PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

Autonomous UAV Takeoff and Landing on Moving Platforms

Submitted by

Anushka Keshri (PES1UG22EC042)
Kiran Easwar (PES1UG22EC123)
Krutharth M C (PES1UG22EC131)
Mohammed Yaseen Lohar (PES1UG22EC163)

August 2024 - December 2025

under the guidance of

Dr. T S Chandar

Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PES University

Bengaluru -560085

FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
PROGRAM B. TECH

CERTIFICATE

This is to certify that the Report entitled

Autonomous UAV Takeoff and Landing on Moving Platforms

is a bonafide work carried out by

Anushka Keshri (PES1UG22EC042)
Kiran Easwar (PES1UG22EC123)
Krutharth M C (PES1UG22EC131)
Mohammed Yaseen Lohar (PES1UG22EC163)

In partial fulfillment for the completion of 7th semester course work in the Program of Study B.Tech in Electronics and Communication Engineering under rules and regulations of PES University, Bengaluru during the period Aug 2024 – Dec 2025. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The report has been approved as it satisfies the academic requirements in respect of project work.

Signature with date	*Signature with date & Seal*	*Signature with date & Seal*
Dr. T S Chandar	*Dr. Shikha Tripathi*	
Internal Guide	*Chairperson*	*Dean -Faculty of Engg. and Technology*

Name and Signature of the Examiners:

Names Signature

DECLARATION

We, **Anushka Keshri, Kiran Easwar, Krutharth M C, Mohammed Yaseen Lohar,** hereby declare that the report entitled, ***Autonomous UAV Takeoff and Landing on Moving Platforms,*** is an original work done by us under the guidance of **Dr. T S Chandar, Professor, Department of ECE,** is being submitted in partial fulfillment of the requirements for completion of project work in the Program of Study B.Tech in Electronics and Communication Engineering.

PLACE:
DATE:

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our project guide, **Dr. Chandar T S**, whose guidance has been nothing short of extraordinary. His exceptional technical expertise, constant encouragement, and unwavering support elevated this project far beyond what we could have achieved on our own. His ability to simplify complex concepts, provide precise direction, and inspire confidence at every stage has been truly remarkable. At every hurdle, his clarity, patience, and insight acted as a guiding light, turning challenges into opportunities for growth. His mentorship has been a driving force throughout this work, and we are deeply grateful for the dedication and brilliance he consistently demonstrated.

I would also like to extend my sincere thanks to the panel members for their valuable suggestions and critical insights during each review. Their inputs helped refine the design, strengthen the methodology, and significantly improve the overall outcome of the project. Their guidance was truly appreciated.

We extend our heartfelt thanks to **Dr. J Suryaprasad**, Vice-Chancellor, PES University for his support. We are also grateful to **Dr. Shikha Tripathi**, Chairperson, Department of Electronics and communication, PES University for her support in facilitating the completion of this project.

Finally, we thank our parents, friends and to all the people who were directly or indirectly involved, for their cooperation and support for guiding and supporting us in completion of this project work.

Anushka Keshri (PES1UG22EC042)
Kiran Easwar (PES1UG22EC123)
Krutharth M C (PES1UG22EC131)
Mohammed Yaseen Lohar (PES1UG22EC163)

ABSTRACT

Autonomous recovery on moving platforms is essential for extending UAV mission endurance and enabling continuous field operations. This work introduces a modular control framework that fuses monocular camera inputs with onboard inertial telemetry to reliably track ground vehicles exhibiting irregular motion. The system utilizes a variable-speed descent logic and a motion-responsive trajectory generator to ensure precise touchdown even during target acceleration. Validated through extensive ArduPilot and Gazebo simulations, the proposed method utilizes System Identification to tune control parameters, achieving centimeter-level accuracy during landing trials. The results demonstrate a resilient architecture capable of handling sensor noise and communication delays, laying the groundwork for real-world deployment on dynamic mobile bases.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

Chapter 1: INTRODUCTION

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have rapidly evolved from being just hobbyist toys or military equipment to becoming essential tools for everyday work. We now see them extensively used for inspecting critical infrastructure, helping in search and rescue missions, delivering packages, conducting surveys and much more.

However, despite their versatility, the utility of multirotor drones is severely restricted by a fundamental physical constraint which is often referred to as the tyranny of the rocket equation. Multirotor UAVs must constantly consume battery to hover in the air to counteract gravity. To increase a multirotor's range, one must increase its battery capacity. However, adding batteries adds mass, lifting that mass requires more thrust, which in turn consumes more energy. This is a vicious cycle where you are not actually increasing the capabilities of the multirotor but instead just making the multirotor bulkier and more prone to failure. Consequently, when designing commercial multirotor engineers are forced to make compromises and trade-offs between flight time, payload capacity, size, and the speed of the multirotor. This ultimately limits their use in real world deployment and prevents their widespread adoption in various industries.

To circumvent this power-to-weight paradox without waiting for a breakthrough in battery technology, we can use a system which combines the Multirotor with a ground-based moving vehicle. This combines the extended range and payload capacity of a vehicle with the agility and aerial perspective of a UAV, enhancing the mission capabilities of both and enabling complex missions which would not be possible with either individually.

The ground-based vehicle can carry the UAV around and then deploy it when necessary, and when the mission of the UAV is finished, it should be able to land back on the vehicle.

An inherent limitation still exists during the landing phase after it completes its mission. Even landing on a stationary target with reasonable accuracy is inherently a non-trivial task requiring multiple sensors to be placed on the UAV, increasing weight and complexity, but through decades of research, multiple solutions have been found to enable the robust landing of the UAV on a stationary target. In our case, this will require the vehicle to stop while the UAV locks onto it and lands safely on top of the vehicle.

However, in the scenario of time-critical applications where the ground vehicle has its own important mission, such as a system of an ambulance carrying a UAV deployed to an area of emergency, stopping the ambulance to wait for the UAV to land on top of it hinders the mission and wastes valuable time, which is not acceptable. Thus, we need the UAV to be able to land on the ground vehicle even when it is moving. This enables both the vehicle and the UAV to complete their missions without hindering each other as a result enhancing the capabilities of the system. To accomplish this, we need to tackle a few challenges namely – State estimation and tracking, control and finally landing policy.

1.1 State Estimation and Tracking

The first challenge lies in finding out accurately where the vehicle is relative to the UAV. To find this out, a common approach is to attach an RGB camera on the UAV facing downward add a visual fiducial marker such as an ArUco tag to the landing surface of the vehicle. Using the fiducial marker, we can calculate the relative distance between the UAV and the vehicle.

This provides us with a good state estimate of where the UAV is with respect to the vehicle, but an inherit problem of this approach is that the state estimate is only as good as the detection of the ArUco tag. This is heavily dependent on environmental factors and if the ArUco tag is not detected, we will lose the state estimate. This is a greater problem with a moving platform as the vehicle can always accelerate out of the frame.

Thus, sensor fusion is essential to have a better state estimator. In this project report we will look at combining the state estimation of the fiducial marker the relative positions of the vehicle and UAV obtained by comparing odometry data of the vehicle with that of the UAV to have a better state estimator that can handle both noise in the system and low detection accuracy of the fiducial marker.

1.2 Control Approach

Once the relative position of the target has been calculated, the UAV must produce accurate motor commands to orient itself and continuously track the moving vehicle. The control problem is nontrivial in that the system must have sufficient robustness to manage three separate kinematic behaviors that can be shown by the ground vehicle, namely -

- The ground vehicle is stationary.
- The ground vehicle is cruising at a constant velocity.
- The vehicle is accelerating.

Each of these cases comes with its own set of challenges and complications. In this project report, we will explore how a cascaded Proportional-Integral-Derivative (PID) control system can handle all three cases and land on the vehicle with acceptable accuracy.

1.3 Landing Policy

The UAV needs to decide how to successfully locate and land on the vehicle after it has completed its mission. A key part of the landing policy will be balancing the rate of descent with the lateral error of the vehicle and what to do when the vehicle accelerates too quickly and moves out of frame of the UAVs camera. In this project report, we will look at an FSM based approach to handle the landing. Eliminating head-of-line blockage at the transport layer is a second reason. Multiple independent streams inside a single connection are now supported natively by QUIC. Packet loss on one stream does not affect other streams because each stream can be provided individually. Browsers frequently retrieve hundreds or even thousands of resources at once. Even with mild packet loss, QUIC's stream-based architecture guarantees better performance.

In this project report, we will develop a framework which will enable the UAV to take off autonomously from the vehicle while it's moving. Both the UAV and the vehicle will be able to complete their own respective missions, then the UAV will be able to land on the moving vehicle autonomously without any compromise in the mission of the vehicle.

Chapter 2: LITERATURE SURVEY

The autonomous landing of UAV'S is a complex yet interesting problem that could benefit us in many ways, sparing us the pain of precise manual control in complex scenarios. This autonomous approach wherein the UAV can make its own decisions involves many steps, including Take-off, localization using the onboard computer, rigorous tuning of the onboard flight controller, system identification of the higher-level control logic, and precision landing on the target platform, be it a stationary platform or a moving vehicle. Due to the constraints on parameters such flight duration, UAV weight, and onboard computing load, it is essential that we choose efficient methods in all areas of project development. There are lots of sensors that are viable for localization with a target such as IR sensors, Cameras, UWB (Ultra-Wide Band) Tags, GPS (Global Positioning System) etc. But each of these sensors has its own pros and cons, the most noticeable one being the range of operation of these sensors. Among these sensors one of the most reliable and efficient sensors for short range localization is the camera sensor that is commonly used in various image detection and tracking projects. A camera sensor can be utilized in various methods depending on the detection algorithm used, such as color detection, using OpenCV to detect physical objects, frame analysis etc., but one such method that stands out in its own unique way is the use of "Fiducial Markers", the detection of fiducial markers is performed at short range and the pose details obtained from the marker have higher accuracy and the detection is quite robust.

To better understand the use and performance of various fiducial markers we look into the work done by Kyas and Springer [1], whose text provides a method for autonomous precision drone landing using a gimbal mounted camera to track a fiducial marker. The authors provide us an idea of the various fiducial markers available for us to use, such as April Tag (48h12, 24h10), Whycode (Original, Ellipse, Multi). The detection and analysis of these markers suggested that April Tag 48h12 provided the most accurate landings (closest distance to the centre), April Tag 24h10 and WhyCode Ellipse had similar, slightly lower accuracy, it also suggested that April Tags are better for short range precision detection and WhyCode performed better at longer ranges, but not as well at shorter ranges, where the April Tags stood out. They introduced a method to solve the problem of the marker moving out of frame , by implementing a search phase where the gimbal tilts up and down continuously and drone yaws in place till the marker is reacquired in the camera frame, after which the drone flies quickly to the targe and localises with it and proceeds to land, the approach of a moving gimbal is beneficial as opposed to lots of other studies that performed detection with a camera fixed downwards. However, adding a gimbal adds weight and mechanical complexity, which is a demerit for smaller drones. Expanding on how these markers are used, Khazetdinov et al. [2] proposed an "Embedded ArUco" system, that explores the commonly faced trade-off between visibility and precision during marker detection. Standard sized markers become harder to detect when the UAV approaches the target during low altitude flight and landing especially, since the marker becomes to enlarged and exceeds the camera frame limits, this hinders the landing approach. The solution employed was to embed a smaller ArUco marker within the larger standard marker, specifically in the centre. This approach allowed them to detect the outer marker from a higher altitude, and when the marker size exceeds the frame limits at lower altitudes the inner marker proceeds to guide the UAV till it touches down. This is a highly effective, low-computation solution, though it relies entirely on the landing pad being pre-printed with this specific pattern. This dual detection approach allowed them to detect from as far as 30m using the outer marker and from as close as 0.2m using the smaller inner marker.

Wicaksana et al. [3] also took a vision-based tracking approach for indoor autonomous navigation, this was because traditional navigation methods such GPS fail indoors where signal locking is difficult. They propose a low cost and effective tracking algorithm using a camera sensor as opposed to expensive Lidar and UWB sensors. Aruco markers provide high precision since the detection extracts the corners of the marker from the images and gets the precise pose estimate, but they have a limited range (0.1-1.5m), whereas object detection algorithms like YOLOv8 have high detection range (0.3-8m), but lack the precision of Aruco markers, this is because the YOLOv8 model presents a bounding box around the target object which may not precisely represent the object’s dimensions. Therefore, they resort to combining both the methods to get a more effective object detection and tracking algorithm, which has the precision of an Aruco marker and the long range of the YOLO8v model, effectively mitigating one algorithm’s flaw using the other. The study concludes by comparing the performance of each algorithm individually and combined. Using only Aruco markers resulted in a low range (0.1-1.5m) high precision landing, using the YOLOv8 model resulted in a longer range (0.3-8m) detection with lesser accuracy, whereas using both these methods combined allowed them to see high precision and range (0.1-8m) at the same time.

Similarly, Ripke and Morelli [4] explore vision-based tracking and landing on a moving UGV (Unmanned Ground Vehicle), this too is a low-cost implementation as it utilises only a camera sensor to achieve this using a classical computer vision algorithm. The landing target is a red circle in this case, a “blob” to be precise. Their algorithm for blob detection is simple, images are converted to HSV colour space and a gaussian blur is applied to remove noise from the images, the target colour is isolated and finally it finds circular contours to determine the (x,y) position of the centre of the blob. The x,y plane lateral error between the centre of the blob and the UAV is minimised using a PID controller, furthermore the authors purposely chose not to landing during the planar localization phase but instead begin decent when the drone is centred exactly above the centre of the blob. The system achieved high precision in 5 successful landings out of 9 attempts from an altitude of 0.9m, whereas it achieved 5 successful landings out of 7 attempts from altitudes of 1.5m and 2.15m, with drawbacks being the drone had to abort landing and restart localization in the x,y plane during windy conditions and due to drift. Wang and McKiver [5] implemented an advanced visual based tracking and landing algorithm using Deep Convolutional Neural Networks for target detection and a linear predictive algorithm for determining the trajectory of the drone. The system combines the popular detection model YOLOv3 and R-CCN to detect landing markers. The feature extraction network was pretrained and the YOLO model was retrained based on landing data to achieve a final landing accuracy of greater than 99%. The system estimates the real time position and relative velocity of the target from the drone. They adopt two methods of control, first one is velocity control which reduces relative velocity between the drone and the platform, second controller is a position controller that reduces the relative position between the drone and the platform. The velocity controller proved to be more robust and tracked the marker at higher speeds, but suffer oscillations. The position controller was much smoother but lacked robustness and couldn’t keep up with higher drone speeds like the velocity controller could. Final landings gave an error of less than 0.3m on static platforms with both controllers. The velocity controller achieved an accuracy of less than

0.3m with the max speed of 4m/s.

