Autonomous Landing of an Unmanned Aerial Vehicle

Siddhy Ganesh Shetty

Delft University of Technology

TU Delft

Autonomous Landing of an Unmanned Aerial Vehicle

by

Siddhy Ganesh Shetty

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 28 October 2022.

Student Number:	5354692	
Project Duration:	December 15, 2021 – October 28, 2022	
Thesis Committee:	Dr. R. T. Rajan,	TU Delft, Supervisor
	Dr. M. Mohammadkarimi,	TU Delft, Daily Supervisor
	Dr. Ir. C. J. M. Verhoeven,	TU Delft, External Supervisor
	Dr. M. Larsen,	Anywi, External Supervisor

[1]The cover photo is by Andreas Dress on Unsplash.

Preface

This thesis is part of completing the Master Program Electrical Engineering at the Delft University of Technology. It is done under the European Union project Airborne Data Collection on Resilient System Architectures. The aim of the project is to increase the probability of safe landing of unmanned aerial vehicle. Taking inspiration from insects on their safe landing, my focus was to use low-complexity algorithms to provide safe landing on an unknown elevated platform.

Drones are an upcoming field of vehicles that is yet to be utilized to the best of its potential. Looking at the experiences my friends have had on landing their drones in GPS-denied environments, it made me curious to research more about it. With my interest in combining machine learning algorithms and control techniques, it was a fun challenge for implementing and increasing my knowledge on autonomous landing.

I am indebted to all the people who have helped me maneuver this journey while seeing to it that I do not crash. I want to thank my main supervisor, Dr. Raj T. Rajan, for his constant support and crucial guidance through the trouble-shooting months. Furthermore, I want to thank my daily supervisor, Dr. Mostafa Mohammadkarimi, for the numerous discussion and feedback sessions to find innovative solutions for the problem.

Lastly, I would like to dedicate this to my mother and my father for being my constant reminder to keep trying my best. To my friends for making this an unforgettable experience.

Siddhy Ganesh Shetty
Delft, October 2022

Abstract

With the advancement in technology, Unmanned Aerial Vehicles have been able to safely maneuver in risky environments and also limited resources. During landing, the UAV should be able to slow down while not affecting its physical design. Hence, it is important to develop highly accurate technique for landing given that there may be space and time constraints. Currently, multiple sensors are being used to increase accuracy which might weigh down some of the smaller UAVs. Furthermore, the increase in the number of sensors will also increase the cost of procurement and the maintenance requirement. But ensuring safe landing in situations of emergency landing and malfunctions in GPS-denied has been a lesser priority while designing UAV platforms. The usage of a single sensor to guarantee higher probability of safe landing is yet on its early stages of development and hence is further looked into. It was observed that insects use Optical Flow to maneuver and this has been used as a method to design UAV's journey and land safely. Therefore, this was used as the sensor in discussion to provide safe landing for UAVs.

Optical Flow is defined as a method of using image intensities to calculate the apparent motion of these image intensities. By dividing each frame, an Optical Flow Difference is calculated which is used as a metric to move the UAV towards the landing platform. Once the UAV has been able to position itself on top of the elevated landing platform, image dilation is used to safely land the UAV. Image dilation or flow divergence can be described as the ratio of the velocity to the height of the ground. One of the methods tested used Image Dilation Method using IMU value of the position and velocity of the UAV. Using the calculated image dilation, the control input to the UAV is generated and the UAV lands. This failed in providing safe landing and it also did not take the vision of the UAV into consideration. Then, Image Dilation Method using Features from Accelerated Segment Test (IDMF-AST) was implemented. This method tracks features observed by the UAV on the landing platform. Using these features, an estimate of the image dilation is calculated. To control the UAV, the estimate of image landing is used. This showed dependency on the landing design platform and the hyperparameters that are used for implementing IDMF.

Different landing designs were tested for different elevated landing platforms. It was observed that concentric circles on a textured landing marker were able to provide the highest probability of safe landing. To tackle the dependency of the hyperparameters, a Classification Model was proposed to find an optimal set of hyperparameters for individual assumed height of the landing platform. The trained model is implemented in the system which provides a set of hyperparameters for an assumed height of the platform, hence making the algorithm an Adaptive IDMF.

The Adaptive IDMF was tested against the original IDMF on the metrics of safe landing probability, time taken to safely land and simulation time. Adaptive IDMF is able to perform better compared to IDMF providing an 190% increase in probability of safe landing, faster safe landing and lesser simulation and compilation time.

Contents

List of Figures

List of Tables

1

Introduction

1.1. Unmanned Aerial Vehicles

Unmanned aerial vehicles (UAVs) are increasingly being used in various operations such as mapping [1], border patrol [2], emergency situations [3], delivery of goods [4], surveillance [5] and agriculture [6]. With the advancement in technology, UAVs have been able to safely maneuver in risky environments and also limited resources. Due to this, UAVs have been able to reduce the requirement of human pilots in risky environments. This increases the motivation to look more into how UAVs can be optimized and utilized to their best potential.

Figure 1.1: Types of UAVs [7]

There are two types of UAVs that have come to exist, fixed-wing and rotary-wing. Fixed wing requires a more complex methodology for landing like a net based recovery as these UAVs require a runway for take off and landing. The advantage of using fixed-wing UAVs is that they are much faster hence used for long distance flights. Furthermore, it is comparatively more stable and can withstand higher winds [8]. Some examples of these designs are aeroplanes/airplanes and gliders. On the other hand, rotary-wing UAVs are more maneuverable and are employed for scanning designated areas. Hence, vertical landing is possible for rotory-wing UAVs [9]. Under rotor-wing type of UAVs, there are helicopters, monocopters and multi-rotors as seen in Figure 1.1. Due to the multiple rotors, the mechanical complexity increases thus reducing the speed and flight range. Nevertheless, Rotor-wing UAVs are able to perform much better in terms of safe landing. Due to the ease of maneuverability, the focus has now switched to rotor-wing UAVs, since fixed-wing UAVs have safety issues due to their belly landing [10].

1.1.1. Autonomous Flight of an UAV

The journey of an UAV's flight can be split into take-off, navigation and landing. Navigation algorithms can either prioritize obstacle avoidance or finding the optimal path in a given environment. Based on this, the metric used for comparison between techniques also varies. But a major factor that affects these algorithms is the environment that the UAV is tested in. Implementing algorithms that are not tuned for the testing environment can lead to crashes and physical impact to the UAV. Thus, it is important to choose the appropriate navigation algorithm. Due to the development of faster processors, pose estimation and path planning, which are used for accurate navigation, have also benefited. This has also improved the communication between the UAV and ground station when there are sensors such as GPS or laser involved [11].

Most researchers are utilizing the fast processors to implement machine learning algorithms for path planning such as deep learning [12], reinforcement learning [13] and model predictive control methods [14]. The main challenge for navigation is accurate localization of the UAV. The main methods of navigation are divided into inertial navigation, satellite navigation, and vision-based navigation[15]. Inertial navigation focus on Inertial Measurement Units (IMU) which sum over time the measurements to provide the position and velocity of the UAV. One of the shortcomings of using IMUs, if the sensor has inherent noisy measurements, the error keeps increasing. Therefore, reduces the accuracy of the values used. To overcome the shortcomings, some researchers use Simultaneous Localization and Mapping (SLAM). SLAM works by using a map to localize the UAV and while generating the map with the help of local information available. With the implementation of faster processors, cheaper and flexible sensors, research on vision-based navigation has become more popular.

Vision-based UAV navigation can be implemented in various stages of the UAV's journey. These being visual localization, obstacle detection and avoidance and path planning. But there are multiple sensors that can be utilized such as monocular cameras, stereo cameras, RGB-D cameras and fisheye cameras. Deciding which sensor is better suited depends on the environment and cost constraint. There is research being done on how these sensors function in constrained environments which are GPS-denied or dimly-lit. Thus, finding an appropriate sensor to maneuver towards the target while avoiding obstacles is key.

1.1.2. Autonomous Landing of an UAV

Landing is a riskier process compared to the other processes due to the requirement of high precision while slowing down the UAV. If the landing method can not guarantee safety with high accuracy, the UAV can crash, impacting its hardware design. Therefore, it is important to develop highly accurate techniques for landing given that there may be space and time constraints along with the mechanical constraints [16]. There are multiple factors that can affect the safety of UAV landing, namely, the environment, the type of landing zone and the sensors used for detecting and landing on the landing marker/zone. It is important for these factors to be taken into consideration before implementing an landing algorithm in a real-life situation. Based on these constraints, it is crucial to simulate the behaviour of the UAV virtually to envision what possible shortcomings exist.

The current research with respect to autonomous landing of UAVs is geared towards on optimizing the vision-based landing with the help of cameras and comparing these with conventional control-based control algorithms. This has increased with the development of accurate and fast image processing algorithms and also the reduced cost of cameras. Furthermore with the increase in Artificial Intelligence, these robust algorithms have been implemented also in the arena of UAV manuevering and landing. There is also an interest in improving precision for moving target or unknown location of target [17] given some constraints in the environment such as windy oceans [18] or GPS-denied environments. Some real-world driven research has been testing UAV landing with the help of camera vision to landing on battery recharging docks or for detecting a safe zone in mountainous or flooded areas. [19]

Few of the current challenges for autonomous landing include the reduced precision due to navigation error, which is fairly significant [20]. Currently, multiple sensors are being used to increase accuracy which might weigh down some of the much smaller UAVs. Furthermore, the increase in the number of sensors will increase the cost of procurement and the maintenance requirement. It might also generally lead to an increase of processing time which can particularly affect the accuracy of safety provided while landing. Hence, finding a balance between the number of sensors and guaranteed safe landing is important with the increased usage of UAVs.

1.2. Problem Statement

With the advancement of technology, the development of autonomous Unmanned Aerial Vehicles (UAVs) has been geared to make UAVs operate in various environments as seen in Figure 1.2. There have been trends towards using multiple sensors for guaranteed safe landing on different types of landing platforms. But ensuring safe landing in situations of emergency landing and malfunctions in GPS-denied environments has been a lower priority while designing UAV platforms. The usage of a single sensor to guarantee higher probability of safe landing is yet on its early stages of development and can be further looked into.

The goal of this research is to design an algorithm for the safe landing of the UAV. To implement this, there are a certain constraints, given that the location of the platform is unknown. Furthermore to detect the landing platform, a single sensor is implemented.

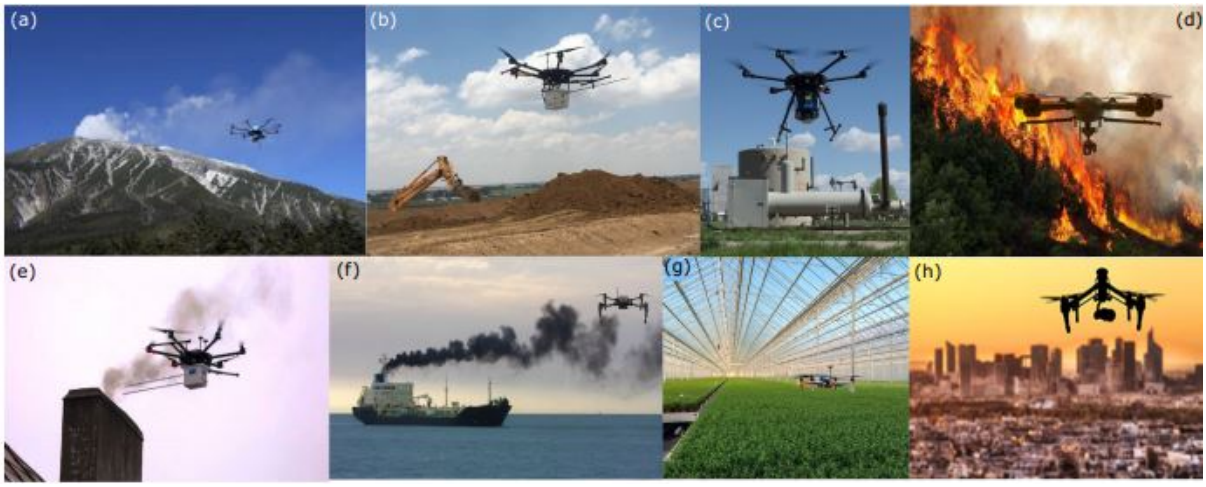


Figure 1.2: Applications of small drones. (a) volcanic research; (b) landfill emission monitoring; (c) poisonous gas emission monitoring in industrial sites; (d) fire detection; (e) residential emissions monitoring; (f) ship emission monitoring(g) precision agriculture; (h) urban air quality [21]

1.3. Thesis Layout

It is important to understand more about each process in the UAV's journey and how it affects the overall safety of the UAV. Thus, Chapter 2 will give an insight about the base algorithms existing and their extensions. It also addresses the source of inspiration that was used as the sensor and methodology for implementing autonomous landing for the UAV.

Chapter 3 will describe the system model and protocol that were used for observing and designing the algorithms to guarantee higher probability of safe landing for an UAV. It talks about how Moving towards the Landing Pad is implemented in simulation. It will explain what is Optical Flow (OF) and then go in-depth about how Optical Flow Difference (OFD) can be used to maneuver the UAV accordingly. It also addresses how image dilation was used to land on unknown height coordinates of the platform. There are two different methods that implement image dilation. The first method uses image dilation with initial values of the UAV's position (IDMI) and then second one implements image dilation using Features from Accelerated Segment Test (FAST) (IDMF-AST). These methods are simulated and tested for different unknown elevated platforms. The possible reasons for the failure of these methods are noted and the problem statement is stated.

In Chapter 4, the shortcomings of the previous methods implemented for safe landing are tackled. First, the design of the landing pad is discussed and how it affects the probability of safe landing. Using this information, an ideal landing design was finalised. It was also observed how the hyperparameters used in FAST with image dilation affected the probability of safe landing. The first attempt was to find an optimal set of parameters for all possible heights of the platform, which failed. Next, there was an attempt to find a set of hyperparameters that that could guarantee safe landing for a smaller range of heights using curve fitting. This method did not give the required results. The last method that was used to find optimal hyperparameters for each possible height of the platform using a Classification Model. The implementation of the model into the system is described in this Chapter.The comparison of the Adaptive IDMF versus the IDMF is then looked into. This uses three metrics, probability of safe landing, time taken for safe landing and time taken for compilation and simulation. Then in Chapter 5, the work is summarized and future work is addressed.

2

Literature Review

To tackle the problem of autonomous landing, it was divided into four parts to understand better the state of technologies used. These are depicted in Figures 2.1 with the primary types of methods available. These divisions were made keeping in mind how each of these categories affect the overall aim of safe journey of an UAV. Individually dividing into these categories were important to understand better how each of these categories could also be combined to improve the behaviour of the UAV.

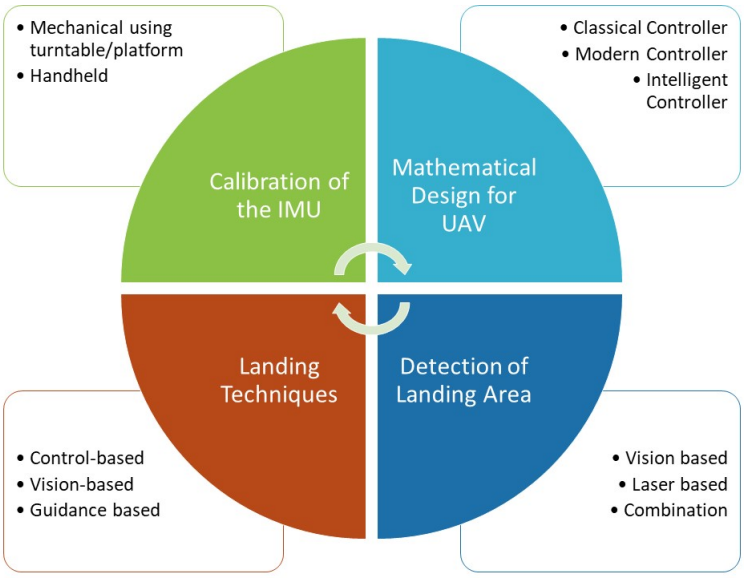


Figure 2.1: Aspects That Affect the UAV's Performance

As stated in Section 1.1.1, accurate localization of the UAV is an important factor for a safe journey of an UAV. An Inertial Measurement Unit (IMU) is an electronic device that uses accelerometers and gyroscopes to measure acceleration and rotation, which can be used to provide position data. If this device is not calibrated properly, there is a possibility of crash. Due to the movement of the UAV and noise that might creep in [22], the error in the measurement keeps increasing over time. Thus, reduces the accuracy which makes IMU unreliable in short runs. Hence, calibration is an important factor for safe landing. After having the necessary calibration done to the sensors, it is important to model and test how the UAV would work in real life. Directly testing on a physical UAV would lead to an expensive and slower process of development. In the research presented by J. Arrigo et al.,[23] the issues of inaccurate hardware implementation was stated. During testing, there were some hardware failures that lead to delays as the replacement had to be imported. Thus, showing how important it is for an accurate modelling and fail-proof algorithms implemented while virtual simulation. This is tackled in the Mathematical Design

for UAV which will also talk about different control algorithms that are implemented for the UAV's motion. Knowing that the simulation is able to mimic the UAV and the control algorithm is robust, then the different methods that are used for detection for an UAV are addressed. For the UAV to be able to detect precisely, there are sensors that need to be attached to the UAV. This is addressed under Detection of the Landing Zone. Once the zone is detected, the path taken by the UAV can be designed by different algorithms. Some methods combine the same sensors used for detection to design the landing path as well. This shows how all these parts are dependent for the safe journey of an UAV.

2.1. Calibration of IMU

Most of the existing research splits into either the usage of a mechanical platform or a hand-held procedure for calibration. For example, G. Panahandeh et al [20], presents an approach for calibrating the accelerometer triad without any mechanical intervention. Instead, the IMU is rotated by hand. Their method uses maximum likelihood estimator to estimate the unknown parameters for calibration and also the orientation of the IMU. The Cramer-Rao bound is derived which is used to compare the values estimated by maximum likelihood estimator. The results show that the Cramer-Rao bound is attained by the mean square error in a few iterations of the simulation. Thus, showing that this method is able to deliver without the need of a mechanical turntable.

Since this paper only focused on the accelerometer triad and would require another set of experimentation for calibrating the gyroscopes, research by Z. Ding et al. [24] was looked into. This paper used Kalman Filter for calibrating both accelerometer and the gyroscope triads. For accelerometer, static positions were defined at which the IMU was placed by hand and for gyroscope, the IMU was rotated along x, y, z directions in both clockwise and anti-clockwise. This was done while maintaining that the rotation angle is large enough to observe the errors in the parameters and the rotation speed was within the permissible range for the gyroscopes. Another paper by D. Tedaldi et al. [25] used static positions (attitudes) for the accelerometers triad for calibration. The measurement samples were then utilized to calibrate the gyroscope. These two methods show how calibration for both accelerometer and the gyroscope can be achieved using static positions.

