memory Recurrent Neural Networks

LSTM is a kind of Recurrent Neural Network (RNN). LSTM is capable of learning long-term dependencies. A RNN is different from a traditional neural network because it introduces a transition weight W to transfer information over time.

In this research, we use an LSTM network to classify the audio signals. As its name suggests, LSTM means that short-term patterns are not forgotten in the long-term [47].

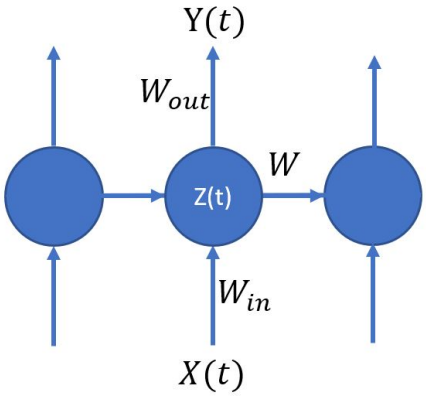


Figure 3.22: Recurrent neural network architecture taking into account the knowledge about the previous runs.

RNN architecture makes use of the previous states of the network. In Figure 3.22 $X(t)$ represents the input. The nodes are connected with arrows. Particular weight is

assigned to the arrow, which is multiplied by the input. Two or more arrows leading to the same node have their values summed in that node. The node shown as a circle represents a hidden layer. The output is equal to $Y(t) = Z(t) \cdot W_{out}$. Recurrent neural networks are used with sequential data.

Figure 3.23: Architecture of an LSTM network for classification in MATLAB® [49].

Figure 3.23 shows the architecture of an LSTM network for the purpose of classification in MATLAB®. Each block in the figure is a representation of a layer in our network. Building a classification LSTM network in MATLAB® consist of creating the following layers:

- A sequence input layer.
- A BiLSTM layer as a sequence layer that learns long-term dependencies between time steps in the data. It is bidirectional because we want the network to learn from the complete time series at each time step.
- A fully connected layer, that is a layer that multiplies the input by a weight matrix and then adds a bias vector.
- A softmax layer that applies a softmax function to its input.
- A classification layer.

3.4 Frameworks (ROS, Gazebo, Matlab, Tensorflow)

To generate data, perform feature extraction, build machine learning models, and conduct a performance analysis on the model, we use two types of frameworks:

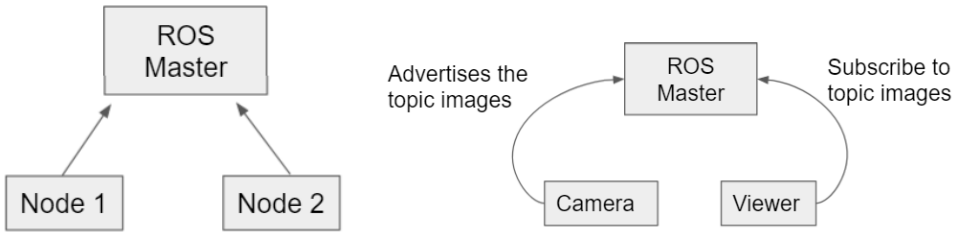
- Frameworks that simulate the quadcopter model and the outdoor environment where the quadcopter is flown. Those are ROS and Gazebo. We give a brief introduction on each.

- Frameworks and platforms that are used to develop and analyze machine learning models. Two frameworks are used Tensorflow and Matlab Deep learning Package.

Simulation Environment Frameworks

In order to generate data relevant to the task or behaviour being studied, we developed multiple ROS packages to control the quadcopter(s) and collect navigational data.

ROS



(a) ROS nodes register at startup with the ROS Master.

(b) ROS node subscribe and publish to topics with the ROS Master

ROS is a meta-operating system. It provides services the user would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. In addition, it provides tools and libraries for building, writing and running code across multiple computers. The primary goal of ROS is to support code reuse in robotics research and development. ROS packages readily available through third parties are used. Those packages are drivers that serve as an interface to the drone(s). That interface allows the acquisition of the drone navigation, inertial measurement unit, and odometry data. It also allows the controlling of the quadcopter by

publishing to topics in the ROS environment.

ROS communication layer consist of a ROS Master node which is a centralized communication server and also acts as a name server for ROS graph resources. The communication layer can be visualized as a graph as shown in Figures figs. 3.24(a) and 3.24(b), where the nodes must register at startup with the Master node. The nodes can then publish and subscribe to topics through the Master node.

We define components and concepts of the ROS graph in the following:

- **Nodes**: Processes that perform computation. For example, one node controls a laser range-finder, one node controls the rotors, one node performs localization, one node performs path planning etc.
- **ROS Parameters**: These are data stored in a central location
- **Topics**: Publish-Subscribe messaging pattern. Nodes can subscribe/publish to topics.
- **Services**: Request-Reply messaging pattern.
- **Bags**: A format for saving ROS messages.

ROS software is organized into packages, which can contain source code, launch files, configuration files, message definitions, data, and documentation

Gazebo

Gazebo is a 3D dynamic simulator. One can create his own world in Gazebo and simulate robots moving around.

Applications and features of Gazebo:

1. Testing robotics algorithms.
2. Designing robots.
3. Multiple physics engines.

4. Wide variety of sensors.

5. Library of robot models and environments.

Gazebo has two processes: a server process that runs the physics loop and generates sensor data and a client process that provides user interaction and visualization of a simulation.

We use a set of ROS packages named gazebo_ros_pkgs, which provides a wrapper around the stand-alone Gazebo. These packages provide necessary interfaces to simulate the drone(s) in Gazebo using ROS messages.

Machine learning Frameworks

Tensorflow, an open-source platform for machine learning, is used to develop and train machine learning models. In addition to TensorFlow, Matlab® is also used for building and training a machine learning model. Specifically using the Deep Learning toolbox in Matlab® to develop an LSTM neural network. In addition to machine learning algorithms, the signal processing toolbox and the Audio toolbox in Matlab® are used to preprocess and feature extraction of the data.

Chapter 4

Recognition of Formation Intentions

UAVs flying in a formation might need to change their formation due to environmental change, modification of the task or some UAVs leaving the formation. There are several reasons for flying in formations, and one might be that a single UAV is limited by the angle of the sensor and cannot cover the targeted area by itself in case of surveillance [50]. A single UAV executing a strike has lots of limitations such as fight range, lethal radius, attack capability, which might reduce the success rate of the whole task. Moreover, for achieving fault tolerance and risk management of a mission, redundancy is required. This chapter focuses on predicting the goal formation that a group of drones is trying to achieve while in the transition phase. By doing so, we might better understand the intentions of a group of a drone and be step ahead of the group in case of malicious intentions. The work done contributes to the state-of-the-art of drone swarm activity recognition. The specific contributions of this chapter include the simulation of drones formations with a real drone model, that is, the AR Drone, and using softmax regression as a machine learning classifier to predict the target formation of the group.

4.1 Related Work

Past research has been conducted on drone activity recognition using machine learning. The components of the feature vectors that have been proposed consist of roll, pitch and yaw angles, velocity and other navigation data. A goal that has been pursued is figuring out the action that a single drone is doing using data obtained from sensor readings. For example, actions being performed by a drone could be hovering, flying forward or flying backward. Recognition using machine learning of single flying drone activity has been conducted by Barták and Vemlelová [19] and by Chen, Hoey, Nugent, Cook and Yu [33] and by Castellini et al. [20], for an aquatic drone. On the other hand, this chapter deals with the identification of drone group behaviour using supervised machine learning. Our interest is to identify the formation being achieved by the collaborative work of a group of drones while it is in progress. This work builds upon an approach introduced in Ref. [51]. Similar research has been conducted about the recognition of robot strategies and formations in robocup by Bruce, Bowling, Browning and Veloso [34], Baez [21] and Trevizan and Veloso [22].

4.2 Framework and Platform

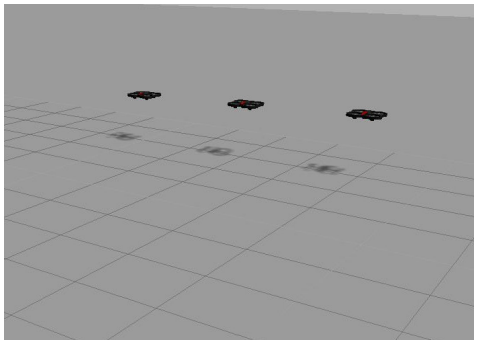


Figure 4.1: Screen capture (Gazebo). Drones shown in hover mode.

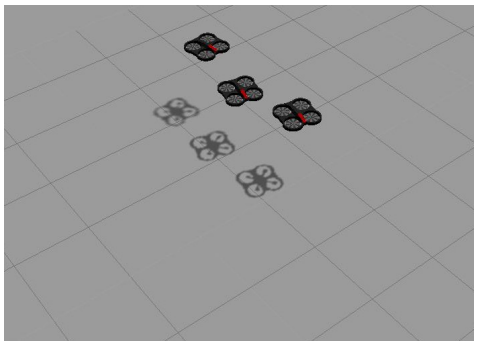


Figure 4.2: Drones final transition to a perpendicular line formation.

Using the Robot Operating System (ROS) version Indigo [52], Gazebo Simulator and ardrone_autonomy driver [53], we simulated drone flights and transitions from one formation to another. Gazebo is a 3D simulator, while ROS serves as an interface to the drones. The ardrone_autonomy [53] is a ROS driver for the Parrot AR-Drone 1.0 & 2.0 quadcopters. The driver allows acquisition of the AR-Drone navigation, Inertial Measurement Unit (IMU), magnetometer and odometry data. The driver also allows controlling quadcopters by publishing to different topics in the ROS environment. A ROS package is created in this work to control a group of drones. Figures 4.1 and 4.2 show screen captures of the Gazebo simulator. The latter figure shows the drones in their final positions after a formation transition has been performed. Using the described environment, we collected data, including the velocity and position of each drone at a rate of 200 data points per second. We preprocess the data to be then submitted to the multiclass classifier, i.e., softmax regression of Tensorflow [54]. We tested the model using different designs of feature vectors. A video of some of the simulations is uploaded to youtube [55].

4.3 Formations

A group of drones is capable of forming several geometric patterns. For the sake of simplicity, we focus on 2D formations. However, the work can be extended to 3D. Using three drones, we form six different patterns, a horizontal line, a vertical line, a

Figure 4.3: Formations simulated.

triangle, a left diagonal and a right diagonal as shown in Figure 4.3. All the trajectories considered in this work are linear. We assume that we can detect the initialization of a formation change by comparing relative acceleration as in Ref. [51]. The initialization of the formation is detected using the rate of change of the linear velocities of the drones since the drones are in hover mode initially, where the velocities are small compared to when the drones are moving. In the Figures 4.4 and 4.5, three drones are shown in their initial positions. The dashed lines represent the path taken by each of the drones. The crosses at the end of the dashed lines show the drones' goal positions. For the vertical line formation in Figure 4.4, all the drones are involved in the formation transition. Figure 4.5 shows the paths taken by the drones to form a triangle formation.

We simulate the transition from each formation in Figure 4.3 to every other formation with a total of 30 transitions.

4.4 Data Preprocessing

Messages are generated for each of the drones at a rate of 200 units per second. The messages contain the positions of the drones and their velocities in the X and Y directions, along with time stamps. Data with time stamps from the three drones were collected during their formation transitions.

The drones' target positions are set in the code. We assume that the data collected can be obtained using radars and other sensors. The numbering of the drones is from left to right as shown in Figures 4.4 and 4.5. For each formation transition, we run the simulations and collect navigation data. We define the capture time as the window we use to predict the intended formation. We set the capture time at two different values. One capture value is 33% of the total duration of the formation transition and another

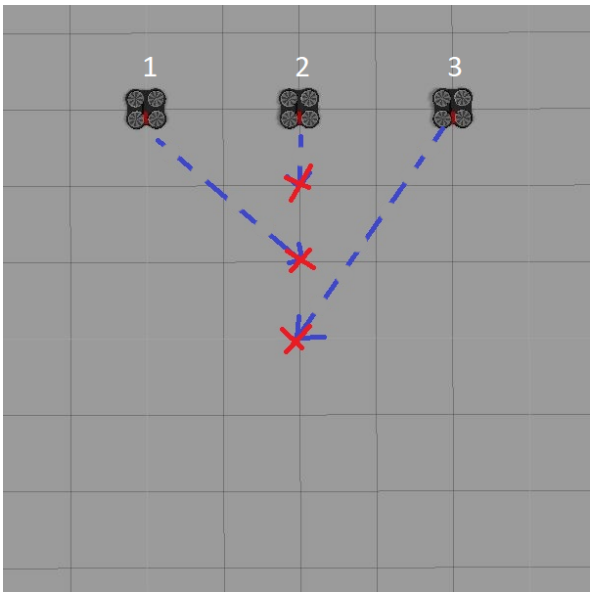


Figure 4.4: Transition from a horizontal line formation to a vertical line formation.

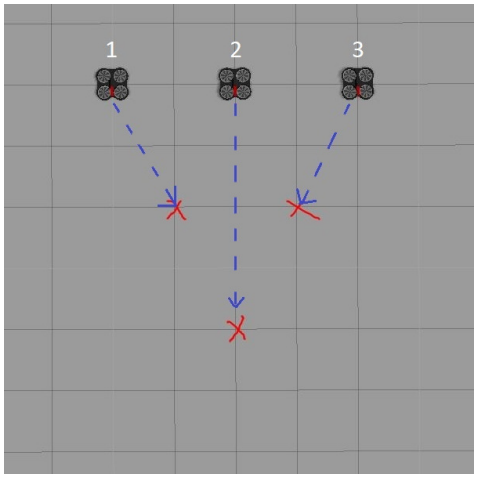


Figure 4.5: Transition from a horizontal line formation to a triangle .

of 66%. We test multiple designs of the feature vector and compare the results of each.

We used 80% of the data collected from the simulation of all formation transitions for training the model. The remaining 20% is used to test the model.

Following the collection of data, we preprocess it and compute the following:

1. The average distance between each drone pair during the capture time.

2. The average midpoint between each of drone pair during the capture time.

3. The average linear velocity of each of the drones during the capture time.

The average midpoint is the average X and Y values of the midpoints of the drones during the capture duration.

We then normalize the average distances, average midpoints and average linear velocities of the drones by scaling them to the range [0-1]. The data is then labelled with the corresponding formation names using one-hot encoding. One hot encoding is a way to represent categorical variables as binary vectors. We use one-hot encoding to avoid confusing the model into thinking that the output categories have some natural ordering, so using a one-hot encoding is a safe bet. As an example of using one-hot encoding, let us consider the case of labelling the data obtained to one of the six formations. We require six bits vectors with only six valid labellings to represent six classes using one hot encoding. Those labellings are [1,0,0,0,0,0], [0,1,0,0,0,0], [0,0,1,0,0,0], [0,0,0,1,0,0], [0,0,0,0,1,0] and [0,0,0,0,0,1].

4.5 Supervised Learning: Softmax Regression

Softmax regression [48] is a multi-class classifier which has been introduced in Chapter 3 and it is a generalization of logistic regression. The labels in Softmax regression can take on K different values. The hypothesis estimates the probability $P[y = k|x]$ for $k = 1, \ldots, K$ where y is a random variable representing a goal formation and x is a test input. In our case, K is equal to 6. We created and trained the model using the TensorFlow library in Python.

The model is trained using the data generated during the simulations of the six formations. We also test the model using part of that data.

4.6 Feature Selection

For feature selection, we identify three methodologies to evaluate and choose whether to add a feature to our model. The approaches used here are based on the fact that the feature is numeric and the output/target is categorical. The three approaches are as follow:

- Using Analysis Of Variance (ANOVA) for feature selection, which is an appropriate method when the features are numeric and the label is categorical [56].
- Mutual information is an application of information gain to feature selection.
- Exhaustive feature selection, a wrapper feature selection method, which uses the performance of a machine learning algorithm to evaluate all possible combinations of features in the data set.

The three approaches identified are all used to evaluate feature importance and to select the features. First, we start with ANOVA to evaluate the correlation between features and targets. Next, we look into using mutual information for feature importance. After which, we use the exhaustive feature selection to choose the best combination.