Once we choose the most effective and optimal detection and tracking methods, the challenge shifts to controlling the UAV to perform object following based on the tracking of the object. It is crucial that we determine the most optimal path that the UAV take to reach the required position and keep up with the tracked object during following, all while ensuring the process happens smoothly with minimal oscillations in the UAV's movement. This is where control strategies and simulation come into play. Before flying real drones, which could crash during experimentation and cause harm to people, property and prove to be very expensive, many researchers turn to simulation environments and virtual flight controllers, to perform tests in a safe, cost- effective environment, all while keeping the testing environment safe. Silva et al. [6] makes use of SITL (Software In the Loop) to emulate UAV firmware (ArduPilot) which allows them to see and utilize UAV sensor data in the simulation, allowing them to test the altitude controller they designed, in simulation first before deploying it to hardware. They created a mathematical model of the drone using "Newton-Euler equations" and utilized ROS nodes to process data obtained from various sensors from the simulation and the processing was done on the onboard computer. They utilized an inner control loop for low level drone angle (roll, pitch, yaw) control and an outer control loop for high level altitude control. The testing of the mathematical model was done against the simulation model in three different scenarios, during take-off, varying altitude and combined manoeuvres such as changing altitude while performing roll and pitch manoeuvres. The simulation results proved that the model behaviour closely matched the SITL environment behaviour corresponding to the roll, pitch, yaw and altitude changes, although the SITL environment was much smoother than the models more responsive behaviour, but there was very little difference between both, proving that SITL is a viable option for simulating UAV missions before deploying on hardware.

Saavedra-Ruiz et al. [7] deploys a SITL simulation to emulate the behaviour before deploying it to software, the system detects the landing pad in the camera frames by comparing them to an existing image template of how the landing pad should look, this was performed and tested with three algorithms namely, ORB (Oriented FAST and Rotated BRIEF), ORB is an efficient, rotation-invariant, and scale-invariant algorithm. Unlike the others, it uses the Hamming distance to compute matches between descriptors, which generally allows for faster computation, SIFT (Scale-Invariant Feature Transform) is designed to be invariant to image rotation and scaling, making it robust for detecting objects even when the drone's perspective changes. It uses Manhattan distance to compute matches between descriptors. Lastly SURF (Speeded-Up Robust Features), like SIFT, it is invariant to scale and rotation and utilizes Manhattan distance for feature matching. It was originally designed to be a faster alternative to SIFT. A Kalman filter was implemented to handle noise and ensure smooth tracking. The controller computes error in 2D making computation much simpler. It minimizes the error between the centre of the marker and the centre of the camera image frame by giving the drone velocity set points, this is commonly known as IBVS (Image Based Visual Servoing). An altitude controller triggered decent only when the marker is properly cantered in the camera frame. The software utilized were Gazebo (physic simulator), ROS (Robotics Operating System) and PX4 SITL UAV firmware. Results suggested that SIFT outperformed both ORB and SRUF in detection accuracy for the centroid and the orientation of the landing pad. The implementation of a Kalman filter significantly reduced estimation errors during the tracing phase. The authors tested for P, PI and PID controllers and PID controllers proved to provide the best control but the authors deliberately chose to use a PI controller as it provided the fastest landing time of approximately 36 seconds, the study concludes that SITL is a viable, safe and relatively realistic option to test UAV mission and deploy them on hardware with minimum fine tuning. On the theoretical side, Rus et al. [8] explores the mathematical modelling and simulation of UAVs, addressing both low level flight control (stabilization and altitude) and high level flight control (object detection and avoidance), the authors proceed to model a drone using the Newton-Euler equations to describe the forces and moments acting on the drone during flight, to simplify the control algorithms the system is linearized at the hover equilibrium of the drone. The purpose of the simulation is to determine an efficient control algorithm to perform obstacle avoidance despite the presence of common constraints such as restricted weight, processing power and battery life of the drone. The paper performs two different obstacle avoidance algorithms, the traditional obstacle avoidance where the drone navigates waypoints while avoiding obstacles by changing the forces of thrust as and when required using the mathematical model equations, the other is a novel approach using LoRa technology and RSSI (Received Signal Strength Indicator) to estimate the distance to obstacles and avoid them in advance. The results concluded that the LoRa based obstacle avoidance provided shorter travel distances and reduced power consumption resulting in improved flight times, in addition to this, LoRa provided better obstacle avoidance and avoided more obstacles as compared to traditional obstacle avoidance.

Roberts and Tayebi [9] propose a text which elaborates on a robust control system for VTOL UAVs that ensures the drone can track the desired flight path even in the presence of unknown disturbances in the environment. They focus on the achievement of global stability i.e. the controller works regardless of the drone's initial conditions. The system follows a three step backstepping approach that includes, adaptive estimation law, that using a "projection mechanism", this estimates the unknown disturbance forces (like wind) while ensuring the estimates remain within realistic, bounded limits so the mathematical model doesn't break (avoiding singularities). The second being attitude and thrust extraction for getting the necessary thrust and attitude data from the virtual acceleration step. The final step being rotation control, in which the system calculates the necessary torque to physically route the drone to the desired attitude derived in the second step. They then propose two control laws, the first one being position tracking that works irrespective of initial conditions but is mathematically complex. The second method guarantees control but with a set of initial conditions and is mathematically simpler. The controllers were tested in a simulation of a 5kg ducted-fan UAV subject to aerodynamic drag and ram-drag forces. The drone was tasked with following a complex spiral trajectory while fighting a constant wind force of [-1, -1, 0] m/s. The first controller successfully forced the system to converge to the desired trajectory with zero error over time. The second controller also achieved the tracking objective. While theoretically more limited, simulations showed it was highly effective with reasonable control gains, making it very practical. The text successfully demonstrated a control method to allow the drone to learn and estimate the wind forces in real time to give the necessary thrust tilts to maintain a precise path.

Smyczyński et al. [10] proposes a flexible control system design for autonomous UAVs that can be adopted on various hardware platforms. The system was specifically tested for UAVs landing on moving vehicles. The system majorly utilizes ROS to create nodes and for multiple components of the UAV system and establishes a publisher subscriber model that allows for efficient data transfer and communication between nodes. The nodes are divided as data handling nodes that acquire raw data, filter and process it, then there are control nodes that run a state machine based on the data received from the data nodes. The system uses a camera sensor to detect a square marker using the “Canny edge detection” algorithm, once the object is detected it switches over to the “Lucas Kanade Optical Flow” algorithm to track the marker. As a result, the system was able to successfully track and follow the marker at a max speed of 1.38 m/s and land on the moving platform. The use of a Raspberry Pi 3 onboard computer was not the best choice as it lacked the computational power to deliver better node performance, which resulted in delayed drone response and slower tracking speeds. To resolve issues related to processing speed as we saw in the previous study, we look into another text, Zheng et al. [11] that addressed the problem of keeping up with change in the camera view that causes the marker to look bigger during decent or motion blur during fast movement, or during varying lighting conditions. Instead of remembering what the marker used to look like and comparing each frame the drone uses an algorithm called STC (Spatio-Temporal Context) where the drone analyses the statistical relation between the pixels that make up the marker and the pixels that makeup the background, this allows it make the same relation in other frames even when the marker does not look like it used to, giving the drone an edge to detect the marker in varying scenarios. The authors implement a scaling update in the algorithm that tells the drone that it is closer to the marker when the marker gets bigger in the frame, and they also utilize a focus mechanism not too different from the human eye, that focuses majorly on the areas of interest in the camera frame and not on the edges, reducing computation power and power drawn from the battery. The system is super-fast and runs at 110 FPS (Frames Per Second), making the algorithm very fast and responsive. The results suggest that the marker was successfully tracked in conditions such as marker being obscured, motion blur, and low light.

Features from SITL and flight controllers allow the UAV to perform an RTL (Return to Launch) feature, that causes the UAV to return to its position of launch with respect to the global frame of reference that uses the GPS, but this feature is only good enough for landing on stationary platforms. It is necessary to research on the challenges of autonomously landing UAVs on moving targets such as boats, cars and ships) using visual based approaches and robust control strategies, to mitigate the need for complex precision sensors such as a high precision GPS. Salagame et al. [12] and Keller and Ben-Moshe [13] worked on robust methodologies for this specific problem. Salagame utilizes a bunch of 5 April Tags with descending sizes, these descending sizes come into play as the drone descends for landing and each marker becomes too big for the camera frame, allowing the drone to localize at very low altitudes (5 - 2m) also and obtain high precision landing. Keller uses aruco markers to determine the pose and distance and localize. Salagame uses a stepped landing approach and switches to smaller markers as it descends lower. Keller uses a “sliding approach” to land diagonally instead of a fixed vertical descent, using the gimbal tilt to keep the target centered in the frame. Both papers implement a PID controller that minimizes the distance between the camera frame center and the marker center, this is also aided by gimbal control, that tilts to keep the marker centered in the camera frame during steep descent angles (up to 60 degrees). Salagame showed successful tests of drone landing in wind speeds of up to 8.33 m/s. Keller displayed a high landing accuracy of 0.01m to 0.05m.

Some researchers like Chang et al. [14] got creative and decided to address the problem of on-board computational load for UAVs by offboarding the data from the UAV to the moving platform, which in turn performs the heavy computation and sends the computed data back to the UAV for localization. The system detects an ArUco marker and uses YOLOv4 for a deep learning-based object detecting algorithm that identifies the marker in the camera image frame using ROI (Region of Interest). It also uses an IMU to update the Kalman filer with velocity data in between image frames to ensure high-rate estimates (50Hz). The control system implements an FSM (Finite State Machine) to perform the trajectory planning, switching from GPS to visual marker following, the switching to a ground effect free trajectory using a "Brachistochrone" curve, which is a trajectory designed to minimize time and avoid ground-effect instability by descending rapidly in open air and then gliding horizontally onto the pad. The combined sensor (sensor fusion) system achieved a RMSE (Root Mean Square Error) of less than 0.08m in position estimation. Guo et al. [15] and Falanga et al. [16] both proposed autonomous systems to land on moving platforms, and enable long endurance flights by allowing the drones to recharge on mobile stations. Guo utilizes a gimbal based two stage tracking algorithm, one stage for long range search and the other for close rang ArUco tracking, keeping the target in view, whereas Falanga uses a downward fixed camera to detect a marker using a PnP (Preserve-n-Point) algorithm. They explored MPC (Model Predictive Control) which uses long sampling intervals at further distance for efficiency and higher sampling rates at short range for precision. They also explore “Minimum Jerk Polynomial Trajectories” in real time, which is said to be a computationally efficient task. The downside of MPC is that it requires a very powerful onboard computer.

Paris et al. [17] addressed another layer of difficulty: turbulent wind. Landing on a moving platform is hard; landing on a moving platform while wind is pushing you is harder. They developed a “dynamic landing” method in which the drone flies in a fast, direct path towards the moving platform, without stopping or hovering above it first. At larger distances the drone localizes using a GPS and when it gets closer it switches to visual based detection method using April tags on the landing platform. They also utilize MPC for predicting a smooth trajectory. The novelty here is the BLSC (Boundary Layer Sliding Controller) which estimates the wind turbulence and actively tunes the motors thrust to prevent the drone from being affected by this turbulence. The results suggested that using the BLSC the drone landed in 3.2 seconds on a stationary platform, which was 8.5 times faster than a standard controller, whereas on a moving turning platform, the drone successfully caught up and landed on it in 6.8 seconds. Subamanian et al. [18] and Saj et al. [19] looked specifically at marine environments and landing on moving ships and naval vessels. At sea the ship or vessel has much disturbances due to random movements generated by the sea, the ship would have 6 degrees of freedom while moving, on top of this there could be strong winds and gusts at sea that the UAV has to take into account for a smooth landing. Both the texts resort to vision-based landing approaches as opposed to using a GPS as that would be too difficult to localize with during varying weather conditions and due to its low precision. They use a downward facing camera to detect for nested April Tags, the larger tag allowing localization from a distance and the smaller one for precision landing when the drone gets close enough. They also implement a method of tracking where the ship is considered as a “fake horizon” as opposed to using a downward facing camera to detect markers. The fake horizon is used as a reference for the drone to stabilize itself and not be distracted by the random rolling movements of the ship deck. They employ visual based marker detection methods that flatten the marker, making it always appear horizontal to the processing computer, even if the ship is tilting. A Kalman filter is used to smooth out the noise. The second approach was to use AI (Artificial Intelligence) to replace standard mathematical PID controllers, where a neural network was trained using RL (Reinforcement Learning), this allowed them to simulate randomized wind speeds and transport delays making the drone learn how to adapt to these complex and unpredictable scenarios. The use of nested markers resulted in a final position error of only 0.1m. The second method was tested in real world using fans to simulate wind turbulence, the AI controller was able to recover from sudden gusts in less than 2 seconds, as opposed to standard controllers that took 15 seconds and successfully landed in windy scenarios where standard controllers have completely failed. Subamanian focused on the visual tracking of a naval base, dealing with the rocking motion of the deck. Saj took a very different approach using "Reinforcement Learning" (RL). The merit of RL is that the drone can learn intuitive behaviors that are hard to code manually. The demerit is the "black box" problem—if it crashes, it is very hard to understand why AI made that decision.

