

Range Extension for Delivery Drones: Autonomous Landing on Non-Cooperative Targets

General Engineering, ME6-624, 2026





Aalborg Universitet
<http://www.aau.dk>

AALBORG UNIVERSITET

STUDENTERRAPPORT

Title:

Range Extension for Delivery Drones:
Autonomous Landing on Non-
Cooperative Targets

Theme:

Sustainable Development Goal 12: Sus-
tainable Consumption and Production

Project Period: Spring 2026

Project Group: ME6-624

Participants:

Thomas Elberg Nielsen

Supervisors:

Dao Zhou

Page Numbers: 53

Date of Completion:

June 9, 2026

Resumé: Bæredygtigt forbrug in-
volverer bæredygtig levering. Forskn-
ing viser at dronelevering har poten-
tiale for at reducere miljøpåvirkning.
Aktuelt er droners rækkevidde be-
grænset til kun at levere i byområder
og forstæder. Dette projekt foreslår
at udvide rækkevidden med offentlig
transport ved at udvikle en kamera-
baseret løsning til landing på ikke-
samarbejdende køretøjer. Løsningen er
evalueret, simuleret og valideret i prak-
tiske forsøg med en Holybro X500 v2
quadrotor drone. Simulering af land-
ing på statiske og dynamiske mål viste
at det var muligt at opnå en præcision
på henholdsvis 2.9 cm og 20 cm. Fy-
siske forsøg opnåede en præcision på
henholdsvis 23 cm og 62 cm. Projek-
tet konkluderer, at visionsbaseret land-
ing på ikke-samarbejdende køretøjer er
mulig, men at konceptet med offentlig
transport kræver yderligere arbejde.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

I want to thank Dao Zhou for accepting to supervise the project. Special thanks go to the members of the student association "AAU Space Robotics" for cooperating with me on the autopilot and simulation software and lending me the
5 physical drone for physical flight tests.

The following programs were used in this project: Overleaf Latex editor, Visual Studio Code, Gazebo simulation environment and QGroundControl Ground Control Station software. Software is written in Python and C++.

10 Generative AI tools were used for: illustrations, grammar correction and coding tasks. Specifically Google Gemini for visual/grammar related tasks and Claude Code for coding tasks.

15 The full repository is available at:
<https://github.com/thomaselberg/drone-software/tree/Development>
Including the algorithms, configuration files, AAU Space Robotics Autopilot software, simulation environment and more.

20 A video from the physical flight test can be accessed here:
https://youtu.be/Zc_a64RxrDg



Thomas Elberg Nielsen
<teni23@student.aau.dk>

Abstract

The rapid growth of e-commerce raises the question of how the associated resource use and environmental impact can be reduced. Drone delivery outperforms traditional methods in low-volume single-parcel rural scenarios, but the short service range of current delivery drones restricts operation to densely populated areas. This work proposes to extend that range by docking with existing public transport vehicles, and identifies the docking maneuver as the open technical sub-problem. A vision-based landing algorithm is developed for autonomous landing on a non-cooperating ground target, combining an ArUco marker, an OpenCV detector, a proportional control law, and a state machine, with a single-axis camera gimbal enabling a terminal slant landing. The gimbal is evaluated against a static-camera baseline in a PX4 software-in-the-loop and Gazebo simulation and validated in flight tests with a Holybro X500 v2 quadrotor. In simulation, the gimbal improved the landing accuracy from 13.1 cm to 2.9 cm on static targets and from 28.6 cm to 20.1 cm on dynamic targets, with engagement durations within the ten-second upper bound derived from rural bus dwell times. Static real flight tests yielded a landing accuracy of 23 cm and a 1.1° angular error. Dynamic tests were negatively affected by an altitude-hold issue and achieved an accuracy of 62 cm and 1.8° . The results indicate that vision-based docking on non-cooperating ground vehicles is feasible but the public transport concept requires further work.

Abbreviations

Abbreviation	Explanation
AAU	Aalborg University
API	Application Programming Interface
B2C	Business-to-Consumer
BRT	Bus Rapid Transit
BVLOS	Beyond Visual Line of Sight
C2B	Consumer-to-Business
C2C	Consumer-to-Consumer
COM	Center of Mass
DDS	Data Distribution Service
DMI	Danish Meteorological Institute
EASA	European Union Aviation Safety Agency
EKF	Extended Kalman Filter
FOV	Field of View
GHG	Greenhouse Gas
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Unit
GWP	Global Warming Potential
Micro XRCE-DDS	Micro eXtremely Resource Constrained Environments – Data Distribution Service
NED	North-East-Down
ROS	Robot Operating System

RTK	Real-Time Kinematic
SITL	Software-in-the-Loop
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-Off and Landing
YOLO	You Only Look Once

Symbols

Symbol	Explanation	Unit
α	Horizontal camera field of view	[rad]
C	Optical axis intersection with the ground plane	[m]
c_u, c_v	Image principal point coordinates	[px]
$\mathbf{d}$	Pixel ray direction in the NED frame	[—]
$\mathbf{d}_c$	Pixel ray direction in the camera frame	[—]
d_D	Downward component of $\mathbf{d}$	[—]
$\mathbf{e}$	Ground error vector	[m]
e_E	East component of the ground error	[m]
e_N	North component of the ground error	[m]
f	Focal length expressed in pixels	[px]
f_x, f_y	Focal lengths along image axes	[px]
$\mathbf{g}$	Ground point beneath the projected pixel ray	[m]
h	Drone altitude above the ground plane	[m]
H	Image height	[px]
h_{low}	Low-altitude descent setpoint	[m]
k	Scaling factor along the pixel ray to the ground plane	[—]
ϕ	Roll angle about the body x -axis	[rad]
ψ	Yaw angle about the body z -axis	[rad]
$\mathbf{R}$	Rotation matrix from the camera frame to the NED frame	[—]

$\mathbf{R}_{\text{cam}}$	Rotation matrix from the camera frame to the body frame	$[-]$
$\mathbf{R}_{\text{drone}}$	Rotation matrix from the body frame to the NED frame	$[-]$
θ	Pitch angle about the body y -axis	$[\text{rad}]$
$\theta_{\text{effective}}$	Combined drone and gimbal camera pitch	$[\text{rad}]$
T	Marker-center ground point	$[\text{m}]$
T_{high}	High-altitude stabilization dwell time	$[\text{s}]$
T_{hold}	Hover hold time after loss of marker lock	$[\text{s}]$
T_{low}	Low-altitude stabilization dwell time	$[\text{s}]$
u, v	Image pixel coordinates	$[\text{px}]$
$v_x(t)$	Target forward velocity in the dynamic-target model	$[\text{m/s}]$
$v_{y,\text{raw}}(t)$	Target raw lateral velocity in the dynamic-target model	$[\text{m/s}]$
W	Image width	$[\text{px}]$
x, y, z	Body-frame coordinates	$[\text{m}]$
$x_{\text{off}}, y_{\text{off}}$	Camera mounting offsets in the body frame	$[\text{m}]$

Table of contents

	1 Introduction	1
	2 Problem analysis	2
	2.1 Last-Mile Delivery	2
5	2.1.1 Mode of Transportation	2
	2.1.2 Environmental Impact	3
	2.2 Delivery Method and Preferences	3
	2.2.1 Rural Density Drawback	4
	2.2.2 Urban Congestion Drawback	4
10	2.3 Autonomous Delivery	4
	2.3.1 Drone Delivery Literature Review	5
	2.3.2 Commercially Implemented Delivery Drones	5
	3 Problem Statement	7
	4 Technical Analysis	8
15	4.1 Delivery Drone Hitch-Hiking Proposal	8
	4.2 Feasibility	9
	4.2.1 Physical Space on Public Transportation Vehicles	9
	4.2.2 Delivery at Destination	9
	4.2.3 Path Planning	10
20	4.2.4 Size and Weight of Packages	10
	4.3 Autonomous Drone Landing Literature Review	11
	5 Project Scope	13
	6 Requirements Specification	15
	6.1 Landing Duration	16
25	6.2 Landing Accuracy	17
	6.3 Landing Robustness	17
	7 Vision-Based Landing Algorithm	19
	7.1 Marker Detection	20
	7.2 Ground-Error Projection	21
30	7.3 Mission Algorithms	23

Table of contents

8	Simulation	28
9	Hardware UAV flight test	30
10	Results	32
11	Discussion	41
5	11.1 Methodological Limitations	41
	11.2 Comparison with Similar Published Works	42
12	Conclusion	43
13	Future Work	45
	Bibliography	46
10	A Static Landing Algorithm	46
	B Static Camera Baseline Simulation Results and Graphs	47
	C ROS 2 Computation Graph Visualized using rqt_graph	51
	D Additional Physical Flight Plots	52

1 Introduction

The electronic commerce (e-commerce) market is growing steadily across the world and more than 80% of Nordic consumers shop online at least monthly [1]. In Sweden, domestic e-commerce grew by 10% in 2025 and online "home electronics" sales outperformed in-store sales for the first time. The share of online sales relative to total retail sales increased for all categories of goods [2]. The rise of e-commerce is accompanied by a rise in package deliveries: in the Nordic countries alone, 854 million parcels were delivered in 2023 [3].

The twelfth Sustainable Development Goal of the United Nations addresses sustainable production and consumption, which includes target 12.2 of sustainable management and efficient use of natural resources [4]. Target 12.2 is a qualitative objective, so assessing the impact from parcel delivery requires measurable proxies. This report quantifies environmental impact using the metrics common to the delivery-systems literature: greenhouse gas (GHG) emissions, energy use, and life-cycle assessment categories such as particulate matter.

Physical and online stores contribute to resource use and emissions alike, while the underlying environmental impact differs in some key areas. Some argue that one van delivering ten packages is less resource-intensive than ten individuals driving private cars to retail stores. Others argue that the increased frequency of returns from online shopping more than offsets that difference. Regardless of where the balance lies, online shopping is steadily overtaking retail shopping and the rise of e-commerce means parcel deliveries are here to stay. This raises the question:

How can the natural resource use and environmental impact from parcel deliveries be reduced?

2 Problem analysis

Logistics at large scale has adopted the fast growth in e-commerce to utilize economies of scale for a majority of the journey. E-commerce companies, like Amazon, have built out logistics infrastructure and large distribution centers, that efficiently moves a high volume of goods with low effort. Less dominant e-commerce businesses typically partner with third-party logistics providers to get volume efficiencies and use existing infrastructure.

2.1 Last-Mile Delivery

Last-mile delivery is much less efficient in terms of cost and environmental impact than the preceding stages of the supply chain. A literature review by Kumar et al. reports that the final leg accounts for up to 53% of the shipping cost and a significant portion of the environmental impact, despite the distance covered being only a fraction of the total journey [5].