ANOVA F-test

ANOVA is a statistical hypothesis test that determines whether the means from two or more samples of data come from the same distribution or not. For k groups and each group is of size $\frac{n}{k}$ the ANOVA F-test is used to calculate the F-value of each as follow:

$$F\text{-}value = \frac{\frac{\text{SSB}}{dF_B}}{\frac{\text{SSW}}{dF_W}}$$

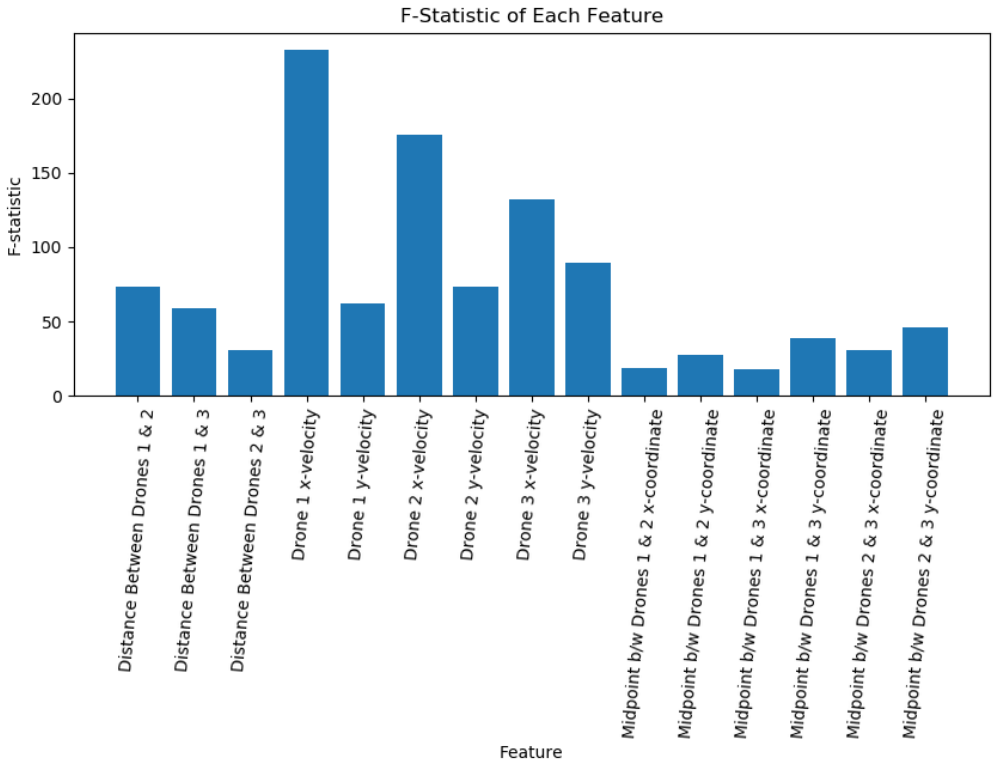


Figure 4.6: The score of the ANOVA for each feature.

Where,

dF_W : The degrees of freedom within groups and is equal to $k - 1$.

dF_B : The degrees of freedom between groups and is equal to $n - k$.

SSW : Sum of squares within groups.

SSB : Sum of squares between groups.

And SSB is:

$$SSB = \frac{n}{k} \sum_{G=1}^{k} (\overline{X_G} - \overline{X})^2$$

Where,

$\overline{X_G}$: Mean of Group G.

$\overline{X}$: The mean of all samples from all groups.

And SSW is:

$$SSW = \sum_{G} \sum_{i} (x_{G_i} - \overline{X_G})^2$$

Where,

x_{G_i} : Sample i in group G.

The ANOVA assumes the data sample of each group to be normally distributed. We performed a test for normality using scipy.stats.normaltest that is based on D'Agostino [57] and D'Agostino et al. [58]. Data was found to be not normally distributed. After which, we used Box-Cox transformation using scipy.stats.boxcox function to transform the features into a normal shape to be used in ANOVA. Each of the features scores is shown in Figure 4.6. However, since not all groups were normally distributed, even after Box-Cox transformation, the ANOVA might not be considered the best option to evaluate the features. We conclude this feature selection methodology by saying that if there were an equal variance between any of the features and the target, we would have removed the feature because it would have no impact on the response. However, because the ANOVA assumes that the group samples are normally distributed, and this is not the case, this evaluation technique should not impact our decision to remove any feature. We will further evaluate the features using mutual information.

Mutual Information

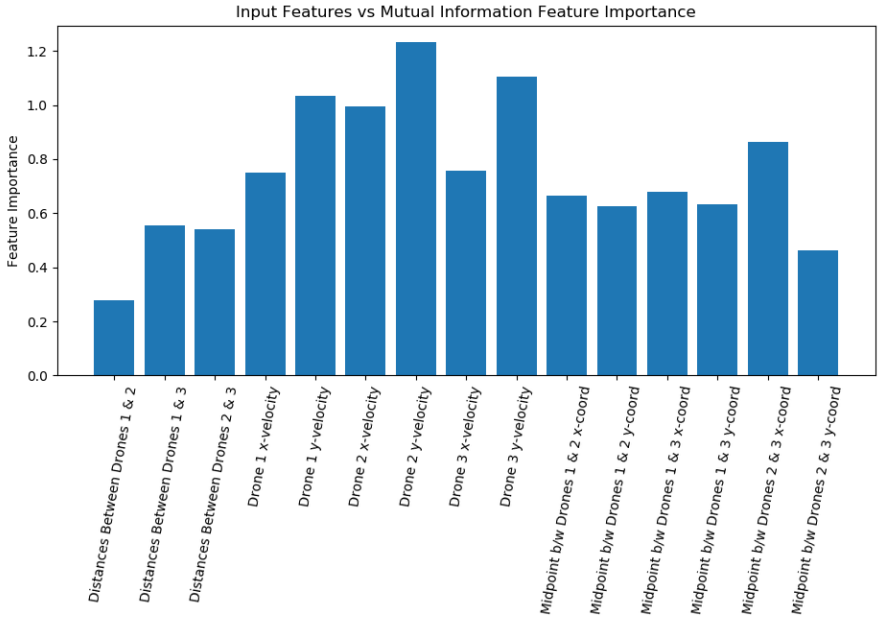


Figure 4.7: The score of the ANOVA for each feature.

In information theory, the surprise or entropy of a random variable is the average level of information or surprise that exists in the variable's possible outcomes. This means that low probability events have higher information or surprise, while high probability events have lower information or are unsurprising. Entropy, also known as Shannon entropy for a discrete random variable X, is defined as in [59]:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot log(P(x_i))$$

Mutual information or information gain is defined by Witten et al. [60] as follow:

> *A quantity called mutual information measures the amount of information one can obtain from one random variable given another.*

It is a measure of dependence or mutual dependence between two variables. Mutual information is stated formally as:

$$I(X;Y) = H(X) – H(X|Y)$$

The estimated mutual information between each feature and the target is then calculated. The mutual information is then used as feature importance. Features with higher mutual information values are considered more important. The feature importance of each of the features is shown in Figure 4.7. The features' importance shows that the drone velocity is the most important feature. However, none of the features are significantly less important than the drone velocity. Therefore we consider using all of the features according to this evaluation technique.

Furthermore, to better show that the selection of all features is reasonable, we evaluate the features using the exhaustive selection method in the next section.

Exhaustive Selection Method

All possible combinations of different features are used in the exhaustive feature selection method. The features are: Average distances between each drone pair, average velocities, and average position of the midpoint between each drone pair.

Using these features, we create all possible combinations of the three features as follow:

1. D1: Using only average distances between each of the drone pairs.
2. D2: Using only average velocities of the drones.
3. D3: Using only average midpoints between each of the drone pairs.
4. D4: Using average distances between the drone pairs and average velocities of the drones.
5. D5: Using average distances and average midpoints between each of the drone pairs.

6. D6: Using average velocities and average midpoints between each of the drone pairs.

7. D7: Using average velocities, average distances and average midpoints between each of the drone pairs.

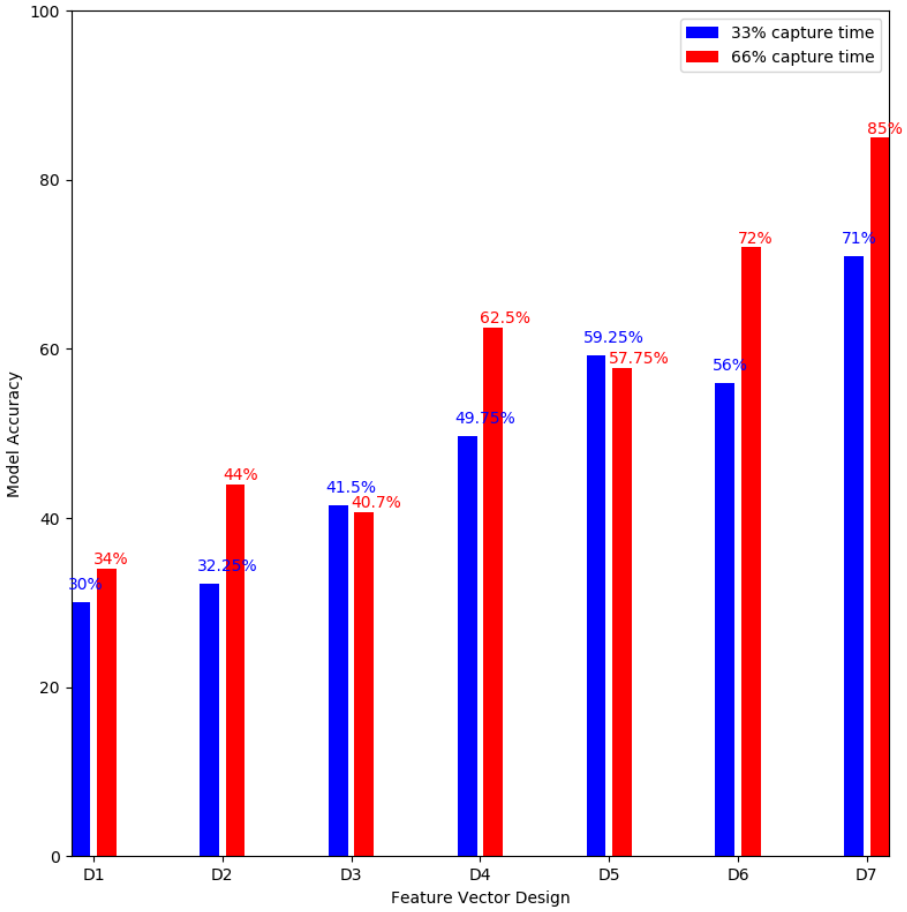


Figure 4.8: Feature vector designs and accuracy of predictions.

We tested the model for each feature vector design and evaluated the accuracy of the prediction of drone formation intentions. The results are then used to evaluate the features and choose the best combination. Finally, we present the results of the accuracy of each of the models with all possible combinations in Figure 4.8. The results

show that using all combinations achieves the best accuracy, and therefore, we proceed accordingly.

4.7 Performance

A correct prediction is when the model predicts the actual formation the drones intend to make while in the transition phase using the navigation data collected during the capture time. The performance evaluation metrics used in this chapter are precision, recall, accuracy, f1-score. We define each of the metric as follow:

Accuracy describes the fraction of the total samples that were correctly classified by the model. The following formula is used to calculate accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Precision tells us what proportion of positive identifications was actually correct. The formula for calculating precision is as follow:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall tells us what proportion of actual positives was identified correctly. Recall is calculated as follow:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score combines precision and recall into a single measure. It is the harmonic mean of precision and recall and is calculated as follow:

$$\text{f1-score} = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Bootstrapping confidence intervals

We perform bootstrapping to calculate the confidence interval for the machine learning model skill, such as accuracy, recall or precision. A confidence interval provides a range of model skills with a likelihood that the model skill falls between the ranges when making new predictions on new data.

We use empirical bootstrapping as a method to estimate the confidence intervals. The idea is to perform computation on the data to estimate the variation of statistics that are themselves computed from the same data. The steps of bootstrapping to calculate the confidence intervals are defined as follow:

1. Let $X = x_1, x_2, ..., x_n$ be a data sample drawn from a distribution F.
2. Let u be a statistic of interest computed from the sample.
3. We perform a sampling with replacement on X to generate X^*. For the new sample X^* to be of same size as X, we sample as many times as the size of X when generating X^*.
4. Compute u^* the statistic of interest from the bootstrap sample X^*.
5. Repeat from Step 3 to generate bootstrap samples along with the statistic u^* as much as wanted.
6. Compute $\delta^* = u - u^*$ for each bootstrap sample.
7. Order δ^* in a non-decreasing order.
8. For a 95% confidence level, we choose the $\delta^*_{0.025}$ at the 97.5 percentile and $\delta^*_{0.975}$ at the 2.5 percentile.
9. The bootstrap 95% confidence interval for u is:

$$[u - \delta^*_{0.025}, u - delta^*_{0.975}]$$

Using bootstrapping, we compute the confidence interval for each of the performance metrics. We do this for performance metrics of experiments done using 33% capture time and 66% capture time.

We calculated the performance metric per class and then used the macro average to combine the scores of each class.

Table 4.1: Class performance metrics (capture time 33%).

Class ID	Formation	Precision %	Precision 95% Confidence Interval	Recall %	Recall 95% Confidence Interval	F1-score %	F1-score 95% Confidence Interval
0	Horizontal Line	71.4	[58.4, 88.3]	76.9	[70, 99.6]	74	[70, 88.4]
1	Left Diagonal	78.4	[69.1, 91.6]	74.6	[57.5, 83.5]	76.5	[66.6, 84]
2	Upward Triangle	69.7	[55.4, 80.1]	69.7	[56.8, 81.5]	69.7	[59.7, 78.1]
3	Vertical Line	86.65	[76.6, 99]	78.6	[64.3, 93.9]	86.9	[84.3, 100]
4	Downward Triangle	59.3	[41.7, 67.1]	69.5	[62.7, 91.3]	64	[56.1, 75.3]
5	Right Diagonal	91.7	[86.7, 99.3]	93	[87.2, 96.5]	92.4	[88.1, 96.2]

Table 4.2: Model class-combined performance metric (33%).

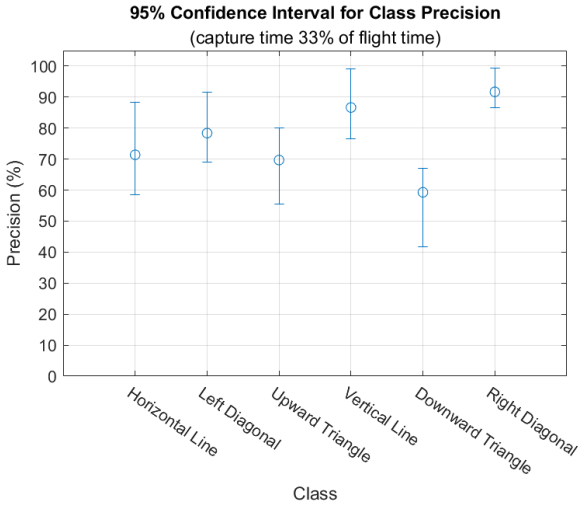
Performance Metric	Value	95% Confidence Interval
Accuracy	76.8%	[73.3%, 82%]
Precision	78%	[75.4%, 83.6%]
Recall	77%	[73.9%, 82.2%]
F1-score	77.2%	[74.5%, 82.8%]

Table 4.3: Class performance metrics (capture time 66%).