Finally, our literature survey narrows down to the problem of visual detection and tracking of the target during low light conditions. Standard cameras are blind in the dark, there are other options besides cameras, such as IR sensors, Lidars etc. that could resolve this issue, but there exists further research on low light target detection using camera sensors by optimising the software part. Lin et al. [20] and Li et al. [21] attempted to solve this using software. The challenge being that traditional vision algorithms fail at night due to low contrast, high noise, and motion blur issues, and using standard brightening techniques like gamma correction can washout the details in the image and also introduce digital noise. Both these papers refuse to utilise active markers such as led lit markers to aid in the dark and aimed to solve this by optimising algorithms alone. The first method is a form of inverted dehazing technique, the authors first inverted the dark image making it look like a foggy day time image, then they apply a dehazing algorithm on it and revert it back, essentially removing the darkness. The other approach adopted was using a Deep Learning Network (ZERNet). Instead of using physical formulas they trained a neural network to estimate an “NE (Nighttime Enlightening) map”, this map applies pixel by pixel S-curve (commonly used in image editing) adjustments to brighten the dark areas and leave the already bright spots like street lights alone. The dehazing algorithm when validated in real world (a nighttime basketball court with 5 lux brightness in this scenario) yielded a result of marker detection rate greater than 95%and ran on real time on an NVIDIA Jetson TX2 onboard computer. The ZERNet method outperformed state-of-the-art low-light enhancers (like Zero-DCE) in tracking accuracy and visual quality metrics. The demerit is that digital brightening adds "noise" (graininess) to the image, which can confuse the landing software.

A hardware alternative was proposed by Kalinov et al. [22] and Badakis et al. [23]. Instead of trying to fix a dark photo, they resorted to a multi sensor fusion system that utilises both visual based approach and IR based approach, essentially combining them to make the best of both worlds. Badakis uses a standard approach of onboard computation, the drone has a camera and an IRLock sensor, that looks for IR beacons, mounted below it, both facing downwards to find the landing pad on the ground. Kalinov inverted the general approach to these problems, by placing active IR markers on the drone and placing a high-resolution camera on the moving rover that sees the drone as it approaches, this removes any error that could arise from the disturbances in the drone's movement and allows for heavy computation to be done on the ground vehicle or platform rather than onboard the done. The camera filters out visible light and only sees the glowing IR pattern, making the landing target pop out even in total darkness. Badakis showed that fusion of sensors can produce a system that is more reliable due to redundancy and that the system can still perform if one of the sensors fails. Kalinov proved that placing the camera on the moving platform and fusing it with an ultrasonic sensor produces very high landing precision (0.0125m). The demerit is that IR sensors can be blinded by other heat sources or direct sunlight during the day, requiring a system that can switch modes.

Lastly, we look at an interesting marker tracking algorithm implemented by Raei et al. [24] that uses the drones onboard camera sensor,1D Lidar and Imu to perform the detection and localization with the moving marker platform. The drone utilises the Lidar to calculate its height from the ground, this is high precision data coming from the Lidar, the camera is used to visually see the marker and the IMU is used to determine the drones mid air pose. The algorithm is very practical , it begins by using the Lidar to estimate height, then the camera detects the marker in the image frame and it begins to calculate the relative angles between the centre of the drone and the centre of the marker in the camera image frame, when the drone pitches or rolls this angle is added then the estimated relative angle from the image processing (this required prior estimation and details of the camera's horizontal and vertical FOV and the pixel dimensions of the image frame) , this allows the system to very efficiently centre the marker in the drone's camera frame, which then tells the drone to begin vertical decent right onto the centre of the marker on the platform. The results show that the sensor fusion method reduced the distance measurement error by approximately 50% compared to camera alone, and the drone successfully tracked and landed on a moving platform at speeds up to 1.5m/s.

2.1 Key Takeaways

The literature clearly indicates that successful autonomous drone landing relies on a blend of robust sensing and predictive control; specifically, the use of "nested" markers is vital for maintaining visual lock from high to low altitudes, while the ability to calculate a "rendezvous point" is critical for accurately intercepting a moving platform, rather than just chasing its current position. Testing is best done safely using Software-in-the-Loop (SITL) before real flight. When dealing with poor lighting, solutions split between digitally enhancing images or using Infrared markers, with IR offering better reliability but adding hardware complexity. Furthermore, the decision between a gimbal camera (less software complexity, more weight) and a fixed camera (more software complexity, less weight) is a core design trade-off, and complex environments often necessitate advanced mathematical control schemes like adaptive tracking to manage unpredictable elements like turbulence.

Chapter 3: PROPOSED METHODOLOGY

The methodology adopted for this project follows a comprehensive simulation workflow designed to validate autonomous takeoff, tracking, and landing on a moving vehicle. We will consider the vehicle to be a rover. Since the system is currently software-only, all components are implemented within a SITL (Software-In-The-Loop) Gazebo environment. This allows for the safe simulation of sensor inputs, rover kinematics, and UAV dynamics. The flow of the mission is shown in Fig 3.1.

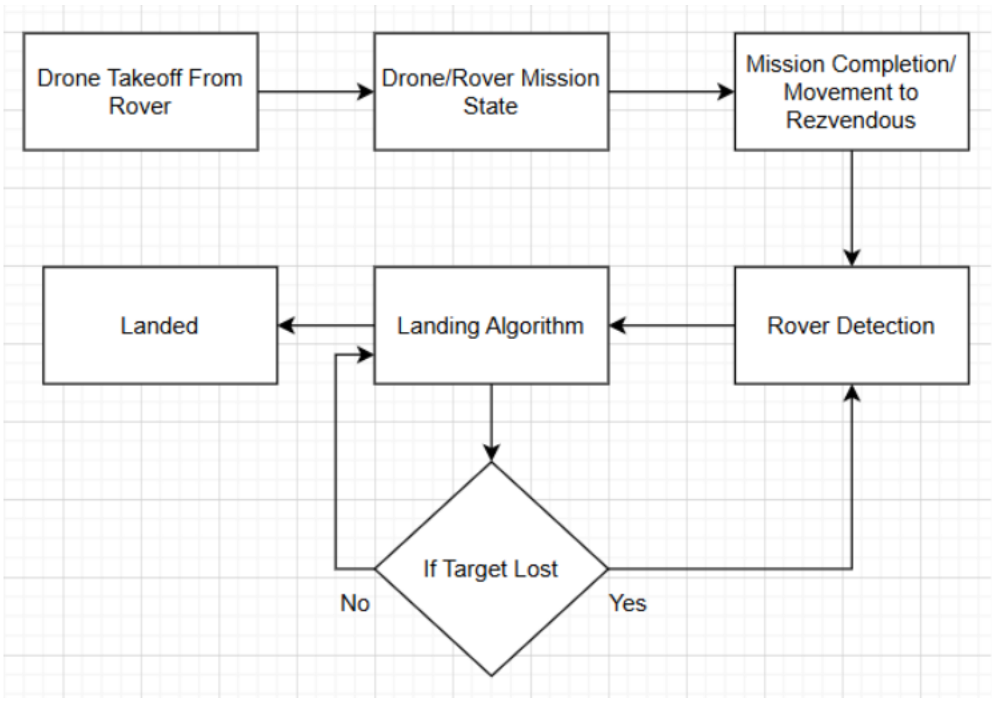


Fig. 3.1 Flowchart of the Mission Plan

The rover begins its mission by moving and performing its own mission. As it starts this journey, the drone that is positioned on the rover prepares for takeoff. The drone lifts off into the air and begins carrying out its own mission, which runs in parallel to the rover's ground movement. While the rover continues to progress on its path, the drone operates independently, relying on its flight systems and sensors to complete its assigned tasks which in our case will be a waypoint mission. Once the drone has finished its mission, the drone must return to the rover. To do this, it first gets a GPS coordinate from the rover and moves towards it, when in visual range it **activates** its detection process, scanning the environment to locate the rover's position. When the rover is successfully detected, the drone initiates its landing sequence, carefully aligning itself above the rover and descending in a controlled manner to ensure a safe landing. If the drone loses sight of the rover during this process, it does not attempt to land immediately. Instead, it rises to a higher altitude to widen its field of view and increases the chances of finding the rover again. This behaviour allows the drone to recover from temporary loss of detection without failing its mission. Once the rover is found again, the drone resumes its landing procedure, gradually lowering itself until it is securely on the rover.

3.1 Coordinate Reference Frames and UAV-Rover System

To rigorously define the state estimation and control laws presented in subsequent sections, we establish three primary orthogonal coordinate frames. All mathematical derivations in this work assume a Right-Handed Coordinate System.

Inertial World Frame (F_w):

- The fixed global reference frame (North-East-Down or NED) originating at the UAV's takeoff location. The ground vehicle's motion and the UAV's global telemetry are defined relative to this static frame.

UAV Body Frame (F_b): A moving reference frame attached to the UAV's center of mass.

- F_B: Points out to the front of UAV.
- Y_B: Points out to the right of UAV.
- Z_B: Points out to the bottom of UAV.
- The transformation from F_W to F_B is governed by the UAV's yaw heading.

Camera Optical Frame (F_C): The reference frame native to the computer vision sensor.

- Z_C: Represents depth (optical axis) pointing towards the ground.
- X_C: Points to the right of the image plane.
- Y_C: Points to the bottom of the image plane.

A critical preprocessing step involves calculating the extrinsic rotation matrix $R_{C\to B}$ to map visual detections from F_C into the control-relevant F_B.

Fig 3.2 shows the UAV and the rover which are modelled in Gazebo to simulate the dynamics of the system. They have the following sensors –

- The UAV has –
    - IMU inertial measurement unit which is noisy.
    - GPS for low accuracy world frame position.
    - Motor RPM sensor on each motor.
    - Bottom mounted Monocular RGB camera for fiducial marker detection.
- The rover has –
    - IMU inertial measurement unit which is noisy
    - GPS for low accuracy world frame position.
    - Wheel encoder to calculate how much the robot has moved.
    - A 40cm 5x5 ArUco tag on the top of the rover.

Both use extended Kalman Filters (EKF) to estimate their position in the F_W Frame.

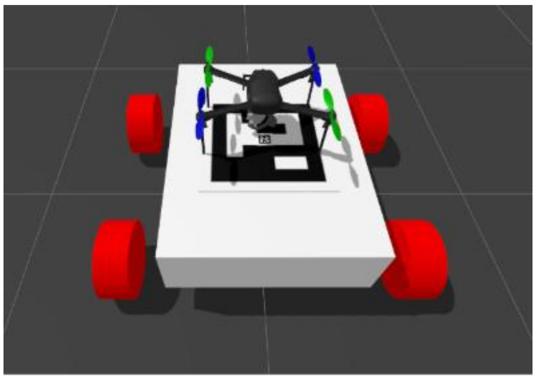


Fig. 3.2 IRIS Drone Resting on the Ground Rover

3.2 Accurate height calculation

The simulation initiates by generating virtual sensor data that mimics real-world hardware constraints. The primary sensor is a simulated monocular camera mounted on the underside of the UAV.

Unlike the approach presented by [26], which relies on a 1D-LiDAR for altitude measurement, our proposed method utilizes Monocular Pose Estimation. The altitude of the UAV is calculated mathematically using the translation vectors derived from the camera extrinsics, eliminating the need for an additional laser rangefinder.

The relationship between the 3D world point P_w and the 2D image pixel p_i

is modelled using the Pinhole Camera Model:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \, . [R|t] . \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \, . . . (3.1)$$

Where K is the intrinsic camera matrix and $t = [t_x, t_y, t_z]^T$ is the translation vector. The UAV's altitude H is derived directly from the z-component of this vector:

$$Z_b \approx t_z \, . . . (3.2)$$

3.3 Marker Detection

The core of the localization system uses the OpenCV library to detect a Nested ArUco Marker configuration placed on the landing deck of the rover. This nested design allows for robust tracking across varying altitudes.

- Outer Marker: ID 10, with a side length of 0.416667 m. Used for high-altitude detection.
- Inner Marker: ID 20, with a side length of 0.04629 m $(1/ 9^{\text{th}}$ of the outer marker). Used for precision terminal landing.

The detection algorithm performs adaptive thresholding to identify candidate square contours, extracts the binary bit code, and matches it against the DICT_5X5_50 dictionary. Once the marker corners are identified, the Perspective-n-Point (PnP) algorithm minimizes the reprojection error to solve for the relative pose.

3.4 State Estimation from Fiducial Marker

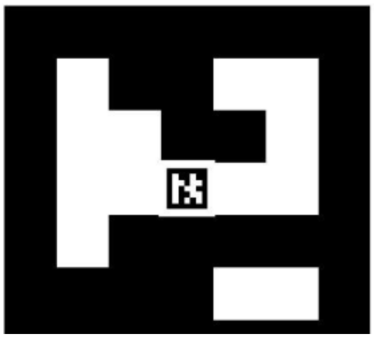


Fig. 3.3 Fiducial Marker used in the Simulation

A lightweight state estimation system fuses visual data from the PnP solver with the UAV's internal telemetry. In the reference methodology [26], the relative position of the rover and UAV is computed by fusing LiDAR distance D with the camera look-angles α and β. This requires explicit calculation of the trigonometric projections (Eq. 3-5 in [26]). In contrast, our system extracts the relative position directly from the translation vector t provided by the PnP solver which uses the marker shown in Fig 3.3. The estimator outputs a state vector $S = [t_x, t_y, t_z,\ \dot{x}, \dot{y}]$ representing the lateral error, longitudinal error, altitude, and relative velocities. This direct extraction of metric data allows the system to handle the "Simulated Reality Gap," where Gazebo's ideal physics often fail to represent the noise found in real-world IMU data. By relying on the visual solution for relative positioning, the system maintains robust tracking even when the rover undergoes constant acceleration.

The UAV generates a closed-loop trajectory based on the rover's estimated motion. The trajectory adapts to different rover motion patterns, such as constant velocity, variable velocity, and constant acceleration.