The high volume of goods transported between distribution centers allows for large vehicles operating at high utilization rates, which brings down cost and environmental impact per item. In the final leg of the delivery journey, the volume drops drastically and the economies of scale are reduced. This is especially true for rural last-mile delivery, which is characterized by per-parcel distribution of individual orders that cannot be consolidated and scaled. Urban last-mile parcel delivery is characterized by higher density and wide adoption of service points and parcel lockers, which limit the share of single-parcel home deliveries [1].

2.1.1 Mode of Transportation

In 2024, 65% of global last-mile deliveries were carried out using vehicles with internal combustion engines [6]. This share is declining steadily as logistics companies electrify their last-mile delivery fleets.

Parcels are mostly delivered by vans, while bicycles and motorcycles are widespread in food delivery services and mail distribution. Drones have additionally been

2.2. Delivery Method and Preferences

explored as a delivery mode, which is elaborated on in subsection 2.3.1.

2.1.2 Environmental Impact

- The environmental impact of a delivery vehicle is the sum of contributions across all stages of the product life cycle. For vehicles, the contributions from two stages dominate: the production phase, covering manufacture of the vehicle and, for electric vehicles, the battery; and the operational phase, which mainly consists of combustion emissions and tire wear.
- A key distinction between electric and diesel vehicles lies in how their environmental impact is distributed across these two phases. For electric vans, the production phase accounts for the majority of the life-cycle impact, largely due to battery manufacturing. For diesel vans, the operational phase dominates because of engine combustion emissions. Both also contribute to road congestion and degradation. Measured over the full life-cycle, GHG emissions are between 12% and 98% lower for electric delivery vans than for diesel counterparts. The operational phase impact of an electric vehicle, however, depends heavily on the local share of renewable electricity generation [7].

2.2 Delivery Method and Preferences

- The method of delivery refers to the destination at which the customer receives the parcel. Across the Nordic countries, the most common method is the "service point", which is a pick-up point often located in a store or kiosk. Home delivery is the preferred method in all Nordic countries except Finland, where parcel lockers are both the most used and the most preferred. Parcel lockers are app-controlled lockers, often located outdoors and available at all times. The discrepancy between preferred and used method often comes down to price. Across all Nordic countries, the most important factor when choosing where to shop was the "possibility to choose delivery method or pickup location", and 26% of Nordic consumers rank "home delivery with less environmental impact" as the most important sustainability factor [1].

2.3. Autonomous Delivery

2.2.1 Rural Density Drawback

Last-mile delivery in urban and rural areas diverges significantly, as population density and thus parcel density differs. The higher density in cities means that delivery vans cover shorter distances per parcel and deliver more parcels per unit of time, reducing both the cost and environmental impact of home deliveries. The distance between service points and parcel lockers is also lower in cities, so the same logic applies to out-of-home deliveries.

In a study of rural e-commerce customers in Poland, Markowska and Marcinkowski found that customers prioritized "Convenience" and "Speed" for direct-to-home couriers, but overwhelmingly chose parcel lockers for "Cheaper Delivery" [8]. The same study shows that over two thirds of rural out-of-home e-commerce deliveries were retrieved by private car, which adds to the total parcel journey.

2.2.2 Urban Congestion Drawback

While urban density reduces cost and emissions per parcel, the same density introduces drawbacks. Delivery vans contribute to road congestion in city centers, where curbside loading and unloading further restricts traffic flow. This negatively affects air quality and natural resource use by raising fuel consumption for every vehicle with an internal combustion engine held up in the surrounding traffic. The congestion also delays all affected road users, so a cost in lost time is borne across the wider transport system.

2.3 Autonomous Delivery

Autonomous delivery is an emerging field in which part of the delivery journey is performed without human intervention. The most notable example is drone delivery, which has been extensively researched as an alternative to current delivery methods. A literature review is conducted to answer the following questions: How does the cost and environmental impact of drone delivery compare to current delivery methods, and in which scenarios are drone delivery advantageous?

2.3. Autonomous Delivery

2.3.1 Drone Delivery Literature Review

Kumar et al. found that GHG emission and energy use was reduced twice as much in urban areas as in rural areas when implementing drone deliveries[5], while Park et al. found that Global Warming Potential (GWP) reductions were ten times larger in rural areas than in urban areas [9]. The large deviation between these studies show that results depend on the chosen context and assumptions. Several studies have found that a synchronized multi-modal model with cooperating truck and drone can reduce cost, time and environmental impact. The systematic review by Garg et al. highlights one such study reporting a 34% reduction in cost and a 30% reduction in CO₂ emissions [10].

When compared to lighter ground vehicles, drones maintain an advantage in single-parcel delivery scenarios. Park et al. report that the GWP per kilometer for a drone is one-sixth that of a motorcycle, and that fine particulate matter emissions are halved. When evaluated using a Life Cycle Assessment combining ten distinct environmental impact categories, the total environmental impact of a drone delivery is 8% of the impact of a motorcycle delivery. Their results show that the drone outperforms both electric and combustion-engine motorcycles, with larger advantages in rural areas than in urban areas [9].

The advantage of drone delivery shrinks rapidly as the number of parcels delivered in a single tour increases. Therefore, van delivery to home, service point and parcel locker remains superior in urban areas if deliveries are consolidated.

The literature suggests that drone delivery outperforms traditional methods in very-low-volume scenarios where single or few parcels are delivered per tour, and that the advantage is largest in rural areas.

Despite tight regulation, multiple companies have received special permissions to deliver parcels using autonomous drones in the USA, Europe and Asia.

2.3.2 Commercially Implemented Delivery Drones

Zipline is the largest drone delivery operator globally, with more than 2 million deliveries completed by early 2026 [11]. The company has performed medical deliveries in several African countries for multiple years and rapidly expanded its American commercial delivery business in 2025. Zipline uses large Vertical

2.3. Autonomous Delivery

Take-Off and Landing (VTOL) aircraft that transition from hovering to glide-style flight, using wings to generate lift and reduce energy consumption. The payload is delivered by hovering at altitude and lowering it on a zipline until it gently reaches the ground, at which point it is detached and the drone returns to base in forward flight. According to Zipline, the median US delivery takes 3 minutes at a cruising speed of 110 km/h. The US fleet has a service-range radius of 16 km with a 3.6 kg payload, while the medical fleet has six times the range with half the payload mass capability.

In China, the largest food delivery service is Meituan. The vast majority of its deliveries are carried out by humans, but since 2017 the company has performed 780.000 drone deliveries across 70 urban routes. Meituan uses hexacopter drones with a 10 km range, a cruising speed of 36 km/h and a payload capacity of 2.4 kg. The payload is delivered to dedicated "drone ports", where the drone lands, releases the parcel and takes off again. The user collects the parcel at the drone port similar to a traditional parcel locker system.

Alphabet's Wing division has performed 750.000 deliveries within US suburbs using VTOL-style delivery drones. The advertised range is 19 km with a cruising speed of 100 km/h, and average flight times are below 4 minutes. The service is mostly used for grocery and food delivery, and the payload is delivered to the user's lawn by gently descending and detaching the package upon ground contact [12].

A common feature of these commercial services is a short service range, which restricts operation to densely populated areas such as suburbs and major cities. To support rural deliveries, the range must be extended significantly.

3 Problem Statement

The problem analysis indicates that drone delivery may reduce natural resource use and environmental impact from parcel deliveries, particularly for rural last-mile deliveries. The existing commercial implementations of drone delivery are,
5 however, limited to urban and suburban areas.

In order to address rural last-mile deliveries, the service range of delivery drones must be increased significantly, which leads to the following question:

- 10 How can the service range of delivery drones be increased to an extent that allows rural parcel deliveries?

4 Technical Analysis

The service range of a delivery drone depends on several factors. Payload mass adds weight, which increases the lift force and therefore the power required to maintain flight. Payload size changes the aerodynamic profile of the drone, so a
5 larger payload increases drag. A larger battery extends the range but also adds considerable mass, increasing the power required to maintain flight. Weather conditions affect the range through both aerodynamics and temperature. Increasing battery capacity extends the range, but at the cost of added mass and thus reduced payload capacity. Using a VTOL-style design with wings reduces
10 the energy required for flight, but increases complexity and size of the drone.

Increasing battery capacity and wing area of a delivery drone offers diminishing returns. This work proposes to reduce the airborne distance the drone must cover by utilizing existing public transport for a portion of the journey.

15 4.1 Delivery Drone Hitch-Hiking Proposal

Public transport already connects much of the Danish rural areas, although the frequency of departures is typically lower in these regions. By docking with a public transport vehicle and "hitch-hiking", the distance covered in flight can be reduced. The achievable reduction depends on several factors, including the
20 distance to the delivery site, the distance to the public transport docking point, the distance from the undocking point to the delivery site, and any transfers in between. By reducing the share of direct flight in the total journey, this concept may enable rural autonomous drone delivery by increasing the service range in multiples rather than percentages.

25 The technical analysis explores two questions: is it feasible to dock drones with public transport vehicles, and how can autonomous vision-based landing be performed?

4.2 Feasibility

In order to evaluate the feasibility of the proposed solution, some practical assumptions are evaluated.

4.2.1 Physical Space on Public Transportation Vehicles

- 5 The proposal assumes that a delivery drone carrying a payload can land on and attach itself to the roof of public transport ground vehicles.

The roof areas of buses and trains are very large compared to the footprint of a single 500-class quadrotor delivery drone, which refers to the side-length in millimeters of the square drawn by the four motor axles. This work proposes
10 that between one and six 500-class delivery drones can realistically be accommodated on a single ground vehicle depending on the size.

Many buses and trains have roof-mounted protrusions such as climate control
15 units. Placing the drone landing spaces behind these units would reduce the aerodynamic penalty of carrying drones, lower the strain on the attachment mechanism, and decrease the risk of collision with airborne objects. Where the vehicle does not have such protrusions, attachment directly to the bare roof is assumed sufficient.

20 Bus shelter roofs are much smaller than the vehicles themselves, but most Danish bus shelters are large enough to accommodate a few delivery drones. Flat landing pads can be retrofitted to shelters with curved or tilted roofs. Train stations are often much larger, but overhead electric lines can prevent docking.

25 Drones placed on top of vehicles pose a risk of hitting and dragging low-hanging tree branches or decorations. Shielding could mitigate this problem, but would add wind resistance even when no drone is attached.

4.2.2 Delivery at Destination

- 30 Existing implementations of drone delivery use different strategies for the final delivery. Some operators lower the payload while hovering, while others land, release the payload and take off again. This work proposes the latter approach in order to minimize size and weight and that the drone navigates to the delivery

4.2. Feasibility

coordinates using Global Navigation Satellite System (GNSS), visually identifies a suitable landing spot, lands, detaches the payload and returns.