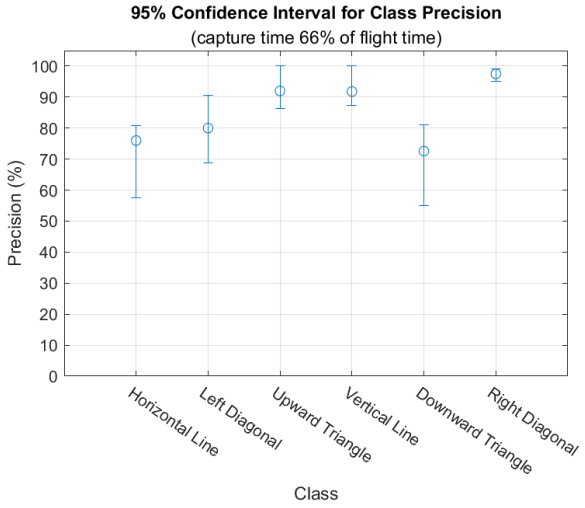
Class ID	Formation	Precision %	Precision 95% Confidence Interval	Recall %	Recall 95% Confidence Interval	F1-score %	F1-score 95% Confidence Interval
0	Horizontal Line	76	[57.5, 80.8]	92.4	[88.3, 100]	83.4	[75.7, 89.2]
1	Left Diagonal	80	[68.9, 90.6]	70	[47, 74.2]	74.6	[62.1, 79.1]
2	Upward Triangle	92	[86.4, 100]	90.7	[82.8, 100]	91.3	[86.6, 99.3]
3	Vertical Line	91.8	[87.4, 100]	75.5	[56, 83.6]	82.9	[74.9, 89.9]
4	Downward Triangle	72.6	[55, 81]	82.4	[71.6, 98.1]	77.2	[68.7, 85.4]
5	Right Diagonal	97.5	[95, 99.2]	97.5	[95, 100]	97.5	[95, 98.8]

Table 4.4: Model class-combined performance metric (66%).

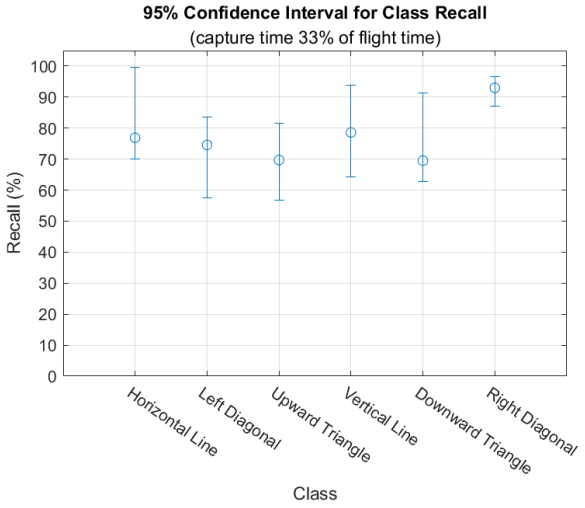
Performance Metric	Value	95% Confidence Interval
Accuracy	84.5%	[79.79%, 87.5%]
Precision	85%	[80.2%, 87.7%]
Recall	84.7%	[80%, 87.6%]
F1-score	84.5%	[79.6%, 87.3%]



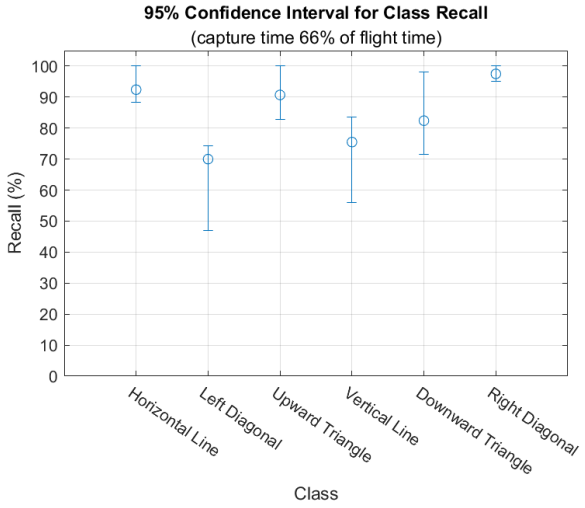
(a) Class precision when capture time is 33% of total flight time.



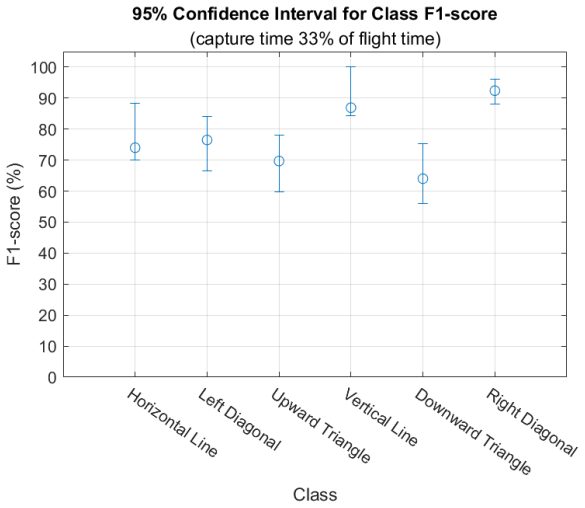
(b) Class precision when capture time is 66% of total flight time.



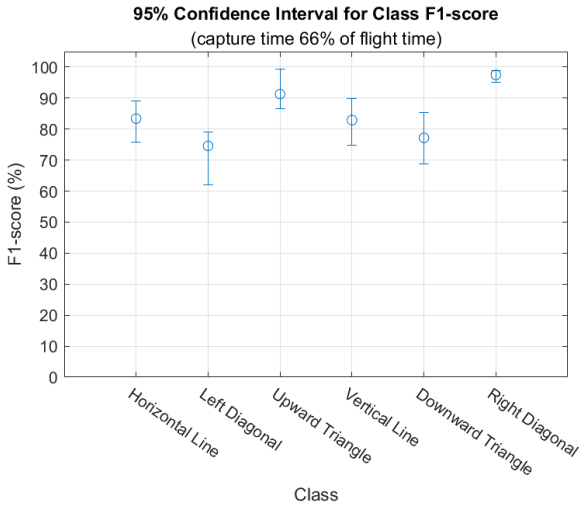
(c) Class recall when capture time is 33% of total flight time.



(d) Class recall when capture time is 66% of total flight time.

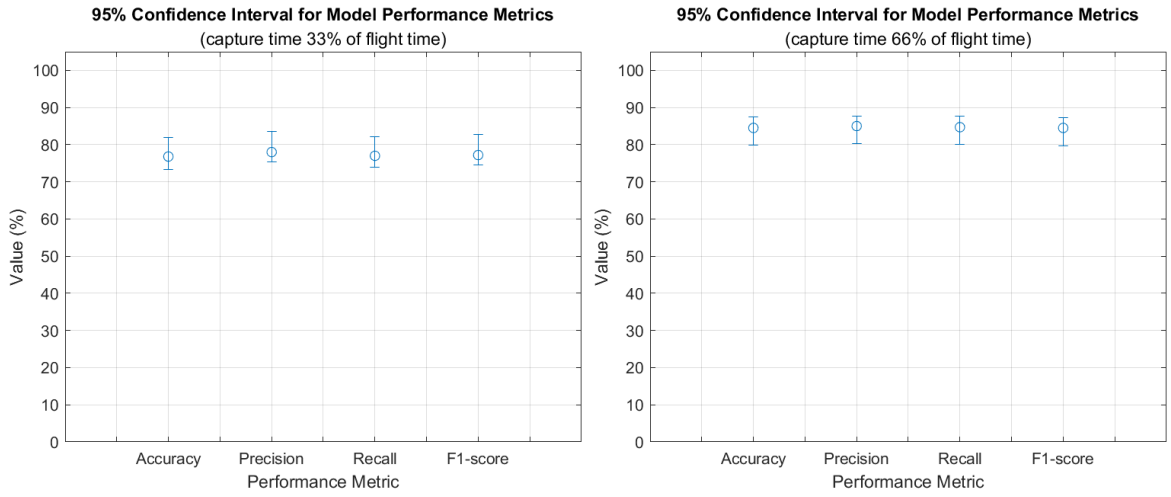


(e) Class F1-score when capture time is 33% of total flight time.



(f) Class F1-score when capture time is 66% of total flight time.

Figure 4.9: Class performance metrics when capture time is 33% and 66%.



(a) Model performance metrics when capture time is 33% of total flight time.

(b) Model performance metrics when capture time is 66% of total flight time.

Figure 4.10: Model performance metrics for each capture time 33% and 66%.

The evaluation results are shown in tables 4.1 to 4.4 and figs. 4.9(a) to 4.9(f), 4.10(a) and 4.10(b). Tables tables 4.1 and 4.3 show the performance metrics corresponding to each of the classes, while tables tables 4.2 and 4.4 show the combined performance metrics of the classes using macro averages. The performance of the model improves about 7% when we increase the capture time from 33% to 66%.

4.8 The dummy classifiers vs the softmax drone formation classifier

We build three dummy classifiers with various strategies as follow:

1. **most frequent dummy classifier**: always predicts the most frequent label in training set.

2. **uniform dummy classifier**: predicts uniformly at random.

3. **stratified dummy classifier**: predicts while respecting the training sets class distribution.

To prove that the classifier described in this Chapter performs better than any of the dummy classifiers, we perform random sampling with a t-test. The performance metrics considered in the hypothesis are accuracy, recall and precision. The null hypothesis and the alternative hypothesis are formally stated as follow:

Hypothesis 0 (Null Hypothesis)**.** *No statistically significant difference between the performance of each of the dummy classifiers and the model trained on the collected and processed navigational data in the task of predicting the formation of a group of drones.*

Hypothesis 1 (Alternative Hypothesis)**.** *There is a statistically significant difference between the performance of each of the dummy classifiers and the model trained on the collected and processed navigational data in the task of predicting the formation of a group of drones.*

When random resampling is used with a standard t-test, it results in a high Type I error [61]. The increased Type I error results from underestimating the variance which is due to the sample being not independent (different training and test sets overlap) [61]. To correct this underestimation of the variance, the corrected resampled t-test takes into consideration the dependency [61].

To compare the two models, we perform random resampling in conjunction with a corrected resampled t-test as suggested by Bouckaert and Frank [61], in which replicability is also considered along with appropriate Type I error and low Type II error. The corrected resampled t-test is:

$$t = \frac{\frac{1}{n}\sum_{j=1}^{n} x_j}{\sqrt{(\frac{1}{n} + \frac{n_2}{n_1})\hat{\sigma}^2}} \tag{4.1}$$

Where,

$x_j = a_j - b_j$

a_j : performance statistic of the machine learning model for the j_{th} iteration

b_j : performance statistic of the dummy classifier for the j_{th} iteration

$\hat{\sigma}^2$: estimate of the variance of n differences

n_1 : number of instances used for training

n_2 : number of instances used for testing

The t-value is calculated for each of the three performance metrics for the three dummy classifiers against the Softmax classifier. It is calculated by resampling 100 times to calculate the performance statistic of the dummy classifiers and the Softmax classifier and then taking the differences as per Equation (4.1).

Table 4.5: t value for each performance metric

	Accuracy	Precision	Recall
Softmax Classifier vs Most-Frequent Dummy Classifier	41.97	72.74	56.70
Softmax Classifier vs Uniform Dummy Classifier	39.15	39.84	39.81
Softmax Classifier vs Stratified Dummy Classifier	45.29	46.51	46.43

The values in Table 4.5 are calculated according to Equation (4.1), and they suggest rejecting the null hypothesis H_0 and favouring the alternative hypothesis H_1 for each metric with a 95% confidence level. Therefore, the Softmax classifier outperforms the dummy classifiers, in which these baseline dummy classifiers learn no patterns at all but rather produce predictions based on specific strategies.

4.9 Summary

We have simulated a group of AR drone quadcopters making six different formations. We collected navigation data during the formation transitions. We tested two different capture times, 33% and 66% of the total time of the formation transition. After the collection, we preprocessed the data and used Softmax regression as a machine learning classifier. We evaluated the features and selected the best combination based on methods of feature evaluation: exhaustive feature selection, the ANOVA F-test and mutual information for feature importance. Two models, one where we used 33% capture time and another where 66% is used, are trained and tested using the data collected during the corresponding capture time. Finally, we presented and evaluated the results of each of the models. After which, we developed three dummy classifiers with three different classification strategies and used those as a baseline to compare our model with. We proved a statistically significant difference exists between the performance of each of those dummy classifiers and the Softmax model trained on the collected and processed navigational data in predicting the formation of a group of drones. Formation transition intentions can be well identified using supervised machine learning algorithms such as Softmax regression using the features we discussed in Section 4.4. By identifying the next move of a fleet of drones, security measures can be taken, or further analysis and predictions can be made. One such analysis is to identify what possible swarm algorithm is a fleet following. Future work includes using neural networks as an alternative to Softmax regression.

Chapter 5

Identification of Drone Payload Using Mel-Frequency Cepstral Coefficients and Neural Networks

In this chapter, we look at a specific drone behaviour to identify whether the drone is carrying some load for delivery or not carrying any load at all. First, we specifically change the physical makeup of the drone by adding a payload to it and altering its weight. We then record the behaviour of the drone and train a machine learning model to identify the physical makeup of the drone (with or without payload).

For a drone to maintain a certain altitude, when having a payload, the propellers need to spin faster, i.e., at a higher frequency than the case where it has no payload. The altitude and coordinates of a drone can be estimated using the approach in [62]. The altitude is an essential factor when determining which models to use.

Work has been done to develop a database of Radio Frequency (RF) signals that are used in the identification of a drone [12]. This valuable information helps us further analyze a drone's activity and better understand the behaviour we seek to classify. Therefore, after identifying a drone, we can use a model trained on data specific for this drone type/model from a database of audio signals described hereafter. The database

to be developed and maintained should contain a wide range of drone audio signals with respective labelling of whether the drone has some load or not, the weight of the payload, the drone model/type, and the drone's altitude. Thus, different models for each specific drone can be developed to classify a drone according to whether it has a payload or not and possibly the weight of the payload at different altitudes.

The chapter is divided into six sections. Related work is reviewed in Section 5.1. In Section 5.2 we present the platform we use in this work. Section 5.3 discusses the data and audio processing techniques we use to extract features from the recorded audio signals. The supervised machine learning algorithm we use to classify the payload that the drone might be carrying is then discussed in Section 5.4. The performance evaluation criteria of the model and the results are presented in Section 5.5. Finally we conclude this chapter in Section 5.7.

5.1 Related Work

Audio features are used in the literature to train machine learning models to classify drones and recognize them. The classification of whether a drone is loaded or not is researched using Micro-Doppler signature and multistatic radars.

An approach to classify loaded or unloaded micro-drones has been developed by Fioranelli et al. [18]. The work is similar to what has been done in [17], in the sense of using micro-Doppler signatures and a multistatic radar. The features used are based on the micro-Doppler signature. The features are the Doppler centroid and Doppler bandwidth. Naïve Bayes and the diagonal-linear variant of the discriminant analysis classifier are used for classification. Two approaches were tested, one in which a centralized classifier was fed sample triples, one component from each radar. In another approach, three separate classifiers are used along with a voting procedure for the final decision.

Audio-based machine learning models in drone applications have been used to clas-

sify and recognize drones. A machine learning sound-based amateur drone detection system has been developed by Anwar et al. [16] for public safety. Their classifier uses MFCCs and Linear Predictive Cepstral Coefficients (LPCCs) as features for a SVM model. The model can classify whether sounds belong to a drone, a bird, a plane or a thunderstorm. A MFCC technique for drone detection using a hidden Markov model as a classifier has been studied by Shi et al. [15]. The results of an experiment show that the model has high recognition rates even in noisy environments.

In this chapter, we classify drone payloads. Deployment of multistatic radars in cities may not be feasible due to the need for line-of-sight from several locations. Therefore, we take the approach of using a sound-based classifier to determine if a drone is loaded or unloaded and estimate the payload weight in case the drone is loaded.

5.2 Framework and Platform

In this work, we use the Parrot Mambo mini quadcopter. The drone was flown indoor at different ranges from a microphone. The frequency response of the MacBook Pro (late 2013) is about 100 Hz-12000 Hz. The equipment used covers the range of frequency band of the propellers and motors, which is 600-6000 Hz [63]. The drone weighs around 71 g after removing the FPV camera. The maximum load we were able to add is 28 g, around 39.4% of the drone weight. We used different payloads: zero, 10, 24, 20 and 28 g. We collected audio signals when the drone was loaded with these weights. Signals were collected in two different modes: when the drone is hovering at one meter in altitude and when the drone flies around at an altitude of one meter for eight seconds. Figure 5.1 shows the Parrot Mambo mini drone flying with payload.