Our proposed methodology utilizes the extrinsic matrix properties. The lateral error e_{lat} and longitudinal error e_{long} are obtained directly from the translation vector t.

$$e_{lat} = t_x \quad ... (3.3)$$

$$e_{long} = -t_y \; ... (3.4)$$

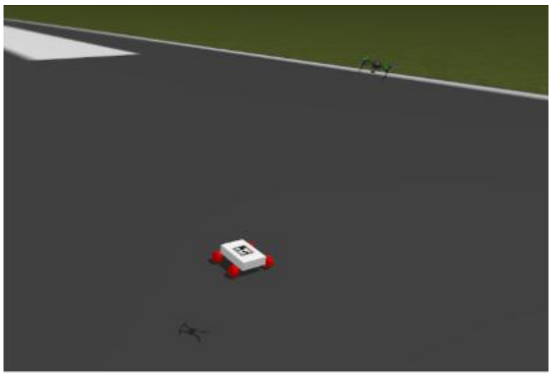


Fig. 3.5 Drone calculating distance from marker

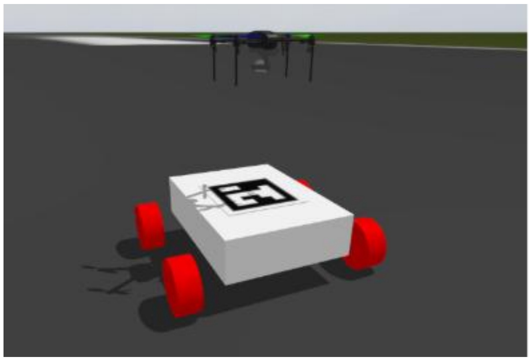


Fig. 3.4 Drone moving towards marker

Note that the y-axis is inverted to align the camera frame with the UAV's Body-NED frame. As the UAV approaches the rover, the controller minimizes these error terms until the UAV is positioned directly over the nested marker (Fig 3.4, Fig 3.5). Vertical descent is initiated only when horizontal alignment meets the required accuracy threshold.

3.5 Simulating Loss of Marker Detection Rate

In practice, the camera continuously detecting the fiducial marker is never assured. Environmental conditions (like changing light conditions or glare, shadows in the field of view, air frame vibration inducing motion blur to the captured video image, occluded markers, etc) invariably lead to False Negatives, which are instances where the camera is capturing an image of the fiducial marker, but the algorithm fails to identify the marker present in the image.

The number of false negatives occurring is dependent on the altitude of the UAV At higher altitudes, the angular resolution of the marker decreases (fewer pixels per marker bit), and the detection of the marker is much more susceptible and affected by environmental conditions. Alternatively, at lower altitudes, the marker occupies more of the sensor frame, and detection is consistent.

To replicate the loss of detections we developed a module that would replace the camera frame with a Null frame (black frame) and pass it to the detection algorithm based on the altitude of the drone.

- Low Reliability Zone (h > 5m): At high altitudes, we limit the actual camera frames sent to 1 Frame Per Second (FPS). This simulates a scenario where the detection fails at a high rate due to a combination of environmental factors and low pixel density.
- High Reliability Zone (h < 2m): In the terminal phase, where the marker is large and distinct, the system permits a high-frequency update rate of 10 FPS, here the detection is mostly reliable due to the marker appearing larger in the image.
- Transition Zone: Between 2m and 5m, the effective detection rate scales linearly, modelling the gradual improvement in feature recognition as the UAV approaches the target.

This explicitly forces the system to handle the loss of visual lock.

3.6 State Estimation during Low Detection Rate – Sensor Fusion

To now find the state of the drone became more challenging as the in the worst case we were getting a useable frame every 1 second. This is not enough to track the rover. So, we now turn to the odometry data that was provided by the EKF on the UAV and the rover.

The rovers odometry offers high-frequency updates but suffers from cumulative drift and also operates in a coordinate frame and might even have a different origin than the UAV. Most importantly it suffers from communication delay as it takes 50ms for the signal to reach the UAV from the rover.

To address these issues a Bias Correction Sensor Fusion algorithm was implemented. This approach treats the visual data not as the primary control signal, but as a sparse "Ground Truth" that is used to calibrate the higher frequency but noisy and biased odometry signal in real-time.

When a real frame is processed and the ArUco marker is detected, the system calculates the Instantaneous Bias (b). This is defined as the vector difference between the position reported by the camera pcam and the difference in current odometry readings P_{odom_diff} .

$$P_{odom\_diff} = P_{rover} - P_{drone} \ . \ . \ . \ (3.5)$$

$$b = p_{cam} - p_{odom\_diff} \ . \ . \ . \ (3.6)$$

This bias vector b effectively captures the coordinate system mismatch and sensor drift at that exact moment. Then when until the next frame where the marker is detected the camera the relative odometry data of the rover and the UAV is compared and that is fed into the control loop. When a new frame is available this process repeats again to handle the noise and drift in the odometry data.

This approach allows the control loop to run at a high frequency i.e. 30Hz driven by the smooth rover odometry, while being corrected at a low frequency 1Hz by the precise error reading form the detection of the marker. At lower altitudes we directly use the vision to calculate the error as the detection rate of the marker is high.

3.7 System Identification

To find the transfer function of the system, we model it as Second-Order Transfer Function with Transport Delay (SOPDT) this captured the dynamics of the drone accurately.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K\omega_n^2}{s^2+2\zeta\omega_n s+ \omega_n^2} e^{-T_d s} \ . \ . \ . \ (3.7)$$

To identify the transfer function, the UAV was first placed in a stable hover above the marker, and a step input was applied to the lateral velocity.

The resulting actual velocity of the drone was recorded. This data was then imported into MATLAB and we used the System identification toolbox to fit the parameters. This was then validated with another dataset that sends random velocity setpoints and tracks the output.

3.8 Control Execution Using Tuned PID Controllers

A cascaded PID controller is used to track the marker first a higher-level controller which takes the error input from the state estimator and gives the velocity commands in x,y,z for the drone to fly in. Then a low level controller converts the velocity commands to motor throttle values.

The high level control actions are executed using discrete-time PID (Proportional-Integral-Derivative) controllers tuned through system identification.

The control law governing the output velocity $u[k]$ for each axis is given by:

$$u[k] = K_p e[k] + K_i \sum_{i=0}^{k} e[i]\Delta t + \frac{K_d(e[k]-e[k-1])}{\Delta t} \quad \ldots (3.8)$$

This loop runs at approximately 30Hz, ensuring stable tracking during the entire landing sequence.

This velocity commands are sent into the controller on the Flight controller where there is another PID loop that converts the velocity commands to actual motor values.

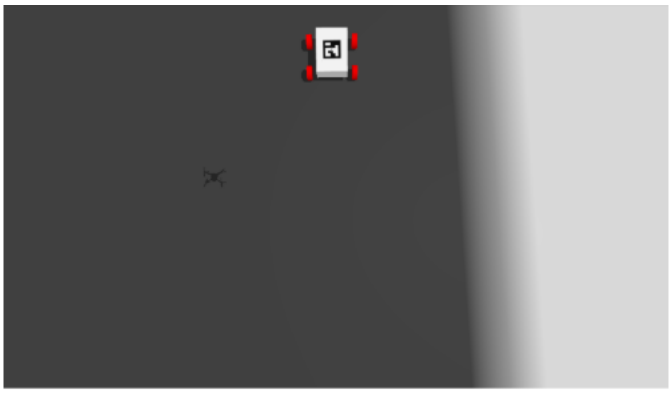


Fig. 3.7 PID Controller attempting to center the marker in the camera frame

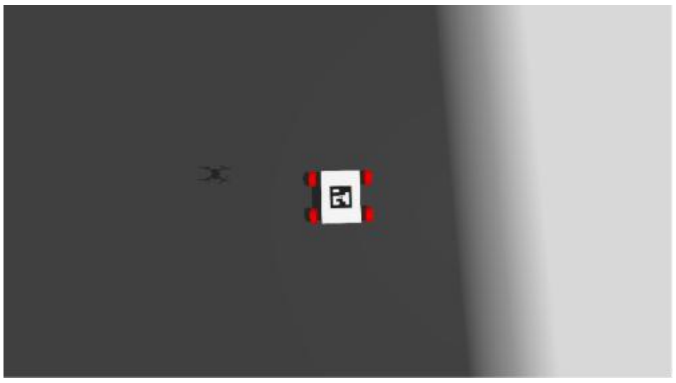


Fig. 3.6 Marker Successfully centered in the camera frame

3.9 Altitude control and Touchdown

The final landing phase is governed by a Variable Speed Descent Logic rather than a static vertical velocity. Landing begins once the marker is detected in the camera frame. We calculate the lateral error of the marker with respect to the rover.

The downward velocity is given by –

$$v_z = v_{base} \times Clamp\left(1.0 - \frac{||e_{lat}||}{e_{threshold}}, 0.1, 1.0\right) \ldots (3.9)$$

Base velocity is based on altitude. When altitude > 2m its set to 1m/s. under that it scales down to 0.1m/s at 0.5m. This is to allow for a quick approach but to ensure a soft landing at the end.

The clamp is based on the lateral error. If the maker is at the centre of the frame, then the clamp will be 1. Which means the Rover is directly under the UAV and it should land as soon as possible. However, if the marker is at the edge of the frame reducing altitude quickly will cause it to go out of frame. So, we reduce the clamp factor linearly with lateral error to reduce the v_z to 0.1.

When the UAV reaches the rover's surface (H < 0.2m), it executes a motor kill command, marking the end of the simulation test (Fig3.8, Fig 3.9).

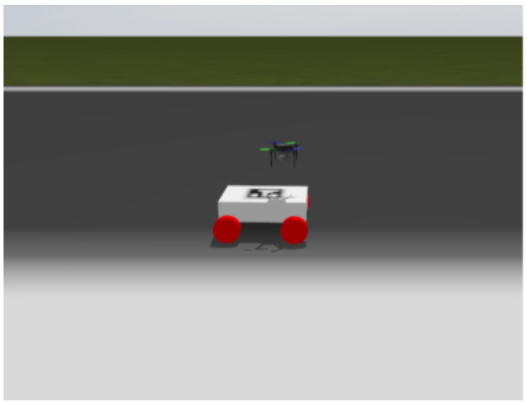


Fig. 3.8 Drone Attempts to land on the moving platform

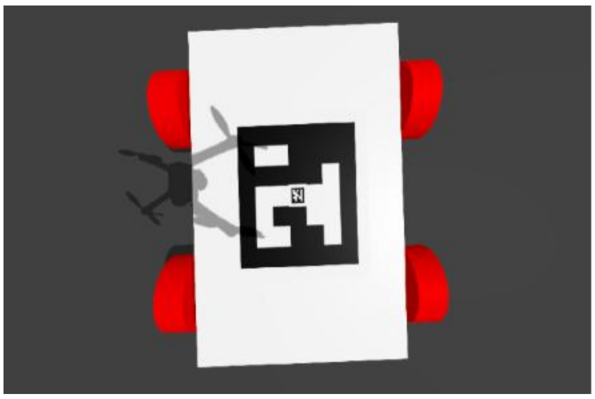


Fig. 3.9 Drone Camera View just before Touchdown

3.10 Handling loss of detection of marker

While the sensor fusion algorithm described in Section 3.8 provides resilience against intermittent data loss, we can still experience complete loss of tracking. This can occur when the rover accelerates aggressively when the UAV is at a low altitude causing it to go out of frame. In this case we increase the altitude steady while using the odometry difference to keep in track. In case we can't still find the rover, we use the GPS on the rover to find where it is and start over the process again.

Chapter 4: IMPLEMENTATION

The implementation of this project is carried out entirely within a software-based simulation environment that combines ArduPilot SITL, Gazebo, MAVSDK, Qgroundcontrol, and OpenCV. ArduPilot SITL provides a virtual flight controller; Gazebo supplies the physics engine and simulated sensors, and MAVSDK allows the python script to communicate with the ArduPilot.

4.1 Simulation Environment Setup

4.1.1 ArduPilot SITL Configuration

For the flight control engine, we used the ArduCopter 4.6.2 firmware within the ArduPilot SITL (Software in the Loop) framework. This was the latest version at the time of this research project and most crucially it includes the most recent EKF3 (Extended Kalman Filter) enhancements, which was vital for our sensor fusion-based approach. Unlike rigid, "black box" flight controllers, ArduPilot exposes a comprehensive parameter tree that allowed us to aggressively tune the UAV’s behaviour without modifying the source code.

The MAVLink protocol was used to establish a communication link between the simulated autopilot and our code. We could then use MAVSDK library natively, allowing our Python-based computer vision, state estimation and control nodes to interact directly with the UAV.

Qgroundcontrol was chosen as a ground station for us to monitor the state of the UAV. During simulation runs, this interface proved to be an invaluable tool, allowing for the human-readable dashboard monitoring of important flight metrics such as relative altitude, battery voltage sag, and GPS satellite lock. Being able to see and obtain that instantaneous visual feedback allowed us to give us real time knowledge of the drone's health and estimator convergence which allowed for astronomically faster debugging to be monitored as opposed to sifting through raw data logs after the flight.

ArduPilot accepts "Offboard" velocity commands over MAVSDK, which allowed us to precisely control the drone to track the rover.

4.1.2 UAV Platform Specification

The simulation framework utilizes the 3DR Iris quadcopter as shown in Fig 4.1, a platform widely recognized as the standard reference model for ArduPilot SITL and Gazebo simulations. We selected this specific airframe because it has a well implemented simulated physics model encompassing inertia tensors, motor thrust curves, and aerodynamic drag coefficients which has been rigorously validated by the community to match real-world flight dynamics. This high-fidelity representation ensures that the control logic developed in the SITL environment encounters the same inertial challenges and response latencies as a physical UAV, making it an excellent candidate for validating complex maneuvers like autonomous landings on moving targets.

Physically, the Iris is modeled as a mid-sized quadrotor with a diagonal wheelbase (motor-to-motor distance) of 550 mm. It is equipped with 10-inch (10x4.7) propellers driven by 850 kV brushless DC motors, resulting in a simulated takeoff weight of approximately 1.28 kg. This mass provides sufficient inertia to test the robustness of our PID controllers against momentum drift during rapid lateral corrections. The wide stance of the landing legs and the low center of gravity also contribute to a stable simulation envelope, allowing us to isolate navigation errors from mechanical instabilities.

Fig. 4.1 The IRIS Drone

4.1.3 Monocular RGB Camera

To facilitate continuous visual tracking, as shown in Fig 4.2 the UAV is equipped with a monocular RGB camera which is attached in the downward position and when the UAV is level The optical sensor was modified from the standard definition SDF parameters to output a high-definition video stream of 1920×1080 pixels. This increased resolution significantly extends the detection range of the Perspective-n-Point (PnP) algorithm, allowing the system to acquire the rover's ArUco marker from higher altitudes while maintaining the precise pixel density required for terminal guidance.