4.2.3 Path Planning

5 Determining the optimal delivery route is a complex problem that involves varying wind conditions, local No-Fly Zones, public transport schedules, and the availability of docking space on a given vehicle or shelter.

10 Denmark provides a public transport routing service called "Rejseplanen". It produces transit plans from public transport timetables, given start and destination GNSS coordinates and either a departure or arrival time. The proposed plans include walking time between points and transfers. Google Maps uses Rejseplanen data for its public transport feature in Denmark and exposes the same feature through an API, which can be queried by a dedicated drone routing algorithm in any country covered by Google Maps' public transport feature.

15 There is no API that allows real-time updates of No-Fly Zones in Denmark, making this an open question that should be addressed in future work. Coordination of delivery drones would have to be incorporated in the route planner to address the limited spaces on vehicles and shelters. Additionally, 20 coordination of multiple drones in the same vicinity would be required, but both issues are left as open problems for future work.

4.2.4 Size and Weight of Packages

A group of students has previously worked on a similar problem and found that 83% of packages were below two kilograms [13]. The same group demonstrated 25 stable flight of a similar delivery drone carrying a 1.8 kg payload, so the mass of a large share of e-commerce parcels is within the capability of the proposed system. The volume constraint is less clearly defined: this project does not design a dedicated payload enclosure, and an exact size limit therefore cannot be stated. Some parcels may exceed the volume a 500-class drone can carry without com- 30 promising ground clearance or aerodynamic stability. It is nonetheless assumed that a significant share of e-commerce parcels fall within both limits.

4.3 Autonomous Drone Landing Literature Review

A literature review was conducted to acquire knowledge about autonomous vision-based landing of quadrotor drones.

5 Miranda et al. evaluated multi-sensor fusion for precision landing [14]. The hardware architecture integrated a DJI Matrice flight controller with an on-board Jetson Nano processing data from a Raspberry Pi camera and Ultra-Wideband (UWB) sensors. The system employed a 12-state Extended Kalman Filter (EKF) that continuously estimated position, orientation, and sensor biases during flight. The importance of the EKF and local target data was demon-
10 strated by systematically isolating individual sensors. When relying solely on GNSS, the drone failed to consistently land on the platform and exhibited errors of up to 1.29 m. When the EKF fused both UWB and computer-vision data, the drone achieved a 100% landing success rate and a mean error of 0.19 m.

15 Keller and Ben-Moshe addressed the challenge of landing on moving targets using only vision-based control [15]. The authors used a commercial drone equipped with a single-axis pitching gimbal and equipped the target with multiple ArUco markers. The study highlights a trade-off between an extended
20 field of view (FOV) when searching for and approaching the target using a forward-facing camera, and keeping the landing pad within the FOV using a down-facing camera. As the drone approaches the target, the algorithm maintains visual lock by adjusting the pitch angle of the camera. The authors report an average landing error of 1%, corresponding to 0.01 m linear distance error on
25 dynamic, cooperative targets.

Falanga et al. used an NVIDIA Jetson running Ubuntu and Robot Operating System (ROS) for high-level pose estimation, communicating speed set-points to a flight controller running ArduPilot [16]. A monocular camera detected an
30 array of ArUco markers placed on the moving ground vehicle. The authors employed a cascade control strategy: the high-level ROS nodes compute the relative pose and feed error values into PI controllers. The controllers generate continuous speed set-points that are passed to the low-level autopilot, which tracks the moving vehicle and lands on a platform mounted on top.

35 Kim et al. presented a vision-based geolocation method that uses the drone altitude, gimbal pan/tilt angles, and camera focal length under a flat-earth assumption.

4.3. Autonomous Drone Landing Literature Review

tion to estimate the absolute ground position of the moving target [17]. If the target exits the camera's FOV, a Kalman Filter is used to point the gimbal at the predicted location and reacquire visual lock.

- 5 Across the reviewed works, accurate vision-based autonomous landing is consistently demonstrated, including landing platforms in motion. The common ingredients are fiducial markers for robust relative-pose estimation, an estimator to bridge brief loss of visual contact, and either a gimbal or sensor fusion to manage the trade-off between a wide, forward-facing FOV for searching and
10 a down-facing FOV for landing. Reported landing errors are in the centimeter range, indicating that the underlying technique is mature enough to support landing on a restricted space even when the target is non-stationary.

However, none of the reviewed works include all of the following: Physical and
15 visual disturbances, an uncooperative target and focus on the landing duration.

5 Project Scope

The technical analysis establishes that the proposal is feasible and that vision-based landing is achievable in principle. Realizing the full proposal nonetheless requires solving four technical sub-problems:

- 5 • First, the behavior and energy use of a drone carrying a payload must be modeled.
- Second, the path-planning and routing problem must be solved while accounting for public transport timetables and their inherent uncertainty.
- Third, robust obstacle avoidance must be developed to prevent collisions.
- 10 • Fourth, rapid and robust docking between the delivery drone and public transport vehicles must be developed.

A fifth-semester mechatronics engineering group at Aalborg University developed a scalable model of a delivery drone and compared it to physical tests. The model and the physical drone showed similar results with an added mass of 1.8 kg, and the physical drone is capable of take-off and maneuvering. A step-input test of 1 m in the z-direction with a load of 1.8 kg suggests that the developed model is sufficiently stable, with less than 2.1% overshoot and a settling time below 3.25 seconds [13].

20 Huang et al. constructed a multi-modal network modeling a drone that uses public transportation to expand its delivery range [18]. The routing problem is formulated as a shortest-path problem bounded by the drone's battery capacity and solved using a Dijkstra-based algorithm. The study models the inherent uncertainty of public transport: bus arrival and departure times are represented as probability distributions, rather than static timetables. By computing upper- and lower-bound transit times, the routing algorithm provides a mathematical guarantee on transfer reliability.

30 A 2025 survey by Merei et al. reviews over one hundred works on Unmanned Aerial Vehicle (UAV) obstacle detection and avoidance across a wide range of sensor modalities and algorithmic families, including machine learning, map-based methods, geometric approaches, potential fields and classical sense-and-avoid techniques [19]. The survey reports that the majority of recent works

address both static and dynamic obstacles in three-dimensional space, and that a substantial share have been validated in real flight rather than only in simulation. The remaining open challenges concern energy-aware planning, no-fly-zone handling, and multi-UAV coordination, rather than basic obstacle avoidance itself.

The cited sources sufficiently cover the first three sub-problems. Therefore, this project addresses the remaining sub-problem: rapid and robust docking of a quadrotor drone to a non-cooperating ground vehicle.

The requirements for the docking maneuver are specified in the next chapter.

6 Requirements Specification

The project scope identifies rapid and robust docking as the open sub-problem and the technical problem is formulated:

- 5 "To what extent can rapid and robust autonomous drone landing be achieved on a non-cooperating ground target using vision-based control?"

In order to land on a non-cooperating target, the target must first be located. A forward-facing camera is advantageous for this stage because it provides a
10 substantially larger field of view. Once the target is found, the drone must continuously update its landing trajectory to account for unpredictable target movement. One way to remove the trade-off between a wide search FOV and a centered landing FOV is to use a single-axis gimbal, as in [15]. Alternative
15 solutions include using an estimator such as the EKF of [14], or switching between cameras with different orientations. Figure 6.1 illustrates the discrepancy between FOV needed when searching for or intercepting the target versus performing a final vision-based landing.

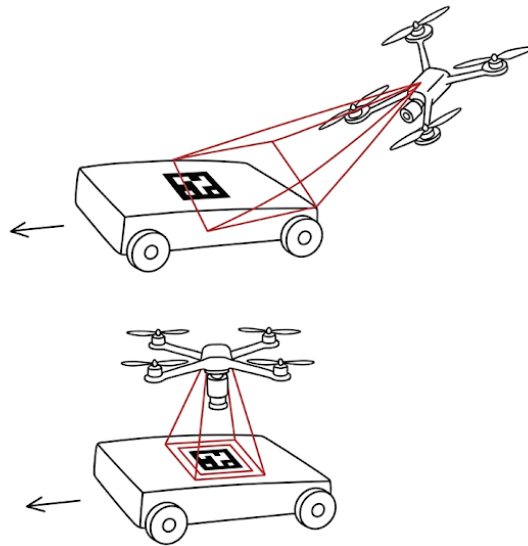


Figure 6.1: Illustration of camera FOV needed in intercepting and landing phases.

In order to validate the developed solution in physical flight, I reached out to

6.1. Landing Duration

the student association "AAU Space Robotics" which is already working with autonomous flight of a suitable Holybro x500 v2 quadrotor frame. They agreed to participate in the project, providing me access to their autopilot software repository and let me perform physical flight tests with the hardware. The drone
5 was equipped with an Intel RealSense camera mounted to point forward and 45 degrees down, which serves as the static-camera baseline in this work. I added a single-axis gimbal by mounting the camera on an axle driven by a small servo motor as Figure 6.2 illustrates:

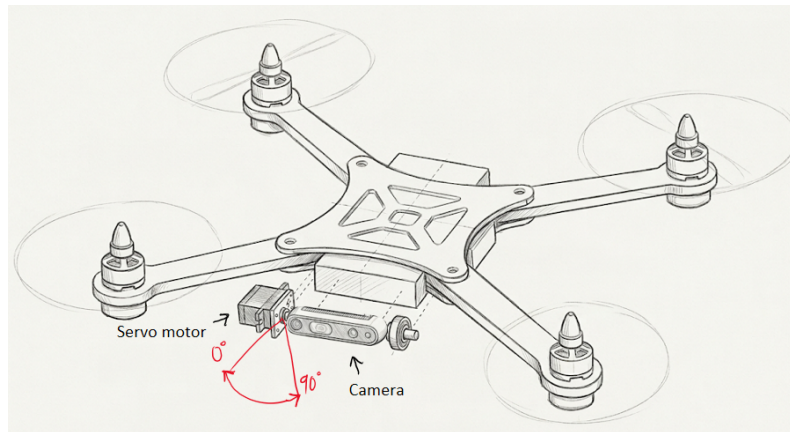


Figure 6.2: Illustration of camera module and single axis gimbal placement.

Before synthesizing and building the proposed single-axis camera tilt, a set of requirements is specified. A simulation framework is then developed based on
10 these requirements to quantify the performance of the tilting camera against the static-camera baseline in chapter 8.

The performance of the proposed solution is assessed against requirements in
15 three categories: duration, accuracy and robustness.