Based on the infrastructure and budget available, decision can be made if mechanical or hand-held calibration can be done. Since the performance of the calibration is also highly dependent on the type of sensor being used, the error may also vary. There is a further possibility of a difference in its behaviour, given there might be a variation in the testing environment of these sensors. Therefore, it is important to find a method that fits perfectly given the sensor.

2.2. Mathematical Design for UAV

In the article by B. Erginer and E. Altug in [26], a classical controller (PD) plus vision based controller was implemented for the controller design of the UAV. In this research, PD helped in attaining the feature coordinates and the vision controller helped in noticing the landing pad to get the required values (pose estimation of the drone) that is fed as desired to the controller. Further more, a decoupled dynamical model was implemented to mimic the behavior of the UAV. This made the modelling not realistic as it was assumed that change in motion along roll, pitch and yaw angles do not affect each other. Noting that this was an older research, an extension was developed. This used Tilt Integral Derivative instead of PD controller. The two controllers were then compared for trajectory tracking. It was observed how TID with a prefilter was able to perform better compared to PD controller.

Some more complex algorithms like neural networks and adaptive neural networks were implemented for better control of the UAV. In the article published by S. Suresh et al. in [27], a direct adaptive neural control was implemented which helped in modelling the uncertainties into the system. But a linearized model of the UAV was designed, thereby limiting the accuracy when compared to a real world implementation. An extension of this research, 6 DOF non-linear model is designed and the stability is analyzed. The backstepping controller is used instead of the two-timescale controller design implemented earlier [27]. This controller when simulated, showed that the tracking error converges to a compact set. To make the controller robust to aerodynamical errors, an adaptive controller based on neural networks is implemented[28]. These algorithms were able to provide high precision results but needed the neural networks to be tuned accurately for these results.

L. Besnard et al. [10], designed a sliding mode controller which is driven by a sliding mode disturbance observer. External disturbances and model uncertainties were implemented to check the robustness of the controller. Moreover, rotor failure was simulated to check if the stability of the UAV is affected. The controller was able to stabilize the vehicle given the uncertainties and failures while remaining within the physical limits. Hence showing that it is possible to have crash-proof algorithms in the case of rotor failures. But these were not tested in the real world and the UAV model was based specifically on the X-4 flyer hence reducing its generality.

The key take-away is that the UAV model that it is used to check the controller design might be too simplified. This reduces the accuracy of how the UAV behave in reality. Therefore, it is important to design a controller keeping in mind that the UAV model is simplified. Furthermore, choosing a good controller will be dependent on the information known about the aerodynamical and modelling errors and how robust the system is preferred to be while not being too energy-extensive.

2.3. Detection of the landing zone

For detection, there are mainly three types of sensors that are used. It is either vision based (using cameras), laser beams (LiDar and RADAR) and a combination or vision and laser beams.

Instances where onboard cameras were installed, April Tag [29] or ArUco Tags [30] were commonly used for detection of the landing area. In the research published by J. Wubben et al. [31], a drone is simulated to be in the range of markers using GPS. It has real world tests for two situations: the first one being a single main ArUco marker and the second being a bigger ArUco marker that leads to the final target of a smaller ArUco marker. This research showed a lesser offset in landing compared to a GPS based systems. However it was observed to have issues in the second type of simulation if there was wind added in the simulation environment.

Figure 2.2: Usage of April Tag for UAV Landing [32]

A. Borowczyk et al. [17], also used an onboard camera along with a mobile phone at the ground vehicle for their implementation. This research used Kalman Filter for relative position and velocity estimation. For detecting the landing target, the AprilTag was used. This methodology was simulated and had real world implementations as well but this was in a controlled environment. This implies that the behaviour of the system is not accounted for if there was any turbulence in the ground vehicle. Furthermore, it also had multiple sensors that needed to work for a successful run. In the research conducted by A. Paris et al. [18], it also implements April Tag for detecting the landing marker. It is noted that the deep-learning (Model Predictive Control) approach had more advantages compared to model-based control to estimate the wind's effects. Thus, it is observed that vision based markers have been able to guarantee safe landing not only in a static marker environment but also in a moving platform along with wind turbulence.

D. Maturana et al. [33], developed a detection algorithm which worked by checking for free and occupied space with the help of volumetric occupancy map. The map was combined with a 3D CNN and tested for an environment with a vegetated terrain. Thus, the drone had to detect safe landing zones with the help of the LiDAR point clouds. The results were comparable with usage of computer vision and showed that the vision is a replaceable feature for safe landing. Y. Shinet al. [34], stated how having the low cost radars on the UAV was able to perform irrespective of the lighting of the environment. Furthermore, this research was able to detect multiple targets along with slope estimation and roughness estimation with lower error compared to the least square method. A downfall of this method is that multiple sensors are required and these might not be easy to attach on smaller UAVs.

Some research combined camera vision along with laser to give better results. J. Kimet et al. [35], had LiDar and an omni-directional camera mounted on the unmanned ground vehicle instead of the UAV. First the UAV reaches within detection range of LiDAR with the help of GPS way point. This method then predicts the location that the ground vehicle will be at and checks if the drone will be within the reachable

boundary. A velocity command is given to the UAV so that it can land safely on the ground vehicle. Compared to conventional landing methods, this proposed method enhanced the overall performance of the UAV.

In the research published by J. Y. Lee et al. [36], the main aim was to reduce the dependency on GPS navigation for drones in urban areas. To ensure this, the problem was divided into two subsystems. The first subsystem, which required to maneuver the UAV close to the landing area, using a laser-guided mechanism. The second subsystem used optical flow generated by the camera vision to guarantee safe landing while avoiding obstacles. By doing so, the dependency of GPS was reduced and showed there are alternative methods that are not very complex.

M. Alijani and A. Osman implemented autonomous landing taking into consideration a mathematical approach [37]. Two stages were used to resolve the landing problem, with the help of both control and computer vision algorithms. YOLOv3 (You Only Look Once, Version 3) was used for target detection in the first stage and by estimating the motion of objects, tracing is done in the second stage. This paper also used the estimation of the relative position of the object as an input for PID controller in the quadcopter. The results were not sufficient to decrease the position measurement and estimation errors hence not competent enough in cases of sensor errors.

Furthermore, M. S. Alam and J. Oluoch provided a description about the types of landing zones that are used and compared how they vary if static or a dynamic landing demarcation is implemented [38]. This paper also listed a detailed comparison of the methodology existing for landing zone detection for a UAV. Hence it is observed from the different research done on detection of landing zones, the type of method used depends heavily on the type of environment the UAV is designed to be tested in.

2.4. Landing Algorithm of the UAV

Once the landing zone is detected, the last part of the UAV's journey is to safely land. The landing algorithm should be robust, keeping in mind the physical constraints of the drone and any turbulent conditions.

A comparison of different landing algorithms were included by the survey done by A. Gautum et al. [16]. The primary classification included the division into control based landing techniques, vision-based techniques and guidance based techniques. Control based landing consists of the conventional control algorithms such as linear, non-linear, intelligent and robust control techniques.

Some key takeways with respect to control based methods were:

- PID may not be the best method when dynamics are available as it does not consider the non-linear nature of the model architecture. For better results, vision based control can be combined with PID control.
- Non-linear control has three types, feedback linearization control, sliding mode control and backstepping control. A shortcoming of feedback linearization is that it might not guarantee stability due to the use of second and third order derivatives. Sliding mode controller has general trend of chattering and high control demand which needs to be considered while finding the optimal parameters. Backstepping controller is a recursive method aimed mostly in making the UAV along a desired trajectory but it requires information about all system states.
- With respect to Intelligent control, the best method was to combine fuzzy logic with PID for better results. For implementing a neural network, prior information is needed for proper tuning thereby increasing the complexity.
- Robust control is not the best suited method for real life implementation as it gives priority to robustness for disturbances but does not take into consideration in the energy expended.

One needs to keep in mind that the working of control based techniques is dependent on good GPS accuracy. But if this can not be guaranteed, the other methods such as vision and guidance based techniques come into play. In computer vision based algorithms, there have been various research done for reducing the error margins such as using neural networks or even using color based markers to recognize the landing zone. But both these came with their own disadvantages such as implementation of

a neural network for training makes the computations complex and color based markings pose high risk of collision with other objects [37]. Thus, vision based techniques are combined with the earlier stated control based techniques to improve safe landing accuracy. Some examples of vision based techniques have been discussed in the Section 2.3 [36].

With respect to guidance based techniques, most methods come under pursuit guidance, time to go polynomial guidance or proportional navigation guidance. Pursuit guidance is based on having the velocity vector of the UAV always directed towards the target. In M. Kim et al. [39], two different guidance laws are proposed. The pursuit guidance law helps in maintaining the waypoint constraint. Once the UAV moves beyond the waypoint, the second guidance law kicks in. Distance error dynamics-based guidance law is then used to hover around the target. Unfortunately. this was not implemented in the real life and is an important test given that this has multiple UAVs that need to maintain distance from the leader.

B. M. Min et al. [9], simulated vision based landing with the help of guidance law for both fixed wing and rotary wing drones. For both type of UAV models, constraints were put for the acceleration and change of acceleration to have robustness in the time-to-go polynomial function used for landing. This research implemented vision-based landing and was successful in providing safe landing in both cases. In the research by A. Borowczyk et al. [17], Proportional Navigation is used for navigating towards the landing area and PID controller is used for landing. In both these papers, it is noted how a combination of different techniques are used to maneuver and land accurately.

2.5. Source of Inspiration

In the previous section, possible methods that can be used individually and combined to guarantee safe journey of an UAV were addressed. Choosing the route to take would determine the constraints and the metric of comparison. A complex algorithm could be implemented that will provide better results but a better comparison would be to look into how the comparatively less complex methods work in the same situation.

To decide on the possible flow of the UAV's journey, a look on how this problem is resolved in the nature seemed insightful. It was observed how flying insects are capable of safely reaching their target without complex image processing algorithms in their brain. These insects are able to maneuver in different types of environments and detect a suitable landing area with the help of neurons inside their brains that are sensitive to light. The last eight decades have produced evidence that flying insects use optic flow (OF), which is produced in their eyes as a result of propulsion, to navigate around their habitats. The translational OF is the angular speed at which contrasting items in the environment pass past the animal in its frame of reference [40].

In the research published by N. Franceschini et al. [41], it is observed how insect-based maneuvering can be extended on micro UAVs to maneuver indoors or in complex environments such as urban areas or rocky hills. Furthermore, how control algorithms can be implemented without being dependent on GPS or on sensors such as RADAR and LiDAR. T. S. Collett in their paper found how Drosophila (Figure 2.3) picked a close target with the help of OF [42]. It is examined how insects around us are able to safely navigate and land while using less complex techniques hence paving a way for humans to try mimicking these methods for implementation on any moving vehicle, specifically UAVs.

Figure 2.3: Fruit Fly/ Drosophila

There are multiple articles that implement insect-inspired algorithms for easy implementation for maneuvering of UAVs [43] [44] [45]. The common factor between these research articles is how bio-inspired algorithms have been quite successful in guaranteeing safe landing. Thus, the usage of OF for maneuvering and detection of landing peaked my interest as it was shown multiple times how it was able to generate similar results to an image processing or multi-sensor implementation.

3

Maneuvering Towards the Landing Platform and Landing on the Platform

3.1. System Model

Figure 3.1 illustrates the considered scenario for maneuvering and landing of the UAV. The details about the system model for maneuvering of the UAV towards the landing platform and its landing on a platform of unknown height is provided in Table 3.1.

Figure 3.1: Top View of the Simulation Environment [46]

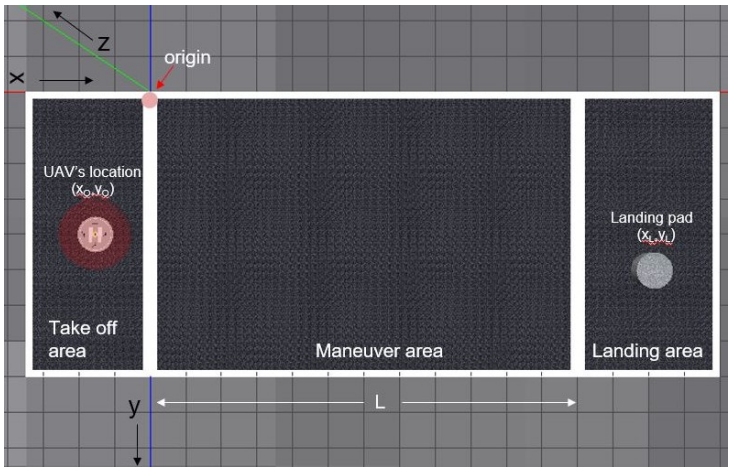


Table 3.1: System Model

1.	**Environment details:** The environment is textured and divided into three areas: take-off, maneuver and landing area. The landing platform has textured surfaces that help in identification.
2.	**Coordinates:** x, y, z axes are along red, blue and green respectively. The meeting point of the three axes is the origin as seen in Figure 3.1.
3	**Location of the UAV and the Landing Pad:** The landing pad is located at (x_L, y_L) and the radius of the landing pad is r meters. The landing area starts beyond L meters. The starting point of the drone is at (x_O, y_O).
4.	**UAV model:** A point based model of a quadcopter is used. It has two camera visions, one facing forward and one facing downward. These sensors are used to mimic Optical Flow sensors.
5.	**IMU on the UAV:** The UAV is aware of its position at all times, using the IMU.

While there are several protocols for landing on a platform with known height [33] [35], there are a few protocols for landing on an unknown height platform using OF sensor. In this thesis, we propose the following landing protocol stated in Algorithm 1.

Algorithm 1 : Protocol for Landing on a Platform with Unknown Height using OF

1. The UAV moves from the take-off area to the landing area by detecting the landing platform using OF.

2. When the UAV is close to the landing platform, the UAV ascends to a predefined height. To reduce the possibility of physical impact while ascension, the UAV should not cross the safety radius. The distance to the landing platform at which the UAV descends has to be more than the safety radius.

3. Using the OF values generated, the UAV is able to position itself closer to the landing platform. The UAV should maneuver such that it is positioned on top of the landing platform. This step is done while keeping its height (z) constant.

4. While keeping its x and y coordinate constant, the UAV will descend from its predefined height on top of the unknown elevated landing platform.

The step 2 of the proposed landing protocol (Algorithm 1) has the UAV ascending to a predefined height once it is close to the landing platform. This method is chosen over having the UAV elevated before it is close to the landing platform. The reason for this step is that the UAV can detect the landing platform better if it is at a lesser height than the landing platform. This is because the elevated platform is captured better while using OF. In Figure 3.2, this is observed as the OF values are smaller in the elevated UAV compared to that of a low-lying UAV.

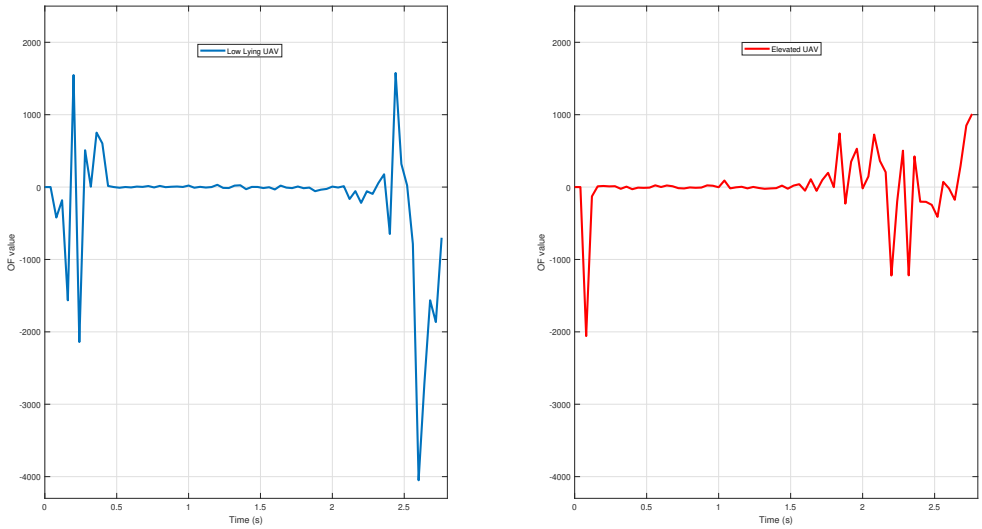


Figure 3.2: Comparison of the OF Values When the UAV starts From the Ground vs Elevated UAV

The known and unknown parameters during maneuvering and landing are given in Table 3.2. The values of parameters used for simulation in this thesis is also given in Table 3.2.

Variable	Simulation Value	Type of Data
Landing Platform Coordinates (x_L, y_L)	(14,5)	Unknown
Radius of the Landing Platform r	0.5m	Unknown
Height of Landing Platform	[0.2,2]	Unknown
Starting Point Coordinates (x_O, y_O)	(-1.5,4)	Known
Landing Area starts L	12m	Unknown
UAV's Linear Velocity	(5,2,0.05)	Known
UAV's Precision Velocity	(0.65, 0.3,0)	Known
Safety Radius	1m	Known

Table 3.2: Parameter Values

3.2. Software Used

The first step of implementing a safe journey for an UAV would be to finalize on the platform of testing. Some popular platforms/software that are used for vehicle simulations are namely ROS + Gazebo [47] [48] and MATLAB + Simulink [49] [50]. Initial testing showed that ROS was not able to provide with stable results. For this reason, the platform of simulation was switched to MATLAB + Simulink.

3.3. Maneuvering Towards the Landing Platform

Before the specifics of take-off and maneuvering are addressed, it is important to understand what optical flow (OF) is and how it can be used for decision making for the UAV.

3.3.1. Optical Flow

Optical flow or motion estimation is a method of using image intensities to calculate the apparent motion of these image intensities. An assumption commonly used while calculating optical flow is that the intensity of the pixel in consideration does not vary between the consecutive frames. Compared to the other vision-based methods, optical flow utilizes not just the trends observed in frames but also takes into account the time information while estimating [51] [52].

The image intensity I is denoted as a function of space (x, y) and time t. With the change of time δt, the space coordinates would have also changed of I to $(x + \delta x, y + \delta y)$ [53]. Since, the intensity is assumed to be constant between consecutive frames,

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{3.1}$$

Using Taylor Series Approximation, the right side of the Equation 3.1 can be written as

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \dots \tag{3.2}$$

Combining the two equations and ignoring the higher order terms, the simplified equation can be written as,

$$\begin{aligned} &\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0 \\ \rightarrow &\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t} = 0 \end{aligned} \tag{3.3}$$

Let $\frac{\delta x}{\delta t} = u$ and $\frac{\delta y}{\delta t} = v$. Thus, Equation 3.3 can be rewritten as,

$$\begin{aligned} \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} &= 0 \\ I_x u + I_y v + I_t &= 0 \end{aligned} \tag{3.4}$$

Where I_x, I_y, and I_t are the spatiotemporal image brightness derivatives, u is the horizontal optical flow and v is the vertical optical flow. To compute the optical flow (u,v), the constraint Equation 3.4 should be solved.