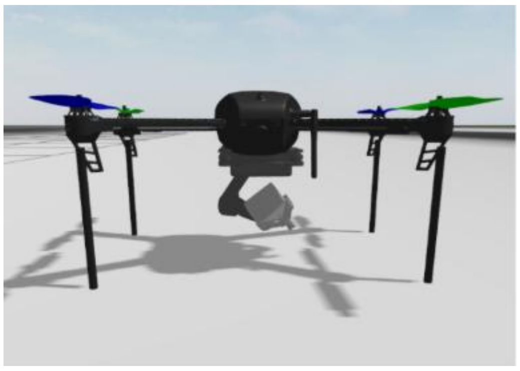


Fig. 4.2 Three Axis Gimbal Mounted Camera

4.1.4 Ground Rover Platform

The ground segment of the simulation is represented by a custom four-wheeled rover as shown in Fig 4.3, this modelled specifically to serve as a mobile landing deck. We defined the vehicle in the Simulation Description Format (SDF) as a robust, rectangular platform driven by a skid-steer differential drive configuration. The main chassis is modelled as a rigid box with dimensions of 1.0 m (length) × 0.75 m (width) × 0.25 m (height). This specific geometry was chosen to provide a flat, predictable landing deck of approximately $0.75\ m^2$ ensuring that the physical contact surface remains predictable during the critical touchdown phase. The vehicle sits on four cylindrical wheels, each with a radius of 0.15 m, which provides sufficient ground clearance for the simulated terrain.

To replicate real-world localization challenges, the rover is instrumented with a sensor suite that mimics the noise characteristics of low-cost hardware. It features a simulated MPU6050 IMU updating at 100 Hz, Wheel encoders measuring the rotation of each wheel and a NavSat (GPS) module operating at 30 Hz. Crucially, the simulation configuration explicitly injects Gaussian noise into the inertial readings—specifically a standard deviation of 0.039 for linear acceleration and 0.00087 for angular velocity. This noise injection is vital for validating that our UAV's state estimator can handle non-ideal telemetry from the ground target.

The IMU and wheel encoder are combined using an Extended Kalman

Filter (EKF) provided by Gazebo which will give us the odometry of the rover. This is then transmitted to the UAV with a delay of 50ms which is done to simulate an extreme case of communication delay for low latency close range communication systems which are commonly used in UAV's such as ExpressLRS.

The ArUco marker mesh is rigidly attached to the chassis via a fixed joint, offset vertically to ensure it remains visible and unobstructed on the dorsal surface.

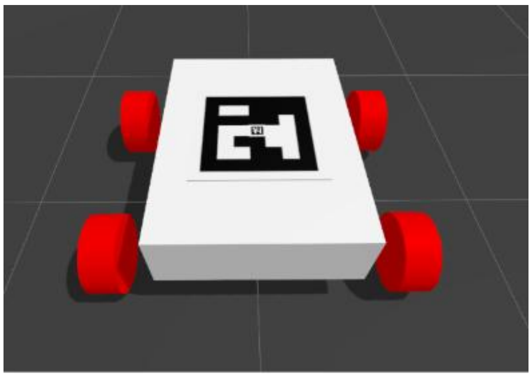


Fig. 4.3The ground rover with the ArUco marker

4.1.5 MAVSDK-Python Control Interface

The MAVSDK-Python library was used to run the mission logic in a Python 3.10 environment. This was chosen instead of pymavlink as it provides a clear, high-level interface that makes communication with the flight controller easier. With MAVSDK, flight behaviors such as arming or sending velocity command can be written in short, readable scripts, without needing to handle the entire MAVLink message format. The library connects to the ArduPilot autopilot through MAVLink over UDP, and it automatically converts Python function calls into the commands required by the flight controller.

An important advantage of MAVSDK is its support for asynchronous programming, with the use of asyncio and threading for tasks that need to run parallelly. In our setup, the drone control logic runs as an asynchronous event loop, allowing flight actions to be awaited without blocking other parts of the mission. At the same time, computer vision tasks such as reading the camera stream and detection of the ArUco markers are handled in separate threads. This arrangement helps keep the vision pipeline responsive and able to update the target error vector continuously, while the MAVSDK loop manages velocity commands and monitors the drone's health in real time.

By combining asynchronous control with threaded vision processing, the system remains responsive and reliable. With the help of this, stable flight can be maintained while the UAV adjusts its position based on visual input, ensuring that autonomous missions are completed efficiently and are practical.

4.2 Vision System Implementation

4.2.1 Simulating Environmental Degradation (Null Frame Injection)

The module subscribes to the /camera/image topic and receives the raw frames of the camera. It then uses the altitude from the UAV and calculates how many frames to pass through. Once the target FPS is calculated, it passes through those frames and fill the rest with NULL frames (black screen). This new stream is then passed through to the Detection module.

4.2.2 ArUco Nested Marker Detection

When a frame is revied by the detection module the following pipeline is executed–

- Preprocessing: The received 3- channel RGB 1080p image is converted into a single-channel gray image. The conversion is performed in order to decrease computational complexity and enhance the detection of edges.
- Candidate Extraction: The cv2.aruco.detectmarkers function applies an adaptive threshold to draw square contours in the frame. The binary bit code is extracted from the candidate images and compared to the previously defined DICT_5X5_50 dictionary.
- Nested Logic: The nested marker configuration was designed to improve both long-range visibility and close-range precision. The algorithm determines which marker to use based on pixel area of the marker size. If both markers are visible, the logic prefers to use the larger marker. However, when UAV gets close it switches to the inner marker. As a result, accurate alignment of the UAV was maintained throughout the approach and descent phases.

4.3 Finding the error term

Using the camera's intrinsic calibration, the system estimates the rover's relative pose by interpreting the spatial arrangement of marker corners. Marker size in the image is used to infer distance, while pixel displacement is translated into lateral and longitudinal offset measurements. These values are forwarded to the state estimator.

4.3.1 By Using Vision

When the ArUco marker is detected the raw pose estimation is found using a Perspective-n-Point (PnP) algorithm. This gives us the translational vector $t_{vec} = [x_c, y_c, z_c]^T$ in the camera's frame. This is then converted to UAVs body frame by applying the following transforms –

$$X_B = -Y_C \;\; ; \; Y_B = X_C \;\; ; \; Z_B = Z_C \ldots (4.1)$$

4.3.2 By using vision along and difference in odometry

We read the odometry from the delayed EKF module which gives us the odometry of the rover with a delay of 50ms to simulate communication delay. We get the odometry of the UAV using the inbuilt EKF on the ArduPilot software. Once we get the first vision frame, we can get the "true" relative positions of rover and UAV. Using this at that instant we can use the difference in the odometry position of UAV and rover. This is then used to calculate the error until a new frame is available.

We have assumed that there will be at least one detected frame in 2 seconds and the drift in odometry position is negligible for that time. Thus, using this we can always have a confident error term to feed to our controller.

4.4 System Identification for Controller Tuning

For the X axis the SOPDT model was identified as the following transfer function –

$$G(s) = \frac{5.1331}{s^2 + 5.5097s + 2.9758\, x\, 10^{-12}} e^{-0.4669s} \ldots (4.2)$$

For the Y axis the SOPDT model was identified as the following transfer function –

$$G(s) = \frac{22.6304}{s^2+22.3437s+6.3982 \times 10^{-6}} e^{-0.5508s} \ldots (4.3)$$

To find these transfer function we used MATLABs system identification toolbox. First, we applied a velocity step to the UAV and using the ArUco detection were able to find the response for x and y axis as shown in Fig 4.4 and Fig 4. 5. Then in order to test how good are model is compared to the actual UAV we decided to validate our models by applying random velocity commands and predicted the output with the transfer function and compared it with the data

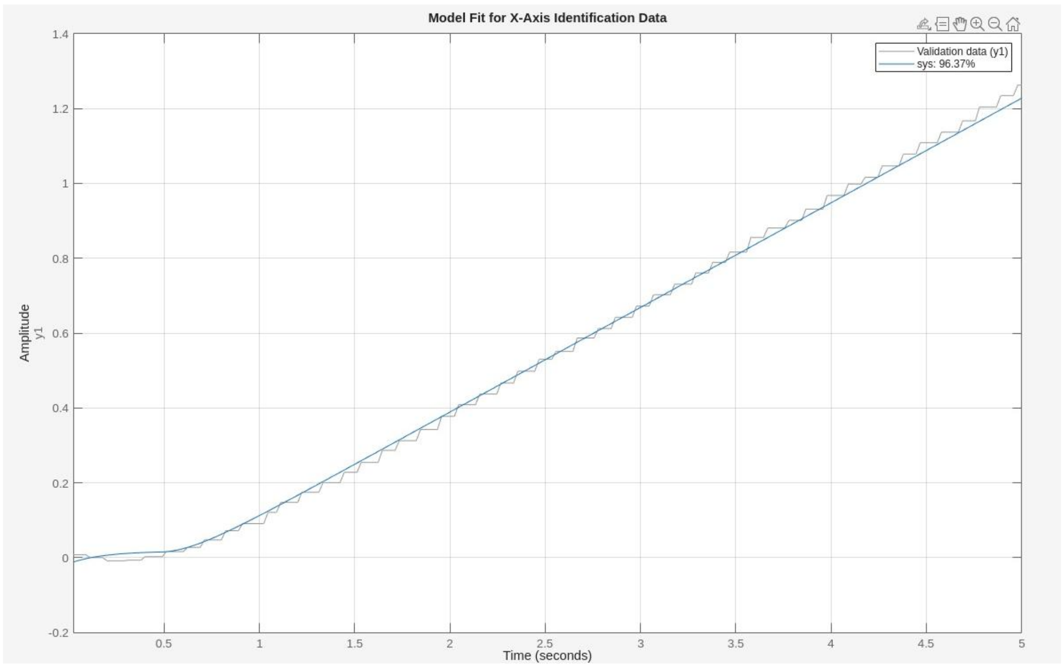


Fig. 4.4 X Axis position displacement for velocity step

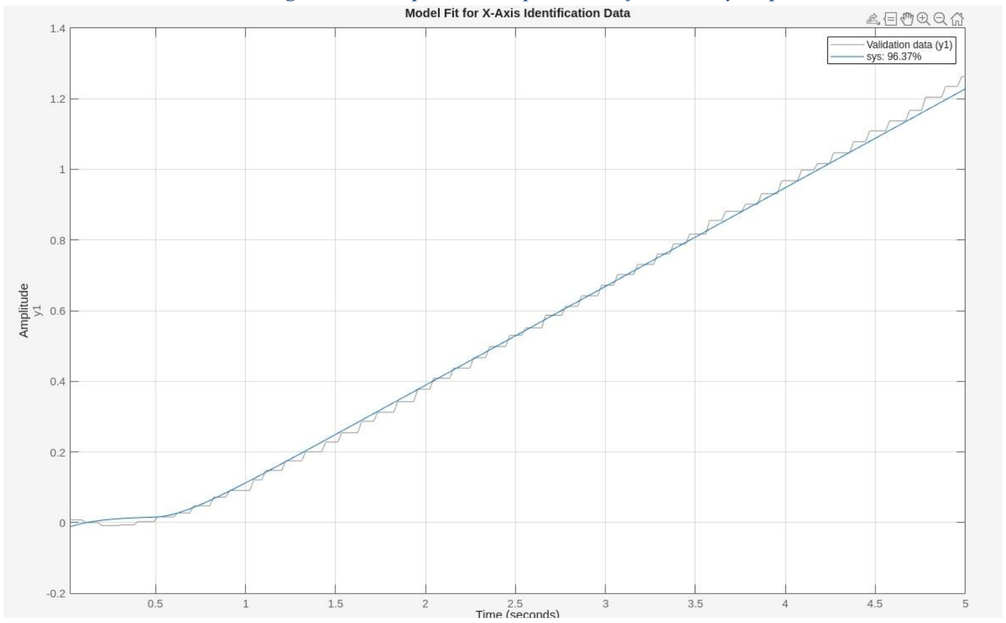


Fig. 4.5 y Axis position displacement for velocity step

recorded form the UAV. This gave us 71.77% validation in x axis (Fig 4.6) and 67.89% in y axis as shown (Fig 4.7).