6.1 Landing Duration

In order to dock with a non-cooperating public transportation vehicle, the full landing sequence must be completed within the vehicle's existing stop duration. This duration varies substantially with vehicle type, with trains often stopping
20 for several minutes and Bus Rapid Transit (BRT) systems often stopping for only a few seconds. In an analysis of bus dwell times, Fricker reviews several earlier dwell-time models and notes that, although the models differ, they all include a

6.2. Landing Accuracy

constant term and additional terms that depend on the number of boarding and alighting passengers [20]. For rural public transport it is reasonable to assume that this number is low, resulting in stop durations of approximately ten seconds.

- 5 The ten-second duration of rural bus stops therefore sets the upper bound for landing duration.

6.2 Landing Accuracy

In order to attach securely to the roof under the high air-resistance forces encountered at vehicle cruising speeds, a physical attachment structure is required.

- 10 The distance between opposing propeller tips is 0.66 m for the drone used in this project. This distance is also the side length of the square drone footprint.

The accuracy requirement is proposed as 10% of the footprint side length, meaning the final linear error between the center of the landing pad and the center of the drone must be below 6.6 cm.

- 15 In addition to linear distance error, the heading of the drone must be aligned with the pad within a certain threshold to allow an attachment mechanism to engage. An angle error of at most five degrees is therefore proposed as the upper bound.

6.3 Landing Robustness

The landing sequence must be robust against real-life disturbances. The vision-based approach leaves the drone vulnerable to changes in lighting and to reflections. The non-cooperating nature of the target means the algorithm cannot reliably predict the target's behavior or assume that it is stationary. Physical forces such as wind gusts further affect the drone.

- 25 The solution is therefore required to be robust against drastic periodic reductions or increases in illumination, strong wind gusts, and the target moving unpredictably during the landing. The magnitude of the disturbances in this work is selected as 200 ms white-outs every 10 s for visual disturbances. The target's forward velocity component is modeled as $v_x(t) = 0.1 \sin(0.2t) + 0.3 \text{ m/s}$, and the lateral component is modeled as a sum of three sinusoids with irrational fre-

6.3. Landing Robustness

quency ratios, $v_{y,\text{raw}}(t) = 0.125 \sin(\frac{\pi}{7}t) + 0.075 \sin(\frac{\sqrt{2}}{3}t) + 0.05 \sin(\frac{e}{11}t)$ m/s. This raw lateral component is dynamically capped to $\pm 25\%$ of the instantaneous forward velocity to mitigate turns steeper than $\pm 14^\circ$. This is done to reflect the long wheel base and thus large turning circle of a bus. Wind disturbances are
5 not simulated, but are inherently evaluated during physical tests.

7 Vision-Based Landing Algorithm

The detector script uses the OpenCV library to calculate the ground error in drone body-frame x and y coordinates, which are published to a separate ROS 2 topic. The mission script uses a proportional controller to minimize the body frame errors in x and y through pitch and roll commands respectively, while a separate proportional controller maintains the desired altitude through the thrust command. A state machine switches between control modes based on the progression of the mission.

To distinguish between the drone's body-fixed coordinate frame (x, y, z) and the global North-East-Down (NED) coordinate frame, the convention illustrated in Figure 7.1 is adopted. In the body-fixed frame, the x -axis points in the drone's forward direction, the y -axis points to the right, and the z -axis points downwards. According to the right-hand rule:

- A positive rotation around the x -axis defines *roll*, denoted by ϕ .
- A positive rotation around the y -axis defines *pitch*, denoted by θ .
- A positive rotation around the z -axis defines *yaw*, denoted by ψ .

The global NED frame is established under a flat-Earth assumption. This approximation is valid for the short-range flight operations analyzed in this work, where the curvature of the Earth can be safely neglected.

7.1. Marker Detection

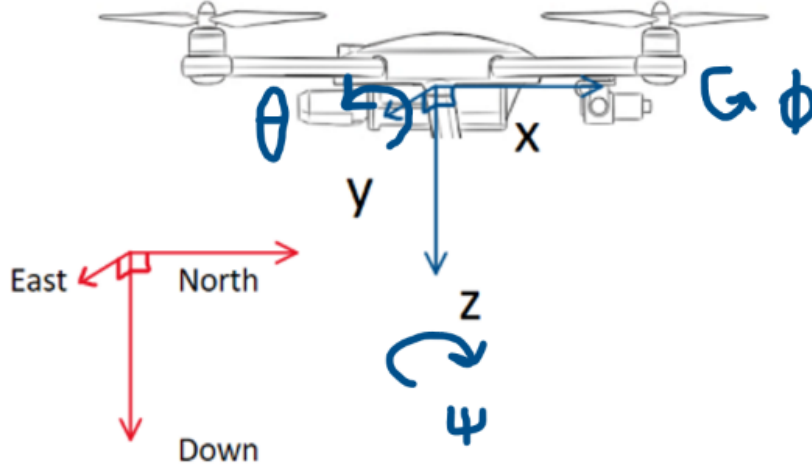


Figure 7.1: Drone coordinate axes (local body frame) relative to the global NED coordinate system.

7.1 Marker Detection

The landing target is identified by an ArUco marker: a square fiducial marker consisting of a black border surrounding a binary grid that encodes an identifier [21]. ArUco markers are detected reliably under perspective distortion and varying scale, and a single marker yields the image positions of all four of its corners, which is sufficient to determine both the centering error and the heading of the target. ArUco markers are widely used for UAV precision landing [22, 23].

A single 4×4 marker with identifier 0 is used. Detection is performed with the ArUco module of the OpenCV library, whose `detectMarkers` routine returns the four corner pixel coordinates of every marker found in an image. Two detector parameters are adjusted from their defaults: the minimum marker perimeter rate is lowered to 0.01 so that small and distant markers are still considered, and the number of pixels sampled per cell during perspective removal is increased to 4 to improve decoding at a slant. The detector subscribes to the camera image, which is published by synthetic camera in simulation and the camera module in real flight and output is either the corner positions of the locked marker or a no-detection result.

7.2 Ground-Error Projection

The detector reports the marker corners in image pixels, but the controller requires the horizontal position error in meters. This section describes the projection from pixel error to horizontal ground error; the geometry is shown in Figure 7.2.

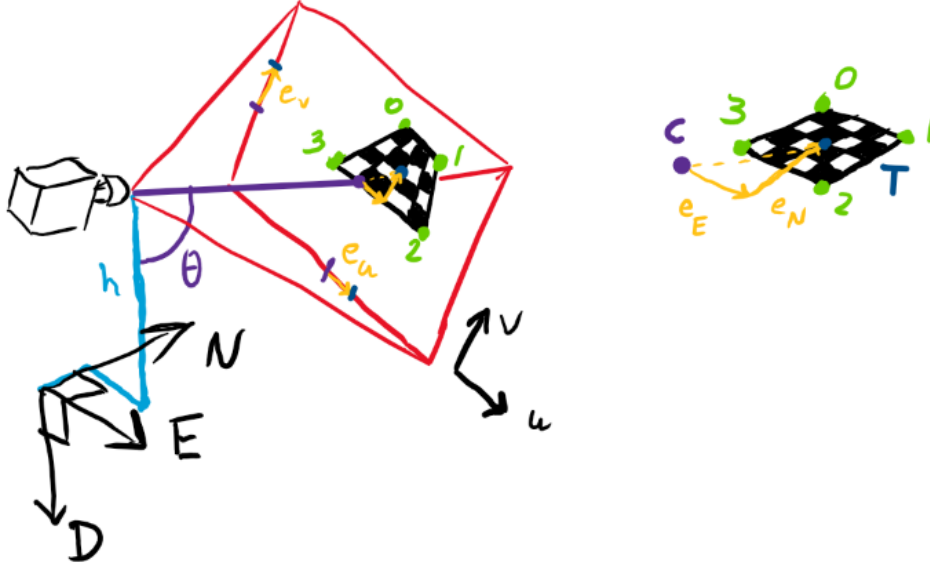


Figure 7.2: Back-projection of the marker-center pixel onto the ground plane. The marker appears as a parallelogram in the image (u, v) frame and as a square in the global NED frame. Each pixel defines a ray from the camera center: the marker-center ray meets the ground at T and the optical axis meets the ground at the optical axis intersection point C . The ground error (e_N, e_E) is the displacement from C to T in North and East components. The ground is assumed flat at $z = 0$, the tilt angle $\theta_{\text{effective}}$ is the combined drone and gimbal pitch, and (e_N, e_E) are expressed in meters.

Camera model. The camera is treated as an ideal pinhole. The image has a width W and height H in pixels and a horizontal field of view α . The focal length expressed in pixels is

$$f = \frac{W/2}{\tan(\alpha/2)}. \quad (7.1)$$

- ¹⁰ The principal point, where the optical axis intersects the image, is taken at the image center $(c_u, c_v) = (W/2, H/2)$.

7.2. Ground-Error Projection

Marker center. The four detected corners are averaged to obtain the marker-center pixel (u, v) . The center is used for the centering error, while the individual corners are used for the heading estimate described below.

- 5 **Back-projection.** A pixel is projected onto the ground in three steps. First, the pixel defines a ray through the camera center. In the camera frame, in which the third axis is the optical axis, this ray has the direction:

$$\mathbf{d}_c = \left[-\frac{v - c_v}{f}, \quad \frac{u - c_u}{f}, \quad 1 \right]^\top. \quad (7.2)$$

Second, the ray is rotated into the global NED frame:

$$\mathbf{d} = \mathbf{R} \mathbf{d}_c, \quad \mathbf{R} = \mathbf{R}_{\text{drone}} \mathbf{R}_{\text{cam}}, \quad (7.3)$$

- 10 where $\mathbf{R}_{\text{drone}}$ represents the rotation matrix from the drone body frame to the global NED frame, and $\mathbf{R}_{\text{cam}}$ is the rotation matrix from the camera frame to the body frame. The camera frame is aligned with the body frame except for a pitch rotation about the body y -axis by the effective camera pitch angle $\theta_{\text{effective}}$:

$$\mathbf{R}_{\text{cam}} = \begin{bmatrix} \cos \theta_{\text{effective}} & 0 & \sin \theta_{\text{effective}} \\ 0 & 1 & 0 \\ -\sin \theta_{\text{effective}} & 0 & \cos \theta_{\text{effective}} \end{bmatrix}. \quad (7.4)$$

- 15 Here, $\theta_{\text{effective}} = 0$ corresponds to the camera pointing straight down, meaning the image plane axes u and v map directly to the drone's body axes y and x respectively, making $\mathbf{R}_{\text{cam}} = \mathbf{I}$. A value of $\theta_{\text{effective}} = \pi/2$ corresponds to the camera pointing straight forward, parallel to the drone's body x -axis.