Figure 5.1: Drone flying with payload on top.

We recorded the audio signal using a Macbook pro (late 2013 model) built-in microphone at a rate of 44100 samples per second. The signals are then processed in MATLAB®. An LSTM model is built using the deep learning package of MATLAB®. Finally, we add white Gaussian noise to the signal for several different SNRs. We evaluated the model.

5.3 Data Preprocessing

The audio signals recorded are imported into MATLAB® for processing. Only specific portions of the signals are taken into account, in which we remove the parts of the signals captured during takeoff and landing. We only keep the signals that that were captured during hovering or moving around. We only consider the cases where the drone is in hover mode or moving around, such as pitching and rolling.

We used 75% of the collected signals for training the model. The rest were used for testing the model. The signals used for testing the model have been combined with white additive Gaussian noise to produce signals with different SNRs.

MFCC extraction is a technique used to capture prominent features from sounds. It is widely used in speech and speaker recognition. The MFCCs represent the spectral envelope of a signal. MFCCs are calculated as in [64] and [65]:

1. Partition the signal into frames where each frame is 30ms.

2. For each frame window the data with a hamming window.

3. Take the Discrete Fourier Transform (DFT) of the frame.

4. Find the magnitude of the Fast Fourier Transform (FFT).

5. Each frame is then passed through mel filter bank.

6. Take the log base 10 of all the filter banks outputs.

7. Take the Discrete Cosine Transform (DCT) of the log filterbank energies.

Mel scale is a perceptual scale that relates the perceived frequency to the actual measured frequency [66]. It scales the frequencies in order to match more closely to what humans hear [66]. The formula for converting the frequency to Mel scale as in [67] is:

$$M(f) = 2595\ log_{10}\left(1 + \frac{f}{700}\right)$$

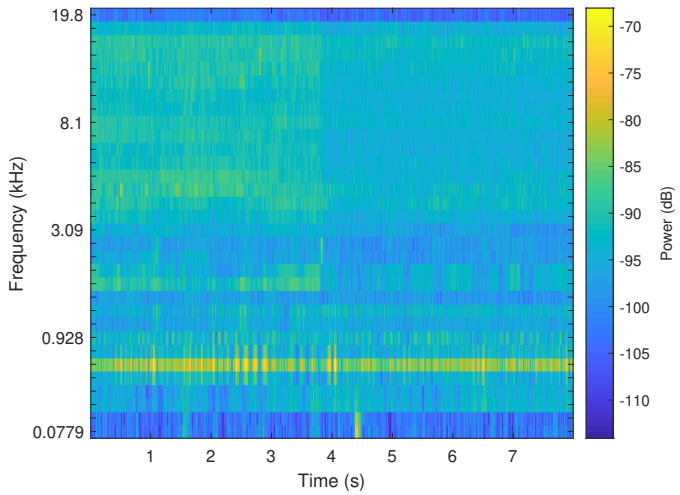
Figures 5.2(a), 5.2(b) and 5.2(c) shows the Mel spectrum of three audio signals of the drone in hover mode. They represent the Mel spectrum when the drone is unloaded, carrying a 20 g payload and carrying 28 g payload. The x-axis represents the time domain of the signal. The y-axis represents the frequency domain from 0.0779 kHz to 19.8 kHz. The colorbar off to the right of each of the figures indicates dB-to-color mapping. As per [64] the Mel spectrogram function is computed as follow:

1. The audio input is buffered into frames of length:

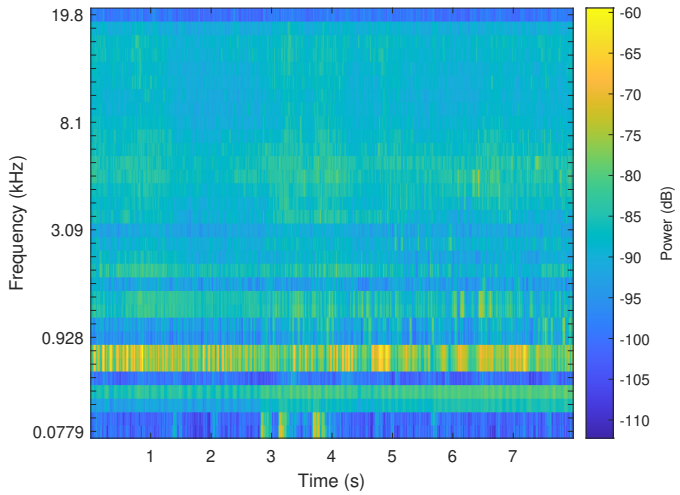
$$fs \cdot 0.03 \text{ samples}$$

2. The frames are overlapped by samples of length:

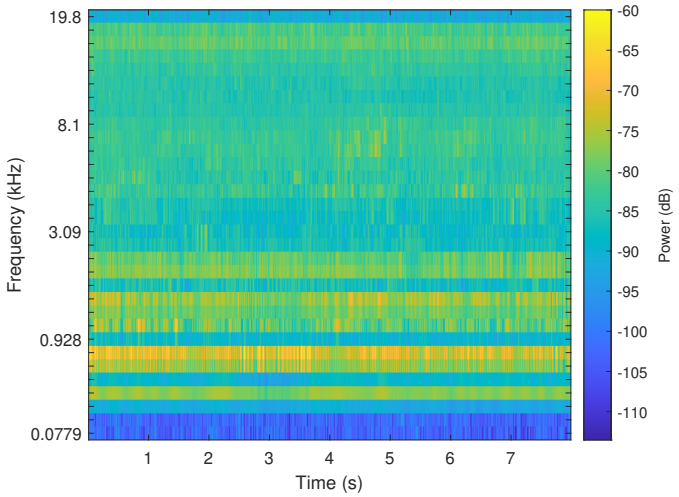
$$fs \cdot 0.02 \text{ samples}$$



(a) Mel spectrum of drone audio signal with no load.



(b) Mel spectrum of drone audio signal with a load of 20 g.



(c) Mel spectrum of drone audio signal with a load of 28 g.

3. A periodic hamming window is applied to each frame.

4. Each frame is then converted to frequency-domain representation.

5. Each frame passes through a mel filter bank.

6. The spectral values from the mel filter bank are then summed.

The sampling rate used is 44,100 kHz. The number of audio frames is calculated as [68]

$$L = \left\lfloor \frac{nRows - winLen}{hopLen} \right\rfloor + 1 \text{ frames} \tag{5.1}$$

where

- nRows is the number of samples in the audio signal,

- winLen is the number of samples in the analysis window and is calculated

$$winLen = fs \cdot 0.03 \tag{5.2}$$

  where fs is the sampling frequency,

- hopLen is the number of samples in the current frame before the start of the next frame,

$$hopeLen = WindowLength - OverlapLength \tag{5.3}$$

- OverlapLength is the number of overlapping samples between adjacent windows.

$$OverlapLength = fs \cdot 0.02 \tag{5.4}$$

Using the MATLAB® audio toolbox, the MFCCs are calculated. The duration of each of the signals (in hover mode) is eight seconds, with 44,100 samples per second. For

each signal, we have 352,800 samples.

$$winLen = 44100 \cdot 0.03 = 1323 \text{ samples}$$

$$OverlapLength = 44100 \cdot 0.02 = 882 \text{ samples}$$

$$hopeLen = 1323 - 882 = 441 \text{ samples}$$

$$L = \left\lfloor \frac{352800 - 1323}{441} \right\rfloor + 1 = 798 \text{ frames}$$

Thirteen MFCCs are computed for each window. The sequence of data items produced is then used as the feature vector.

5.4 Supervised Learning: Long Short-Term Memory Recurrent Neural Networks

We use an LSTM network to classify the audio signals according to their respective payload. As its name suggests, LSTM means that short-term patterns are not forgotten in the long-term [47].

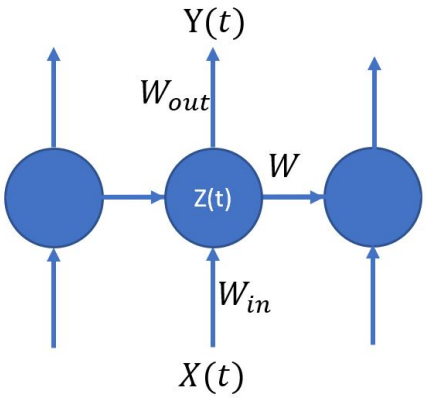


Figure 5.2: Recurrent neural network architecture taking into account the knowledge about the previous runs.

LSTM is a recurrent neural network architecture. The distinct feature of a recurrent neural network is that it takes into account knowledge about the previous runs [47]. Figure 5.2 shows how the RNN architecture makes use of the previous states of the network. $X(t)$ is the input. The nodes are connected with arrows. An arrow has a particular weight and is multiplied by the input. When two or more arrows lead to the same node, their values are summed in that node. The node shown as a circle represents a hidden layer. The output is equal to $Y(t) = Z(t) \cdot W_{out}$. What makes it different from a classical neural network is that it introduces a transition weight W to transfer information over time, as shown in Figure 5.2. Recurrent neural networks are used with sequential data. In this work, the MFCC features extracted for each frame are used to train the LSTM network.

Figure 5.3: Architecture of an LSTM network for classification in MATLAB® [49].

Figure 5.3 shows the architecture of an LSTM network for the purpose of classification in MATLAB®. Each block in the figure is a representation of a layer in our

network. Building a classification LSTM network in MATLAB® consist of creating the following layers:

- A sequence input layer.
- A BiLSTM layer as a sequence layer that learns long-term dependencies between time steps in the data. It is bidirectional because we want the network to learn from the complete time series at each time step.
- A fully connected layer, that is a layer that multiplies the input by a weight matrix and then adds a bias vector.
- A softmax layer that applies a softmax function to its input.
- A classification layer.

The weights are the classes predicted by the LSTM network. Therefore the weights are the labels we use in the training data.

5.5 Performance and Evaluation

The accuracy of our model is evaluated. Accuracy is defined as the following ratio

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

A correct prediction is when the model predicts the actual payload of the drone for an eight-second flight time, using the audio signal that it produces. In addition to accuracy, precision and recall are also calculated and used to evaluate the model. Recall and precision are defined as follow:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision represents the proportion of positive identifications that were actually correct, while recall represents the proportion of actual positives that were identified correctly.

A confidence interval is calculated for each of the evaluation metrics. A confidence interval provides a range of model skills with a likelihood that the model skill falls between the ranges when making new predictions on new data.

The confidence interval is estimated using empirical bootstrapping. The steps of bootstrapping to calculate the confidence intervals are defined as follow:

1. Let $X = x_1, x_2, ..., x_n$ be a data sample drawn from a distribution F.

2. Let u be a statistic of interest computed from the sample.

3. We perform a sampling with replacement on X to generate X^*. For the new sample X^* to be of same size as X, we sample as many times as the size of X when generating X^*.

4. Compute u^* the statistic of interest from the bootstrap sample X^*.

5. Repeat from Step 3 to generate bootstrap samples along with the statistic u^* as much as wanted.

6. Compute $\delta^* = u - u^*$ for each bootstrap sample.

7. Order δ^* in a non-decreasing order.

8. For a 95% confidence level, we choose the $\delta^*_{0.025}$ at the 97.5 percentile and $\delta^*_{0.975}$ at the 2.5 percentile.

9. The bootstrap 95% confidence interval for u is:

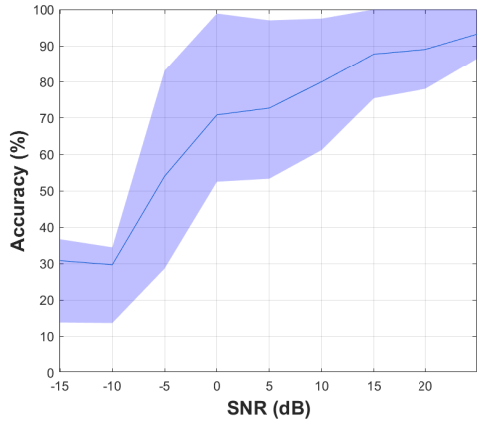
$$[u - \delta^*_{0.025}, u - delta^*_{0.975}]$$

We evaluated two models in two different experiments. In the first experiment, we trained and tested a neural network using only three weights. In the second experiment, we tested a neural network using five different weights. The two experiments weights are shown in table 5.1. We did this to see the variation in performance when we have more granularity.

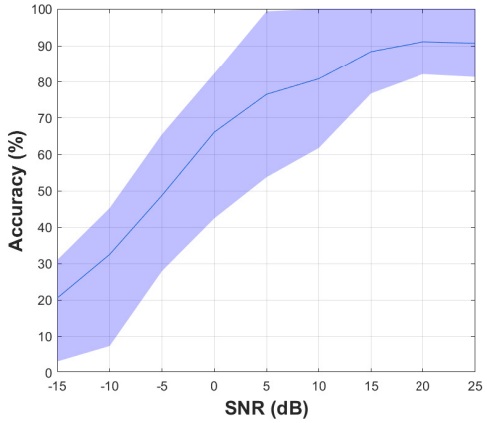
Evaluation of the Models performance vs. SNR is shown in figs. 5.4(a) to 5.4(f). The model using only three weights performance is shown in figs. 5.4(a), 5.4(c) and 5.4(e), while the performance of the model using five different weights is shown in figs. 5.4(b), 5.4(d) and 5.4(f). The accuracy of both models gets approximately to an interval of [80%, 100%] at an SNR of 20 dB. The precision of both models gets to an interval of [80%, 100%] at an SNR of 15 dB. The recall of the model using three weights gets to an interval of [80%, 100%] at SNR of 20 dB while the model using five weights gets to the same interval at SNR of 15 dB. The performance of both models is almost the same at higher SNR. The performance of the model using three weights is much better than that of the model using five weights at low SNR. The result shows that as we add more weights to have finer and more granular labelling, the performance drops at lower SNR. The large signal bandwidth (20 kHz) explains the required high SNR (20 dB) to reach the peak performance.

Table 5.1: Weights used in each experiment.

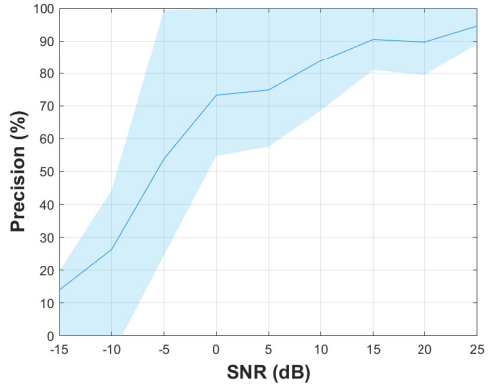
Weights in first experiment	Weights in second experiment
No Load	No Load
20g	10g
28g	20g
	24g
	28g



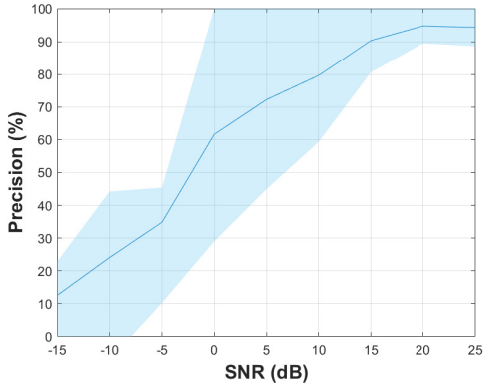
(a) Model Accuracy vs SNR with a 95% confidence interval (when using three weights)



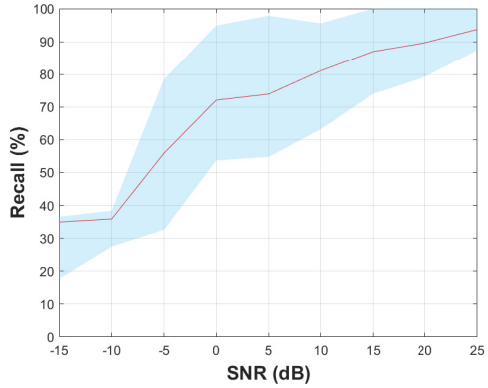
(b) Model Accuracy vs SNR with a 95% confidence interval (when using five weights)



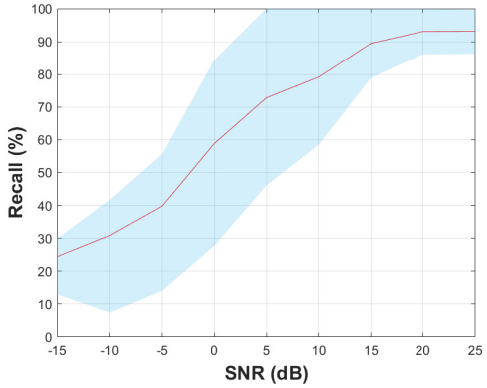
(c) Model Precision vs SNR with a 95% confidence interval (when using three weights)



(d) Model Precision vs SNR with a 95% confidence interval (when using five weights)



(e) Model Recall vs SNR with a 95% confidence interval (when using three weights)



(f) Model Recall vs SNR with a 95% confidence interval (when using five weights)

Figure 5.4

5.6 The dummy classifiers vs the LSTM network for drone payload classification

We build three dummy classifiers for classifying the weight of the drone's payload with the following strategies:

1. **most frequent dummy classifier**: always predicts the most frequent label in training set.