Since a vision sensor is used in the simulation, it is necessary to implement a optical flow estimation method to generate the required horizontal and vertical optical flow, u and v. There are different methods available for OF estimation such as the Lucas-Kanade method [54] and the Horn–Schunck method [55] and also the adapted versions such as Kanade-Lucas-Tomasi (KLT) feature matching algorithm [56].

Furthermore, MATLAB + Simulink has a Computer Vision Toolbox, which includes an Optical Flow block. This block takes an input of an image and gives the output of u and v based on the estimation method chosen (Lucas-Kanade method or the Horn–Schunck method). Both these methods do come with their advantages and disadvantages. Lucas-Kanade is better suited for smaller changes in motion, whereas Horn-Schunk method is a global method thus being less robust to noise. For this implementation, Lucas-Kanade method was chosen as the optical flow estimation technique as the object in consideration (the landing platform) is not a moving platform.

3.3.2. Using Optical Flow Difference for Maneuvering

It is observed that when the landing platform is detected, there is a change in the magnitude of the Optical Flow by a factor of around 1.5 times. This was also observed in the research published by C. S. Royden et al. [57]. This information is used to steer the UAV in the required direction. The image is divided into half, and the optical flow values are summed for each half. The side which has a higher value of optical flow, the UAV is steered in that direction.

For example, in Figure 3.3, one can notice the red lines on the elevated platform. The yellow line was added externally for the ease of demarcating the two sides. For generating the red lines of motion, the velocity values generated at each iteration is compared and then used to generate the coordinate points for the lines of motion. On calculating the sum of the velocity estimates for the left and right halves of the image, the value for the left side was 79278 whereas for the right side the value was 115542. Therefore, the difference can be calculated between the two halves, which is then used to maneuver the drone. This process is referred to as Optical Flow Difference. For this simulation, the difference has been calculated by subtracted the sum of the optical values of the right side from that of the left side.

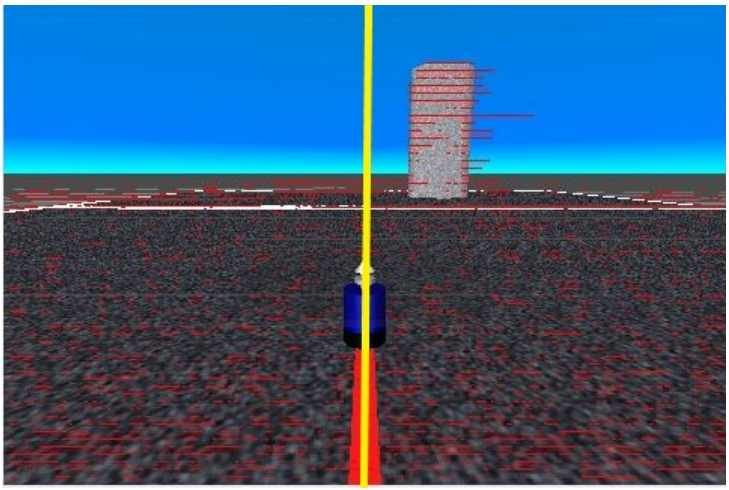


Figure 3.3: View from the Forward Camera of the Landing Pad

As stated in Algorithm 3.1, the UAV is assumed to be a point-based system. Thus, based on the optical flow difference, if the output from the sign block is positive, UAV moves towards the left side, i.e. V_y is negative. Also, if the output from the Sign block is negative, drone will move towards the right side, i.e, V_y is positive. These linear velocities were then integrated to generate the position of the UAV. Using a similar mechanism, the angular velocity is used to calculate the orientation of the drone. This mechanism is depicted in Figure 3.4.

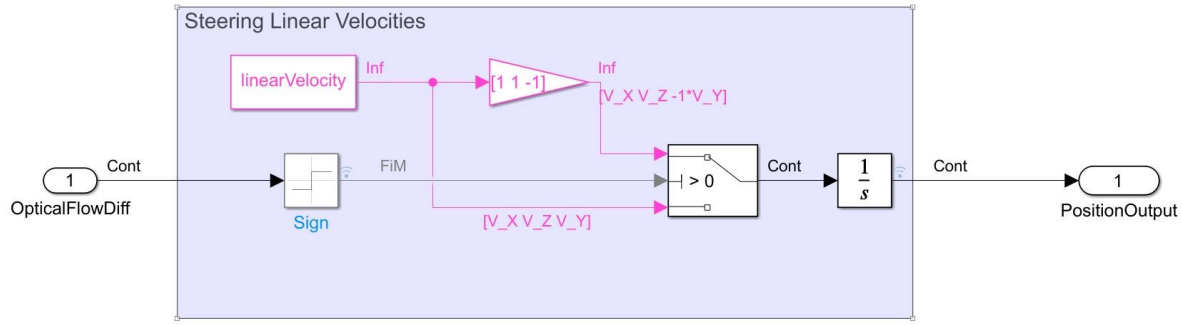
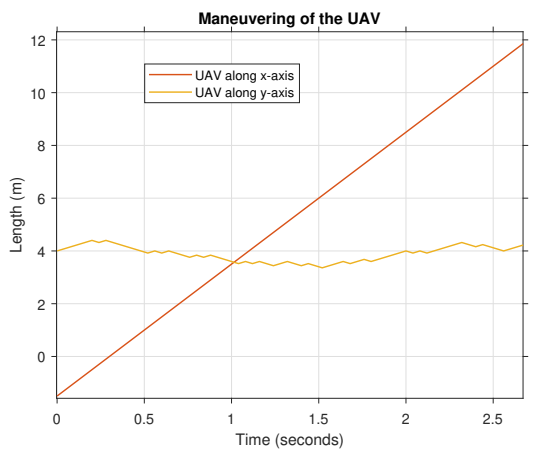


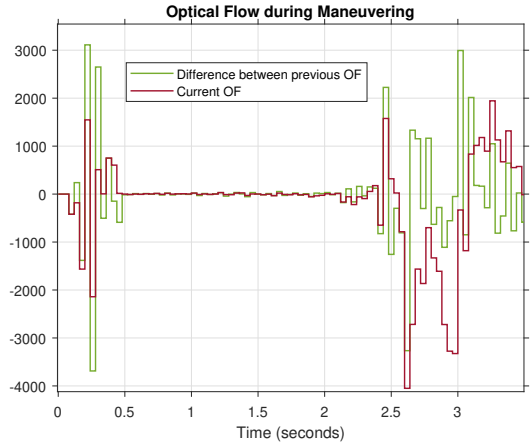
Figure 3.4: Block Diagram of Maneuver System

Once the landing platform is detected, the UAV needs switch to the next step of maneuvering on top of the landing platform. But this switch should occur once the UAV has come close to the landing platform and not when it first detects it. To ensure this, the optical flow values were observed to see how the output changes as the UAV comes closer to the landing platform. There was a constant increase in the OF values as the UAV inched closer to the platform.

To compare how much of a jump occurs, the difference between the previous value and the current OF value was compared. On multiple runs for different heights of the landing platform, it was noted that 1 meter before the landing platform the difference was constant with the magnitude around 2.6×10^6. Thus, it was decided to take this value as a threshold, such that when the UAV generates this difference, it will stop maneuvering forward and go to the next steps required. This threshold is dependent on a few parameters such as the starting location of the UAV, the location of the landing platform and also the texture used on it.



(a) Movement of the UAV using OF towards to Landing Platform



(b) Variation of the Optical Flow value as the UAV comes closer to Landing Platform

Figure 3.5: Behaviour of the UAV while Moving Towards the Landing Platform

As the UAV inches closer to the platform, there is a spike in the OF values, as seen in Figure 3.5. The OF values were divided by 1000 for the ease of observing the variation. When the UAV is around 5 meters away from the landing platform the spikes in OF value is observed. The initial spike within 0.5 seconds is due to the UAV crossing the white border that defined the take-off area. The UAV, till 2 seconds, tends to move away from the landing platform, showing that the landing platform was too far for the vision to detect it. Since the maneuver area is textured and not the surrounding area, the UAV is able to stay on the area and not move away from the environment. Thus, showing that the textured maneuver area is important for the UAV to be able to successfully maneuver across towards the landing area.

3.3.3. Ascension of UAV

Once the platform is detected, the next step is to increase the height of the UAV. This is done so that once the UAV has detected the landing platform, the UAV can observe the landing platform from the bottom facing camera as well. To do so, an integrator was used along with a difference calculator, as a reference tracker for the height of the UAV. The reference height (z_o) provided was kept as 2.5 meters. Since the landing platform is assumed to be at a maximum height of 2 meters (h_u), 2.5 (z_o) was kept as the height the UAV can be at to view the landing platform clearly. This is possible since it is assumed that the UAV is a VTOL and the IMU on the UAV is accurate as stated in Table 3.1.

3.3.4. Detection using Downward Facing Vision

There is a possibility that the UAV's downward vision does not have the landing platform in its vision. This is because the processes till now were dependent on a forward facing vision but after ascension the system needs to switch to a downward facing one. Thus as a fail-proof, using the downward vision, the UAV tries to maneuver towards the landing platform to ensure that the platform is insight. TO guarantee that the landing platform is in sight, the jump in OF value is observed as the UAV inches closer to the platform. Knowing that the difference in value once the landing platform is observed is around 1.4-1.5 times, the block was conditioned to implement till the optical flow value differed by this range. In Figure 3.6, it is observed as the UAV inches closer to the landing platform, there is a spike in the OF values. Thus, as soon as the OF value reaches more than 18800, it is assumed that the landing platform is now insight. To implement this, the linear velocity was reduced compared to the previous implementation as it is known that the landing platform is close by. The velocity used was *UAV's Precision Velocity* from Table 3.2.

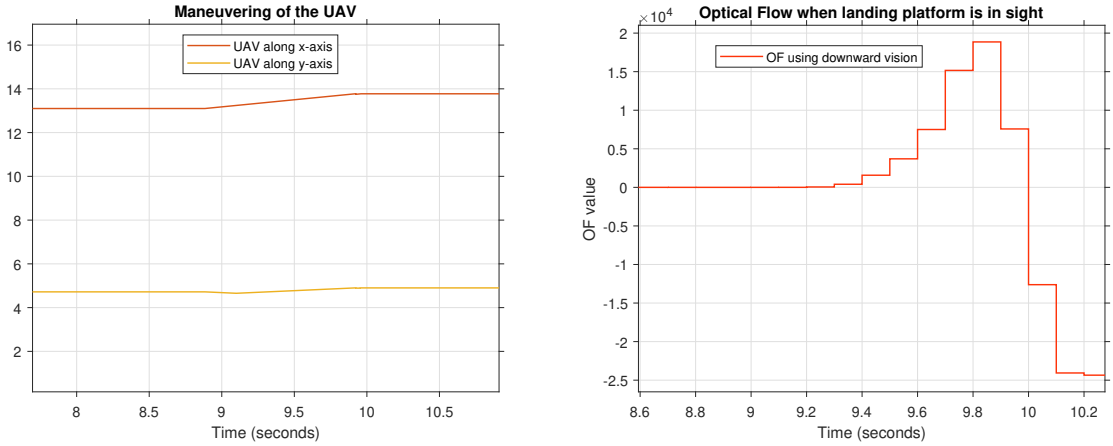


Figure 3.6: Using Downward Vision to Maneuver the UAV towards the Landing Platform

For precise maneuvering of the UAV, the possible movements that the UAV can implement included up and down along with the existing left and right. Thus, given the downward frame, there are two OF values generated, one along y-axis i.e, left or right and one along x-axis i.e, up or down. Using these values, the UAV can accordingly decide to move. In this implementation, it is assumed that the UAV can either move left, right, up or down but no diagonal movement occurs.

Figure 3.7 shows how the OF values are used to maneuver the UAV on top of the landing platform. The Steering Linear Velocity areas depict the usage of the OF value to generate the actual velocity of the UAV. If the OF value for left or right is greater than zero, that means the landing platform area is more on the left. Thus, the UAV should move more on the left. If the OF value for x-axis is greater than zero, that means there is more landing platform in front. Hence, the UAV should move forward. But to choose if the UAV should move in x-axis or y-axis, a comparison is done of the magnitude of the OF values. The value which is higher in magnitude translates to the UAV being further away along that axis. Using this, the UAV decides to either move along x or y axes.

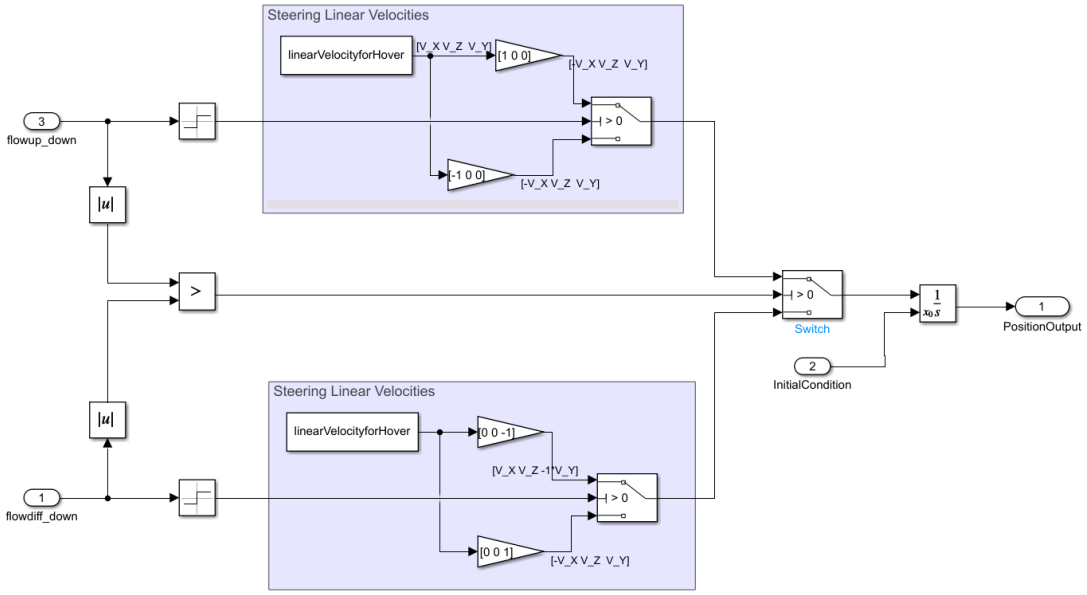


Figure 3.7: Block Diagram Depicting the Positioning Method

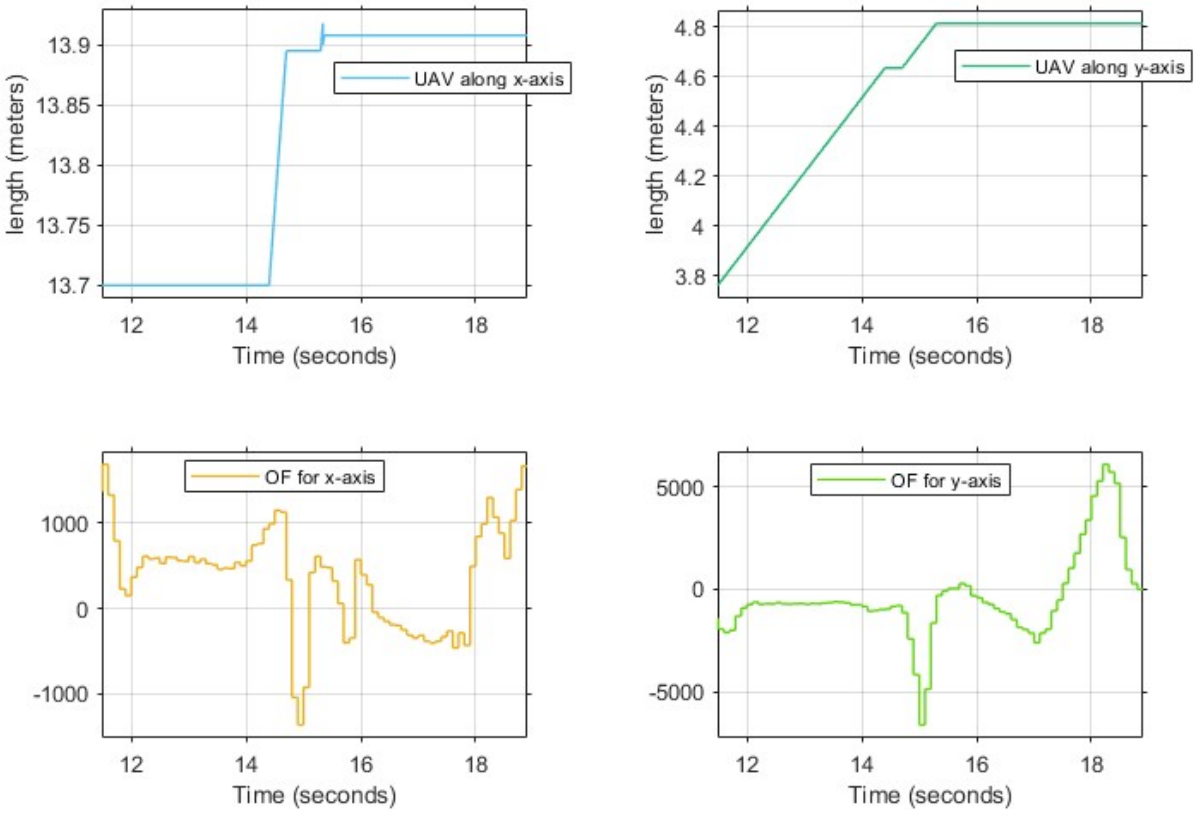


Figure 3.8: Using downward vision to maneuver the UAV towards landing platform

Figure 3.8 depicts how at around 15 seconds, the OFD values drop much lower (488 and 293 for x and y respectively) compared to the expected range of around 1000s. This shows that the UAV has positioned itself on top of the platform in such a manner, that the difference between the the two halves along x and y axes is much less. This is interpreted as the UAV must have been able to position itself at the center of

the landing platform. Using this, the system then proceeds to the next block in the UAV's journey.

To observe how this works along with the maneuvering towards the landing platform, it was assumed that the height of the landing platform is 0.6. Thus, an integrator control was implemented for landing the UAV once it was detected to be on top of the landing platform.