- 20 Third, the ray is intersected with the ground plane. With the camera at altitude h and writing d_D for the downward component of $\mathbf{d}$, the ray reaches the ground after a scaling:

$$k = \frac{h}{d_D}, \quad \mathbf{g} = k \mathbf{d}, \quad (7.5)$$

so that $\mathbf{g}$ is the ground point beneath the pixel, expressed relative to the camera.

- 25 **Ground error.** The back-projection is applied to two pixels: the marker center, giving the ground point T , and the principal point, whose ray is the optical axis and which gives the optical axis intersection with ground C (see Figure 7.2). The

7.3. Mission Algorithms

ground error is their difference,

$$\mathbf{e} = \mathbf{T} - \mathbf{C}, \quad (7.6)$$

and its horizontal components (e_N, e_E) are passed to the controller. Defining the error relative to the optical axis intersection rather than the drone position means it vanishes precisely when the marker center lies on the optical axis.

5

To ensure that the drone's Center of Mass (COM)—rather than the camera itself—stabilizes directly above the target, the visual error is compensated for the physical camera mounting offsets. The camera's body-frame offsets $(x_{\text{off}}, y_{\text{off}})$ relative to the COM are rotated by the target's relative heading and added to the visual error. This shifts the visual tracking setpoint, driving the drone's COM directly over the target center.

10

Heading. The marker also provides the relative heading needed to align the drone with the pad. The marker's top edge, between corners 0 and 1, points in the forward direction of the target. The midpoints of the front and back edges are back-projected to the ground, and the vector between them gives the marker heading in the world frame. Subtracting the drone yaw yields the relative yaw error, which is published alongside the horizontal linear error components.

15

20 7.3 Mission Algorithms

The complete delivery mission is summarized in algorithm 1 as a high-level sequence of phases, deliberately omitting timers and thresholds.

7.3. Mission Algorithms

Algorithm 1: Full delivery mission

Input: delivery point coordinates; public-transport route; drone GNSS location

```
/* Outbound:  base  $\rightarrow$  delivery area                                */
Take off from base
Flight to departure location
Land on pad and wait for the scheduled vehicle
Recognize vehicle using computer vision
Hover above anticipated stop position
Dock onto the vehicle roof
Ride to the stop nearest the delivery point
Undock from vehicle and take off
Fly to the delivery coordinates
Identify a suitable spot and land
Release the parcel
Take off

/* Return:  delivery area  $\rightarrow$  base                                    */
Fly to the bus stop nearest the delivery point
Land on the shelter pad and wait for the scheduled vehicle
Recognize vehicle using computer vision
Hover above anticipated stop position
Dock onto the vehicle roof
Ride to the bus stop nearest the base
Undock and take off
Fly to base and land
```

The full delivery mission requires multiple autonomous landings, where some are static pad landings and some are non-cooperating vehicle landings.

- ⁵ The control logic for landing on a static target is given in Appendix A as a simplification of the code implementation. For the vehicle docking and undocking, a more comprehensive six-phase plan is needed. Figure 7.3 illustrates the entire docking scenario:

7.3. Mission Algorithms

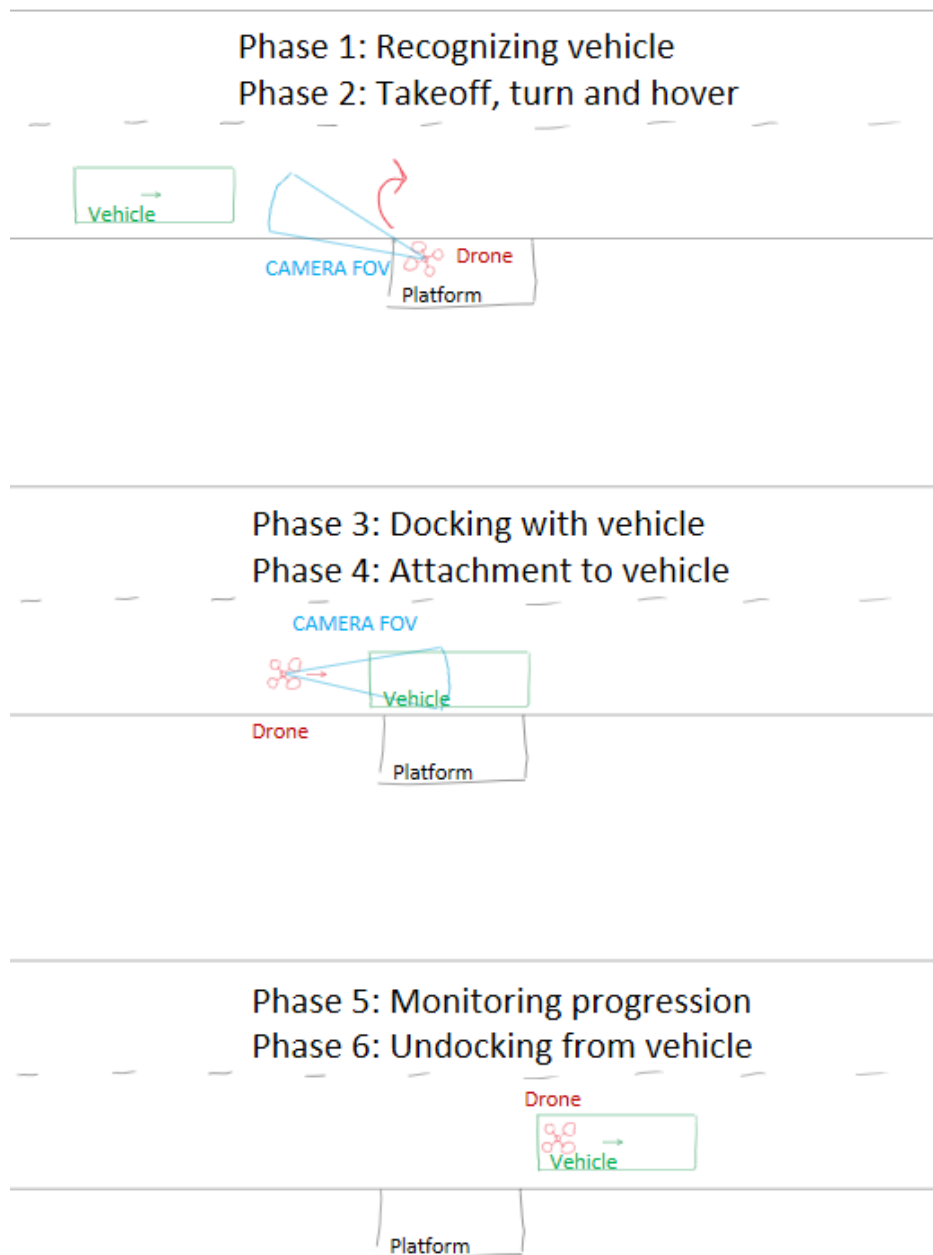


Figure 7.3: Visual overview of the six phases of the dynamic bus landing sequence: (1) Computer vision vehicle recognition, (2) Hovering above the anticipated stop position, (3) Docking using the vision-based landing algorithm, (4) Secure mechanical attachment, (5) Monitoring of journey progression, and (6) detachment at the planned stop nearest the final delivery.

This project focuses on the autonomous docking in Phase 3, so the remaining five phases are left as open problems to be solved in future work.

7.3. Mission Algorithms

The algorithm for docking with the non-cooperating vehicle is given in algorithm 2.

Algorithm 2: Landing on dynamic target

Input: ground error (e_x, e_y) and lock flag from the ArUco detector;
relative yaw ψ ; altitude h ; camera mode $\in \{\text{gimbal}, \text{static}\}$

/ State machine */*

```

switch state do
  case idle do
    | after a start-up delay, send arm;  $\rightarrow$  arming
  case arming do
    | once armed, command take-off to takeoff_alt;  $\rightarrow$  take-off
  case take-off do
    | when  $h \approx \text{takeoff\_alt}$ , enable autopilot hover with roll, pitch, yaw
    | and thrust commands from mission script;  $\rightarrow$  search
  case search do
    | Hold position at takeoff_alt and wait for the moving target to
    | enter the field of view
    | when the marker is locked and the forward component of ground
    | error is positive,  $\rightarrow$  stabilize-high
  case stabilize-high do
    | minimize ground error and relative yaw while holding altitude at
    | takeoff_alt
    | after a dwell of  $T_{\text{high}}$ ,  $\rightarrow$  descend
  case descend do
    | minimize ground error and relative yaw while descending
    | when  $h \leq h_{\text{low}}$ ,  $\rightarrow$  stabilize-low
  case stabilize-low do
    | minimize ground error and relative yaw while holding altitude at
    | descend_alt
    | after a dwell of  $T_{\text{low}}$ ,  $\rightarrow$  terminal-landing
  case terminal-landing do
    | if camera mode is gimbal then sweep the gimbal from  $45^\circ$  to
    | straight down while minimizing ground error and relative yaw.
    | After the sweep, thrust is reduced and the drone lands
    | else descend blind at target extrapolation of true target state upon
    | switch to terminal-landing (simulation of static camera baseline)
  case not visually locked do
    | if visual lock returns, resume the stored state; otherwise hover for
    |  $T_{\text{hold}}$ , then land

```

7.3. Mission Algorithms

Two parts of the implementation are shown directly: Listing 7.1 gives the ground error control law, and Listing 7.2 gives the gimbal sweep that defines the proposed slant landing.

Listing 7.1: Visual-servo control law from `vision_landing_mission.py`

```
#Rotate ground error from world frame to drone body frame
5 err_fwd  =  err_x * cos(yaw) + err_y * sin(yaw)
  err_side = -err_x * sin(yaw) + err_y * cos(yaw)

#Proportional control law, saturated to the command range
  cmd_pitch = clamp(kp * err_fwd, -1.0, 1.0)
10 cmd_roll  = clamp(kp * err_side, -1.0, 1.0)
```

Listing 7.2: Forward gimbal sweep mapping from `vision_landing_mission.py`

```
# Sweep logical value from 0 to -1.0
  dt_sweep  = time.monotonic() - terminal_start
  frac      = min(dt_sweep / slant_sweep_time, 1.0)
  target_val = 0.0 - frac * 1.0

15 # Map logical value to raw servo command
  servo_val = down_cmd + (45_cmd - down_cmd) * (target_val + 1.0)
  servo_cmd = max(-1.0, min(1.0, servo_val))
```

The complete drone software repository is available on GitHub at <https://github.com/thomaselberg/drone-software/tree/Development>, and the visualized ROS 2 computation graph showing the network of nodes and topics can be seen in Appendix C.