2. **uniform dummy classifier**: predicts uniformly at random.

3. **stratified dummy classifier**: predicts while respecting the training sets class distribution.

We now show that the LSTM network built performs better than any of the three dummy classifiers for the task of classifying the weight of a drone payload. Formally, we state the null and the alternative hypotheses as follow:

Hypothesis 0 (Null Hypothesis)**.** *No statistically significant difference between the performance of each of the dummy classifiers and the model trained on the collected and processed acoustic data in the task of identifying the weight of a drone's payload.*

Hypothesis 1 (Alternative Hypothesis)**.** *There is a statistically significant difference between the performance of each of the dummy classifiers and the model trained on the collected and processed acoustic data in the task of identifying the weight of a drone's payload.*

To compare each dummy classifier with the LSTM network, we perform random resampling in conjunction with a corrected resampled t-test as suggested by Bouckaert and Frank [61]. The corrected resampled t-test is the same one as Equation (4.1):

Table 5.2: t value for each performance metric

	Accuracy	Precision	Recall
LSTM network vs Most-Frequent Dummy Classifier	1.5287	3.4532	3.0102
LSTM network vs Uniform Dummy Classifier	2.5975	2.9732	3.0090
LSTM network vs Stratified Dummy Classifier	2.6657	2.9425	2.9374

The t-value is calculated according to Equation (4.1) for accuracy, recall and precision performance metrics between the dummy classifiers and the LSTM network. The values of the t-tests are shown in Table 5.2. The values suggest rejecting the null hypothesis H_0 and favour the alternate hypothesis H_1 with a confidence level of 95% for each of the performance metrics and dummy classifiers except for the case of LSTM vs. most-frequent dummy classifier, where we cannot reject the null hypothesis in terms of accuracy as a performance metric with a t-value of 1.5287. However, at a lower confidence level of 90%, t-values suggest rejecting the null hypothesis for all metrics and dummy classifiers. Furthermore, since the t-values of two out of the three performance metrics suggest rejecting the null hypothesis at a confidence level of 95% and all the t-values of all performance metrics suggest rejecting the null hypothesis at a confidence level of 90%, we conclude by saying that the LSTM network outperforms the dummy classifiers in the task of identifying the weight of the drone's payload.

5.7 Summary

We recorded audio signals for the Parrot Mambo drone while it was in hover mode and moving around, in both cases at an altitude of one meter. Then MFCCs are extracted from those signals as features to be used with supervised machine learning. We used 75% of the data to train a LSTM network, while we used the other 25% to

produce more signals with different SNRs. The model has been tested using the signals produced with different SNRs. Model performance vs. SNR results has been presented. We found that the peak performance is reached as SNR reaches 20 dB. We conclude by saying that the payload of the drone can well be identified using the MFCCs and LSTM neural networks. We also noticed that as we add more granularity to weight labelling, performance drops at lower SNRs. We then created three dummy classifiers with different classification strategies and compared their performance to the LSTM network. We found that the LSTM model outperformed the three dummy classifiers.

Future work includes using different drone models with different payloads to compare the machine learning model's performance when using heavier drones and heavier payloads with that of using lighter ones. In addition, there are several conditions to test the model against in future work. One necessary condition is to see how altitude and distance from the acoustic sensor affect the model's performance.

Chapter 6

A Reverse Turing Like Test for Quad-copters

An important piece of information that anti-drone systems seek is whether a drone is autonomous or operated by a human. We develop a reverse Turing test to tell apart a human pilot from an autopilot in this work. A reverse Turing test is when the roles of humans and computers are reversed with respect to the original Turing test. We develop a model using a neural network and features extracted from flights executed by an autonomous drone or controlled by a human operator. Every flight path taken requires the drone to avoid an obstacle that manifests specific attributes. Features from the flight are extracted to distinguish between an autonomous drone and a human operator. Our method achieves an average of 97.8% accuracy, 91% precision and 68.5% recall on imbalanced data used for training and testing the model.

We review related work in Section 6.1. Next, we look at the main idea of the reverse Turing test for quadcopters. Section 6.3 discusses the framework and the simulations. Features and feature selection are introduced in sections 6.4 and 6.5. The problem of classification on imbalanced data is discussed briefly in Section 6.6. Simulation results and analysis are presented in Section 6.7. We then compare the neural network developed to the dummy classifiers in Section 6.8. Finally, we conclude this chapter in

Section 6.9.

6.1 Related Work

The application of the Turing test to UAVs has been studied in the literature. The work by Young in [24] provides four protocols to evaluate autonomous systems flying UAVs. The author suggests that human observers, a group of subject matter expert reviewers, monitor the UAVs performing multiple mission tasks. Moreover, the paper mentions some cues to differentiate between whether an aerial vehicle is piloted or autonomous. Note that a piloted aircraft, as defined by the author, is one where a pilot or operator is either physically on-board or remotely controls it. In both cases, the operator is directly providing real-time flight control inputs and thus operating the aircraft.

In [25], Herńandez-Orallo has divided the system evaluation types in artificial intelligence to be either task-oriented or ability-oriented. Autonomous navigation systems in drones are considered specialized systems that require task-oriented evaluation. Furthermore, the author has defined three types of behavioural evaluation for task-oriented evaluation: human discrimination, problem benchmarks and peer confrontation. In human discrimination, the assessment is made by observation. The Turing test falls under this type of evaluation. The idea is to evaluate the system by human judges or by comparing those systems to humans.

The reverse Turing test is related to the Turing test. An example of a reverse Turing test is the CAPTCHA [26]. It is a test to distinguish a bot from a human. In this test, a program tells whether the user is a bot or a human. Many variations of this test have been proposed, such as Activity Recognition CAPTCHA [27].

In [28], Kalik and Prokhorov present an implementation of the Turing test in the automotive field. Rather than assessing the intelligence of a computer program, the authors extend the scope of the test to the domain of intelligent automotive vehicles,

consider different formats for such a test, and study various measures produced by these tests. Both real-world environments and simulated environments have been considered in their research. An interesting part of the research is where the authors suggest using a passive interrogation method, which departs from the original Turing test by removing the interaction between the interrogator and subject and converting the interrogator's role to an observer role. The authors also mention the possibility of implementing a computerized classifier system, placing humans out of the testing loop. It is the latter approach we adopt in this work.

Another variation of CAPTCHA using speech is studied in [29] by Kochanski, Lopresti and Shih. The paper presents a test that depends on the fact that humans can better recognize distorted speech far more robust than automatic speech recognition techniques.

A reverse Turing test for detecting machine-made texts is proposed in [30] by Shao, Uchendu and Lee. The authors study the classification task of distinguishing between human-made vs. machine-made texts. Their study utilizes financial earning reports, research articles, and chatbot dialogues. Their model achieved an 84% accuracy.

Underwater vehicle metric assessment concerning autonomous capabilities has been studied by Insuarralde et Lane [31]. The authors' review approaches to assess autonomous behaviours in UMVs. They provide a baseline study for methodologies to assess maritime autonomy so that metrics can be defined for UMVs. Also, the authors present a statistical comparison of existing assessment frameworks for autonomy in different domains. The authors in [32] also review the current approaches to assess autonomous behaviour in self-governed vehicles. The paper also proposes metrics for undersea autonomy to identify the degree of autonomy in an AUV.

6.2 Reverse Turing Test for Quad-Copters

Path planning is based on mission planning. The goal of a mission planner is to generate and execute a plan to accomplish a specific navigational task. Examples of such tasks are exploration, coverage, surveillance and tracking.

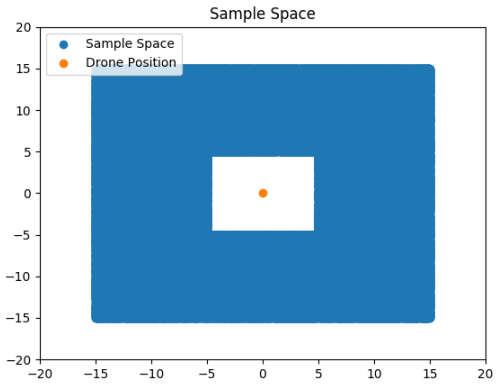
We seek to classify a drone operator that faces an obstacle and has to maneuver around it. The intuition behind this is that by facing an obstacle, the operation of the drone will exhibit certain features that the machine learning model can use to distinguish between a human pilot and an autonomous system.

We assume the navigational data is available through hacking or radars. The first objective is to identify a set of attributes that can be used to differentiate between autopilot and a human operator. In [24] the author lists several cues that a human observer might consider to distinguish whether a human or an autonomous system operates an aerial vehicle. The cues are as shown in Table 6.1.

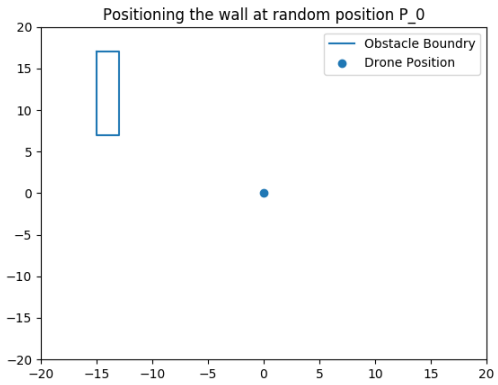
Table 6.1: Cues that can help in differentiating between a human operator of the drone and autopilot.

Cue
Slowness of mission task execution
Unsteadiness of flight maneuvers
Use of 2-3 dimensional rectilinear versus curvilinear trajectories
Discrete or step like vs. continuous & smooth, incremental attitude or position change
Lack of precision of flight maneuvers
Severe or abrupt changes in attitude or position
Failure to complete flight maneuvers or mission tasks
Situational awareness of hazards, obstacles and other aircraft flying in proximity to the evaluated UAV
Manifestation of inadequate, or inappropriate, flight behaviour in response to unplanned/ unanticipated changes in mission tasks or scope & operational environment.

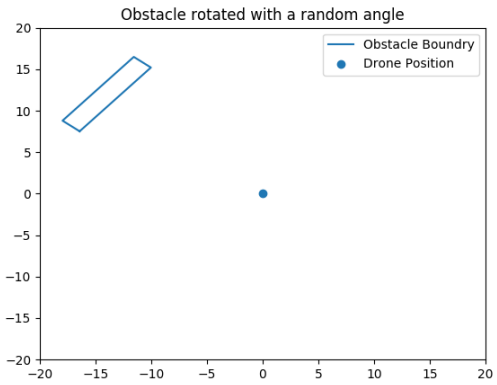
To ensure that the drone has to maneuver around the obstacle for both, human pilot



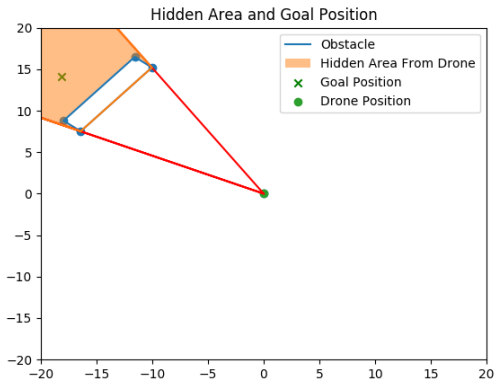
(a) Obstacle sample space shown in blue.



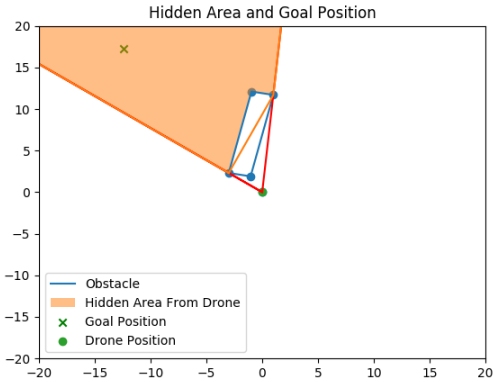
(b) Placing the obstacle in a random position.



(c) Rotating the obstacle by a random angle around the z-axis.



(d) Finding the hidden area sample space (shown in orange) and placing the goal randomly in it.



(e) Rotating the obstacle by a random angle around the z-axis.

Figure 6.1: Curvature of different paths.

and autopilot, the simulations are developed as follow:

1. A drone is positioned in the center of a $w \times h$ grid map.

2. A random position P_o is chosen from the sample space shown as the blue region shown in Figure 6.1(a).

3. An obstacle of the dimensions $obs_x \times obs_y \times obs_z$ that is, $depth \times width \times height$ respectively, is placed in the randomly selected position as in Figure 6.1(b).

4. The obstacle is rotated with a random angle α along the z-axis (yaw rotation). An example can be seen in Figure 6.1(c).

Table 6.2: Values chosen for the grid and obstacle size parameters.

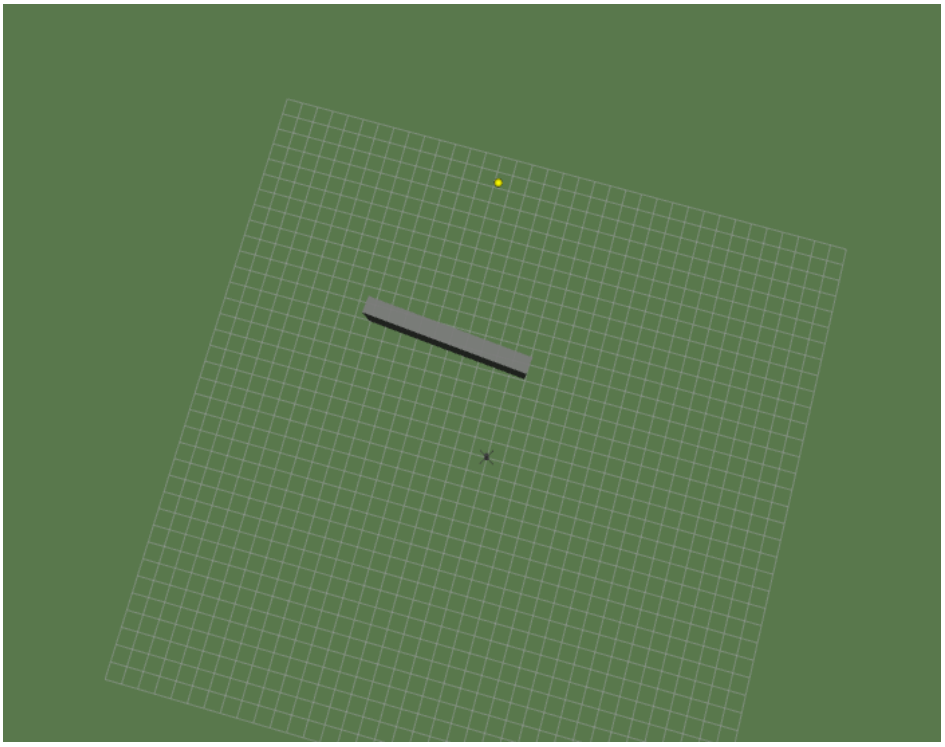
w	20
h	20
obs_x	1
obs_y	10
obs_z	5

5. Finally, the goal position is placed in the hidden area, as shown in Figure 6.1(d). Finding the hidden area is a problem related to the visibility polygon. Rays are shot towards the four corners of a rectangle. We then compute the angles of all possible combinations of the four rays. That is, we choose the widest angle among $\binom{4}{2} = 6$ angles. The hidden area lies behind the corners with the widest angle. In case opposite corners had the widest angle, as in Figure 6.1(e), the area of the obstacle polygon is removed from the hidden area polygon. The values of the parameters w, h, obs_x, obs_y and obs_z are chosen for the simulations to be as shown in Table 6.2.