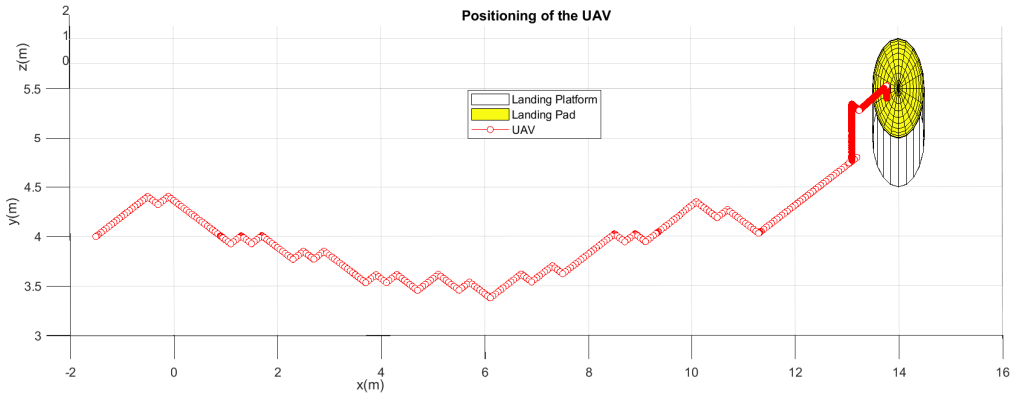


Figure 3.9: Maneuvering of the UAV on top of the landing platform

Figure 3.9 illustrates the maneuvering of the UAV on top of the landing platform. There are multiple data points once the UAV is elevated depicted by the darker red traces. This shows that the UAV is maneuvering slower on top of the landing platform versus when it was maneuvering towards the platform. It is also observed how the UAV is not positioned exactly in-line with the center of the landing platform, but since the landing platform is bigger, this does not affect the landing process.

3.4. Landing of the UAV

After the UAV has positioned itself on top of the landing platform, the method of image dilation is used to safely land the UAV. The focus of this section will be on landing that is reducing the z-coordinate of the UAV to match the unknown height of the platform. Thus, x and y coordinate of the UAV is set to a constant and does not vary in this process.

3.4.1. Image Dilation Method using IMU (IDMI)

M. Alkowatly et al. [44] used the calculation of image dilation to update the decreasing z-coordinate hence mimicking the movement of landing. This implementation requires knowledge about the starting position of the UAV and the initial velocity as well. Image dilation or flow divergence can be described as the ratio of the velocity to the height of the ground.

$$D = \frac{\text{velocity or rate of change of height}}{\text{height}} \tag{3.5}$$

The system was first redefined to have z and velocity along z (V_z) as the states and a control input as μ.

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mu(t) \\ y(t) &= [x_2(t)/x_1(t)] = D \\ \mathbf{x} &= [x_1, x_2]^T = [z, V_z]^T \end{aligned} \tag{3.6}$$

$$\Rightarrow D = \frac{z}{V_z} \tag{3.7}$$

For calculating the control input, a PI controller was used with D* as the desired dilation.

$$\mu = K_p (D^* - D) + K_i \int (D^* - D)\, dt \tag{3.8}$$

By keeping the image dilation constant at $D = -k$, the dynamics of the UAV can be controlled. With different positive values of k, the dynamics of the UAV can be varied. For example, the larger the k is, the faster the states converge to zero. But this can lead to physical consequences. D is a negative value as V_z will be a negative value. If D becomes a positive value during the simulation, the UAV tends to ascend instead of landing. For implementing this method, k was chosen as 0.5. The proportional and integral gains were kept at $K_p = 16$ and $K_i = 10$ respectively.

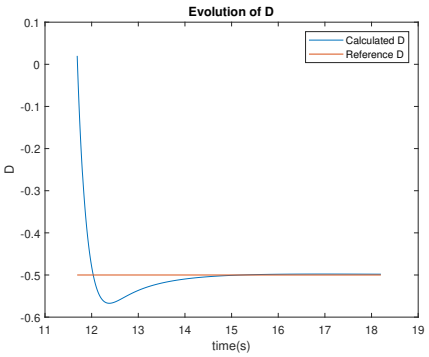


Figure 3.10: Image Dilation with reference of -0.5

In Figure 3.10, the calculated image dilation was able to safely reach the reference value in 3 seconds after landing started. Though the simulation was run in 3D World linked with Simulink, for the ease of understanding, a plot was generated with a cylinder with the top as a filled circle mimicking. But on observing the behaviour of the UAV in Figure 3.11, the drone tends to go through the platform and does not stop at the landing pad. The drone stopped once the height was almost at zero. Thus, showing that this algorithm failed to recognize the elevated platform. Furthermore, this algorithm is independent of the view that the camera sees. Hence an alternate method would be to utilize the optical flow algorithm discussed before to detect the height better.

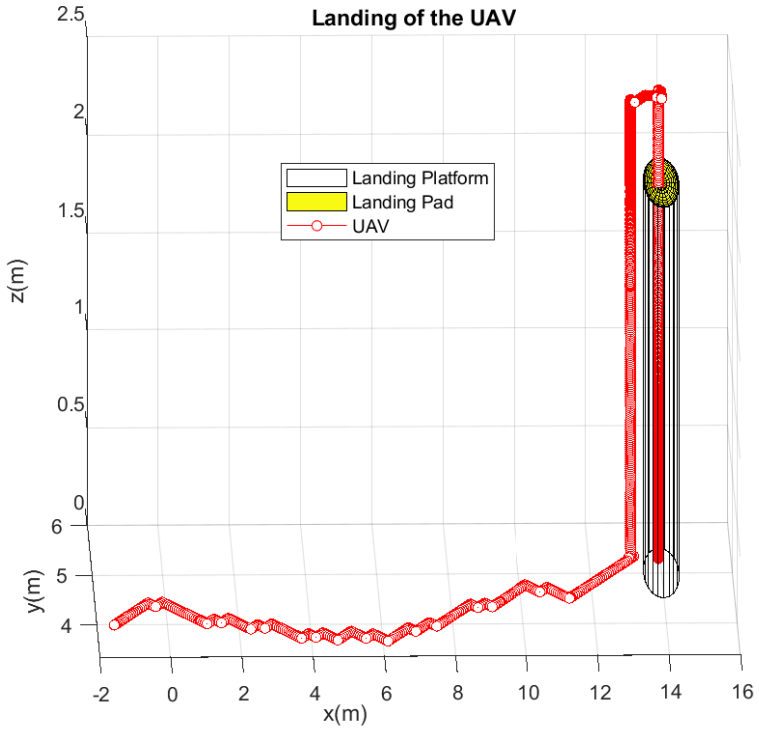
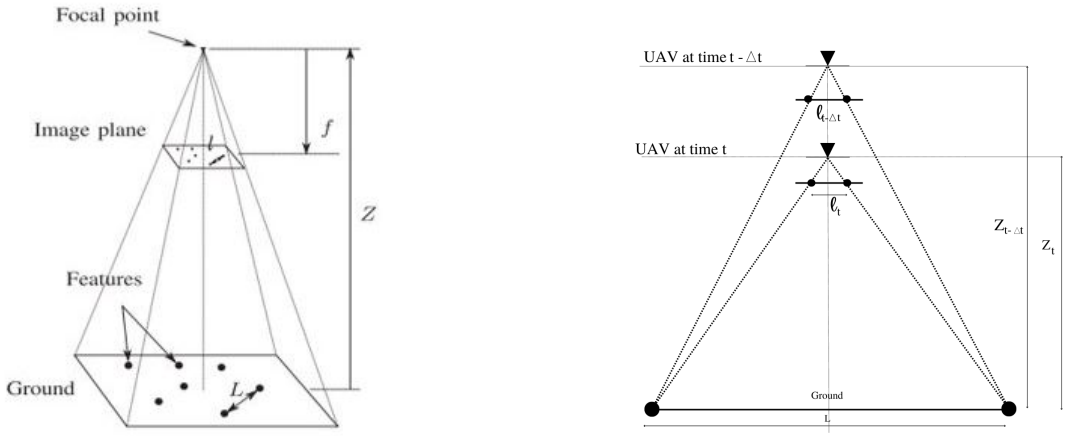


Figure 3.11: Landing using Image Dilation and Initial Height and Velocity

3.4.2. Image Dilation Method using Features from Accelerated Segment Test (IDMF-AST)

Instead of using the UAV's location for calculating image dilation, an estimate can be calculated by using the camera vision. In the research published by H. W. Ho et al., [58] textured points of the platform were used to estimate the image dilation. This paper used the estimation method for safe landing while the landing area is on the ground hence this will be tested for elevated landing platforms.

To detect these points on the landing pad and to track these, Features from Accelerated Segment Test(FAST) is implemented along with the Lucas Kanade tracker [59]. This helps in detecting points at a certain time instant. The change in the position of the points is tracked using Lucas Kanade tracker. It is assumed that the maximum bi-directional error that these feature points face at the next time instant will be 10 and the total number of points being tracked are a maximum of 50.



(a) A pinhole camera model showing the actual size and image size connecting the two features indicated by L and l

(b) Change in the distance between features with time

Figure 3.12: The feature correlation from the view of the camera

In Figure 3.12a, due to the camera vision, the distance between the features will be different from the actual distance. Thus, using similar triangle congruence, a relation can be made between the actual height difference from the UAV to the ground and the distance between the features.

$$\frac{L}{Z_t} = \frac{l_t}{f} \tag{3.9}$$

Similarly, for the previous time instant $t - \Delta t$, another correlation can be made.

$$\frac{L}{Z_{t-\Delta t}} = \frac{l_{t-\Delta t}}{f}, \tag{3.10}$$

Figure 3.12b illustrates the change of the size of the projected lines in the image plane, from $l_{t-\Delta t}$ to l_t when the UAV is moving toward the ground, from $Z_{t-\Delta t}$ to Z_t. Using similar triangles, the following relation can be written.

$$\frac{Z_t}{Z_{t-\Delta t}} = \frac{l_{t-\Delta t}}{l_t} \tag{3.11}$$

Equation 3.7 can be rewritten for each time step t as

$$\begin{aligned} D_t &= \frac{[Z_t - Z_{t-\Delta t}]}{\Delta t}\frac{1}{Z_t} \\ \text{where } \frac{[Z_t - Z_{t-\Delta t}]}{\Delta t} &\text{ is the rate of change of height or } V_z \\ \Rightarrow D_t &= \frac{1}{\Delta t}\left[1 - \frac{Z_{t-\Delta t}}{Z_t}\right] \end{aligned} \tag{3.12}$$

Using Equation 3.11, image dilation D can be instead calculated by the distance between the features.

$$\begin{aligned} D_{est_t} &= \frac{1}{\Delta t}\left[1 - \frac{l_t}{l_{t-\Delta t}}\right] \\ D_{est_t} &= \frac{1}{\Delta t}\left[\frac{l_{t-\Delta t} - l_t}{l_{t-\Delta t}}\right] \end{aligned} \tag{3.13}$$

Assuming there are N lines generated at each time step, the estimated image dilation D can be described as

$$\widehat{D} = \frac{1}{N}\sum_{i=1}^{N} D_{est_{t_i}} \tag{3.14}$$

The calculated D is taken to be an average to reduce the possibility of any outliers affecting the system drastically. For implementing this method, the control algorithm used in the previous section was implemented but the control input was based on the estimated D_{est} instead. Thus, the control input is calculated using D_{est}.

$$\mu = K_p\left(D^* - D_{est}\right) + K_i \int \left(D^* - D_{est}\right) dt \tag{3.15}$$

The more detailed implementation of image dilation is enumerated in Algorithm 2.

Algorithm 2 : IDMF-AST implementation

1. Since the view of the landing pad will change once the drone gets closer, a reset of the points that are being tracked needs to happen. For this implementation, when the number of points that were able to successfully be tracked reduces to below a value of θ_p, then the points are then based on the current view of the landing pad.
2. For calculating l (Equation 3.9), if points are either horizontally or vertically aligned with a deviation of 3, the points were considered to be in a line.
3. The Δt (Equation 3.13) was set to match the step size of the camera view update.
4. The paper in reference did state that once the camera is really close to the ground, this algorithm fails to be accurate [58]. The paper implemented a PID controller once they observed that the UAV was about 0.2-0.3 meters away from the ground. In this particular simulation, to recognize that the camera was too close to the ground, the number of times the algorithm was reset is considered instead. A threshold value of θ_r was set.
5. Once the algorithm resets more than θ_r times consecutively, the system would no longer use the D_{est} but use its current height and slowly drop down by h_{sl} meters.
6. If the drone was able to land on the platform before h_{sl} meters, then it is considered as a successful run with safe landing. It is considered safe as the descent of h_{sl} meters is very slow thereby not inducing any physical impact on the UAV.

Specific values that were used for the simulation are tabulated in Table 3.3.

Variable Name	Variable	Value
Number of Points Tracked	θ_p	25
Consecutive Reset Threshold	θ_r	30
Time Step	Δt	0.1
Proportional Gain	K_p	16
Integral Gain	K_i	10
Desired D	D^*	-0.5
Safe Landing Descent	h_{sl}	0.4 meters

Table 3.3: The Hyperparamters value

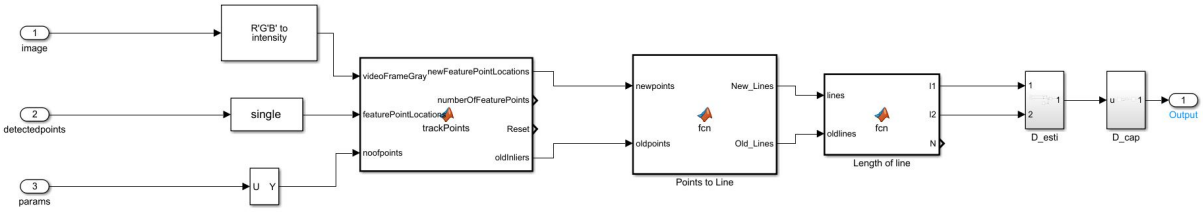
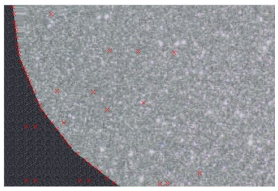
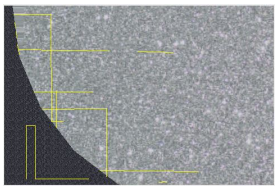


Figure 3.13: Simulink Diagram of $\hat{D}$

Figure 3.13 shows how the estimation of the image dilation is implemented in Simulink. On implementing image dilation, the UAV was not able to reach the platform by h_{sl} = 0.4 meters. This is also observed in Figure 3.15. For this simulation a textured platform was implemented but without any design or demarcations on it as seen in Figure 3.14. There is a possibility that a different type of landing pad design might work better for point tracking. This was based of how most of the features selected for tracking had a tendency to be on the edge of the platform till the edge was not visible. The next chapter will work on expanding this method for working on unknown elevated landing platforms.



(a) Features detected that are being tracked



(b) The lines generated using the points observed

Figure 3.14: The feature tracking and distance calculation

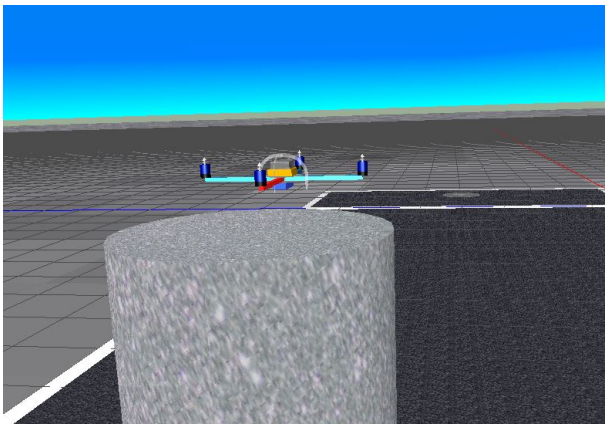


Figure 3.15: UAV hovering on implementation of Landing using Image Dilation and FAST

There have been more complex control algorithms implemented for better stability of the UAV while landing. H. W. Ho et al. [60], worked on height-based gains for the PID controller. This was done to reduce the oscillations due to the control gains and then tested in windy conditions. H. W. Ho et al. [58], then extended this research to calculate adaptive gains which were then implemented to reduce the oscillations observed close to the ground. Y. Zhou et al. [61], implemented Extended Incremental Nonlinear Dynamic Inversion (EINDI) to reduce the oscillations and it is able to show better results than the previous runs and was also tested for different starting heights. Unfortunately this implementation is a starting base for the above stated algorithms and was not able to provide a better probability of safe landing for varying heights of the platform. Thus, future research would be towards finding an improved algorithm based on IDMF-AST.

3.5. Problem Formulation

Given the vast number of methods available and the sub-processes that affect the accuracy of safe autonomous landing of UAVs, it is important to define the problem in detail. The aim of this research is to propose a new method that will perform autonomous landing of an UAV on a platform whose height is unknown, using a low complexity algorithm.

The aim of this thesis is encapsulated as follows:

1. Design a safe autonomous landing algorithm that considers Optical Flow vision.
2. The designed algorithm should be able to land safely on an unknown elevated landing platform.
3. Criteria that the proposed algorithm should meet comparable to existing methods, such as IDMF:
    - Higher probability of safe landing while decreasing the probability of crash.
    - Comparable time taken for safe landing if not faster.
    - Computational load while simulating should not affect simulation time.

4

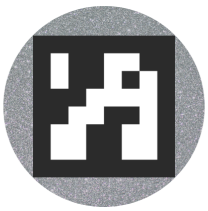
Improved IDMF

The earlier investigation hinted that the IDMF algorithm might work better on a landing pad which have some design or demarcations. Therefore, in this chapter, we first study the performance of the IDMF algorithm with optimized marker design. Moreover, there is a discussion into how the hyperparameters implemented in IDMF affect the probability of safe landing. The improved IDMF algorithm with ideal hyperparameter selection is then proposed. Finally, the Adaptive IDMF algorithm for landing of a UAV on a landing platform with unknown height is proposed.

4.1. IDMF with Optimized Marker Design

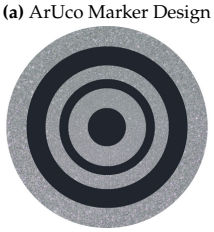
To tackle the problem of choosing a better landing pad design, the state of the art landing designs are tested for varying heights of the platform. These being ArUco [30], Bullseye, Fourier Circles Markers and a landing pad with no design as seen in Figure 4.1. These specific markers have been designed for easy detection using camera vision. Furthermore, these designs have been used for decoding the height between the landing platform and the UAV using image processing methods. These designs are compared with the textured landing design to see how the performance varies with the given algorithm.

Figure 4.1: Different Marker Designs Tested for Different Heights

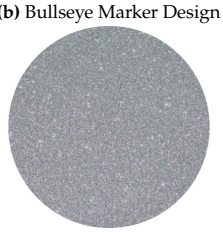


(a) ArUco Marker Design

(b) Bullseye Marker Design



(c) 3 concentric circles on textured area



(d) Clear textured landing pad

Three possible types of landing are possible. Safe landing is defined to start when the number of consecutive resets are more than a defined threshold. Based on this, the types of landing are:

1. **Safe Landing:** when the UAV is able to reach the landing pad within the h_{sl}.[1]

[1] the value of h_{sl} is taken from Table 3.3

2. **Hover:** when the UAV is not able to reach the landing pad within the h_{sl}.
3. **Crash:** when the number of consecutive resets does not reach the threshold before the landing pad.