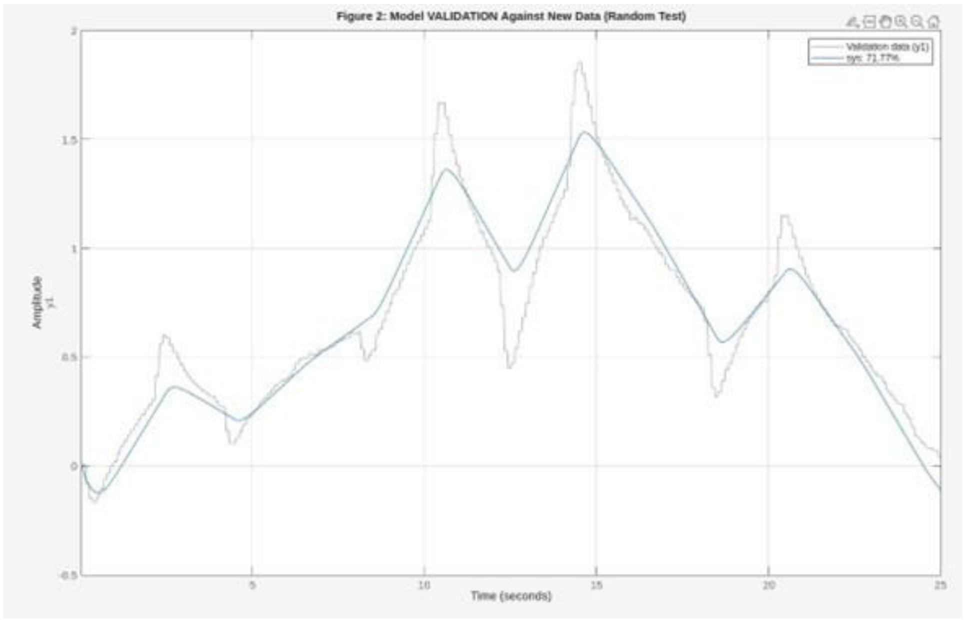


Fig. 4.6 Model Validation for random velocity along X axis

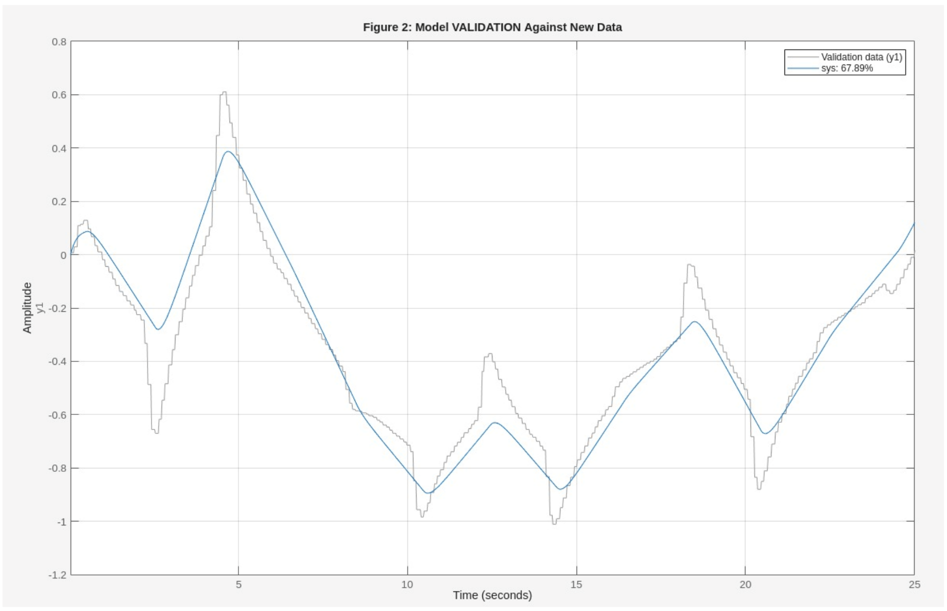


Fig. 4.7 Model Validation for random velocity along Y axis

4.5 PID Control Execution

Now that we had the transfer function, we could then tune the high-level PID controller using MATLAB's PID toolbox. After tuning we found the best PID values for X axis to be

$$K_p = 1.2043;\ K_i = 0.1539;\ K_d = 0.4124 \ldots (4.4)$$

This gave a rise time of 0.673s setting time of 14.6s and an overshoot of 9% as shown in Fig 4.8 Similarly for y axis –

$K_p = 1.3435$; $K_i = 0.1741$; $K_d = 0.4324$ . . . (4.5)

This gave a rise time of 0.369s setting time of 2.91s and an overshoot of 14.6% as shown in Fig 4.9 We found that these PID values offered a rapid response to track the rover at higher velocities and also had a good accuracy when coming for the landing.

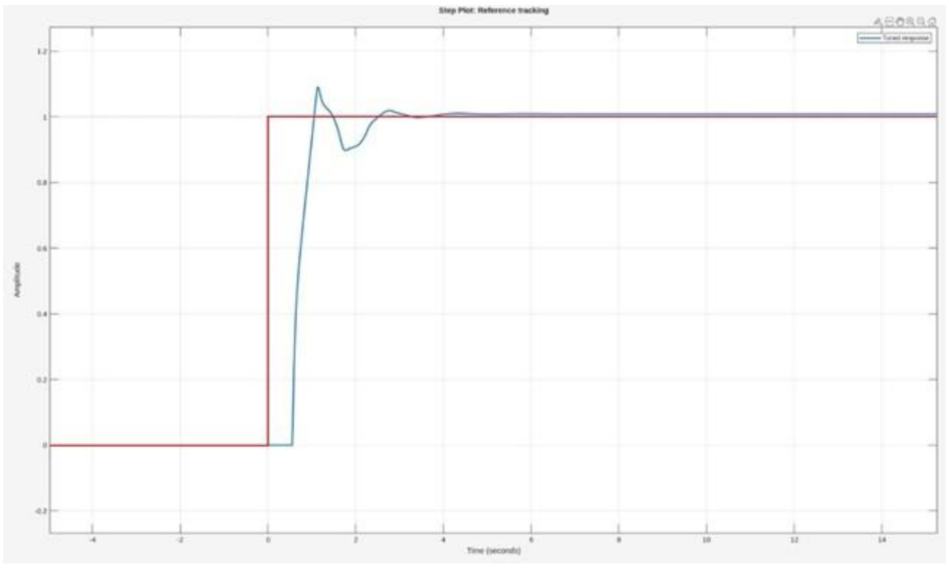


Fig. 4.8 Y Axis step response of tuned PID controller

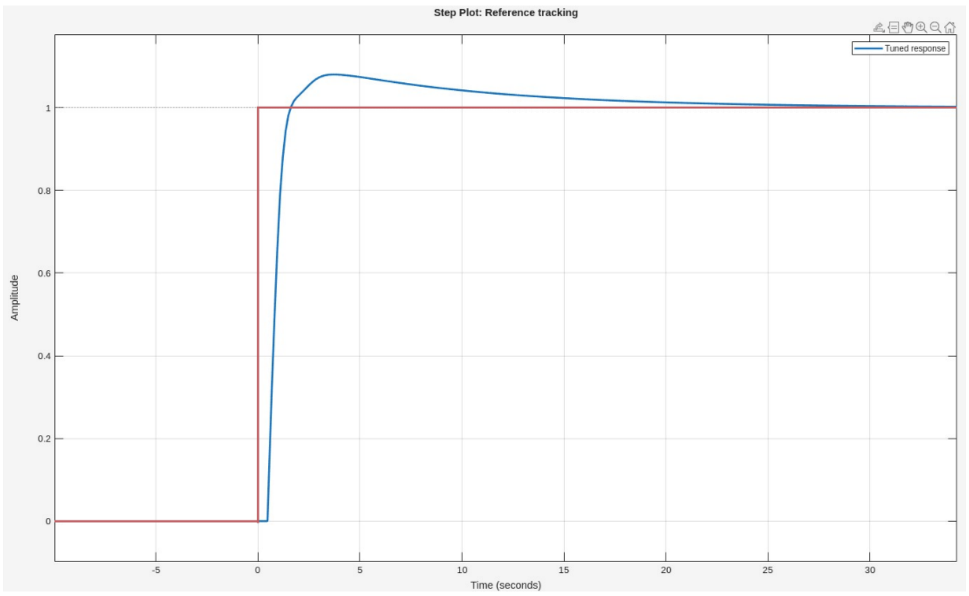


Fig. 4.9 X Axis step response of tuned PID controller

4.6 Descent onto the marker

The rate of decent depends on two factors the altitude of the drone and the lateral error of the rover. To achieve soft landing but still land at a quick pace the descent velocity setpoint is scaled directly with the UAV's altitude. The system commands a maximum descent rate of 1.0 m/s when the UAV is above the braking threshold (2.0 m), employing a linear interpolation function. The descent velocity is smoothly tapered down below this threshold altitude value to a minimum floor of 0.1 m/s at the terminal altitude (0.5 m), which helps to avoid hard impacts and minimize flight time.

To make sure the rover is always in sight of the UAV we introduce a clamp factor to the rate of decent. This is scaled from 1 when the rover is aligned perfectly underneath the drone to a factor of 0.1 when the rover is at the edge of the camera frame.

At an altitude of less than 0.02m the drone will disarm landing on the rover.

4.7 Rover Motion Design

The rover follows scripted motion profiles implemented through Gazebo's model controllers. These profiles include uniform-speed motion, velocity-varying behavior, and constant-acceleration movement. Each scenario challenges the UAV to maintain accurate tracking despite changes in rover dynamics, thereby validating the robustness of the implemented landing strategy. The rover was designed to have 4 wheels, and was fitted with Ackerman steering, to ensure stability of the platform the drone will land on. It was not fit with suspension so as to reduce the tilt the platform undergoes. The rover was able to achieve speeds of 20m/s, without any instabilities, and was able to accelerate at speeds of $2m/s^2$ without any turns. Missions were conducted on it while it was:

- Stationary
- Under constant velocity
- Under constant acceleration
- under variable acceleration with a variance of $0.5m/s^2$

Mission Plans were made surrounding these parameters.

4.8 Hardware Ready Architecture

Although the current implementation is entirely simulation-based, the architecture is designed for future deployment on physical hardware with minimal modification. A Pixhawk-class flight controller, a downward-facing camera, a Raspberry Pi 5 or Jetson companion computer, a wheeled rover platform and an IR-assisted nested marker setup can directly replace the simulated components. Because MAVLink and ArUco remain identical in both simulation and hardware, the transition pathway is straightforward.

Chapter 5: RESULTS AND ANALYSIS

The autonomous detection, tracking, and landing capabilities of the Unmanned Aerial Vehicle (UAV) were rigorously evaluated in simulation.

A critical finding consistent across all test conditions was that the maximum altitude of the detection of ArUco marker 10 m. At this range, the Outer marker was detected, above this altitude detection of marker was unreliable.

5.1 Mission Scenario

Here in Fig 5.1, we can see the entire mission as it was intended. At the start both rover and UAV are moving together, after 10 seconds the UAV is deployed to perform a waypoint mission (visit two GPS coordinates autonomously). After it has successfully finished its waypoint mission, it starts moving towards the GPS coordinate of the rover. Once it gets visual lock on the rover, it starts its decent. After 13.6 seconds it lands on the rover with a lateral error of 2.73cm as shown in Fig 5.2. This mission plan was tested as the rover was moving at a constant 2m/s.

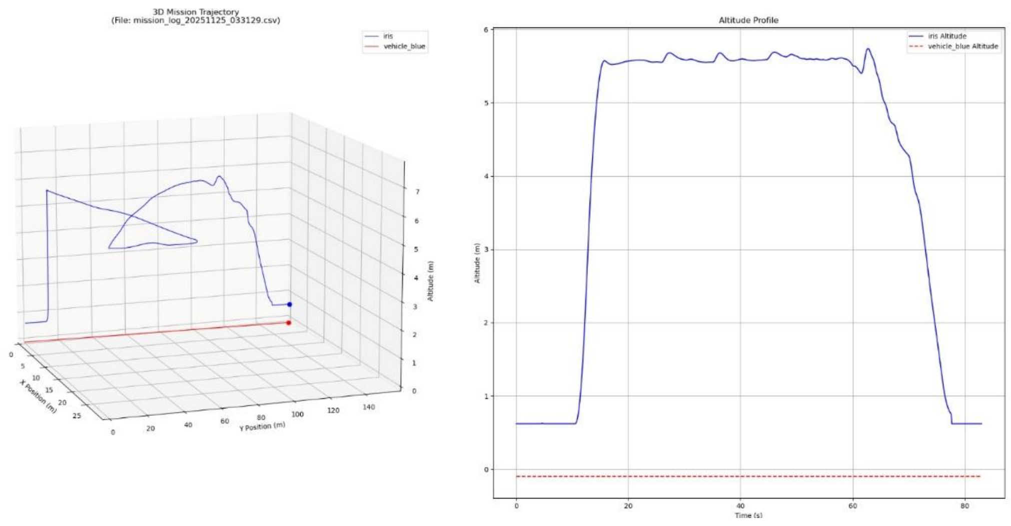
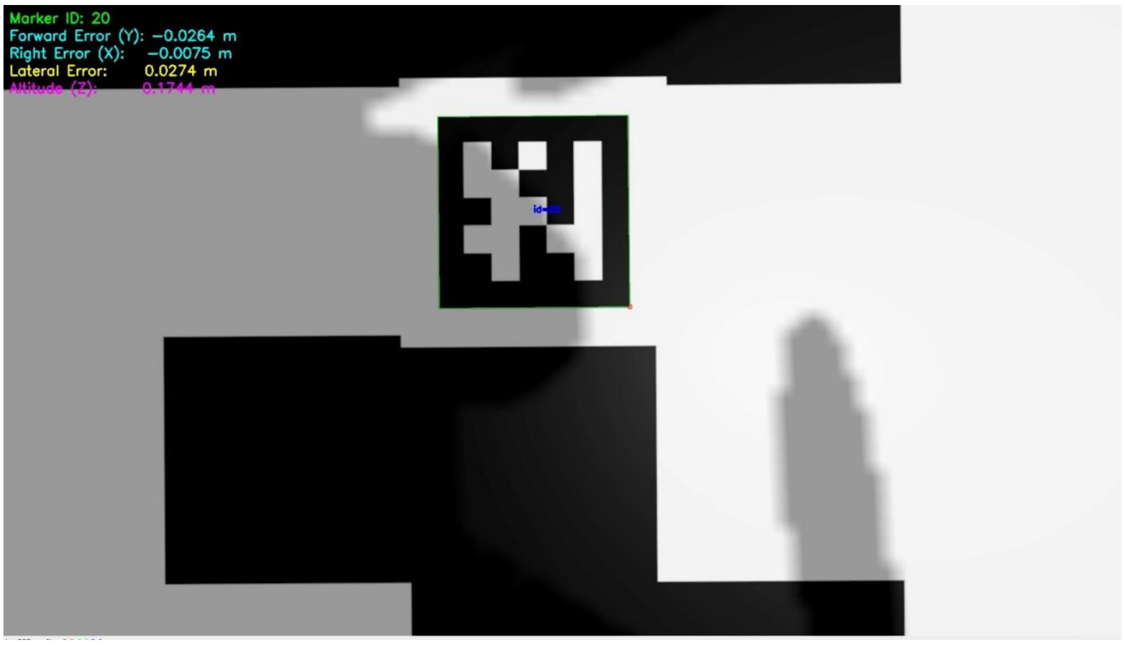


Fig. 5.1 3D mission trajectory and altitude plot.



```
Marker ID: 20
Forward Error (Y): -0.0264 m
Right Error (X):   -0.0075 m
Lateral Error:      0.0274 m
Altitude (Z):       0.1744 m
```

Fig. 5.2 Drone landing on Rover

5.2 Rover Moving in Circle

In Fig 5.3 the rover moves in a circle and the UAV has to land on it. Initially it oscillates while following the trajectory due to the low detection rate of the camera but later on it is able to get a lock and land on the rover with a lateral error of 7.82cm in a time of 32 seconds from detection.