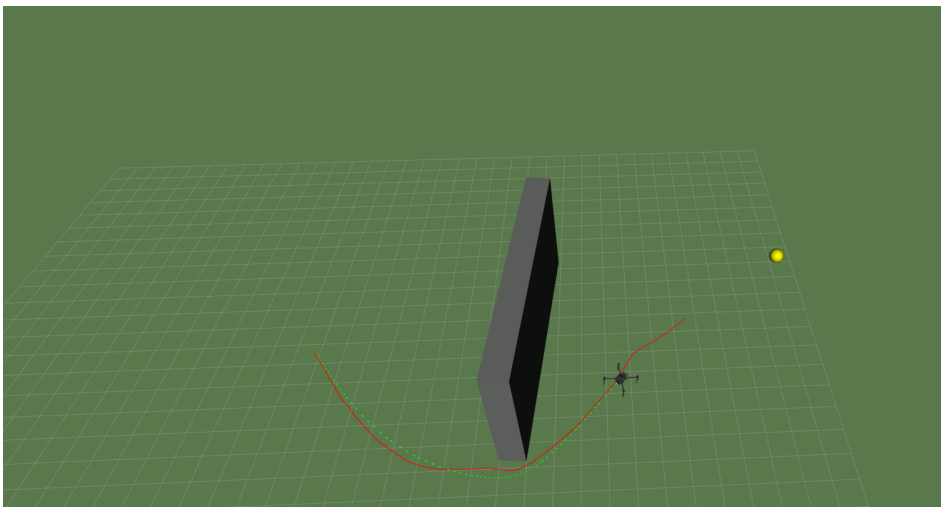
6.3 Framework and Simulation

The simulation is executed using ROS Melodic, Gazebo 9 and RViz (a 3D visualization tool for ROS) on a Ubuntu Bionic machine. The Gazebo is an open-source 3D robotics simulator with a robust physics engine. Moreover, MAVROS, a MAVLink extendable communication node for ROS with proxy for Ground Control Station, is installed along with PX4-avoidance node from [69].

Two different types of simulations are developed. The first type is where we have an autopilot controlling the drone with obstacle avoidance capabilities and using a local planner in [69]. A goal is positioned that the autopilot must maneuver around the obstacle to reach the goal position. All the files are generated using a shell script that controls several python scripts to create the XML files that define the environment, obstacle, initial drone position and goal position. The autopilot is then started, and an instance of the simulation runs. The process repeats for a thousand simulations. While a simulation runs, we collect navigational data through a ROS node explicitly developed for this research. A screenshot of the simulator is shown in Figures 6.2(a) and 6.2(b). A video of the autopilot simulation can be seen in [70].



(a) A top view of the simulation. The obstacle is the grey wall placed and rotated randomly along with the goal.



(b) Screenshot as the drone controlled by the autopilot is reaching its goal. The goal is shown as the yellow sphere.

Figure 6.2: Simulation screen captures.

Figure 6.3: Logitech Gamepad F310.

The second type of simulation is a human pilot controlling the drone through Logitech Gamepad F310 (see Figure 6.3). The user controls the drone altitude and yaw using the left stick. Moving the left stick up would lift the drone, and moving the stick down would decrease its altitude. Moving the left stick right or left would cause the drone to yaw in the respective direction. Moving the right stick forward would cause a forward pitch movement while moving it down would cause a pitch backward. To create a roll movement, the user can move the right stick, either right or left.

In both types of simulations, gravity is $-9.8\frac{m}{s^2}$. Gravity is the only external force considered. We imposed a condition on human pilots that they cannot fly over the obstacle, but they must avoid it by moving around the obstacle. This restriction was imposed because the avoidance algorithm in the autopilot is always set to avoid an obstacle by moving around it and not fly over it [69].

Another restriction on the drone's maximum speed is imposed on the drone configuration for a human pilot. The reason for doing so is that the obstacle avoidance algorithm limits the maximum speed, which is approximately 3 m/s (due to the cost

of computing the avoidance path). By imposing this restriction, we are achieving a more realistic and better comparison as using a more capable computing module would eliminate this restriction.

6.4 Features

Using the ROS node developed, we capture each of the following attributes at each simulation instance at a frequency of 50 message/sec:

- timestamp
- x-position of the drone
- y-position of the drone
- z-position of the drone
- x-velocity of the drone
- y-velocity of the drone
- z-velocity of the drone
- A set of quaternions of the drone angular positions

While some of the cues mentioned in Table 6.1 are not considered, we now dive deeper into some of them and quantify/measure the attribute or property discussed. To build our reverse Turing test for quad-copters and extract appropriate features for our model we focus on the following cues:

- **Unsteadiness of flight maneuvers**

  The author in [24] defines steady flight to occur when the components of the translational velocity vector of an aircraft, when expressed in the aircraft-fixed frame, are constant in time and when the components of the angular velocity

vector, when described in the aircraft-fixed frame, are constant in time. The standard deviation of velocity and velocities of roll, pitch and yaw are features that can be good metrics of unsteadiness of flight maneuvers.

Velocity along x-axis and the y-axis are captured in the messages collected through ROS node. The magnitude of the total velocity is calculated as follow:

$$v_{Total} = \sqrt{(v_x)^2 + (v_y)^2} \tag{6.1}$$

Where v_x is the velocity along the x-axis, v_y is the velocity along the y-axis and v_{Total} is the magnitude of the total velocity. The standard deviation of the magnitude of the total velocity:

$$\sigma_{v_{Total}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_{Total_i} - \overline{v_{Total}})^2} \tag{6.2}$$

Where N is the number of samples taken at a sampling frequency of 50 samples/second.

We capture the roll ϕ , pitch θ and yaw ψ using the ROS node developed for data collection. We calculate their velocities as follow:

$$\phi' = \frac{d\phi}{dt},\ \theta' = \frac{d\theta}{dt},\ \psi' = \frac{d\psi}{dt} \tag{6.3}$$

The standard deviations of roll, pitch yaw velocities are as follow:

$$\sigma_{\phi'} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\phi'_i - \overline{\phi'})^2}$$

$$\sigma_{\theta'} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\theta'_i - \overline{\theta'})^2} \tag{6.4}$$

$$\sigma_{\psi'} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\psi'_i - \overline{\psi'})^2}$$

- **Use of 2-3 dimensional rectilinear versus curvilinear trajectories**

  Standard deviation of the curvature would capture the difference between 2-3 dimensional rectilinear versus curvilinear trajectories.

  Let $r(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ be the position vector representing the position of the drone at time t.

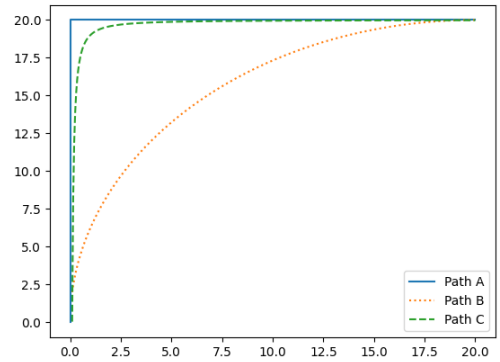
  The curvature is calculated as follow:

$$\kappa(t) = \frac{\mid x''(t)y'(t) - x'(t)y''(t) \mid}{[(x'(t))^2 + (y'(t))^2]^{3/2}} \tag{6.5}$$

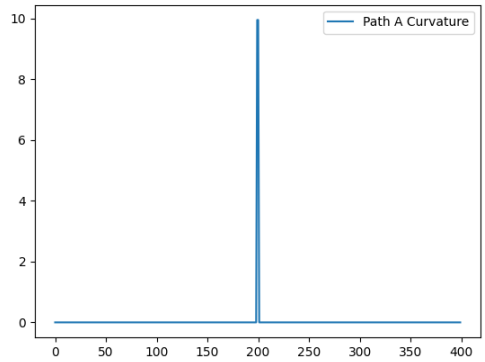
  Where x' and x'' are the first and second derivatives of the x-coordinate of the position vector r with respect to time. And y' and y'' are the first and second derivatives of the y-coordinate of the position vector r with respect to time.

  The standard deviation of the curvature κ is:

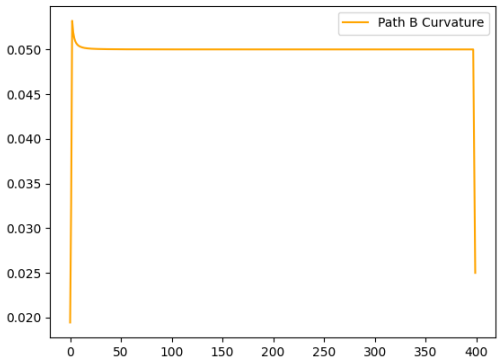
$$\sigma_{\kappa} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\kappa_i - \overline{\kappa})^2} \tag{6.6}$$



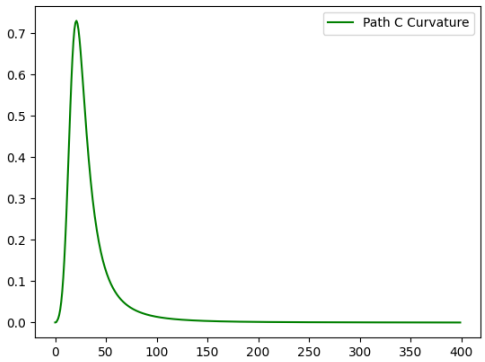
(a) Three different paths representing rectilinear and curvilinear trajectories.



(b) Curvature of Path A.



(c) Curvature of Path B.



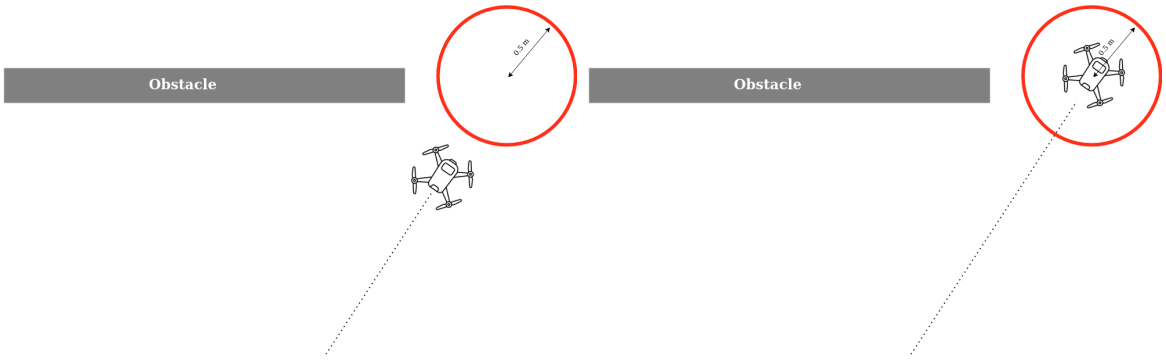
(d) Curvature of Path C.

Figure 6.4: Curvature of different paths.

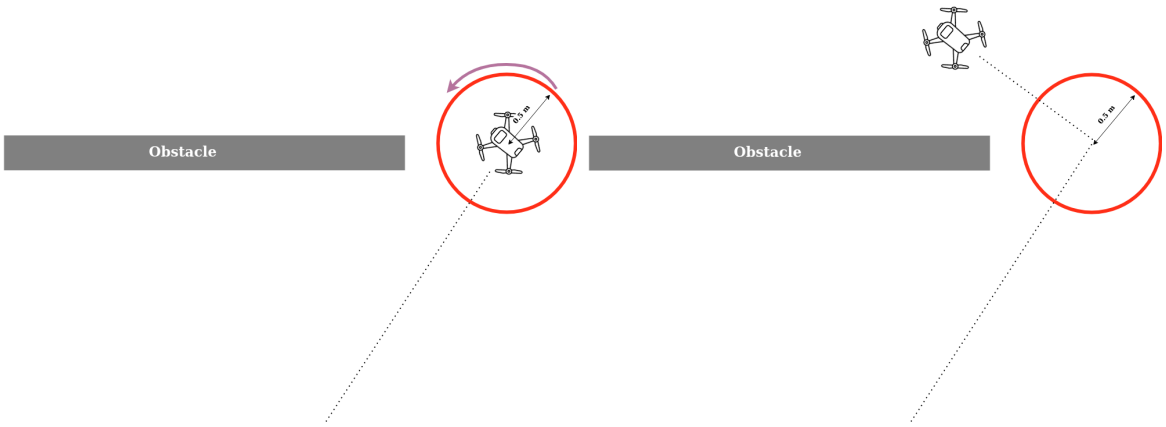
Figures 6.4(b), 6.4(c) and 6.4(d) show the curvature calculated for each sample on the paths shown in Figure 6.4(a).

- **Discrete or step-like vs. continuous and smooth, incremental attitude or position change**

  Discrete or step-like vs. continuous and smooth incremental attitude or position changes are captured within a couple of features such as the number of pauses, the standard deviation of the magnitude of acceleration and standard deviation of roll, pitch and yaw acceleration. We define each of the features that capture this cue as follow:



(a) Drone entering the pause zone. (b) Done reached the pause zone center.



(c) Drone made a yaw rotation. (d) Drone left the pause zone.

Figure 6.5: A pause taken by the drone in a rectilinear trajectory.

1. Number of pauses: This corresponds to the number of pauses during its flight from the initial position to the goal position. The pause is defined as

a time when the drone has stayed within specified proximity β of a specific location in space for more than a predefined time γ. The intuition is that human pilots would make more pauses than an autopilot, especially with rectilinear trajectories where the pilot would stop the drone, rotate then progress. The drone's max speed is 3 m/s, and its max yaw speed is $\frac{\pi}{2}$ rad/s. A pause, rotate, then turn would approximately take $1-2$ seconds to leave the proximity of a circle with a radius of 0.5 m. Once the center of the drone crosses the red circle it would take the drone $\sim 0.2s$ to reach the center of the pause zone shown as a red circle in figures 6.5(a) and 6.5(b) and the rotation would take $\sim 0.5s$ for a $\frac{\pi}{4}rad$ yaw rotation such as the one in Figure 6.5(c). The drone would take $\sim 0.25s$ to exit the pause zone Figure 6.5(d). This is a total of approximately one second; however, this duration does not consider the time the human pilot takes for planning and executing the maneuvers using the joystick. In our experiment, the parameter β is set to 0.5 meters and γ is set to two seconds.

2. Standard deviation of the magnitude of acceleration First we calculate the acceleration along each of the x-axis and y-axis.

$$x'' = \frac{dx'}{dt},\ y'' = \frac{dy'}{dt} \tag{6.7}$$

Where x' is the velocity along the x-axis and y' is the velocity along the y-axis.

The magnitude of total acceleration:

$$acc_{Total} = \sqrt{(x'')^2 + (y'')^2} \tag{6.8}$$

Standard deviation of the magnitude of acceleration:

$$\sigma_{acc_{Total}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(acc_{Total_i} - \overline{acc_{Total}})^2} \tag{6.9}$$

3. Acceleration of each of roll, pitch and yaw:

$$\phi'' = \frac{d\phi'}{dt},\ \theta'' = \frac{d\theta'}{dt},\ \psi'' = \frac{d\psi'}{dt} \tag{6.10}$$

Standard deviations of roll, pitch yaw acceleration are as follow:

$$\sigma_{\phi''} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\phi_i'' - \overline{\phi''})^2}$$

$$\sigma_{\theta''} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\theta_i'' - \overline{\theta''})^2} \tag{6.11}$$

$$\sigma_{\psi''} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\psi_i'' - \overline{\psi''})^2}$$

- **Severe or abrupt changes in attitude or position**

Severe and abrupt changes can be captured in features such as the standard deviation of the magnitude of the jerk (a derivative of acceleration) and standard deviation of each roll, pitch and yaw jerk.