The markers were tested for 5 different heights of the platform. The implementation followed IDMF-AST as summarized in Algorithm 2. The hyperparameters used were referenced from Table 3.2. The type of landing for each design type was noted in Table 4.1.

Table 4.1: Test Results of Different Marker Designs

Height (m) / Type of landing	Bullseye	ArUco	Clear	Concentric Circles (3)
1	safe	hover	crash	safe
0.8	crash	safe	hover	safe
0.6	hover	hover	crash	crash
0.4	safe	hover	safe	safe
0.24	safe	crash	hover	safe

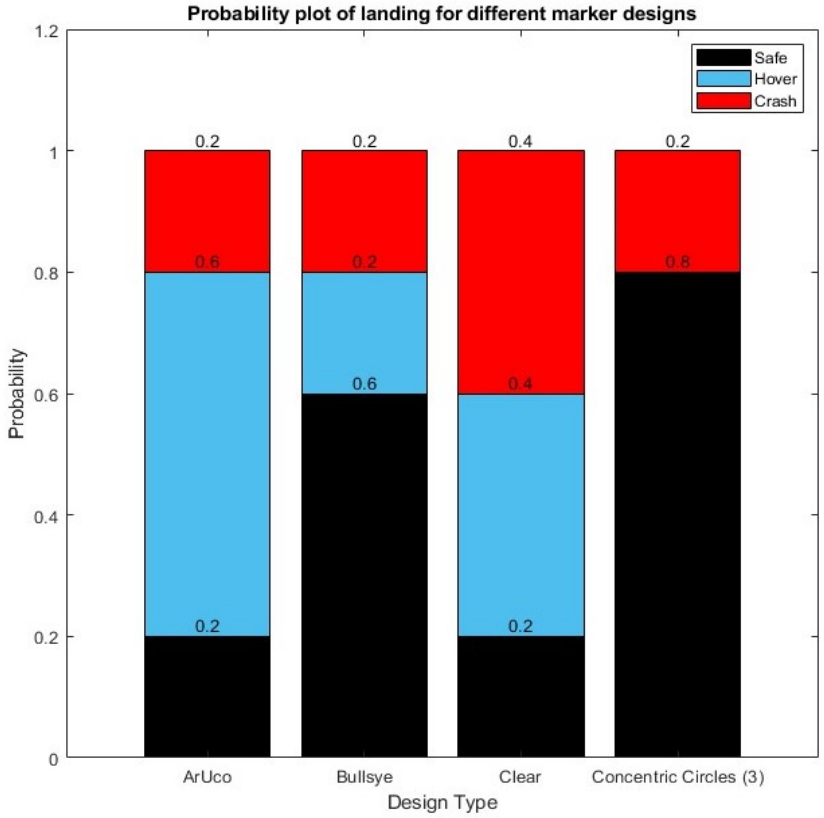


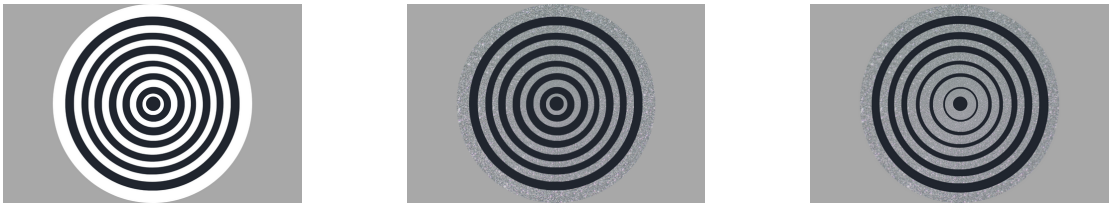
Figure 4.2: Probability of Each Type of Landing for The Different Marker Designs for The Case of Unknown Height in the Range of [0.24,1]

The probability of the three type of landing are illustrated in Figure 4.2. It is observed how concentric circle design is able to provide safe landing for most of the varied height's of the landing platform compared to the Bullseye, ArUco and clear design. A reason why Bullseye and ArUco were not able to perform stems from the absence of texture and lesser points to track once the camera gets very close to the landing pad. This result also is comparable to the research done by S. Lange et al. [62]. This paper also had an implementation of a concentric circle landing pad design over other designs and were able to estimate the position of the UAV accurately using image-processing.

To check if the existence of texture or the varying thickness of the circles in the design was the reason why the concentric circles worked better, another round of testing was conducted for three types of design;

black and white concentric circles, textured concentric circles, reducing width textured concentric circles. These designs were tested from 3 concentric circles to 8 concentric circles. An example of how these designs vary is depicted in Figure 4.3. The individual results are tabulated in A.1. The summary of these datapoints are charted in Figure 4.4.

Figure 4.3: Example of the Designs for 7 Concentric Circles

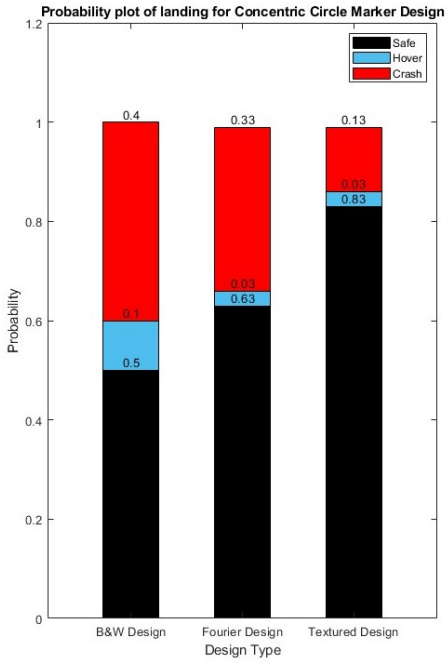


(a) Black and white circles (7)

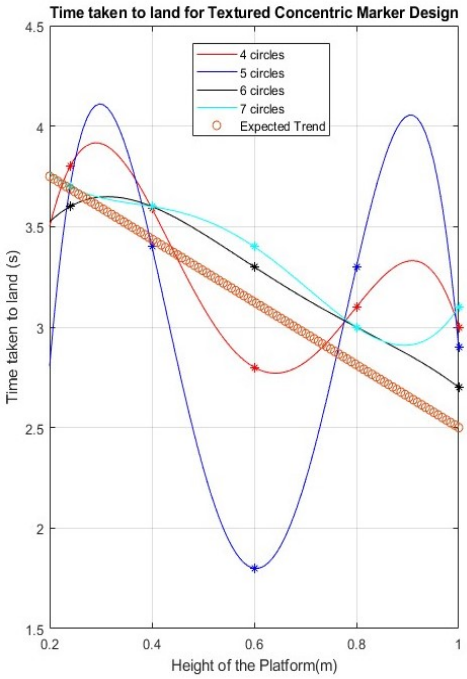
(b) Textured concentric circles (7t)

(c) Fourier Design (7f)

Figure 4.4: Test Results for Concentric Circle Designs



(a) Probability of Landing Types for Concentric Circle Marker Design



(b) Trend for the Time Taken to Safely Land

It is observed in Figure 4.4a that the textured designs were able to perform much better compared to the others as these designs could guarantee a higher probability of safe landing. This could be due to the picking up of textures even when the vision is very close to the landing platform.

The results of the successful runs were plotted as shown in Figure 4.4b. An expected trend was that as the height of the platform increases, time taken to land should reduce. This is plotted in figure as "Expected Trend". Figure 4.4b shows the attempt to observe if any design was able to follow the "Expected Trend" while being able to land safely. It was noted that 6 concentric circles had a constantly faster landing time compared to 7 concentric circles and was more consistent compared to 5 concentric circles. Thus, the next set of testing was limited to 6 concentric circles design only.

During these tests, it was detected how the hyperparameters: the PID gains (K_i, K_p), the number of points that are being tracked (θ_p) and the reset threshold (θ_r) had an impact on the type of landing that would take place without changing the height of the platform. The next section will look into this impact.

4.2. Improved IDMF with Ideal Hyperparameter Selection

It was observed that hyperparameters were affecting the type of landing the UAV would go through. Thus, the first attempt was to find an optimal set of parameters for all possible heights of the platform $h \in [h_l,h_u]$ which would provide safe landing. This is addressed in Section 4.2.1.

The next attempt was to find optimal hyperparameters that could guarantee safe landing for a smaller range of $[h_l,h_u]$. Curve fitting was used to find hyperparameter values (θ_p) that can guarantee safe landing for $h \in [h_a,h_b]$ where $h_a,h_b \in [h_l,h_u]$ and h_b=h_a+Δh. The details on curve fitting is provided in Section 4.2.2.

The last attempt was to find optimal hyperparameters for each possible height of the platform $h \in [h_l,h_u]$. A classification model was designed for finding optimal hyperparameters for each possible height between the range of $[h_l,h_u]$. A reason why classification model was required over curve fitting is that the three types of landing were considered compared to just safe landing in curve fitting. More on how the model was designed and implemented in the system is detailed in Section 4.2.3.

4.2.1. Finding Optimal Hyperparameters for the Unknown Height Range of $[h_l,h_u]$

To observe if there are any optimal hyperparameters for safe landing, few more rounds of simulations were done. This data has varying PID gains (K_i, K_p), different heights of the platform and a range of number of points (θ_p) that are tracked by FAST while keeping the reset threshold constant. The reset threshold was kept a constant as it is calculated based on the instances when the number of points needed to be tracked is not met. The values in consideration are summarized in Table 4.2.

Variable Name	Variable	Value
Number of points tracked	θ_p	[10,15,20,25,30]
Consecutive Reset Threshold	θ_r	20
Platform Height	$[h_l,h_u]$	[0.2,2]
Proportional Gain	K_p	4 or 8
Integral Gain	K_i	4 or 9 or 14

Table 4.2: The Range of Values Tested for Optimal Hyperparameters for Height Range $[h_l,h_u]$

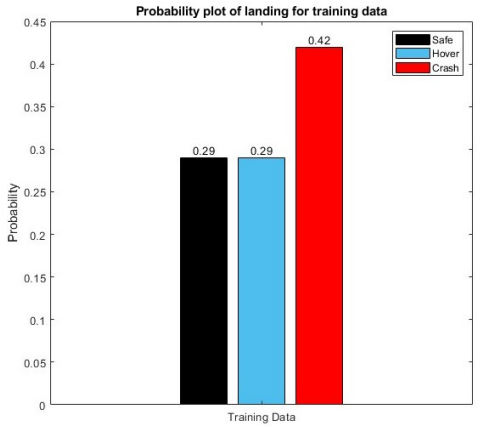


Figure 4.5: Probability of the Types of Landing for the Training Data Generated

As observed in the Figure 4.5, the probability of crash is quite high given the comparatively small range of landing platform height. Thus, the aim will be to find optimal parameters that would reduce the probability of crash. First attempt was to look into if there are any patterns in the training data and if there are any set of hyperparameters work for a smaller range of heights for the landing platform.

4.2.2. Finding Optimal Hyperparameters for the Unknown Height Range [h_a,h_b] where h_b=h_a+Δh using Curve Fitting

A polynomial curve fitting was employed to see if there were any [h_a,h_b] where h_a,h_b $\in$ [h_l,h_u] and h_b=h_a+Δh that a certain range of values of θ_p would provide with safe landing. To implement curve fitting, the coefficients that best fit the data points for varying heights were calculated. The method used was the best fit using least-squares for the degree n. These were then plotted against the actual data points to see how well the the curve can estimate all the safe landing hyperparameter values available. The curve generated should be able to give the range of θ_p for an assumed range of elevated platform height. The considered hyperparameter values for curve fitting are summarized in Table 4.3.

Variable Name	Variable	Value
Number of points tracked	θ_p	[10,15,20,25,30]
Consecutive Reset Threshold	θ_r	20
Platform Height	[h_l,h_u]	[0.2,2]
Difference between h_a,h_b	Δh	0.2
Proportional Gain	K_p	4 or 8
Integral Gain	K_i	4 or 9 or 14
Degree of the polynomial	n	4

Table 4.3: The Range of Values Used in Curve Fitting for Finding Optimal Hyperparameters for Height Range [h_a,h_b]

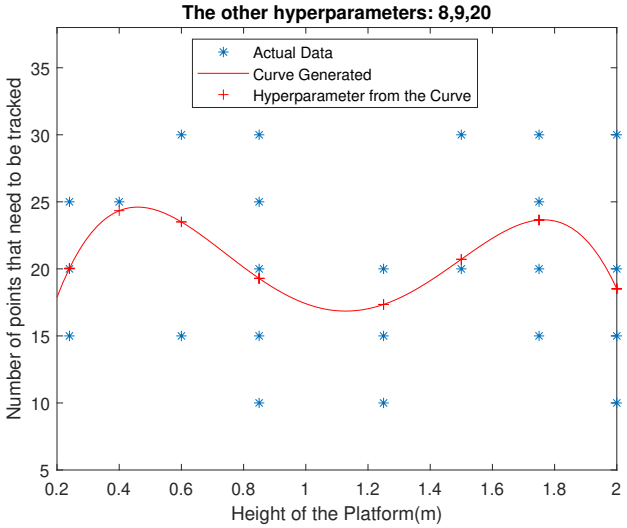


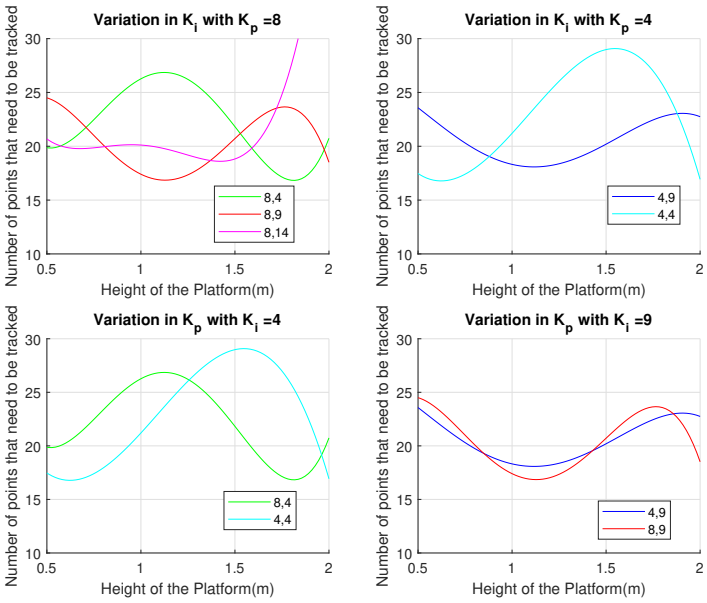
Figure 4.6: Curve Fitting Results for θ_p that provide safe landing

In Figure 4.6, θ_p for safe landing were illustrated versus the height of the platform. This was tested for the hyperparameter values from Table 4.4. The range of θ_p to be tracked for safe landing for an elevated platform of [h_a,h_b] meters can be inferred from Figure 4.6. Unfortunately, on testing the range of θ_p for the considered range of [h_a,h_b] lead to a crash. Thus, the idea of using curve fitting for optimal hyperparameters fails. Furthermore, this model does not consider cases when the landing is not safe which might have affected the test case.

Variable Name	Variable	Value	Type of Data
Range of Height Considered	[h_a,h_b]	[0.6,0.8]	Known
Number of Points Tracked	θ_p	[20,24]	Known
Platform Height	h	0.75 m	Unknown

Table 4.4: The Values Considered to Test Curve Fitting for Finding Optimal θ_r

Figure 4.7: Trends on Changing the PID Gains



Using the curves generated per trial, there was an attempt to observe trends if there are changes in one of the hyperparameters. The trials specifically looked into how changes in the gains affected θ_p that provide safe landing. Figure 4.7 encapsulates the trials in consideration with the first and second quadrant looking into variation in K_i and variation in K_i in the third and fourth quadrant. As seen in Figure 4.7, there is no consistent pattern between changing the PID gains and the number of points (θ_p) to be tracked for safe landing. This shows that a generalized method for safe landing given a range of elevated landing platforms does not help fix the problem of a high probability of crashes. Thus, a more individualistic method should be generated that can take into consideration not only hyperparameters that give safe landing but also crash and hover. The next attempt was to use a Classification Learner Toolbox to see if a trained model can predict what type of landing a set of parameters can give.

4.2.3. Finding Optimal Hyperparameters for An Assumed Height of the Platform Varying in the Range of [h_l,h_u] using Classification Model

Curve fitting takes into consideration only hyperparameters that provide safe landing. Hence, a classification model will be able to provide better suited set of hyperparameters using all the types of landing possible. The aim of this model is to find hyperparameters that provide safe landing for the UAV.

Before the Classification Model is incorporated with the Simulink System, 2 steps, training and testing is implemented on the Classification Model to guarantee finding optimal hyperparameters. Training a model entails using a pre-existing labeled dataset including the output sent to the model. The model learns possible biases that lead to the required output or category in this case. For testing the model, new labeled data is sent to the trained model and the model attempts to classify/categorize the data. This output is then compared with the expected output of the labeled data to calculate how accurate the trained model is.

The hyperparameters used and their values in the labeled dataset are encapsulated in Table 4.5. The type of landing was numerically categorized as 1 or 2 or 3 which was then sent to the model along with the hyperparameter values. The mapping of the numeric values to the type of landing is defined below.

1. **Crash:** 1
2. **Hover:** 2.
3. **Safe Landing:** 3.

Variable Name	Variable	Value
Number of Points Tracked	θ_p	[10,15,20,25,30]
Consecutive Reset Threshold	θ_r	[10,15,20,25,30]
Platform Height	$[h_l, h_u]$	[0.2,2]
Proportional Gain	K_p	[2,20]
Integral Gain	K_i	[2,20]
UAV's Starting Height	z_o	2.5 or 3
Classification Category	y	1 or 2 or 3

Table 4.5: The Training Data for Classification Model with varying UAV starting height

Training the Landing Classifier using Height of Descent (HOD)
There was an attempt to train a classifier which takes HOD (height the UAV starts to descent - height of the platform) and the other hyperparameters. The details on the inputs required for training is encapsulated in Figure 4.8. The values of the hyperparameters are referred from Table 4.5. The training setup and the accuracy of the training and testing of the model is summarised below. The confusion matrices are illustrated in Figures 4.9 and 4.10.