8 Simulation

The simulation extends the PX4 software-in-the-loop (SITL) and Gazebo environment set up by AAU Space Robotics with a synthetic vision add-on developed in this project. The flight dynamics are provided by PX4 SITL and Gazebo environment using the Holybro X500 quadrotor airframe with a downward distance sensor. PX4 and the ROS 2 nodes exchange data through the Micro eX-
5 extremely Resource Constrained Environments - Data Distribution Service (Micro XRCE-DDS) bridge, and ROS 2 uses the Gazebo simulation clock.

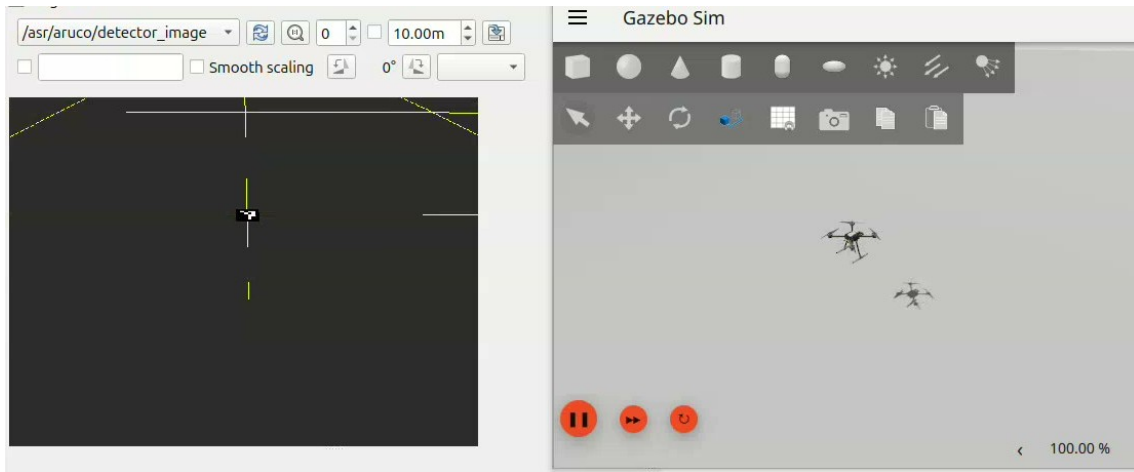


Figure 8.1: This screenshot shows the detector_image next to the Gazebo graphical user interface. Here, the drone is approaching the static target, which is not yet detected. Both marker and ground level grid lines are drawn according to drone attitude, three-dimensional position in the world frame and the gimbal pitch angle. The Gazebo window shows the drone from a third-person view with the shadow indicating the ground plane.

10 The synthetic camera Python script draws an accurate warped depiction of the ground and the ArUco marker according to the effective camera orientation, which combines the drone attitude and the gimbal angle. The synthetic image is published to a ROS 2 topic that a separate ArUco detector algorithm consumes. Figure 8.2 shows the detector recognizing the marker drawn by the synthetic
15 camera node alongside the same code recognizing a real marker captured by the camera module during flight test.

In simulation, tilt and thrust commands are passed to the autopilot like in phys-



Figure 8.2: Successful detection of the 4×4 ArUco ID 0 marker in simulation and in physical flight. Because the synthetic camera and the real camera publish to the same ROS 2 topic and are geometrically identical, the detector code is unchanged between the two environments.

ical flight, but instead of spinning the motors, the flight controller passes the commands on to Gazebo for computation. The autopilot’s controller gains and motion limits are configured in `thyra_params_sim.yaml` by AAU Space Robotics, and the corresponding file for the physical flight hardware is `thyra_params.yaml`.

- 5 The two configurations share identical inner-loop and position-controller gains in order to ensure direct portability of the control logic, while setup calibrations, tilt angle limits and certain parameters diverge to reduce risk of destroying the single available drone.
- 10 The complete parameter files, together with the mission thresholds and timers used in both simulation and physical flight, are available in the GitHub repository. The synthetic camera’s resolution (1280×720 px), focal length ($f_x \approx 910.7$ px, $f_y \approx 910.8$ px), and principal point are matched exactly to the intrinsic calibration parameters of the physical Intel RealSense camera. The physical
- 15 flight tests were conducted at a downsampled resolution of 848×480 px, trading spatial detection accuracy for a higher and more stable frame rate. Running the camera at the full 1280×720 px resolution caused thermal throttling on the Raspberry Pi companion computer in successive runs.
- 20 To add disturbances, the synthetic camera is set to output a completely white image for 0.2 s every 10 s, simulating reflections or strong light sources that intermittently blind the vision system. Wind disturbance is not simulated as the environment used in this work does not support this without developing a dedicated airframe model. Robustness to wind is instead assessed in the hardware
- 25 flight test, where the drone is exposed to real wind conditions.

9 Hardware UAV flight test

The physical flight tests were conducted on a private farm in a remote area to comply with regulation of minimum 150 meter horizontal distance to residential, commercial, industrial, or recreational areas. Additionally, the European
5 Union Aviation Safety Agency (EASA) Remote Pilot Certificate of Competency for the Open subcategories A1 and A3 was achieved prior to the flight tests.

The mean wind was negligible, but sporadic gusts of up to 7 m/s from varying directions were reported by the Danish Meteorological Institute (DMI) for the
10 region, which is consistent with the conditions perceived during flight.

The Holybro x500 v2 quadrotor drone lent from AAU Space Robotics was used for the flight tests. Three views of this drone can be seen in Figure 9.1:

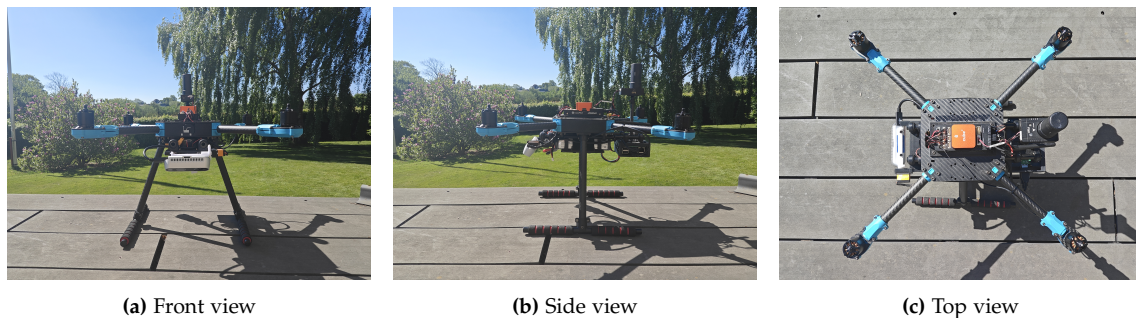


Figure 9.1: Orthogonal views of the used Holybro x500 v2 quadrotor UAV.

15 Power and signal connections are illustrated in a combined hardware block diagram in Figure 9.2.

The required hardware for the flight test include a Ground Control Station, remote control, RF link, RTK base station, WiFi router and ArUco marker. These
20 are photographed and labeled in Figure 9.3.

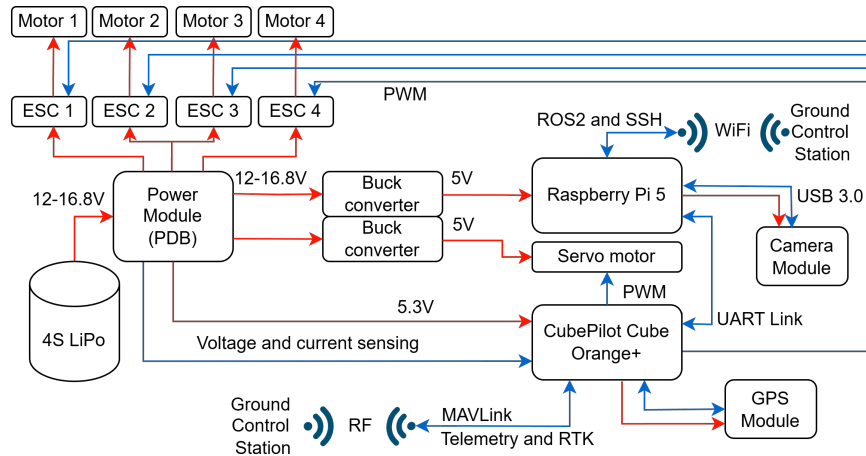


Figure 9.2: Hardware block diagram showing power connections in red and signal telemetry paths in blue.

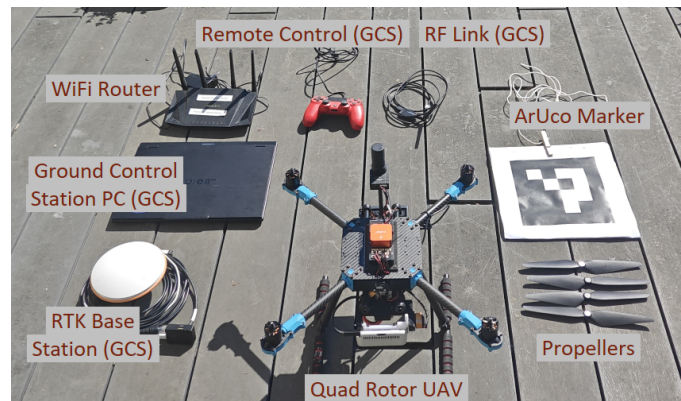


Figure 9.3: Physical experimental setup with labels

10 Results

The results are presented for both static and dynamic target landings from simulations and physical flight tests. A summary of the average landing performance metrics across all scenarios is provided in Table 10.1.

Table 10.1: Summary of simulation and physical flight test landing performance.

Target	Environment	Camera Mode	Lin. Error (cm)	Rot. Error	Duration (s)
Static	Simulation	Static Baseline	13.1	0.06°	6.0
	Simulation	Gimbal	2.9	0.04°	9.0
	Physical Test	Gimbal	22.8	1.1°	17.6
Dynamic	Simulation	Static Baseline	28.6	9.5°	8.8 ^a
	Simulation	Gimbal	20.1	4.4°	10.3 ^a
	Physical Test	Gimbal	62.1 ^b	1.8° ^c	10.6

^a Net tracking durations after subtracting the initial search state duration.

^b Large error caused by altitude hold issue.

^c Rotation error excluding two outliers.

5 The average accuracy of simulated static target landing was 2.9 cm for the gimbal camera versus 13.1 cm for the static camera baseline. The rotation errors are 0.04 and 0.06 degrees, respectively. The average duration from first lock to final landing was 9 seconds for the gimbal camera runs and 6 seconds for the static camera baseline. The discrepancy happens during the terminal-landing state
 10 due to the gimbal sweep duration being longer than the interception to the true state from the static baseline algorithm. Figure 10.1 shows the ten overlaid runs of the gimbal camera static target simulations. The corresponding results for the static camera baseline are presented in Figure B.1.