The jerk along the x-axis and y-axis:

$$x''' = \frac{dx''}{dt},\ y''' = \frac{dy''}{dt}$$

The magnitude of the total jerk:

$$jerk_{Total} = \sqrt{(x''')^2 + (y''')^2}$$

Standard deviation of of the magnitude of total jerk

$$\sigma_{jerk_{Total}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(jerk_{Total_i} - \overline{jerk_{Total}})^2}$$

Standard deviation of each of roll, pitch and yaw jerk:

$$\sigma_{\phi'''} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\phi_i''' - \overline{\phi''})^2}$$

$$\sigma_{\theta'''} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\theta_i''' - \overline{\theta''})^2} \tag{6.12}$$

$$\sigma_{\psi'''} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\psi_i''' - \overline{\psi''})^2}$$

6.5 Feature Selection Method

We use a filter-based feature selection method among the 14 features listed. Specifically, we use the ANOVA F-test. The F-test is used to calculate the ratio between variances. ANOVA is a test that determines if the means from two or more groups are different by using F-tests to test the equality of the means statistically. The ANOVA is suitable for classification problem [56, p. 242].

Using the sklearn library SelectKBest function for feature selection, we run the

Table 6.3: Top k features ranked.

Top k	**feature(s)**
14	number of pauses, standard deviation of each of curvature, magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration, roll jerk, pitch velocity, pitch acceleration , pitch jerk , yaw velocity, yaw acceleration and yaw jerk.
13	standard deviation of each of curvature, magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration, roll jerk, pitch velocity, pitch acceleration, pitch jerk, yaw velocity, yaw acceleration and yaw jerk.
12	standard deviation of each of curvature, magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration, roll jerk, pitch velocity, pitch acceleration, pitch jerk, yaw velocity and yaw acceleration.
11	standard deviation of each of curvature, magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration, roll jerk, pitch velocity, pitch acceleration, pitch jerk and yaw velocity.
10	standard deviation of each of curvature, magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration, roll jerk, pitch velocity, pitch acceleration and pitch jerk.
9	standard deviation of each of magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration, roll jerk, pitch velocity, pitch acceleration and pitch jerk.
8	standard deviation of each of magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration, roll jerk, pitch velocity and pitch acceleration.
7	standard deviation of each of magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration, pitch velocity and of pitch acceleration.
6	standard deviation of each of magnitude of velocity, acceleration, jerk, roll velocity, roll acceleration and pitch velocity.
5	standard deviation of each of acceleration, jerk, roll velocity, roll acceleration and pitch velocity.
4	standard deviation of each of acceleration, jerk, roll velocity and pitch velocity.
3	standard deviation of each of acceleration, roll velocity and pitch velocity.
2	standard deviation of each of acceleration and roll velocity.
1	standard deviation of roll velocity.

function with f_classif (ANOVA) to select the top k features. We treated k as a variable from 1 to 14 to see how the performance varies with different k features. The top k features, along with the feature names, are listed in Table 6.3. The features are ranked. We build a model for each top k feature and analyze the results accordingly.

6.6 Classification on Imbalanced Data

A deep feed forward neural network model is built for classification. The neural network is defined with two densely connected hidden layers of 16 nodes each, a dropout layer to reduce over-fitting, and an output sigmoid layer that returns the probability of the drone's operator being a human. The hyperparameters, epochs and batch size are 100 and 2048, respectively. The data collected was imbalanced. For autopilot, 1000 simulations were done. However, we collected data from 10 people of around 50 simulations for human pilots, where each participant contributed in 5 simulations.

Without considering the need for compensation in imbalanced data in a binary classifier, the model might learn the following behaviour: giving '0' as a class output because it is correct more often than giving '1'.

When there is a highly imbalanced dataset, in which the number of examples in one class vastly outnumber the examples in another, class weights can be used in Keras to help the model learn from the imbalanced data, by doing so, the model will pay more attention to examples from an under-represented class.

6.7 Simulation Results and Analysis

Firstly, let us define the evaluation metrics using True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) quantities.

1. Precision is the ratio of TP to the sum of TP and FP.

$$Precision = \frac{TP}{TP + FP} \tag{6.13}$$

2. Recall is the ratio of TP to the sum of TP + FN.

$$Recall = \frac{TP}{TP + FN} \tag{6.14}$$

3. Accuracy is the ratio of number of correct predictions to the total number of predictions.

$$Accuracy = \frac{\text{Number of correct Predictions}}{\text{Total number of predictions}} \tag{6.15}$$

4. Area Under the ROC Curve (AUC) which is the area under the Receiver Operating Characteristic Curve (ROC). ROC is a graph showing the performance of a classification model at all classification thresholds. The ROC curve plots True Positive Rate (also known as Recall) vs False Positive Rate (also known as the false alarm rate). False Positive Rate (FPR) is defined as follow:

$$FPR = \frac{FP}{FP + TN} \tag{6.16}$$

   ROC is commonly used to present results for binary decision problems in machine learning. It mainly describes how good the model is at predicting the positive class.

5. Precision-Recall Curve (PRC) and more specifically the area under PRC is another evaluation metric we consider. A precision-recall curve plots the precision and the recall for different thresholds, much like the ROC curve. In the case of imbalanced data, the ROC presents an overly optimistic view of the performance

of the algorithm, and a better alternative is to use the PRC for such data as per Davis et Goadrich [71].

We built 14 models using each of the top k features from Table 6.3. We evaluate them according to the described aforementioned evaluation metrics. The confidence interval is estimated using empirical bootstrapping. The steps of bootstrapping to calculate the confidence intervals are defined as follow:

1. Let $X = x_1, x_2, ..., x_n$ be a data sample drawn from a distribution F.

2. Let u be a statistic of interest computed from the sample.

3. We perform a sampling with replacement on X to generate X^*. For the new sample X^* to be of same size as X, we sample as many times as the size of X when generating X^*.

4. Compute u^* the statistic of interest from the bootstrap sample X^*.

5. Repeat from Step 3 to generate bootstrap samples along with the statistic u^* as much as wanted.

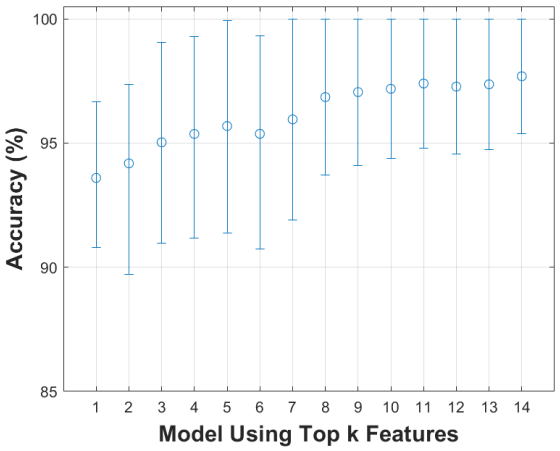
6. Compute $\delta^* = u - u^*$ for each bootstrap sample.

7. Order δ^* in a non-decreasing order.

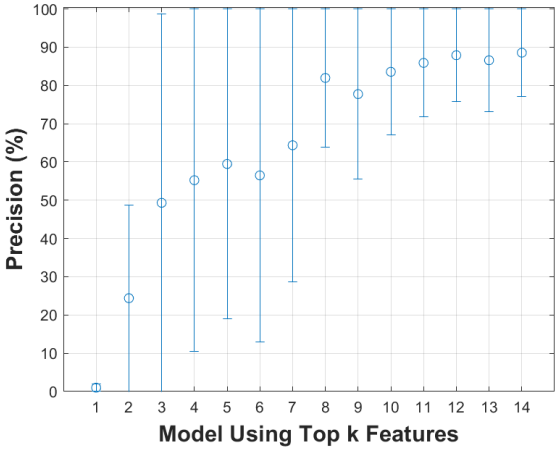
8. For a 95% confidence level, we choose the $\delta^*_{0.025}$ at the 97.5 percentile and $\delta^*_{0.975}$ at the 2.5 percentile.

9. The bootstrap 95% confidence interval for u is:

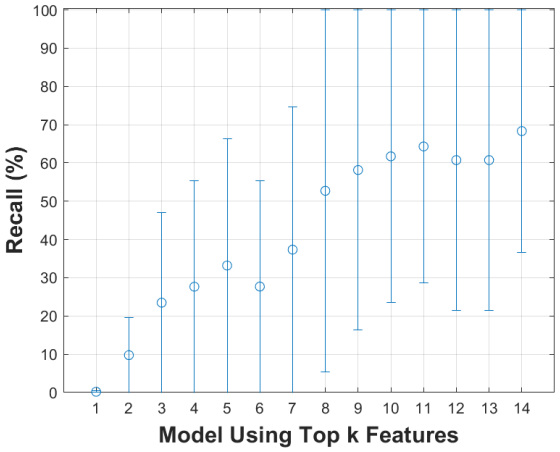
$$[u - \delta^*_{0.025}, u - delta^*_{0.975}]$$



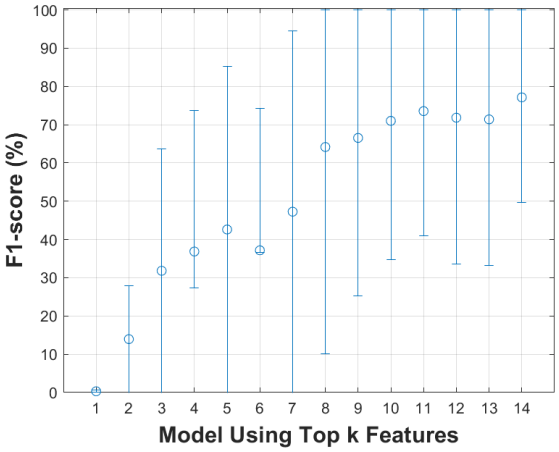
(a) Accuracy of each model with a 95% Confidence Interval.



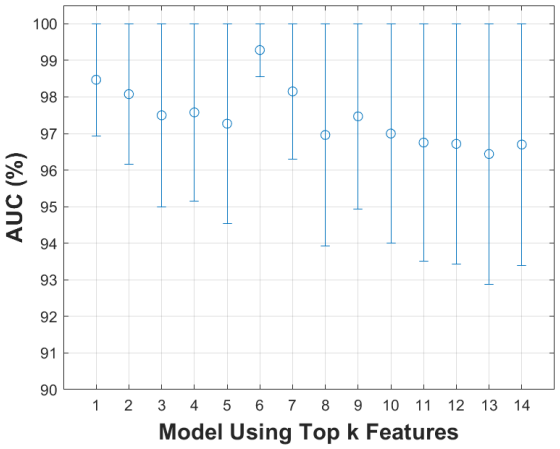
(b) Precision of each model with 95% Confidence Interval.



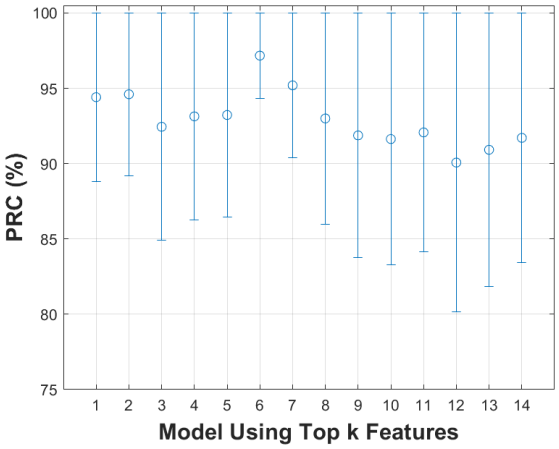
(c) Recall of each model with 95% Confidence Interval.



(d) F1-score of each model with 95% Confidence Interval.



(e) AUC of each model with 95% Confidence Interval.



(f) Area under PRC of each model with 95% Confidence Interval.

The results can be seen in Figures 6.6(a), 6.6(b), 6.6(c), 6.6(e), 6.6(f). Most of the models have high accuracy. That is due to the imbalance of the data. Accuracy here can be deceiving and what matters is the recall, which tells us what proportion of actual human pilots was identified correctly. The model using the 14 features have the best recall of an average of 68% and a 95% confidence interval of [38%, 100%]. In addition, with 95% confidence level the model using the 14 features achieves high accuracy [95%, 100%], high precision [88%, 100%], high AUC [93%, 100%] and high PRC [92%, 100%].

6.8 The dummy classifiers vs the neural network for drone pilot classification

We build three dummy classifiers for classifying the pilot of a drone. The classifiers and their strategies are as follow:

1. **most frequent dummy classifier**: always predicts the most frequent label in training set.
2. **uniform dummy classifier**: predicts uniformly at random.
3. **stratified dummy classifier**: predicts while respecting the training sets class distribution.

We show that the neural network built performs better than any of the three dummy classifiers. The null and alternative hypotheses are as follow:

Hypothesis 0 (Null Hypothesis)**.** *No statistically significant difference between the performance of each of the dummy classifiers and the model trained on the collected and processed navigational data in the task of identifying whether the pilot of a drone is a an autopilot or a human.*

Hypothesis 1 (Alternative Hypothesis)**.** *There is a statistically significant difference between the performance of each of the dummy classifiers and the model trained on the collected and processed navigational data in the task of identifying whether the pilot of a drone is a an autopilot or a human.*

The dataset used for training and testing is imbalanced, in which we had 57 data points for human pilots and a thousand data points for autopilot. We show four different tables for the t-values calculated. Table 6.4 shows the result when we performed a stratified resampling, Table 6.5 show the t-values in the case where we do a non-stratified random resampling, Table 6.6 shows the results where we perform a non-stratified resampling without replacement and finally Table 6.7 shows the results where we perform a stratified resampling without replacement. When using stratified or non-stratified without replacement, the results vary very slightly, which agrees with what Bouckaert and Frank have found in [61].

Table 6.4: t-value for each performance metric using stratified resampling with replacement

	Accuracy	Precision	Recall
Neural network vs Most-Frequent Dummy Classifier	2.9533	2.7256	0.9762
Neural network vs Uniform Dummy Classifier	18.3606	2.5247	0.9550
Neural network vs Stratified Dummy Classifier	5.7894	2.4842	0.9651

Table 6.5: t-value for each performance metric using non-stratified resampling with replacement

	Accuracy	Precision	Recall
Neural network vs Most-Frequent Dummy Classifier	2.8512	2.6072	1.0058
Neural network vs Uniform Dummy Classifier	19.5416	2.4182	0.9652
Neural network vs Stratified Dummy Classifier	5.4665	2.3937	0.9987

Table 6.6: t-value for each performance metric using non-stratified resampling without replacement

	Accuracy	Precision	Recall
Neural network vs Most-Frequent Dummy Classifier	3.0744	2.8547	0.9627
Neural network vs Uniform Dummy Classifier	19.9541	2.6503	0.9809
Neural network vs Stratified Dummy Classifier	5.7897	2.6598	0.9766

Table 6.7: t-value for each performance metric using stratified resampling without replacement

	Accuracy	Precision	Recall
Neural network vs Most-Frequent Dummy Classifier	2.8089	2.9480	0.8103
Neural network vs Uniform Dummy Classifier	17.1214	2.7402	0.8027
Neural network vs Stratified Dummy Classifier	5.7208	2.6941	0.8027

The t-value is calculated according to Equation (4.1) for accuracy, recall and precision performance metrics between the dummy classifiers and the neural network used

for the task of identifying whether the pilot is an autopilot or a human pilot. The values of the t-tests with different resampling strategies are shown in tables 6.4 to 6.7. The values of each of the dummy classifiers vs. the neural network using accuracy or precision as performance metrics suggest rejecting the null hypothesis H_0 and favour the alternate hypothesis H_1 with a confidence level of 95%. However, with the t-values obtained using the recall performance metric, we cannot reject the null hypothesis at a confidence level of 95%. As a reminder, recall represents the ratio of pilots that we correctly identified as humans to all pilots that were actually humans. Two performance metrics suggest rejecting the null hypothesis, while one performance metric shows we cannot reject the null hypothesis. Using a voting system where all performance metrics have equal weights, we reject the null hypothesis and favour the alternate hypothesis.