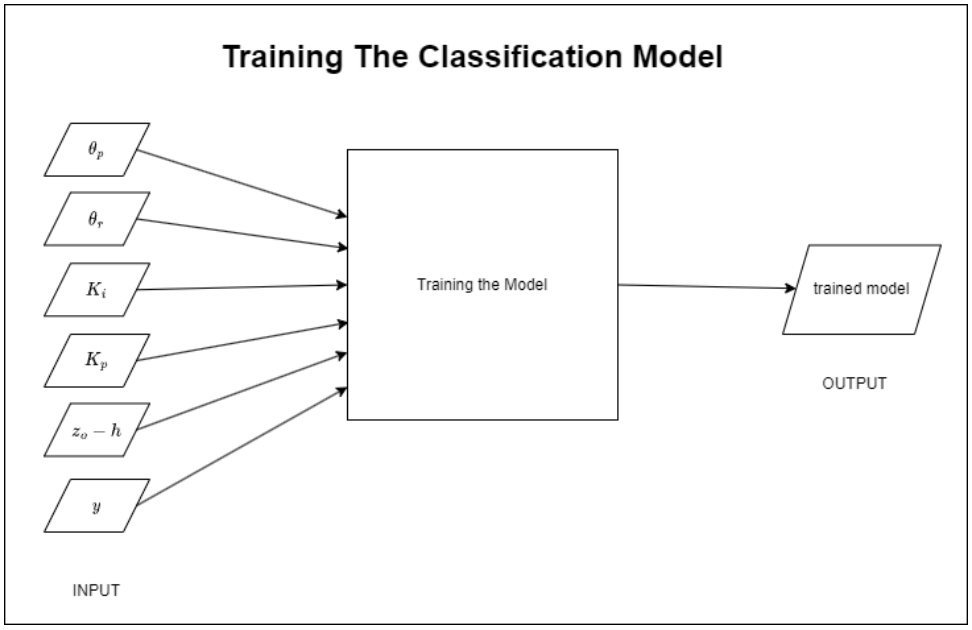


Figure 4.8: The Hyperparameters Required for Training the Classifier using HOD

1. **Data available:**
   (a) Training Data Size: 916
   (b) Training and Testing Data Division: 85% and 15%
2. **Best Model:** Fine Trees
   (a) Validation Accuracy: 53.9%
   (b) Test Accuracy: 56.2%

Figure 4.9: Fine Trees Model (for HOD): Validation Results

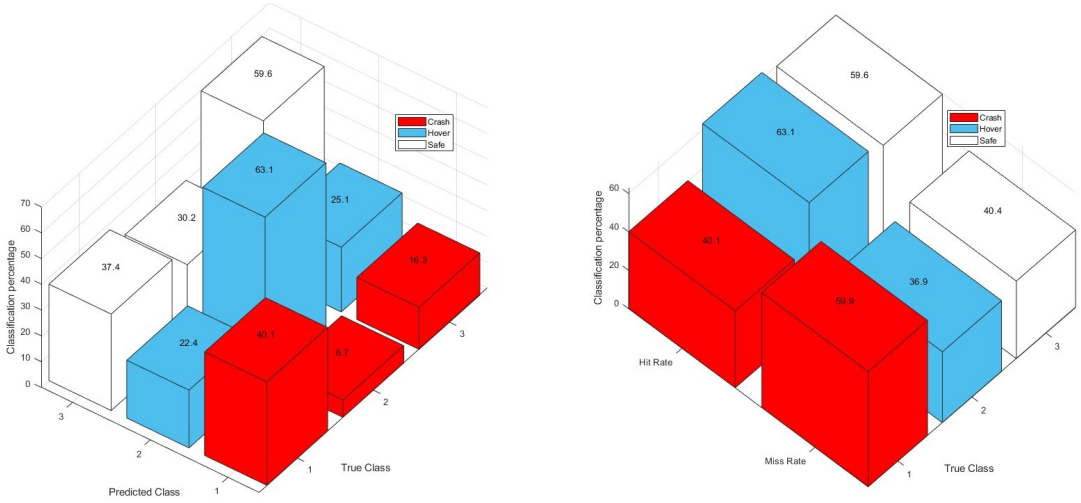
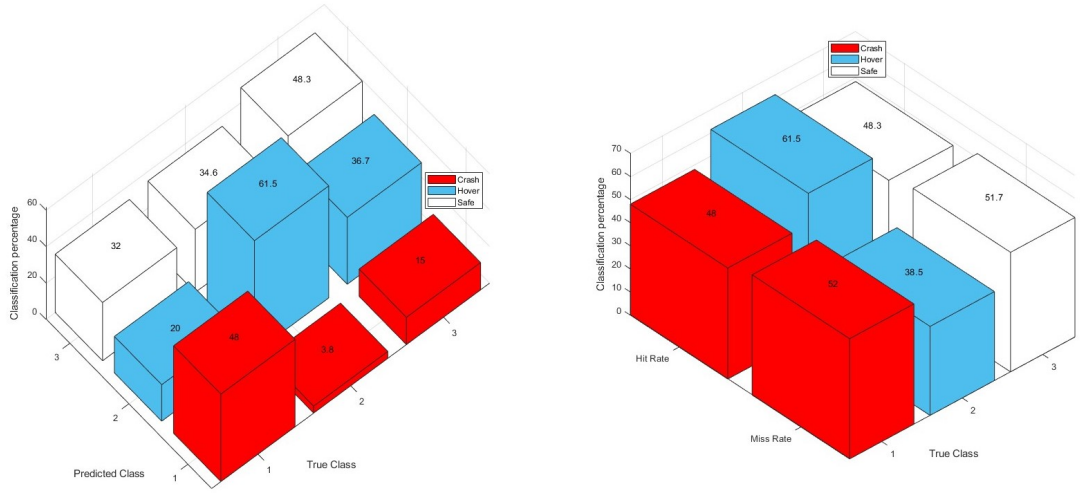


Figure 4.10: Fine Trees Model (for HOD): Test Results



Figures 4.9 and 4.10 shows the the confusion matrix which explains how often is there a possibility of the model classifying a set of hyperparameters accurately. On the left side, the plot has the probability of True Class and Predicted Class as the x and y axes respectively. As stated earlier, the landing classification categories of crash, hover and safe were numerically categorized as 1, 2 and 3 respectively. So if a set of hyperparameters were classified as safe landing (3) but the true class is crash (1), then this would be categorized under 37.4%. The right side of the figure gives the Hit Rate and the Miss Rate which is the rate of a correct classification and wrong classification respectively for each class.

Even though the accuracy rates were more than 50% for Fine Trees model, the Hit Rate for crash (1), is lesser than 50%. Even though the dataset is quite large, the model is not able to have better accuracy in classifying crashes. Thus, a more detailed look into a better way of training the model with some feature translation was necessary for better accuracy results.

Training the Landing Classifier using Height of Platform and Different UAV Starting Heights

The next attempt was to split HOD as height of the platform and the UAV's height at the start of descent. This was done as the accuracy was low compared to the data available for training and there might be more dependency on the starting height of the UAV. Thus, the variables used for training are depicted in Figure 4.11 and the values considered are from Table 4.5. The data used and the accuracy of the training

and testing of the model are summarised below. The confusion matrices are described in Figures 4.12 and 4.13.

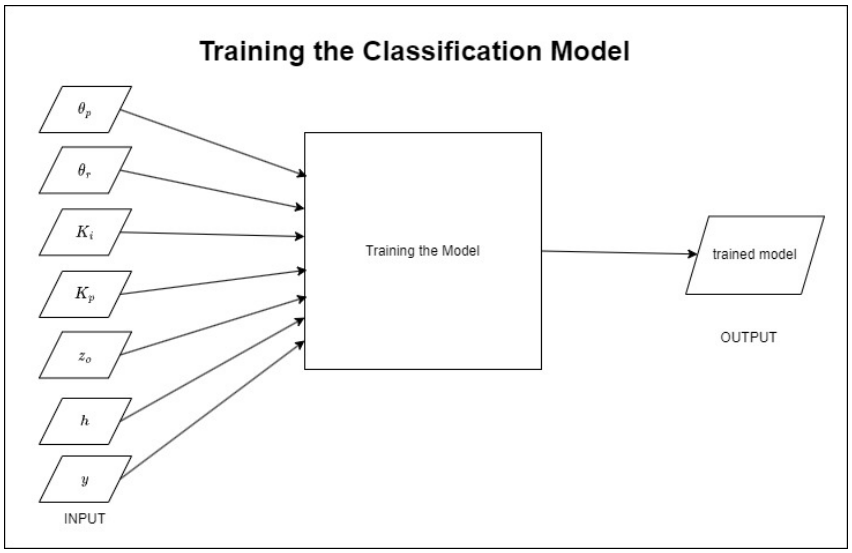


Figure 4.11: The Hyperparameters Required for Training the Classifier using h and Different z_o

1. **Data available:**
   (a) Training Data Size: 916
   (b) Training and Testing Data Division: 85% and 15%
2. **Best Model:** Bagged Trees, an Ensemble Model
   (a) Validation Accuracy: 60.0%
   (b) Test Accuracy: 60.0%

It is observed in Figures 4.12 and 4.13 that the accuracy for true crash classification is at 60%. This is quite similar in value to using a coin toss to predict if the set of hyperparameters would lead to crash or safe landing or hover. This was not expected with a dataset of almost 1000 as the classification model should have been able to give better accuracy results. Thus, splitting the height parameter was yet not able to give better results.

Figure 4.12: Bagged Trees Model (for h and Different z_o): Validation Results

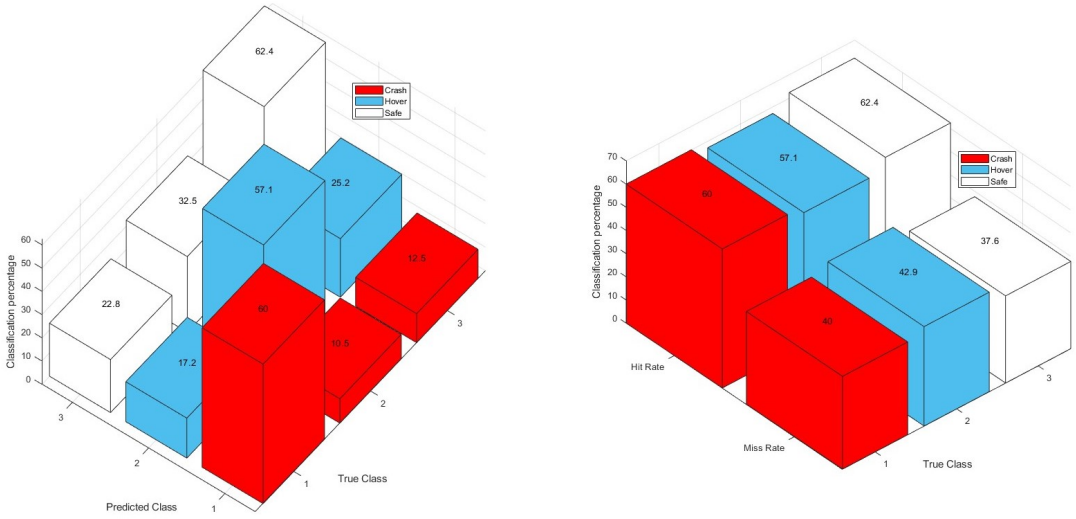
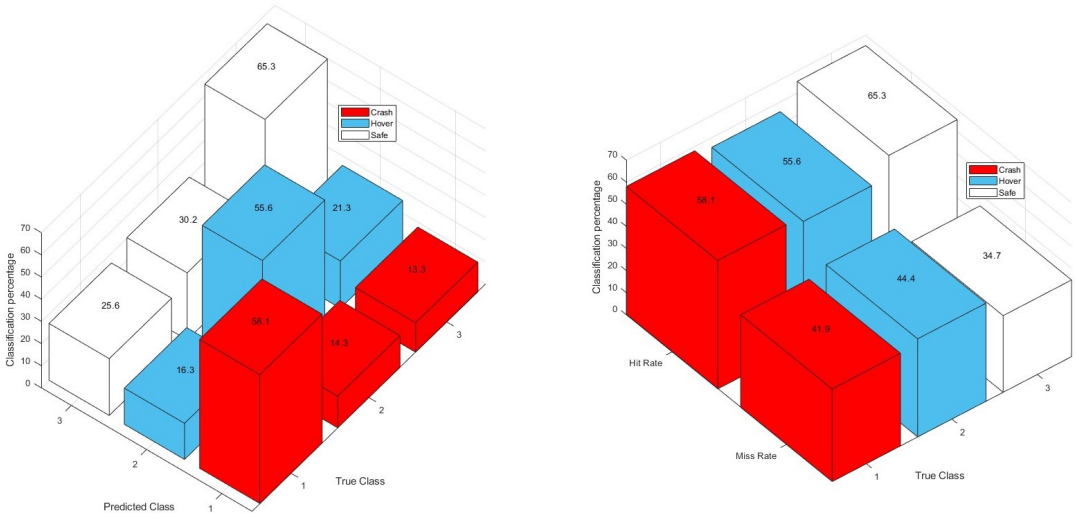


Figure 4.13: Bagged Trees Model (for h and Different z_o): Test Results



To find possible flaws, it is important to know more about the classifier model and data traits this mode tends to amplify. Ensemble methods work by implementing different type of machine learning algorithms to get better accuracy. Bagged types of models specifically by choosing different subsets of the training data and working on different classifiers for these subsets. These subsets of training data are not balanced thus it tends to prioritize correctly classifying the majority class. On examining the training data generated, it was observed that there was not an equal amount of data for each class. The current data had approximately 40% of safe landing, 40% of hover and only 20% of crash landing. This might have affected the training of the model and hence affecting the accuracy of crash classification.

Some possible ways of addressing this issue are by undersampling or oversampling it [63]. For this case, removing the data to implement undersampling was counter-intuitive as the total data available was less compared to other applications of classification model training. On the other hand, for oversampling, getting only crash data was quite time-taking as the probability of crash in a single round of test is an average of 30%. A common oversampling method is SMOTE (Synthetic Minority Over-sampling Technique) [64]. Thus, the next run implemented SMOTE to balance the dataset.

Training the Landing Classifier using Height of Platform and Constant UAV Starting Height
Compared to the previous attempts, the starting height of the UAV (z_o) was made a constant at 2.5 meters. Thus, the parameters defined to the Classification Model are depicted in Figure 4.14.

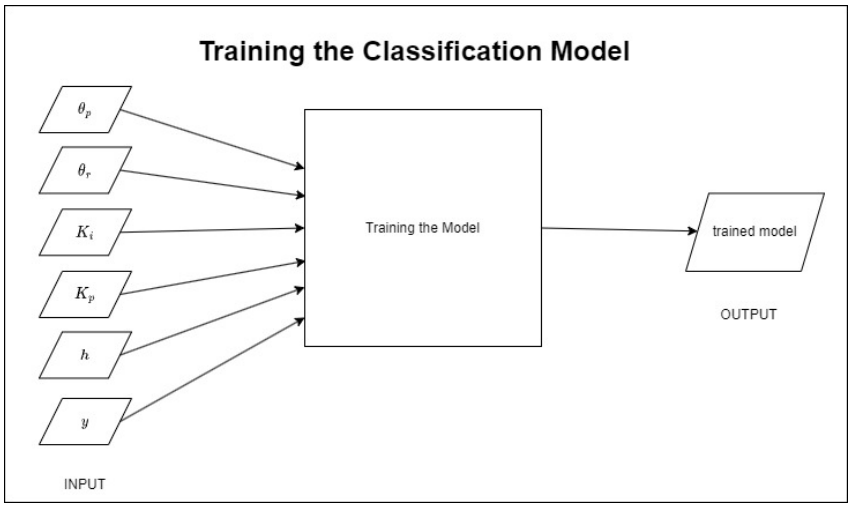


Figure 4.14: The Hyperparameters Required for Training the Classifier using Height of Landing Platform

Details about the data used are summarised below.

1. **Data available:**
   (a) Training Data Size: 800
   (b) Training and Testing Data Division: 85% and 15%
   (c) SMOTE was used to extend the data classified as crash.
2. **Miscalculation cost:** double the penalty when true class of crash is classified as safe landing. This is done as a miscalculation of a crash as any other type of landing will be harmful for the UAV.
3. **Features being sent to the model are:**

Table 4.6: The Training Data for Classification Model with constant UAV starting height

Variable Name	Variable	Value
Number of points tracked	θ_p	[10,15,20,25,30]
Consecutive Reset Threshold	θ_r	[10,15,20,25,30]
Platform Height	h	[0.2,2]
Proportional Gain	K_p	[2,20]
Integral Gain	K_i	[2,20]
Classification Category	y	1 or 2 or 3

In Table 4.7, the accuracy of each model is summarized to show how different the accuracy is between the ensemble models compared to SVM and KNN methods. The highlighted method is the one which had the highest accuracy rates for both validation and testing.

Table 4.7: Accuracy Results of the 25 available models with adjusted cost matrix

Model Name	Validation Accuracy	Test Accuracy
Fine Tree	63.8%	70.7%
Medium Tree	53.2%	50%
Coarse Tree	46.8%	47.4%
Linear Discriminant	46.2%	46.6%
Quadratic Discriminant	52.6%	51.7%
Guassian Naive Bayes	50%	55.2%
Kernel Naive Bayes	45.7%	47.4%
Linear SVM	49.8%	50.9%
Quadratic SVM	54.3%	51.7%
Cubic SVM	59.6%	60.3%
Fine Guassian SVM	62.9%	63.8%
Medium Guassian SVM	58.4%	55.2%
Coarse Guassian SVM	48.9%	50.0%
Fine KNN	67.9%	69.8%
Medium KNN	56.5%	46.6%
Coarse KNN	48.8%	50.0%
Cosine KNN	54%	49.1%
Cubic KNN	56.4%	46.6%
Weighted KNN	69.3%	69.8%
Boosted Trees	57.4%	59.5%
Bagged Trees	76.0%	79.6%
Subspace Discriminant	49.4%	52.6%
Subspace KNN	43.0%	39.7%
RUSBoosted Trees	55%	60.3%
SVM Kernel	56.8%	56.9%

Figures 4.15 and 4.16 shows how the accuracy with penalized cost for miscalculation of crashes as safe gives a higher Hit Rate. As the accuracy has become much better compared to the previous trials, this Bagged Trees model is used for the integration with the system model.