15 The altitude plot on Figure 10.1 shows the division into two distinct descending phases: the initial descent to low altitude hold and the terminal landing. The linear error is decreasing in two phases: the first being the descent phase, where the drone is maintaining point C directly on top of point T (see Figure 7.2) while reducing altitude. The second phase is the gimbal sweep, where the servo tilt
 20 is forcing point C away from point T, leading to a forward error, which is minimized by the controller. When the sweep is finished, the drone is hovering above

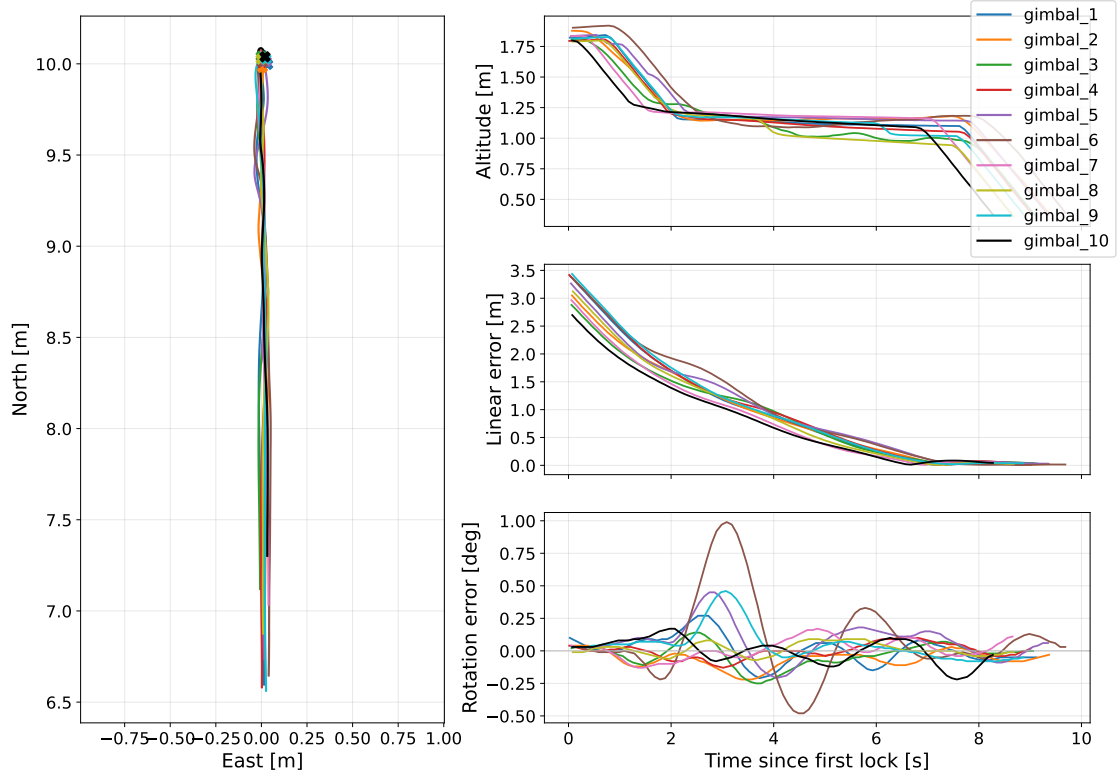


Figure 10.1: Ten runs of gimbal camera static landings divided into four plots: Horizontal drone coordinate traces and timeseries plots of altitude, linear error and rotation error

the target with no ground error, which the plot clearly shows for all runs around the six-to-seven second point.

Figure 10.2 shows the distribution of final linear and rotation errors as well as the engagement duration from first lock to touchdown. It also shows errors and commands normalized to the North distance covered. The corresponding results for the static camera baseline are presented in Figure B.2.

The errors and commands normalized to North distance covered show that the error decreases linearly with North distance covered, even with a much larger pitch command in the descent phase than during the gimbal sweep. Decreasing the sweep duration would increase the pitch commands during the gimbal sweep and accelerate the reduction of the linear error, so it would show as a piecewise linear function. In the static camera baseline, a near-linear error reduction is similarly observed, but with a significant increase in pitch once the true state is handed at the start of the terminal landing phase.

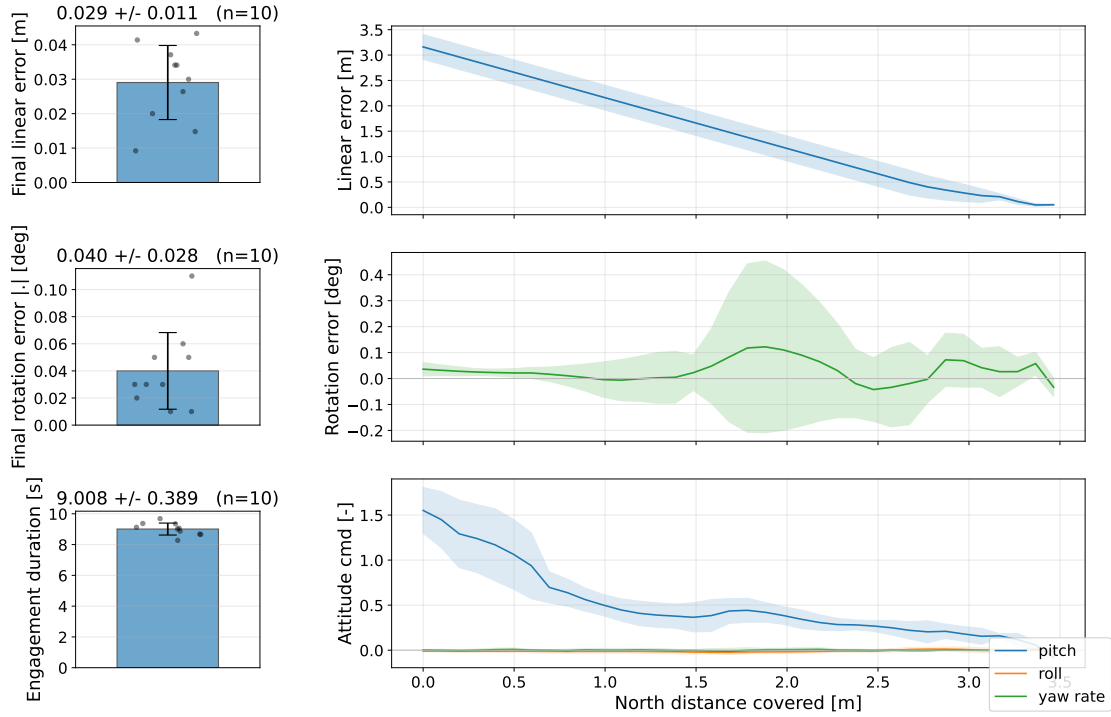


Figure 10.2: Ten runs of gimbal camera static landings represented as an average line with one standard deviation bands around it.

The dynamic target simulations deviated more between the gimbal camera and the static camera baseline. The dotted line traces on Figure 10.3 show the continuous but somewhat unpredictable target paths. The simulation randomization of target path is not seeded, meaning the runs are not reproducible. The altitude plot shows multiple runs losing target lock during the descent state, shown as horizontal lines and a following continuation of the descent. The overlaid runs also reveal the forward error threshold required to initiate the high-altitude-stabilize state. The linear error is seen steadily increasing until the T point overtakes the C point. From that point, the descent is initiated and the two-phase error reduction at descent and the following gimbal sweep is observed from the plot. The simulations reveal that the gimbal camera algorithm struggles to keep up with the dynamic target after the sweep. This is caused in the very last moments before touchdown, where the camera is so close to the marker that small movements make the marker leave the frame and thus preventing detection and subsequent error calculation. The same phenomenon occurs with last-moment rotation errors, where the drone touches down in the

moment the target turns, leading to a large rotation error. Two examples of such are gimbal_4 and gimbal_5 runs.

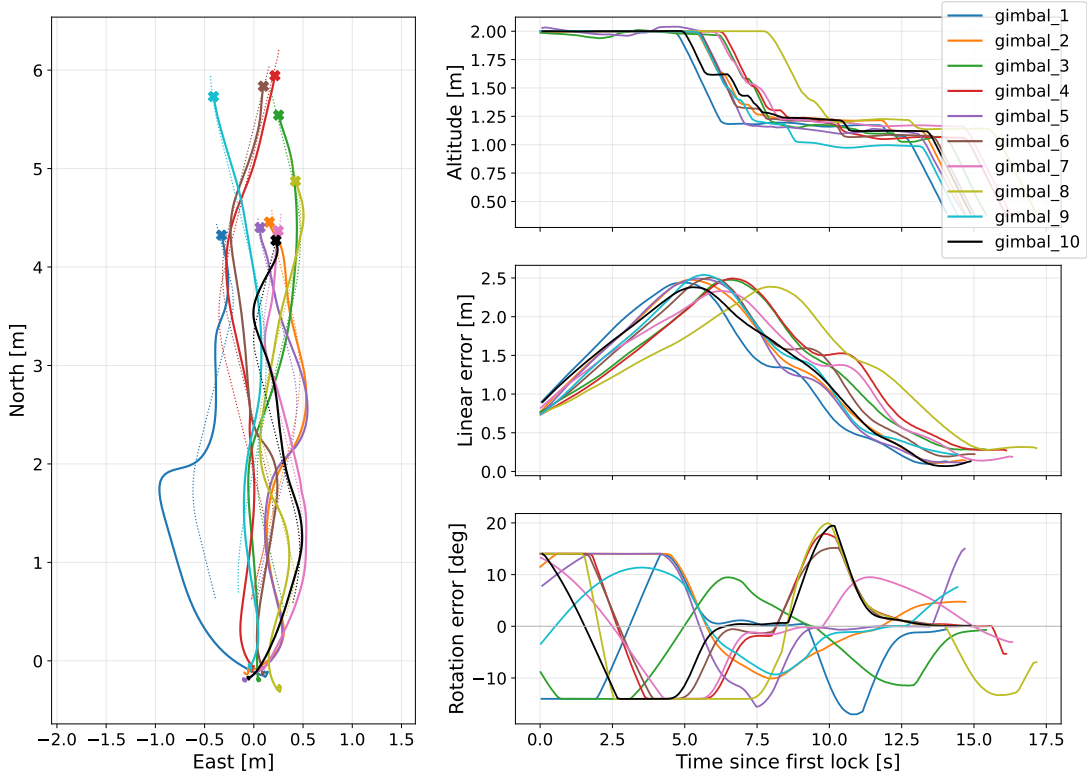


Figure 10.3: Ten runs of gimbal camera dynamic landings overlaid in four plots: Horizontal global coordinate traces of target (dotted lines) and drone (full lines) with an x indicating the final landing position. The time series plots show altitude, linear error and rotation error from first detection to touchdown

The static camera baseline (Figure B.3) shows the linear error decreasing until the terminal landing state, at which point the blind drone diverges from the unpredictable target as it extrapolates the target state linearly. Similarly, the rotation error diverges from the moment the static camera baseline enters the terminal-landing state. The blind terminal-landing state of the static camera baseline results in an average linear error of 28.6 cm compared with 20.1 cm for the gimbal camera. The final rotation error of the static camera baseline is also much larger at 9.5° compared to 4.4° for the gimbal camera. Engagement duration was notably shorter at 13.6 seconds for the static camera baseline compared to 15.3 seconds for the gimbal camera. However, the average waiting time in the search state of 5.0 seconds for the gimbal camera and 4.8 seconds for the static

camera should be subtracted from that, giving a final average engagement duration of 8.8 seconds for the static camera and 10.3 seconds for the gimbal camera. The average run with standard deviation bands is shown in Figure 10.4 and the corresponding static camera results are shown in Figure B.4.