6.9 Summary

Classifying the drone operator, whether a human pilot or an autonomous system, can be done with high accuracy. Enhancing the identification model by gathering more human-operated drone data is something we look forward to post-pandemic, as only ten people were involved in the experiment to collect human pilot data in this work. We then used three strategies to build three dummy classifiers and used those as a baseline to compare the neural network's performance to those. We showed that the neural network built outperforms the dummy classifiers.

Future work includes developing a model to identify the autonomous system operating the drone. In this case, different autonomous systems would be used to operate the same type of drone. Data would be collected. Features would be identified post statistical analysis of the data collected. Finally, develop a model to use those features. Another future work includes identifying the class of algorithms controlling a swarm of drones.

Chapter 7

Conclusion

7.1 Summary

In chapters 4 and 5 we have measured the response or behaviour of a drone or a group of drones to a change in one of the three main components we introduced in Chapter 1, namely, the physical makeup of the done, environment and program controlling the drone.

Specifically, in Chapter 4 we monitored the behaviour of a group of drones when the only component changing is the program controlling the drones to form different formations. The behaviour is monitored through the collection of navigational data. The navigational data is further processed for feature extraction and evaluation. The data of the selected features are then used to train a linear model for classification based on Softmax regression. Finally, the model is trained to classify the data into one of the formations. The idea is to try to predict the formation while in transition. We used capture times of 33% and 66% of the total flight time of making the formation. Two models are trained on different capture times. The performance of each model is evaluated and presented.

Similarly, in Chapter 5 we monitored the behaviour of a drone as the physical makeup of the drone was altered by adding and removing payloads of different weights.

At the same time, both the environment and code controlling the drone remained unaltered. We collected audio signals for behaviour monitoring. The signals are then processed for features extraction. An LSTM neural network is then trained and evaluated. The evaluation is based on signals collected and not used for training. Those signals are further used to generate more signals with different SNR. We trained two neural networks, one only using three different payloads and a more granular one using five different payloads. Finally, the performance of each of the models is evaluated, analyzed and presented.

In Chapter 6, we monitored behaviour as the program controlling a drone changes; while the physical makeup of the drone remains unchanged. However, the environment does change in which we place the obstacle in a different location in both training and testing data-sets. The program controlling the drone is either set to manual, where a human pilot controls the drone, or to an autonomous flight mode. We collected navigational data to monitor the behaviour of the drone. The data is then processed for features extraction. The features are then ranked using the ANOVA f-test. Various machine learning models are built by training different neural networks using different combinations of features based on their rank. All models' performances are then evaluated, analyzed and presented.

7.2 Main Results

We have built three different machine learning models in chapters 4 to 6, we evaluated them with various performance metrics. The specific task of each model was different. However, they are all supervised learning algorithms that aim to do some classification. In Chapter 4, we developed a model that can predict the intended formation of a group of drones while in transition. While in Chapter 5 we used acoustic features to train an LSTM neural network to classify the weight of a drone's payload. The third task we took in Chapter 6 is to classify the pilot of a quadcopter, that is, to classify whether the

pilot is a human pilot or an autopilot using features processed from navigational data. In each of the chapters 4 to 6 we compared the performance of each of the models built to three dummy classifiers with three different classification strategies. Using those dummy classifiers as a baseline, we have shown that each of the advanced models we built has a better performance and is statistically significantly different from the baseline dummy classifiers.

7.3 Future Work

Future work will mainly focus on varying two out of the three components: Program/task, drone and environment. In the task of payload weight identification, we would vary the environment, besides the drone physical makeup, by conducting experiments indoor and outdoor, train an LSTM network on both environments and evaluate the model. This would also include an assessment of different types of noise on the performance of the model. Moreover, the distance between the drone and the acoustic sensor plays a significant role which should also be assessed. As for the group formation identification, we would also vary the environment by adding wind fields and obstacles. Also, the future work related to group formation identification will include non-linear and evasive trajectories.

Furthermore, future work includes developing a model to identify different autonomous systems operating the drone. In this case, different autonomous systems would be used to operate the same type of drone. Again, data will be collected, and features will be identified post statistical analysis of the data collected. Finally, develop a model to use those features to identify the specific autonomous system being used. In addition, future work will include more advanced capabilities to the mode where a human pilots the drone. Such capabilities include collision avoidance. Other future work includes using different drone models with different payloads to compare the machine learning model's performance when using heavier drones and heavier payloads

with lighter ones. There are several conditions to test such a model against in future work. One important condition is to see how altitude and distance from the acoustic sensor affect the model's performance.

Bibliography

[1] U. Nehmzow, *Robot behaviour: design, description, analysis and modelling*, Springer Science & Business Media (2008).

[2] M. Deruyck, J. Wyckmans, L. Martens, and W. Joseph. *Emergency ad-hoc networks by using drone mounted base stations for a disaster scenario*. In *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–7, (2016).

[3] S. Raghavan, *Saudis say oil pipeline was attacked by drones, possibly from Yemen*, The Washington Post (2019).

[4] J. Zitser, *A rogue killer drone'hunted down'a human target without being instructed to, UN report says*, Business Insider **30** (2021).

[5] R. News. *Drone leads to controlled airspace closure at Wellington Airport*, (2018). Available at `https://www.rnz.co.nz/news/national/375671/drone-leads-to-controlled-airspace-closure-at-wellington-airport`.

[6] L. Mikelionis. *Drug cartels using drones to smuggle drugs at border*, (2018). Available at `https://www.foxnews.com/us/drug-cartels-using-drones-to-smuggle-drugs-at-border`.

[7] Darryndavis. *Correctional officials raise concern over drones smuggling contraband into Kingston-area prisons*, (2019). Available at `https://globalnews.ca/news/5074018/drones-smuggling-contraband-into-prisons-kingston/`.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg (2006).

[9] A. Traboulsi and M. Barbeau. *Recognition of drone formation intentions using supervised machine learning.* In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 408–411. IEEE, (2019).

[10] A. Traboulsi and M. Barbeau. *Identification of Drone Payload Using Mel-Frequency Cepstral Coefficients and LSTM Neural Networks.* In *Proceedings of the Future Technologies Conference*, pages 402–412. Springer, (2020).

[11] A. Traboulsi and M. Barbeau. *A Reverse Turing Like Test for Quad-copters.* In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 351–358. IEEE, (2021).

[12] M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, *RF-based drone detection and identification using deep learning approaches: An initiative towards a large open source drone database*, Future Generation Computer Systems **100**, 86–97 (2019).

[13] X. Shi, C. Yang, W. Xie, C. Liang, Z. Shi, and J. Chen, *Anti-drone system with multiple surveillance technologies: Architecture, implementation, and challenges*, IEEE Communications Magazine **56**, 68–74 (2018).

[14] G. Ding, Q. Wu, L. Zhang, Y. Lin, T. A. Tsiftsis, and Y.-D. Yao, *An amateur drone surveillance system based on the cognitive Internet of Things*, IEEE Communications Magazine **56**, 29–35 (2018).

[15] L. Shi, I. Ahmad, Y. He, and K. Chang, *Hidden Markov model based drone sound recognition using MFCC technique in practical noisy environments*, Journal of Communications and Networks **20**, 509–518 (2018).

[16] M. Z. Anwar, Z. Kaleem, and A. Jamalipour, *Machine learning inspired sound-based amateur drone detection for public safety applications*, IEEE Transactions on Vehicular Technology **68**, 2526–2534 (2019).

[17] J. M. de Wit, R. Harmanny, and G. Premel-Cabic. *Micro-Doppler analysis of small UAVs.* In *2012 9th European Radar Conference*, pages 210–213. IEEE, (2012).

[18] F. Fioranelli, M. Ritchie, H. Griffiths, and H. Borrion, *Classification of loaded/unloaded micro-drones using multistatic radar*, electronics letters **51**, 1813–1815 (2015).

[19] R. Barták and M. Vomlelová. *Using machine learning to identify activities of a flying drone from sensor readings.* In *The Thirtieth International Flairs Conference*, (2017).

[20] A. Castellini, G. Beltrame, M. Bicego, J. Blum, M. Denitto, and A. Farinelli. *Unsupervised activity recognition for autonomous water drones.* In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 840–842, (2018).

[21] S. Baez, *Predicting opponent team activity in a RoboCup environment*, arXiv preprint arXiv:1503.01446 (2015).

[22] F. W. Trevizan and M. M. Veloso. *Learning opponent's strategies in the RoboCup small size league.* In *Proc. AAMAS*, volume 10. Citeseer, (2010).

[23] B. Nassi, R. Ben-Netanel, A. Shamir, and Y. Elovici, *Game of drones-detecting streamed POI from encrypted FPV channel*, arXiv preprint arXiv:1801.03074 (2018).

[24] L. A. Young. *Feasibility of Turing-Style Tests for Autonomous Aerial Vehicle 'Intelligence'.* In *AHS International Specialists' Meeting on Unmanned Rotorcraft, Chandler, AZ*, (2007).

[25] J. Hernández-Orallo, *Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement*, Artificial Intelligence Review **48**, 397–447 (2017).

[26] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford. *CAPTCHA: Using hard AI problems for security.* In *International conference on the theory and applications of cryptographic techniques*, pages 294–311. Springer, (2003).

[27] E. R. Vimina and A. U. Areekal. *Telling computers and humans apart automatically using activity recognition.* In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 4906–4909. IEEE, (2009).

[28] S. F. Kalik and D. V. Prokhorov. *Automotive Turing test.* In *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, pages 152–158, (2007).

[29] G. Kochanski, D. Lopresti, and C. Shih, *A reverse Turing test using speech*, (2002).

[30] J. Shao, A. Uchendu, and D. Lee. *A reverse Turing test for detecting machine-made texts.* In *Proceedings of the 10th ACM Conference on Web Science*, pages 275–279, (2019).

[31] C. C. Insaurralde and D. L. Lane, *Metric assessment of autonomous capabilities in unmanned maritime vehicles*, Engineering Applications of Artificial Intelligence **30**, 41–48 (2014).

[32] C. C. Insaurralde and D. M. Lane. *Autonomy-assessment criteria for underwater vehicles.* In *2012 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 1–8. IEEE, (2012).

[33] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, *Sensor-based activity recognition*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42**, 790–808 (2012).

[34] J. Bruce, M. Bowling, B. Browning, and M. Veloso. *Multi-robot team response to a multi-robot opponent team.* In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 2281–2286. IEEE, (2003).

[35] S. Writers. *LDS unveils SpectroDrone to find explosives*, (2016). Copyright - Copyright (c) by United Press International. All Rights Reserved; Last updated - 2016-11-17.

[36] H. Elkholy. *Dynamic Modeling and Control of a Quadrotor Using Linearand Nonlinear Approaches.* Master's thesis, (2014). American University in Cairo.

[37] T. Lee, M. Leok, and N. H. McClamroch. *Geometric tracking control of a quadrotor UAV on SE (3).* In *49th IEEE conference on decision and control (CDC)*, pages 5420–5425. IEEE, (2010).

[38] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. *RotorS—A modular gazebo MAV simulator framework.* In *Robot operating system (ROS)*, pages 595–625. Springer, (2016).

[39] S. M. LaValle, *Planning algorithms*, Cambridge university press (2006).

[40] A. H. Kai M. Wurm. Octomap, 3D mapping framework, available at `https://octomap.github.io/`.

[41] T. Baumann. *Obstacle Avoidance for Drones Using a 3DVFH* Algorithm.* Master's thesis, (2018). Swiss Federal Institute of Technology in Zurich.

[42] J. Borenstein, Y. Koren, et al., *The vector field histogram-fast obstacle avoidance for mobile robots*, IEEE transactions on robotics and automation **7**, 278–288 (1991).

[43] I. Ulrich and J. Borenstein. *VFH+: Reliable obstacle avoidance for fast mobile robots.* In *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*, volume 2, pages 1572–1577. IEEE, (1998).

[44] I. Ulrich and J. Borenstein. *VFH/sup*: Local obstacle avoidance with look-ahead verification.* In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 3, pages 2505–2511. IEEE, (2000).

[45] S. Vanneste, B. Bellekens, and M. Weyn. *3DVFH+: Real-Time Three-Dimensional Obstacle Avoidance Using an Octomap.* In *MORSE@ STAF*, pages 91–102, (2014).

[46] J. Brownlee. *Parametric and Nonparametric machine learning algorithms*, (2020). Available at `https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms`.

[47] N. Shukla and K. Fricklas, *Machine learning with TensorFlow*, Manning Shelter Island, Ny (2018).

[48] *Softmax regression.* Available at `http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression`.

[49] *LSTM architecture Matlab.* Available at `https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html`.

[50] D. Luo, W. Xu, S. Wu, and Y. Ma. *UAV formation flight control and formation switch strategy.* In *2013 8th International Conference on Computer Science & Education*, pages 264–269. IEEE, (2013).

[51] M. Barbeau, *Recognizing Drone Swarm Activities: Classical versus Quantum Machine Learning*, Digitale Welt **3**, 45–50 (2019).

[52] *ROS-Indigo-Wiki.* Available at `http://wiki.ros.org/indigo`.

[53] *ardrone_autonomy.* Available at `https://ardrone-autonomy.readthedocs.io/en/latest/`.

[54] *TensorFlow.* Available at `https://www.tensorflow.org/`.

[55] *Simulating drones formations transitions.* Available at `https://www.youtube.com/playlist?list=PLdnNs8O76_ExxyDjz0KbuH0ZvLQE4Z8uh`.

[56] M. Kuhn and K. Johnson, *Feature engineering and selection: A practical approach for predictive models*, CRC Press (2019).

[57] R. B. d'Agostino, *An omnibus test of normality for moderate and large size samples*, Biometrika **58**, 341–348 (1971).

[58] E. S. Pearson, R. B. D "'AGOSTINO, and K. O. Bowman, *Tests for departure from normality: Comparison of powers*, Biometrika **64**, 231–246 (1977).

[59] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press (2016).

[60] I. H. Witten, E. Frank, and M. A. Hall, *Data mining: practical machine learning tools and techniques*, Elsevier (2011).

[61] R. R. Bouckaert and E. Frank. *Evaluating the replicability of significance tests for comparing learning algorithms.* In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 3–12. Springer, (2004).

[62] B.-H. Sheu, C.-C. Chiu, W.-T. Lu, C.-I. Huang, and W.-P. Chen, *Development of UAV tracing and coordinate detection method using a dual-axis rotary platform for an anti-UAV system*, Applied Sciences **9**, 2583 (2019).

[63] N. Intaratep, W. N. Alexander, W. J. Devenport, S. M. Grace, and A. Dropkin. *Experimental study of quadcopter acoustics and performance at static thrust conditions.* In *22nd AIAA/CEAS Aeroacoustics Conference*, page 2873, (2016).

[64] *Mel Spectrogram MATLAB.* Available at `https://www.mathworks.com/help/audio/ref/melspectrogram.html#mw_6cd51158-d7d7-4ee4-b11c-270ff5a4775e`.

[65] *Crypto - MFCC.* Available at `http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/`.

[66] P. Nair. *The dummy's guide to MFCC*, (2018). Available at `https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd`.

[67] *Mel scale*, (2019). Available at `https://en.wikipedia.org/wiki/Mel_scale`.

[68] *MFCC Matlab.* Available at `https://www.mathworks.com/help/audio/ref/mfcc.html`.

[69] *PX4/PX4-Avoidance.* Available at `https://github.com/PX4/PX4-Avoidance`.

[70] A. Traboulsi. *Autopilot Simulations using PX4, RVis.*, (2021). Simulation video available at `https://youtu.be/M_Rrnh3G9mQ`.

[71] J. Davis and M. Goadrich. *The relationship between Precision-Recall and ROC curves.* In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, (2006).