Figure 4.15: Bagged Trees Model (updated cost matrix): Validation Results

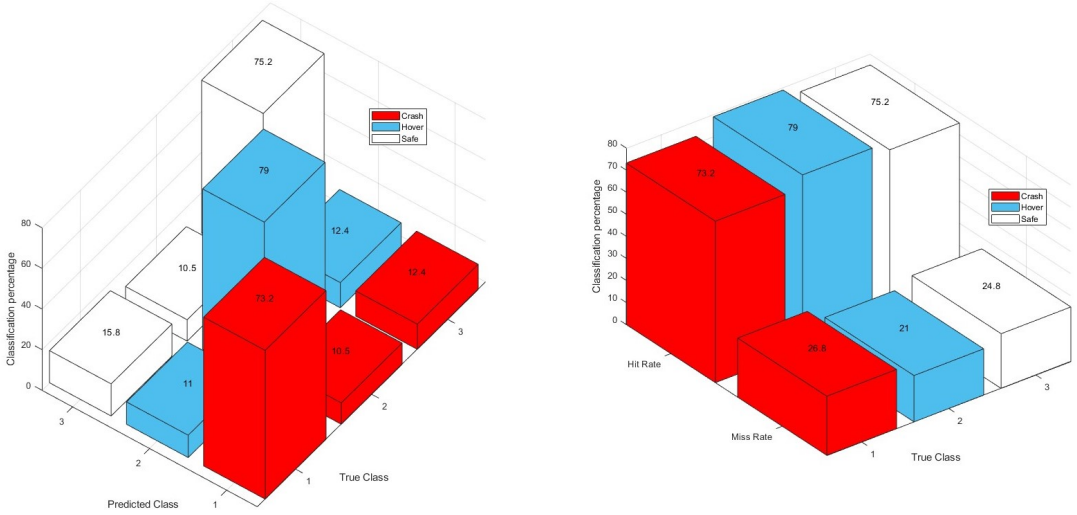
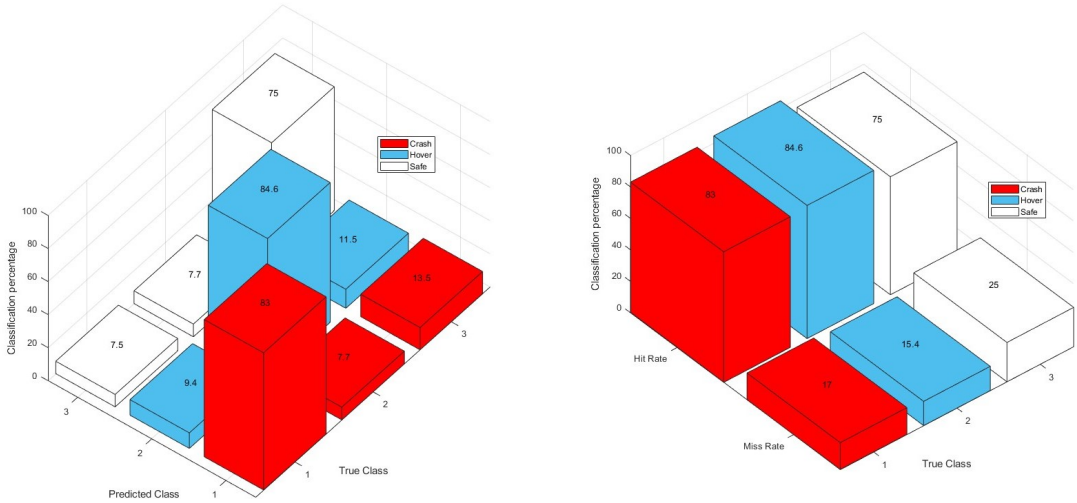


Figure 4.16: Bagged Trees Model (updated cost matrix): Test Results



4.3. Adaptive IDMF

The system model can now use the trained classifier to adaptively set the values for the hyperparameters based on an assumed height. As stated in Table 3.1, the UAV is aware of its position with the help of the IMU. Thus, this value should be taken into consideration by the trained model to give an updated assumption of the height of the platform. Detailed description of how the classifier is incorporated and implemented is addressed in the Algorithms 3 and 4.

4.3.1. Incorporation of the Landing Classifier in the System Model

Before landing commences, there is a set of ideal hyperparameters generated. It is assumed that the maximum height that the landing platform (h_u) can take is 2 meters as the starting height of descent of the UAV (z_o) is 2.5 meters. During the training, using feature ranking (Chi2), it was observed how PID gains (K_p, K_i) had a higher importance score than number of points tracked and the reset threshold. This

information was used to divide the hyperparameter finding process into 2 halves. The steps taken to generate the ideal set of hyperparameters is encapsulated in Algorithm 3. The inputs and the output of the classification model needed is defined in Figure 4.17. Using the classification output, the set of hyperparameters which provide safe landing for the UAV is sent to the Simulink System.

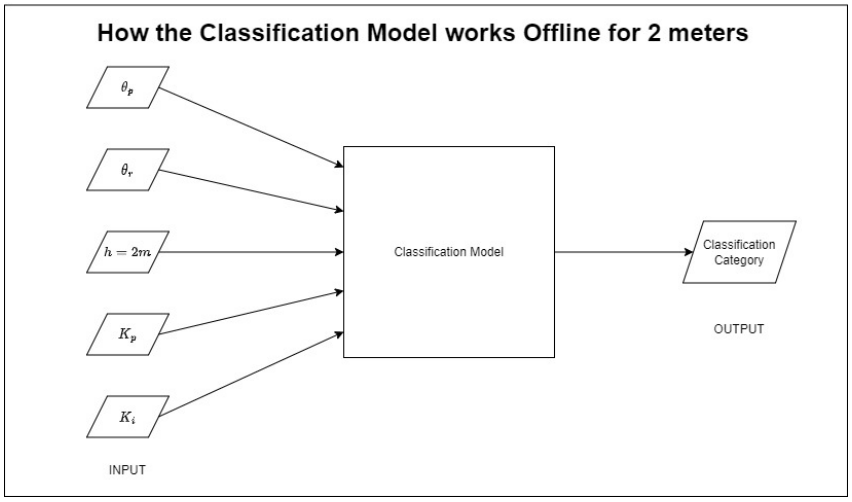


Figure 4.17: Working of the Classification Model Offline

Algorithm 3 : Finding the optimal hyperparameters offline

Ensure: classificationmodel is loaded from Classification Learner
$K_p, K_i \leftarrow [2, 20]$ ▷ the range of values PID gains can take
$\theta_p, \theta_r \leftarrow [10, 30]$ ▷ the range of values points tracked and reset threshold can take
height $\leftarrow 2$ ▷ maximum value of the platform
gain matrix $\leftarrow ndgrid$(range of values K_p, K_i) ▷ creates a grid with all combinations of K_p&K_i
$X \leftarrow$ [height, gain matrix, min(θ_p), min(θ_r)]
$y \leftarrow$ classificationmodel.predictFcn (X)
$X = [X, y]$
for j = 1: length(X) **do**
  if $y(j)$ == safe landing **then**
    $op1 = X(j)$
  end if
end for
$op1_{(K_p,K_i)} \leftarrow$ smallest gain values ($op1$)
range matrix $\leftarrow ndgrid$(range of values θ_p, θ_r)
$X \leftarrow$ [height, $op1_{(K_p,K_i)}$, range matrix]
$y \leftarrow$ classificationmodel.predictFcn (X)
$X = [X, y]$
for j = 1: length(X) **do**
  if $y(j)$ == safe landing **then**
    $op2 = X(j)$
  end if
end for
$op2_{small} \leftarrow$ smallest values of θ_p, θ_r ($op2$) ▷ Optimal smallest-valued hyperparameters that provide safe landing

Algorithm 3 also addresses how the smallest values are selected when there are multiple sets of hyperparameters providing safe landing. The smallest values are taken because this will limit the drastic variation when the hyperparameter updation occurs. Figure 4.18 shows the variables that are

needed for adaptive hyperparameter generation. This depicts adaptive hyperparameters function as a black box which needs three inputs, that is, the UAV's current location, the current reset count and the set of hyperparameters that are already generated. Algorithm 4 talks in detail on how the optimal hyperparameters are generated in the adaptive hyperparameter function.

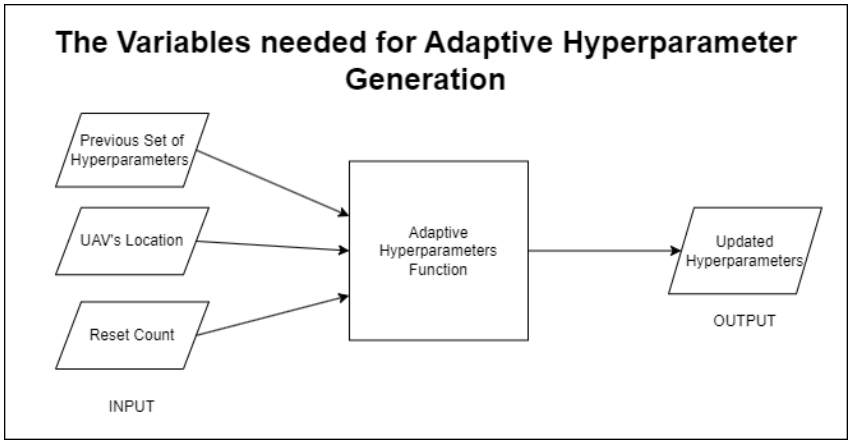


Figure 4.18: Basic Working for Adaptive Hyperparameter Generation

Algorithm 4 : Generating adaptive hyperparameters while landing

Ensure: classificationmodel is loaded, ideal hyperparameters ($op2_{small}$) is sent to system
$K_p, K_i \leftarrow [2, 20]$ ▷ the range of values PID gains can take
$\theta_p, \theta_r \leftarrow [10, 30]$ ▷ the range of values points tracked and reset threshold can take
gain matrix $\leftarrow ndgrid$(range of values K_p, K_i)
currenth $\leftarrow$ height of the UAV
estimatedh $\leftarrow$ height used for optimal hyperparameters
currentreset $\leftarrow$ current value of reset
safereset $\leftarrow \theta_r$ from $op2_{small}$
delh $\leftarrow 0.1$ ▷ the Δh considered for the hyperparameters to get updated
if currenth - estimatedh == delh && currentreset $\leq$ safereset **then**
  $X \leftarrow$ [currenth - delh, gain matrix, min(θ_p), min(θ_r)]
  $y \leftarrow$ classificationmodel.predictFcn (X)
  for j = 1: length(X) **do**
    if $y(j)$ == safe landing **then**
      $op1 \leftarrow X(j)$
    end if
  end for
  $op1_{(K_p,K_i)} \leftarrow$ smallest gain values ($op1$)
  range matrix $\leftarrow ndgrid$(range of values θ_p, θ_r)
  $X \leftarrow$ [currenth - delh, $op1_{(K_p,K_i)}$, range matrix]
  $y \leftarrow$ classificationmodel.predictFcn (X)
  $X \leftarrow [X, y]$
  for j = 1: length(X) **do**
    if $y(j)$ == safe landing **then**
      $op2 \leftarrow X(j)$
    end if
  end for
  $op2_{small} \leftarrow$ smallest values of θ_p and θ_r ($op2$)
end if
$op2_{small}$ is sent to the system. ▷ smallest-valued hyperparameters for safe landing

4.3.2. Implementing Adaptive IDMF

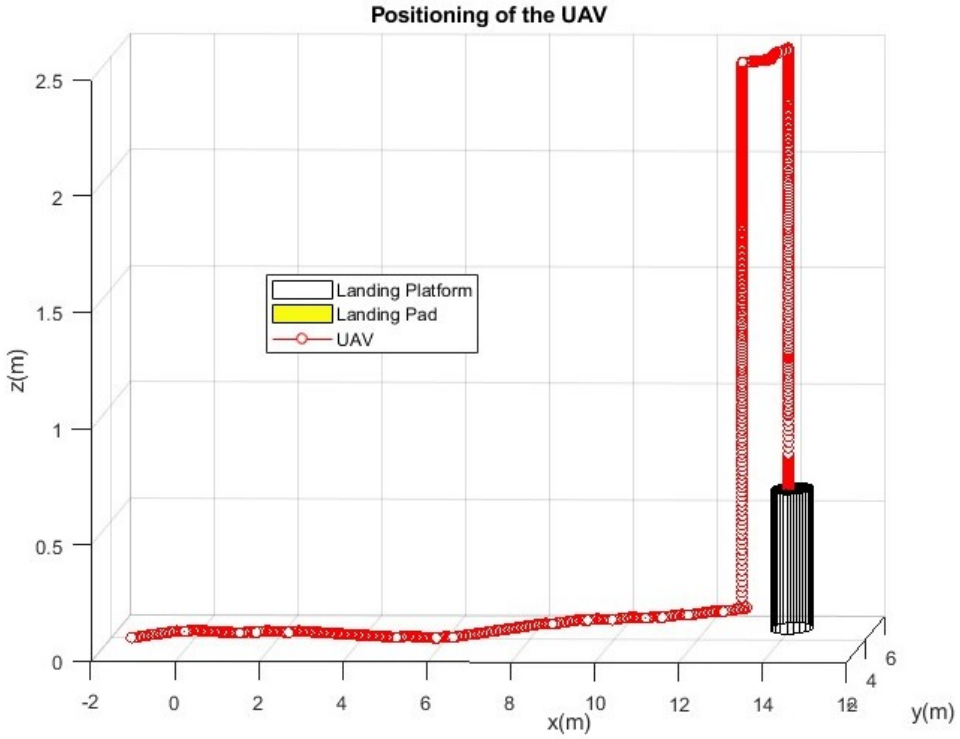


Figure 4.19: Results on Implementing Adaptive Hyperparameters

Figure 4.19 shows an example on how the landing algorithm works on a platform of 0.6 meter height. The system parameter values were taken from Table 3.2. When the UAV is close to the landing platform, it is observed how there are multiple iterations showing that the UAV starts to safe land. Due to the adaptive hyperparameter selection, there is an estimate of the height of the landing platform that is also generated. The next section will compare the Adaptive IDMF with the previously tested methods such as IDMF.

4.4. Adaptive IDMF vs IDMF

In this section, the performance of the proposed Adaptive IDMF is compared versus the IDMF algorithm in terms of the following metrics:

1. Safe landing probability
2. Time taken to safely land
3. Simulation Time

4.4.1. Safe Landing Probability

The Adaptive IDMF was tested for the range of $[h_l,h_u]$ = [0.2,2] meters of the landing platform with the landing design as the 6 concentric circles on textured background. It is observed how there is only one instance that the UAV crashes. This is might be due to a obstructed vision of the landing platform when the platform is of the height 1.6 meters to 2 meters. The results are tabulated in Table 4.8. Adaptive IDMF is able to outperform the IDMF algorithm as the probability of safe landing is higher compared to a constant hyperparameter IDMF as seen in Figure 4.20.

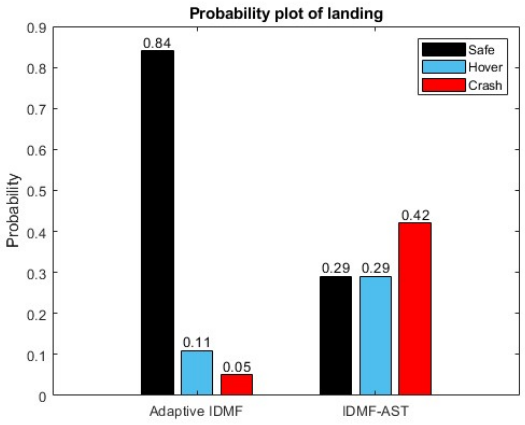
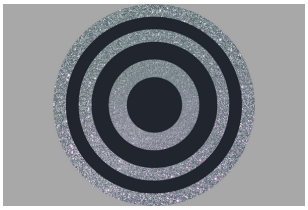


Figure 4.20: Probability of Landing Comparing Adaptive IDMF and IDMF Algorithm

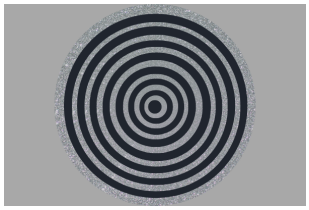
Height	Landing
2.0	safe
1.9	safe
1.8	safe
1.7	safe
1.6	crash
1.5	safe
1.4	safe
1.3	hover
1.2	safe
1.1	safe
1	safe
0.9	safe
0.8	safe
0.7	safe
0.6	safe
0.5	safe
0.4	safe
0.3	safe
0.2	safe

Table 4.8: Landing Results for Improved IDMF

Therefore, it can be inferred that the Adaptive IDMF is able to attain the goal of improving the probability of safe landing. The permanence of the proposed Adaptive IDMF is compared versus the IDMF algorithm for the two landing designs in Figure 4.21.



(a) 3 concentric circles



(b) 8 concentric circles

Figure 4.21: Landing Platform Designs Used to Test Adaptive IDMF

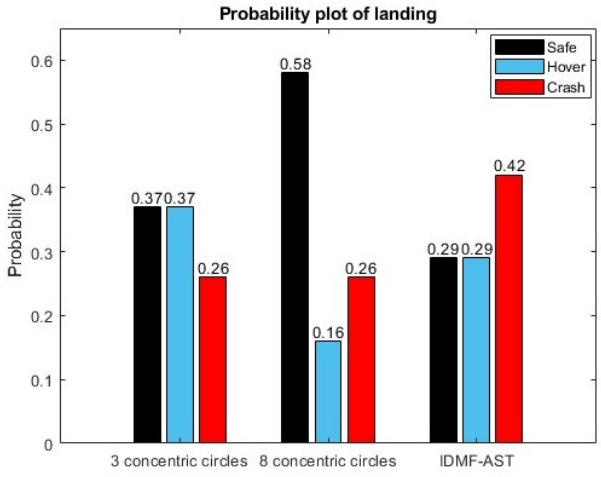


Figure 4.22: Probability Plot of the Adaptive Algorithm with Different Landing Designs

Figure 4.22 shows how Adaptive IDMF is able to provide higher probability of safe landing even though the classification model is tested on a different landing design. Hence, using Adaptive IDMF is beneficial compared to a constant hyperparameter implementation of IDMF.

4.4.2. Time Taken to Safely Land

To observe if Adaptive IDMF impacts the time taken to land, the average time taken for the safe landing of IDMF algorithm was compared with the time taken to land using Adaptive IDMF. The individual data points are tabulated in A.2. This comparison is summarized in Figure 4.23. As seen in figure, Adaptive IDMF has a lower time measurement. Therefore, Adaptive IDMF is able to provide not only higher probability of safe landing but also faster safe landing. At $h = 2$ meters, it is observed that the average landing time is higher compared to that of Adaptive IDMF. This is due to the hyperparameter selection that is done offline. As observed in Algorithm 3, the minimum values of the hyperparameters specifically PID gains are selecting. Thus, making it slower for the initial height assumption of the landing platform.