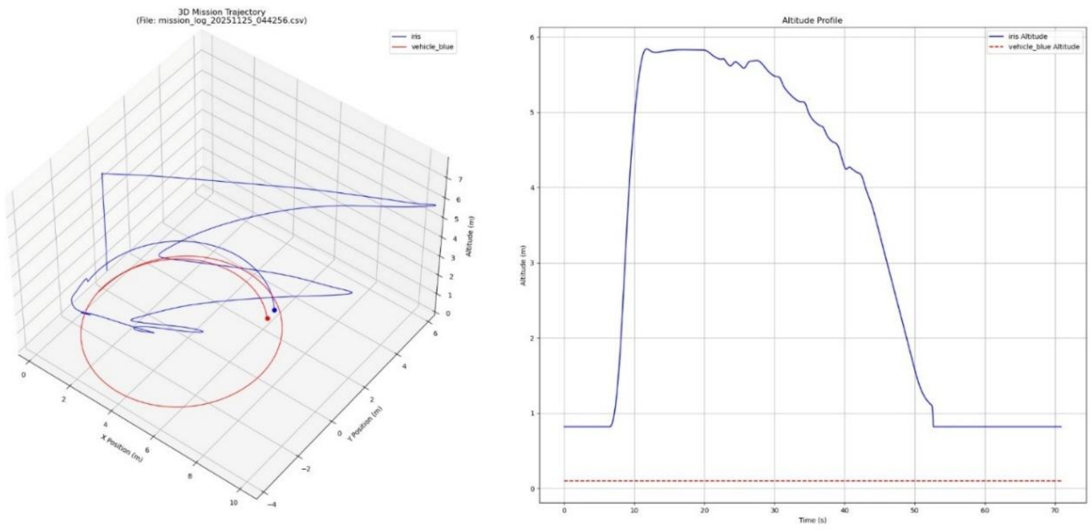


Fig. 5.3 Landing when rover is moving in a circle

5.3 Tracking during changing velocity

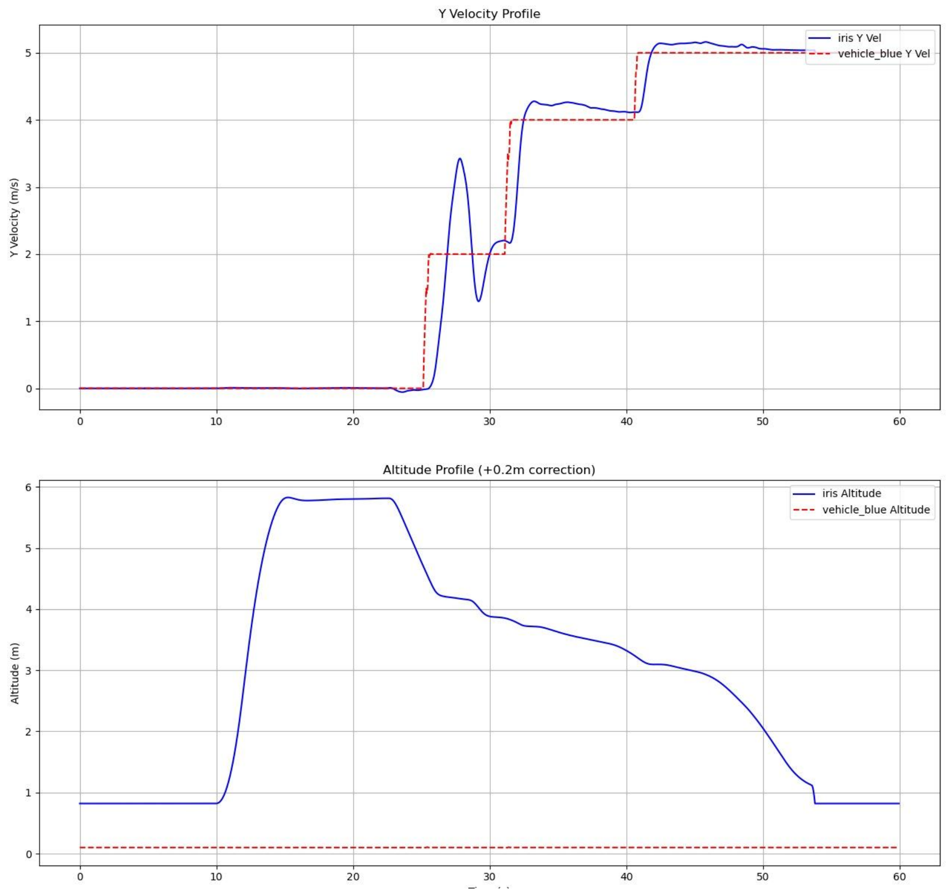


Fig. 5.4 The trajectories of the drone(blue) and the rover (red) during an accelerative mission plan

Here we keep increasing the velocity until the UAV landed. The UAV was able to handle changes in velocity even when it was only 3m above the rover. Finally, the max speed when landing is possible is 5m/s at which it had a landing accuracy of 10.16cm.

Chapter 6: CONCLUSION

This research successfully solves and demonstrates a robust solution for "Autonomous UAV takeoff and landing on moving platforms". We successfully developed a simulation environment comprising of a collaborative UAV and vehicle system, in which the UAV utilizes vision-based tracking, that was refined using system identification to tune the high level PID controller for better performance in tracking. This was further developed on by offboarding the vehicle's odometry data to the UAV which would help it localize with the vehicle in case there is a loss of camera frames due to whatever reason, which was demonstrated by introducing noise and delay to mimic real world scenarios. This was successfully demonstrated in simulation and yielded good results wherein the UAV successfully lands on the rover at high speeds in both straight line following and in curved paths, with high precision.

However, these tests we're performed in ideal weather conditions and further development of the project would be necessary in varying wind conditions and low light conditions as well.

Chapter 7: FUTURE SCOPE

While the foundational principles and performance metrics have been thoroughly validated in the simulation environment, the long-term goal necessitates the transition to a physical platform. The immediate and primary focus for future development is the hardware porting and validation of the current architecture. This transition will require using an onboard flight controller in tandem with a dedicated companion computer (e.g., using a modern, low-power system like NVIDIA Jetson or similar) for vision processing. Essential hardware components include a monocular RGB camera, a reliable NavSat positioning system, IMU and wheel encoders with low drift. Further research efforts will concentrate on algorithmic enhancements aimed at improving dynamic operational capabilities.

The tracking strategy can be significantly improved by migrating from simple trajectory following to more sophisticated Model Predictive Controller (MPC) or risk-aware path planning. The intent here would be for the UAV to proactively predict the future state of the rover, specifically during turns or sudden accelerations of the rover to develop smoother capture trajectories as well as reduce control effort during the terminal part of the capture. Future work will include the application of a dynamic-window approach (DWA) or a similar local planning method to allow the UAV to handle immediate and out-of-expectation obstacles easily and reaction time for sudden changes of the rover speed.

Another area for improvement is the visual robustness of the detection system for deployment in real-world outdoor environments and low-light conditions. For instance, extreme environmental conditions such as glare (specular reflection), changes in illumination or even partial occlusion of the marker by dust or shadow may severely affect the reliability of ArUco detection. To mitigate these environmental effects, we can investigate adaptive thresholding algorithms, applying tuning of filtering techniques (e.g., selective tuning of the system prediction logic) in a motion-aware vision context, and leveraging other techniques for marker detection, including lightweight Convolutional Neural Networks (CNNs). These detection techniques typically have increased robustness compared to classical computer vision under a variable exposure environment.

Experimenting with alternative fiducial markers or using stereo cameras for better depth perception may also the maximum detection range of 10m. The eventual outcome of these efforts will be real world field trials in commercial applications that can ultimately evaluate the system’s performance, repeatability, and endurance. Successfully addressing these challenges provide a robust solution for autonomous landing on moving vehicles, allowing UAVs to be utilize in new fields as well as enhancing their capabilities in existing applications.

Chapter 8: REFERENCES

[1] M. Kyas and J. Springer, "Autonomous Drone Landing with Fiducial Markers and a Gimbal-Mounted Camera for Active Tracking," *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*, Italy, 2022, pp. 243-247.

[2] A. Khazetdinov, A. Zakiev, T. Tsoy, M. Svinin and E. Magid, "Embedded ArUco: a novel approach for high precision UAV landing," *2021 International Siberian Conference on Control and Communications (SIBCON)*, Kazan, Russia, 2021, pp. 1-6.

[3] H. B. Arie Wicaksana, R. Mardiyanto and A. N. Irfansyah, "Drone Position Tracking System based on Object Detection and ArUco Marker for Autonomous Navigation Applications," *2024 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, Mataram, Indonesia, 2024, pp. 131-135.

[4] Taylor Ripke, Tony Morelli, 0, "Autonomous Landing of a Drone using Smart Vision", *International Journal of Engineering Research & Technology (IJERT),* Volume 07, Issue 11 (2018).

[5] J. Wang and D. McKiver, "Precision UAV Landing Control Based on Visual Detection," *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, Shenzhen, China, 2020.

[6] D. H. C. Silva, M. F. Santos, M. F. Silva, A. F. S. Neto and P. Mercorelli, "Design of Controllers Applied to Autonomous Unmanned Aerial Vehicles Using Software In The Loop," *2019 20th International Carpathian Control Conference (ICCC)*, Krakow-Wieliczka, Poland, 2019, pp. 1-6.

[7] M. Saavedra-Ruiz, A. M. Pinto-Vargas and V. Romero-Cano, "Monocular Visual Autonomous Landing System for Quadcopter Drones Using Software in the Loop," in *IEEE Aerospace and Electronic Systems Magazine*, vol. 37, no. 5, pp. 2-16.

[8] C. Rus, E. Lupulescu, M. Leba and M. Risteiu, "Advanced Mathematical Modeling and Control Strategies for Autonomous Drone Systems," *2024 25th International Carpathian Control Conference (ICCC)*, Krynica Zdrój, Poland, 2024, pp. 1-6

[9] A. Roberts and A. Tayebi, "Adaptive position tracking of VTOL UAVs," *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Shanghai, China, 2009, pp. 5233-5238.

[10] P. Smyczyński, Ł. Starzec and G. Granosik, "Autonomous drone control system for object tracking: Flexible system design with implementation example," *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, Miedzyzdroje, Poland, 2017, pp. 734-738.

[11] Zhihui Zheng, Haiping Wei, Bo Tang and Bin Zhou, "A fast visual

tracking method via spatio-temporal context learning for unmanned rotorcrafts fixed-pointed landing," *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, Nanjing, 2016, pp. 2006-2010.

[12] A. Salagame, S. Govindraj, and O. S. N, "Precision Landing of a UAV on a Moving Platform for Outdoor Applications," *arXiv (Cornell University)*, Jan. 2022.

[13] A. Keller and B. Ben-Moshe, "A Robust and Accurate Landing Methodology for Drones on Moving Targets," *Drones*, vol. 6, no. 4, p. 98, Apr. 2022.

[14] Chang CW, Lo LY, Cheung HC, Feng Y, Yang AS, Wen CY, Zhou W. "Proactive Guidance for Accurate UAV Landing on a Dynamic Platform: A Visual-Inertial Approach." *Sensors (Basel)*. Jan 2022.

[15] Guo, K., Tang, P., Wang, H., Lin, D., and Cui, X., "Autonomous Landing of a Quadrotor on a Moving Platform via Model Predictive Control", *Aerospace*, vol. 9, no. 1, Art. no. 34, 2022.

[16] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico and D. Scaramuzza, "Vision-based autonomous quadrotor landing on a moving platform," *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, Shanghai, China, 2017, pp. 200-207.

[17] A. Paris, B. T. Lopez and J. P. How, "Dynamic Landing of an Autonomous Quadrotor on a Moving Platform in Turbulent Wind Conditions," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 9577-9583.

[18] S. P. V. Subamanian, S. M. Subramanian, P. Muthiah and J. M. A. Shajahan, "Autonomous Drone Landing on a Moving Naval Base using Vision-Based Robot Control," *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Tenerife, Canary Islands, Spain, 2023, pp. 1-8.

[19] Saj, Vishnu, Bochan Lee, Dileep Kalathil, and Moble Benedict. "Robust Reinforcement Learning Control for Vision-Based Ship Landing of VTOL-UAVs." *Journal of the American Helicopter Society (2025).*

[20] S. Lin, L. Jin, and Z. Chen, "Real-Time Monocular Vision System for UAV Autonomous Landing in Outdoor Low-Illumination Environments," *Sensors*, vol. 21, no. 18, p. 6226, Sep. 2021.