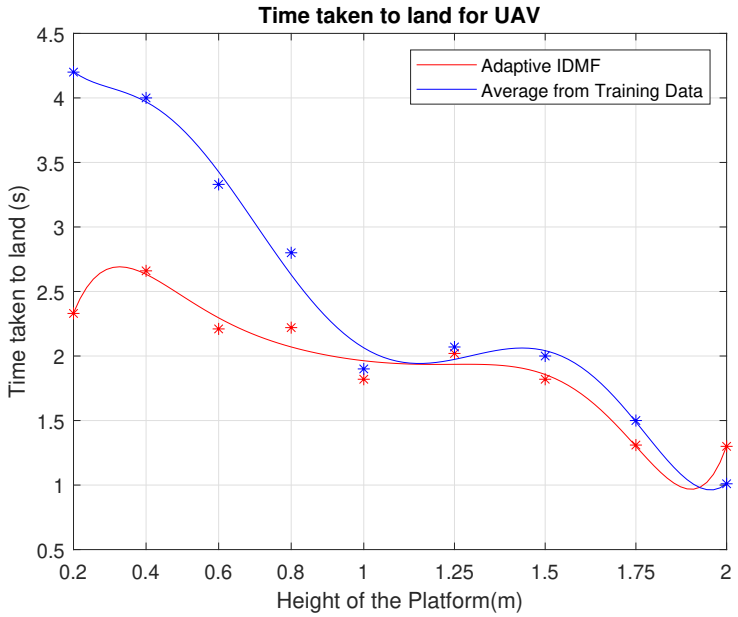


Figure 4.23: Performance of the Adaptive IDMF vs IDMF in terms of Time Taken

4.4.3. Simulation Time

Due to the adaptive nature of the proposed algorithm, there might be a possibility of a slower run time of the Simulink system. To observe this, the two methods were compared in terms of the time taken for initialization, compilation and total run time. The results are tabulated as follows:

Phase	Adaptive IDMF (seconds)	IDMF (seconds)
Total time	390.549	396.987
Simulation Phase	373.241	377.722
Compile Phase	16.079	17.090
Initialization Phase	1.207	2.083
Termination Phase	0.0203	0.091

Table 4.9: Simulation Time Comparison

In Table 4.9, it is noted how Adaptive IDMF is able to perform much faster compared to IDMF in all phases of the simulation. One thing to be noted, when the Adaptive IDMF is compiled for the first time, to combine the trained model in the system takes on average 10 times the time in compile phase (149

seconds). But this is due to the limitation of the software. Therefore, once the trained model is loaded, this will not affect the behaviour of the system.

Thus, it can be inferred from three metrics used, Adaptive IDMF is able to perform better compared to IDMF in terms of higher safe landing probability, faster safe landing and lesser simulation and compilation time.

5

Conclusion and Future Work

5.1. Conclusion

The research of implementing OF for Autonomous Landing of an UAV provided the following insights:

- There is research geared towards incorporating the mechanisms that insects around us implement for safe landing of UAVs. By doing so, the complexity of the algorithm implemented reduces.
- OF can be used for detection and maneuvering towards the landing platform using the jump in its value once the platform is in view.
- Usage of IDMF-AST is better suited for safe landing compared to IDMI when there is a possibility of an elevated landing platform.
- The improved IDMF looked into different landing designs which affected the probability of safe landing. The trends observed on changing the hyperparameters given a landing design affected the outcome at a higher randomness. Thus, finding the right hyperparameters was crucial.
- Using a Classification Model along with iterative hyperparameter selection proved to be the best way to identify the ideal hyperparameters that can guarantee safe landing given an assumption of the landing platform height.
- The Adaptive IDMF was able to double the probability of safe landing compared to IDMF and also reduce the probability of crash occurring.

5.2. Future Work

Some areas of this project that will be looked into are:

- Expand the algorithm such that it can guarantee safe landing with a varying location of the landing platform while using OFD.
- Observe how the system reacts in the case of turbulent wind conditions.
- Implement other control algorithms such as Sliding Mode Control and Linear Quadratic Regulator in place of PID and investigate the difference in the safe landing probability.

Bibliography

[1] A. Skondras, E. Karachaliou, I. Tavantzis, *et al.*, "Uav mapping and 3d modeling as a tool for promotion and management of the urban space," *MDPI*, May 2022. [Online]. Available: https://www.mdpi.com/2504-446X/6/5/115.

[2] N. Ahmadian, G. J. Lim, M. Torabbeigi, and S. J. Kim, "Smart border patrol using drones and wireless charging system under budget limitation," *Computers Industrial Engineering*, vol. 164, p. 107 891, 2022, ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2021.107891. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835221007956.

[3] C. Kyrkou and T. Theocharides, "Emergencynet: Efficient aerial image classification for drone-based emergency monitoring using atrous convolutional feature fusion," *CoRR*, vol. abs/2104.14006, 2021. arXiv: 2104.14006. [Online]. Available: https://arxiv.org/abs/2104.14006.

[4] J. Reed, *Amazon prime air is set to start drone deliveries this year*, Jun. 2022. [Online]. Available: https://www.aviationtoday.com/2022/06/16/amazon-prime-air-drone-deliveries-2022/.

[5] A. Chriki, H. Touati, H. Snoussi, and F. Kamoun, "Uav-based surveillance system: An anomaly detection approach," pp. 1–6, 2020. DOI: 10.1109/ISCC50000.2020.9219585.

[6] C. Yinka-Banjo and O. Ajayi, "Sky-farmers: Applications of unmanned aerial vehicles (uav) in agriculture," G. Dekoulis, Ed., 2019. DOI: 10.5772/intechopen.89488. [Online]. Available: https://doi.org/10.5772/intechopen.89488.

[7] M. Stewart and S. Martin, "In: Unmanned aerial vehicles unmanned aerial vehicles: Fundamentals, components, mechanics, and regulations," in Dec. 2020, pp. 1–70, ISBN: 978-1-53618-900-1.

[8] A. Crosby, *Fixed-wing vs multirotor drones: Which is better?* Apr. 2022. [Online]. Available: https://geonadir.com/fixed-wing-vs-multirotor-drones-which-is-better/.

[9] B.-M. Min, M.-J. Tahk, H.-C. Shim, and H. Bang, "Guidance law for vision-based automatic landing of uav," *International Journal of Aeronautical and Space Sciences*, vol. 8, pp. 46–53, Jun. 2007. DOI: 10.5139/IJASS.2007.8.1.046.

[10] L. Besnard, Y. B. Shtessel, and B. Landrum, "Control of a quadrotor vehicle using sliding mode disturbance observer," pp. 5230–5235, 2007. DOI: 10.1109/ACC.2007.4282421.

[11] A. Nemati, M. Sarim, M. Hashemi, *et al.*, "Autonomous navigation of uav through gps-denied indoor environment with obstacles," Jan. 2015. DOI: 10.2514/6.2015-0715.

[12] R. P. Padhy, S. Verma, S. Ahmad, S. K. Choudhury, and P. K. Sa, "Deep neural network for autonomous uav navigation in indoor corridor environments," *Procedia Computer Science*, vol. 133, pp. 643–650, 2018, International Conference on Robotics and Smart Manufacturing (RoSMa2018), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.07.099. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918310524.

[13] H. X. Pham, H. M. La, D. Feil-Seifer, and L. Van Nguyen, "Reinforcement learning for autonomous uav navigation using function approximation," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2018, pp. 1–6. DOI: 10.1109/SSRR.2018.8468611.

[14] D. Saccani and L. Fagiano, "Autonomous uav navigation in an unknown environment via multi-trajectory model predictive control," in *2021 European Control Conference (ECC)*, 2021, pp. 1577–1582. DOI: 10.23919/ECC54610.2021.9655166.

[15] Y. Lu, Z. Xue, G.-S. Xia, and L. Zhang, "A survey on vision-based uav navigation," *Geo-spatial Information Science*, vol. 21, no. 1, pp. 21–32, 2018. DOI: 10.1080/10095020.2017.1420509. [Online]. Available: https://doi.org/10.1080/10095020.2017.1420509.

[16] A. Gautam, P. Sujit, and S. Saripalli, "A survey of autonomous landing techniques for uavs," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 1210–1218. DOI: 10.1109/ICUAS.2014.6842377.

[17] A. Borowczyk, N. Tien, A. Nguyen, D. Nguyen, D. Saussie, and J. Le Ny, "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *IFAC-PapersOnLine*, Nov. 2016.

[18] A. Paris, A. Tagliabue, and J. How, "Autonomous mav landing on a moving platform with estimation of unknown turbulent wind conditions," Jan. 2021. DOI: 10.2514/6.2021-0378.

[19] *Unmanned aerial vehicles - ready for take-off?* [Online]. Available: https://www.dhl.com/global-en/home/insights-and-innovation/thought-leadership/trend-reports/unmanned-aerial-vehicles.html.

[20] G. Panahandeh, I. Skog, and M. Jansson, "Calibration of the accelerometer triad of an inertial measurement unit, maximum likelihood estimation and cramér-rao bound," pp. 1–6, 2010. DOI: 10.1109/IPIN.2010.5646832.

[21] J. Burgués and S. Marco, "Environmental chemical sensing using small drones: A review," *Science of The Total Environment*, vol. 748, p. 141 172, 2020, ISSN: 0048-9697. DOI: https://doi.org/10.1016/j.scitotenv.2020.141172. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S004896972034701X.

[22] M. Alijani and A. Osman, "Autonomous landing of uav on moving platform: A mathematical approach," pp. 1–6, 2020.

[23] J. Arrigo, V. Chen, Z. K. Guha, A. Kruthiventy, R. Valencia, and B. Lai, "A visionless autonomous quadcopter landing," in *2019 IEEE MIT Undergraduate Research Technology Conference (URTC)*, 2019, pp. 1–4. DOI: 10.1109/URTC49097.2019.9660468.

[24] Z. Ding, H. Cai, and H. Yang, "An improved multi-position calibration method for low cost micro-electro mechanical systems inertial measurement units," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 229, pp. 1919–1930, Aug. 2015. DOI: 10.1177/0954410014562262.

[25] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for imu calibration without external equipments," pp. 3042–3049, 2014. DOI: 10.1109/ICRA.2014.6907297.

[26] B. Erginer and E. Altug, "Modeling and pd control of a quadrotor vtol vehicle," pp. 894–899, 2007. DOI: 10.1109/IVS.2007.4290230.

[27] S. Suresh and N. Kannan, "Direct adaptive neural flight control system for an unstable unmanned aircraft," *Applied Soft Computing*, vol. 8, no. 2, pp. 937–948, 2008, ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2007.07.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156849460700083X.

[28] T. Lee and Y. Kim, "Nonlinear adaptive flight control using backstepping and neural networks controller," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 4, pp. 675–682, 2001. DOI: 10.2514/2.4794. [Online]. Available: https://doi.org/10.2514/2.4794.

[29] E. Olson, "Apriltag: A robust and flexible visual fiducial system," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3400–3407, Jun. 2011. DOI: 10.1109/ICRA.2011.5979561.

[30] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, pp. 2280–2292, Jun. 2014. DOI: 10.1016/j.patcog.2014.01.005.

[31] J. Wubben, F. Fabra, C. T. Calafate, *et al.*, "A vision-based system for autonomous vertical landing of unmanned aerial vehicles," pp. 1–7, 2019. DOI: 10.1109/DS-RT47707.2019.8958701.

[32] M. Mondal, S. Shidlovskiy, and D. Shashev, "Camera assisted autonomous uav landing," *Proceedings of the 30th International Conference on Computer Graphics and Machine Vision (GraphiCon 2020). Part 2*, 2020. DOI: 10.51130/graphicon-2020-2-3-82.

[33] D. Maturana and S. Scherer, "3d convolutional neural networks for landing zone detection from lidar," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, pp. 3471–3478, Jun. 2015. DOI: 10.1109/ICRA.2015.7139679.

[34] Y. Shin, S. Lee, and J. Seo, "Autonomous safe landing-area determination for rotorcraft uavs using multiple ir-uwb radars," *Aerospace Science and Technology*, vol. 69, Jul. 2017. DOI: 10.1016/j.ast.2017.07.018.

[35] J. Kim, S. Woo, and J. Kim, "Lidar-guided autonomous landing of an aerial vehicle on a ground vehicle," in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2017, pp. 228–231. DOI: 10.1109/URAI.2017.7992719.

[36] J. Y. Lee, A. Y. Chung, H. Shim, C. Joe, S. Park, and H. Kim, "Uav flight and landing guidance system for emergency situations," *Sensors*, vol. 19, no. 20, 2019, ISSN: 1424-8220. DOI: 10.3390/s19204468. [Online]. Available: https://www.mdpi.com/1424-8220/19/20/4468.

[37] M. Alijani and A. Osman, "Autonomous landing of uav on moving platform: A mathematical approach," in *2020 International Conference on Control, Automation and Diagnosis (ICCAD)*, 2020, pp. 1–6. DOI: 10.1109/ICCAD49821.2020.9260498.

[38] M. Shah Alam and J. Oluoch, "A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (uavs)," *Expert Systems with Applications*, vol. 179, p. 115 091, 2021, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115091. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421005327.

[39] M. Kim and Y. Kim, "Multiple uavs nonlinear guidance laws for stationary target observation with waypoint incidence angle constraint," *International Journal of Aeronautical and Space Sciences*, vol. 14, pp. 67–74, Mar. 2013. DOI: 10.5139/IJASS.2013.14.1.67.

[40] J. Kennedy, "The visual responses of flying mosquitoes," *Proceedings of the Zoological Society of London*, vol. A109, pp. 221–242, Aug. 1939. DOI: 10.1111/j.1096-3642.1940.tb00831.x.

[41] N. Franceschini, F. Ruffier, J. Serres, and S. Viollet, "Optic flow based visual guidance: From flying insects to miniature aerial vehicles," in Jan. 2009, ISBN: 978-953-7619-41-1. DOI: 10.5772/6491.

[42] T. Collett, "Insect vision: Controlling actions through optic flow," *Current biology : CB*, vol. 12, R615–7, Oct. 2002. DOI: 10.1016/S0960-9822(02)01132-6.

[43] P. Xie, O. Ma, and Z. Zhang, "A bio-inspired approach for uav landing and perching," Aug. 2013, ISBN: 978-1-62410-224-0. DOI: 10.2514/6.2013-5108.

[44] M. Alkowatly, V. Becerra, and W. Holderbaum, "Bioinspired autonomous visual vertical control of a quadrotor unmanned aerial vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 38, pp. 249–262, Feb. 2015. DOI: 10.2514/1.G000634.

[45] M. Tropea and A. Serianni, "Bio-inspired drones recruiting strategy for precision agriculture domain," in *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2020, pp. 1–4. DOI: 10.1109/DS-RT50469.2020.9213516.

[46] Mathworks-Robotics, *Mathworks-robotics/obstacle-avoidance-using-camera: Obstacle avoidance for uav/ugv using optical flow algorithm calculated using the vehicle's front view camera.* [Online]. Available: https://github.com/mathworks-robotics/obstacle-avoidance-using-camera.

[47] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, pp. 2149–2154.

[48] M. Quigley, K. Conley, B. Gerkey, *et al.*, "Ros: An open-source robot operating system," vol. 3, Jan. 2009.

[49] MATLAB, *version 9.12.0 (R2022a)*. Natick, Massachusetts: The MathWorks Inc., 2022.

[50] S. Documentation, *Simulation and model-based design*, 2022. [Online]. Available: https://www.mathworks.com/products/simulink.html.

[51] P. Turaga, R. Chellappa, and A. Veeraraghavan, *Advances in video-based human activity analysis: Challenges and approaches*, Jun. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245810800075.

[52] J. Lee and A. Bovik, "Chapter 19 - video surveillance," in *The Essential Guide to Video Processing*, A. Bovik, Ed., Boston: Academic Press, 2009, pp. 619–651, ISBN: 978-0-12-374456-2. DOI: https://doi.org/10.1016/B978-0-12-374456-2.00022-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123744562000220.

[53] D. Fleet and Y. Weiss, "Optical flow estimation," in *Handbook of Mathematical Models in Computer Vision*, N. Paragios, Y. Chen, and O. Faugeras, Eds. Boston, MA: Springer US, 2006, pp. 237–257, ISBN: 978-0-387-28831-4. DOI: 10.1007/0-387-28831-7_15. [Online]. Available: https://doi.org/10.1007/0-387-28831-715.

[54] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," vol. 81, Apr. 1981.

[55] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981, ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(81)90024-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370281900242.

[56] C. Tomasi and T. Kanade, "Detection and tracking of point features," International Journal of Computer Vision, Tech. Rep., 1991.

[57] C. S. Royden and M. A. Holloway, "Detecting moving objects in an optic flow field using direction- and speed-tuned operators," *Vision Research*, vol. 98, pp. 14–25, 2014, ISSN: 0042-6989. DOI: https://doi.org/10.1016/j.visres.2014.02.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0042698914000455.

[58] H. W. Ho, G. C. H. E. de Croon, E. van Kampen, Q. Chu, and M. Mulder, "Adaptive control strategy for constant optical flow divergence landing," *CoRR*, vol. abs/1609.06767, 2016.

[59] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," vol. 2, Nov. 2005, 1508–1515 Vol. 2, ISBN: 0-7695-2334-X. DOI: 10.1109/ICCV.2005.104.

[60] H. W. Ho, G. Croon, and Q. Chu, "Distance and velocity estimation using optical flow from a monocular camera," *International Journal of Micro Air Vehicles*, vol. 9, pp. 198–208, Mar. 2017. DOI: 10.1177/1756829317695566.

[61] Y. Zhou, H. Ho, and Q. Chu, "Extended incremental nonlinear dynamic inversion for optical flow control of micro air vehicles," English, *Aerospace Science and Technology*, vol. 116, 2021, ISSN: 1270-9638. DOI: 10.1016/j.ast.2021.106889.

[62] S. Lange, N. Sünderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments," Jul. 2009, pp. 1–6.

[63] W. Badr, *Having an imbalanced dataset? here is how you can fix it.* Dec. 2020. [Online]. Available: https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb.

[64] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002. DOI: 10.1613/jair.953. [Online]. Available: https://doi.org/10.16132Fjair.953.

A

Appendix

A.1. Concentric Circles Data

Table A.1: Test Results for Different Types of Concentric Circles

Height	3t	3	3f	4t	4	4f	5t	5	5f
1	crash	safe	safe	safe	safe	safe	safe	safe	safe
0.8	crash	crash	safe	safe	crash	safe	safe	safe	crash
0.6	safe	hover	crash	safe	safe	safe	safe	crash	safe
0.4	safe	crash	safe	safe	crash	safe	safe	crash	crash
0.24	safe	hover	safe	safe	safe	crash	safe	safe	safe

Height	6t	6	6f	7t	7	7f	8t	8	8f
1	safe	safe	crash	safe	crash	crash	crash	crash	safe
0.8	safe	crash	safe	safe	safe	hover	safe	hover	safe
0.6	safe	safe	crash	safe	crash	crash	hover	safe	crash
0.4	safe	safe	safe	safe	safe	safe	safe	safe	crash
0.24	safe	safe	safe	safe	crash	safe	crash	crash	safe

A.2. Time Taken to Land using Adaptive IDMF vs IDMF

Table A.2: Time Taken for Different Elevated Platforms

Height	Adaptive IDMF (s)	Average for Training Data (s)
2.0	1.3	1.01
1.75	1.31	1.5
1.5	1.82	2
1.25	2.02	2.07
1	1.82	1.9
0.8	2.22	2.8
0.6	2.21	3.33
0.4	2.66	4
0.2	2.33	4.2