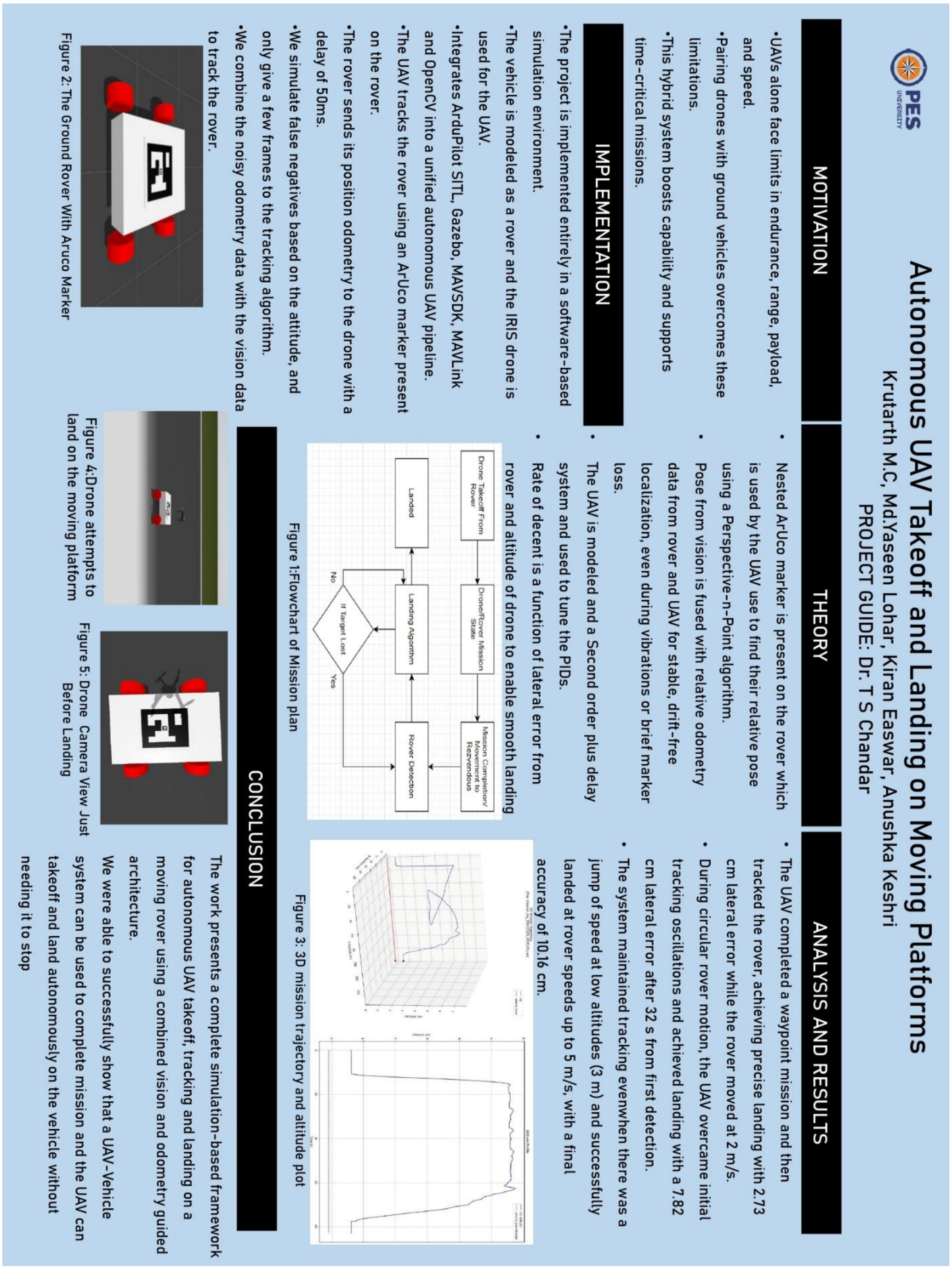
[21] Li, Yuezhou & Niu, Yuzhen & Xu, Rui & He, Yuqi. (2024), "Zero-Referenced Enlightening and Restoration for UAV Nighttime Vision", in *IEEE Geoscience and Remote Sensing Letters*, vol 21, Dec 2023.

[22] I. Kalinov, E. Safronov, R. Agishev, M. Kurenkov and D. Tsetserukou, "High-Precision UAV Localization System for Landing on a Mobile Collaborative Robot Based on an IR Marker Pattern Recognition," *2019 IEEE 89th Vehicular Technology Conference (VTC 2019-Spring)*, Kuala Lumpur, Malaysia, 2019, pp. 1-6.

[23] G. Badakis, M. Koutsoubelias and S. Lalis, "Robust Precision Landing for Autonomous Drones Combining Vision-based and Infrared Sensors," *2021 IEEE Sensors Applications Symposium (SAS)*, Sundsvall, Sweden, 2021, pp. 1-6.

[24] H. Raei, Y. Cho and K. Park, "Autonomous landing on moving targets using LiDAR, Camera and IMU sensor Fusion," *2022 13th Asian Control Conference (ASCC)*, Jeju, Korea, Republic of, 2022, pp. 419-423

Chapter 9: POSTER



Autonomous UAV Takeoff and Landing on Moving Platforms

Krutarth M.C, Md.Yaseen Lohar, Kiran Easwar, Anushka Keshri
PROJECT GUIDE: Dr. T S Chandar

MOTIVATION

- UAVs alone face limits in endurance, range, payload, and speed.
- Pairing drones with ground vehicles overcomes these limitations.
- This hybrid system boosts capability and supports time-critical missions.

IMPLEMENTATION

- The project is implemented entirely in a software-based simulation environment.
- The vehicle is modeled as a rover and the IRIS drone is used for the UAV.
- Integrates ArduPilot SITL, Gazebo, MAVSDK, MAVLink and OpenCV into a unified autonomous UAV pipeline.
- The UAV tracks the rover using an ArUco marker present on the rover.
- The rover sends its position odometry to the drone with a delay of 50ms.
- We simulate false negatives based on the attitude, and only give a few frames to the tracking algorithm.
- We combine the noisy odometry data with the vision data to track the rover.

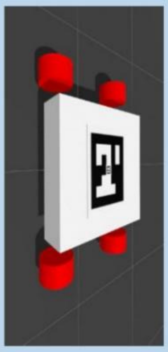


Figure 2: The Ground Rover With Aruco Marker

THEORY

- Nested ArUco marker is present on the rover which is used by the UAV use to find their relative pose using a Perspective-n-Point algorithm.
- Pose from vision is fused with relative odometry data from rover and UAV for stable, drift-free localization, even during vibrations or brief marker loss.
- The UAV is modeled and a Second order plus delay system and used to tune the PIDs.
- Rate of decent is a function of lateral error from rover and altitude of drone to enable smooth landing

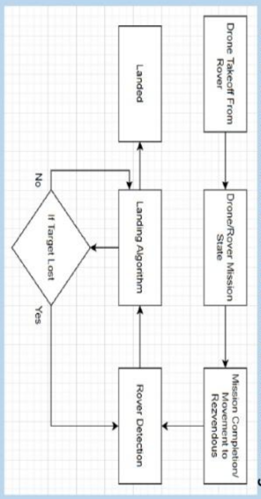


Figure 1:Flowchart of Mission plan

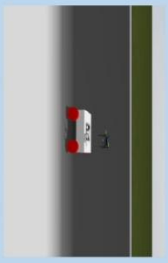


Figure 4:Drone attempts to land on the moving platform

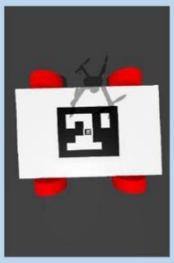


Figure 5: Drone Camera View Just Before Landing

ANALYSIS AND RESULTS

- The UAV completed a waypoint mission and then tracked the rover, achieving precise landing with 2.73 cm lateral error while the rover moved at 2 m/s.
- During circular rover motion, the UAV overcame initial tracking oscillations and achieved landing with a 7.82 cm lateral error after 32 s from first detection.
- The system maintained tracking evenwhen there was a jump of speed at low altitudes (3 m) and successfully landed at rover speeds up to 5 m/s, with a final accuracy of 10.16 cm.

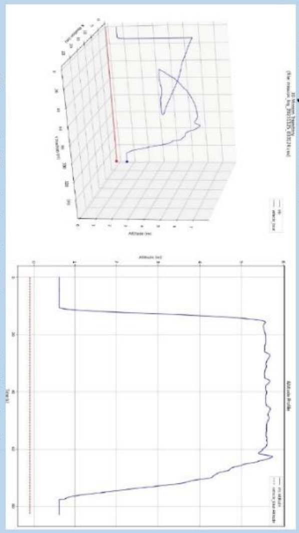


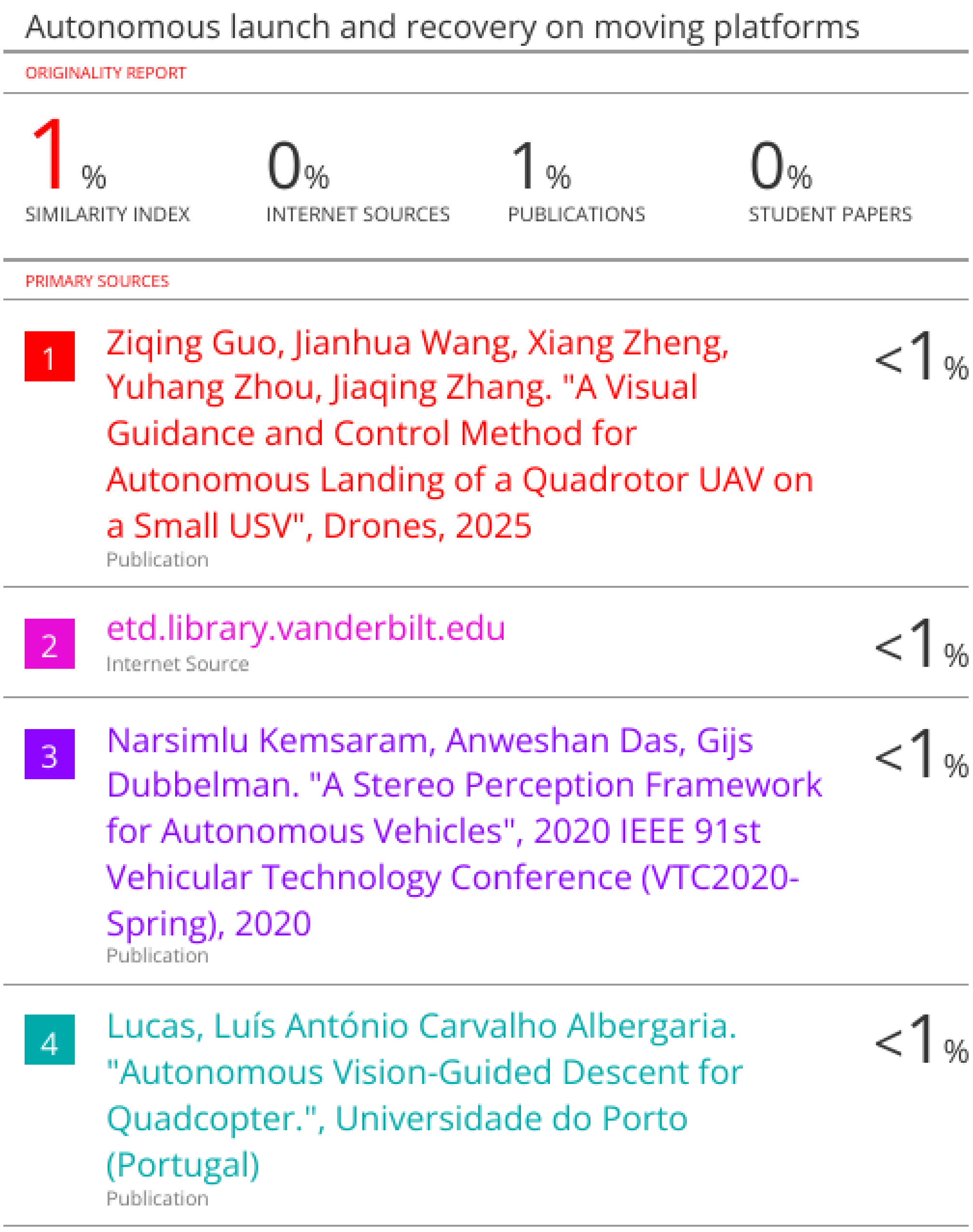
Figure 3: 3D mission trajectory and altitude plot

CONCLUSION

The work presents a complete simulation-based framework for autonomous UAV takeoff, tracking and landing on a moving rover using a combined vision and odometry guided architecture.

We were able to successfully show that a UAV-Vehicle system can be used to complete mission and the UAV can takeoff and land autonomously on the vehicle without needing it to stop

Chapter 10: PLAGIARISM REPORT



Autonomous launch and recovery on moving platforms

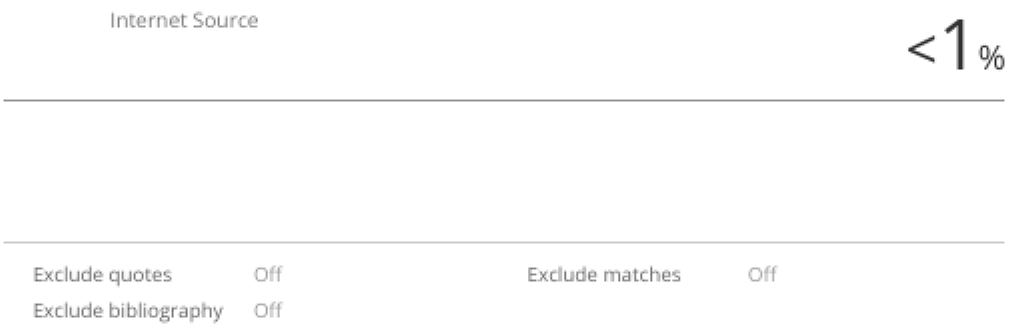
ORIGINALITY REPORT

1%	0%	1%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ziqing Guo, Jianhua Wang, Xiang Zheng, Yuhang Zhou, Jiaqing Zhang. "A Visual Guidance and Control Method for Autonomous Landing of a Quadrotor UAV on a Small USV", Drones, 2025 Publication	<1%
2	etd.library.vanderbilt.edu Internet Source	<1%
3	Narsimlu Kemsaram, Anweshan Das, Gijs Dubbelman. "A Stereo Perception Framework for Autonomous Vehicles", 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), 2020 Publication	<1%
4	Lucas, Luís António Carvalho Albergaria. "Autonomous Vision-Guided Descent for Quadcopter.", Universidade do Porto (Portugal) Publication	<1%

5	Maghareh, Amin. "Nonlinear Robust Framework for Real-time Hybrid Simulation of Structural Systems: Design, Implementation, and Validation.", Purdue University, 2017 Publication	<1 %
6	www.coursehero.com Internet Source	<1 %
7	Submitted to Queen Mary and Westfield College Student Paper	<1 %
8	Joshua Springer, Marcel Kyas. "Autonomous Drone Landing: Marked Landing Pads and Solidified Lava Flows", International Journal of Semantic Computing, 2024 Publication	<1 %
9	ASHRAF S., SABER A., TAREK T.. "MIXED-REALITY ENVIRONMENT FOR ONLINE SYSTEM IDENTIFICATION OF NONLINEAR SYSTEMS", The International Conference on Applied Mechanics and Mechanical Engineering, 2008 Publication	<1 %
10	Lecture Notes in Computer Science, 2015. Publication	<1 %
11	docplayer.net Internet Source	<1 %
12	www.kennisplatformtunnelveiligheid.nl	



Internet Source <1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		