5

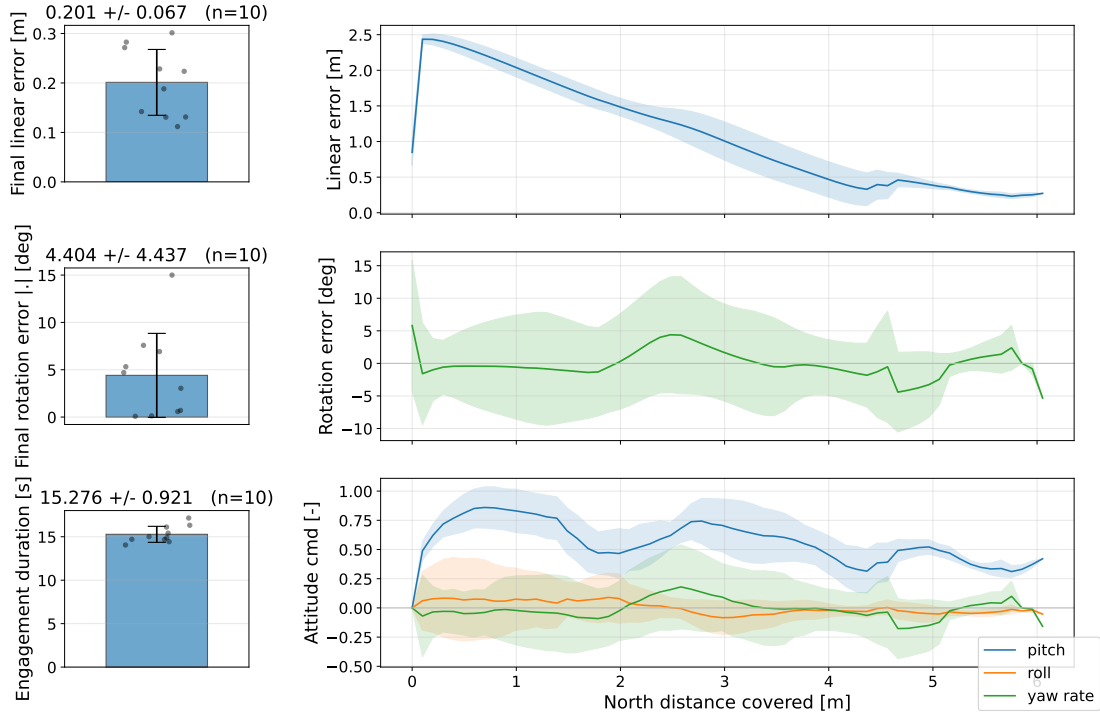


Figure 10.4: Ten runs of gimbal camera dynamic landings visualized as an average line with one standard deviation bands around it

The dynamic target average commands diverge much more in the lateral and yaw directions for both the gimbal camera and the static camera. The pitch commands maintain an elevated level until touchdown, which is expected because the target movement is increasing the forward component of the error. The gimbal camera results show the average linear error converging to around 20 cm and the static camera converges similarly before the linear extrapolation causes a divergence in the last moments of the terminal-landing state.

10

The results from static target physical flight tests are shown in Figure 10.5 with ten overlayed runs.

15

The average final linear and rotation error was 23 cm and 1.1° , respectively. The

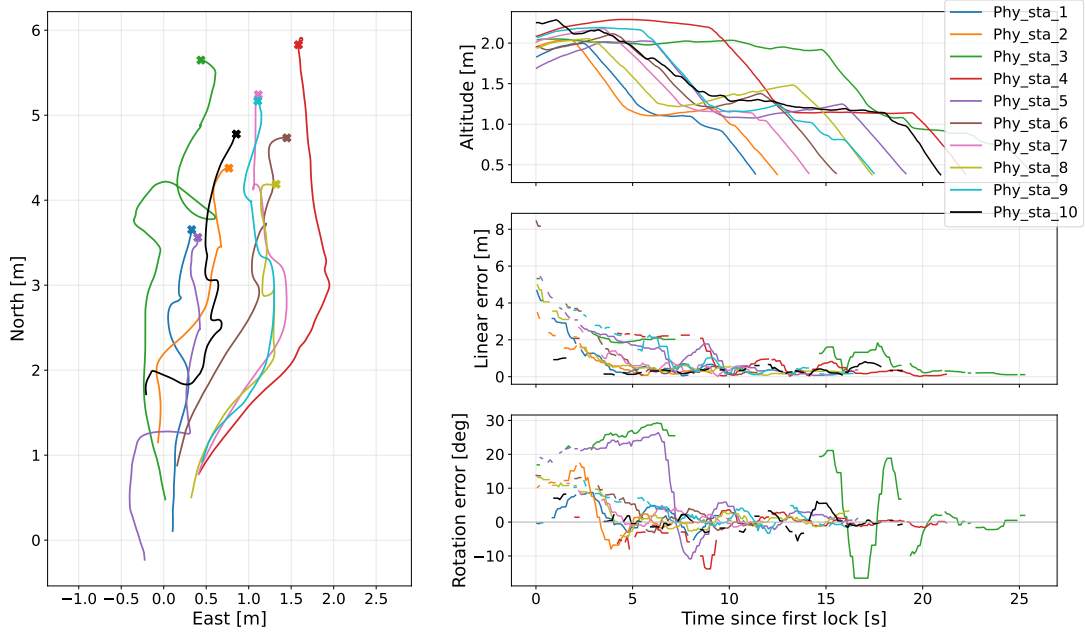


Figure 10.5: Ten runs of physical flight static landings overlayed in four plots: Horizontal global coordinate traces with an x indicating the final landing position. The time series plots show altitude, linear error and rotation error from first detection to touchdown

landing durations were significantly longer than the simulations due to lower search gain and a slower descent profile to mitigate risks.

The dynamic target tests were plagued with the autopilot failing to hold altitude in every single run. The algorithm is identical regardless of what the target does from the descent phase, so the root of the problem is outside the companion computer scope. The ten overlayed runs are presented in Figure 10.6.

The dynamic target landing altitude plot shows the descent profile as a piecewise linear function, where the middle section is supposed to be horizontal, meaning the altitude is held constant by the proportional altitude controller at 1.5 m. The faulty descent profile makes the drone touch down while trailing the target during the gimbal sweep at an average linear error of 62 cm. The logs reveal it was almost entirely a forward error in the drone body frame, suggesting the error was entirely due to the mentioned altitude hold issue. The average rotation error was 5.1° , but removing two outliers brings the error down to an average of 1.8° . Those two outliers were likewise caused by premature landing due to the altitude hold problem.

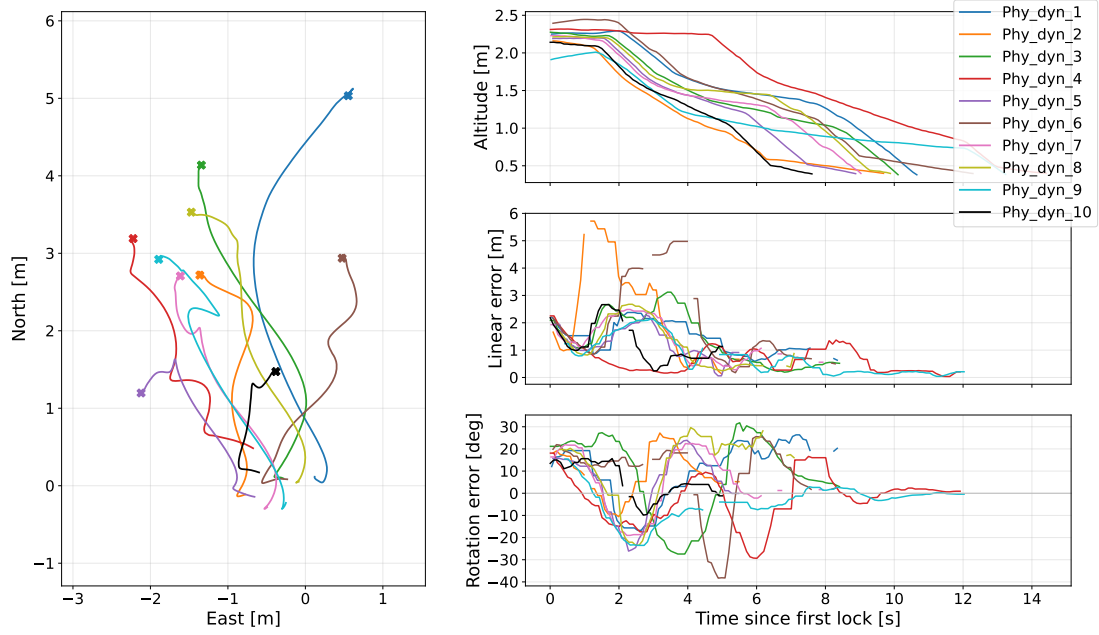


Figure 10.6: Ten runs of physical flight dynamic landings overlaid in four plots: Horizontal global coordinate traces with an x indicating the final landing position. The time series plots show altitude, linear error and rotation error from first detection to touchdown

Figure 10.7 shows a single dynamic landing performed in the physical flight test. The target was moved in a sinusoidal pattern manually at ground level and from beneath the drone and moving forward, somewhat matching the dynamic target in simulations. The time series plots show altitude and commands with the current state represented by the colored bands. The commands plotted show the pitch command being saturated at 1.0 for more than two seconds. This is due to the tilt limits being set very conservatively to minimize risk in physical flight. Easing safety restrictions will decrease the duration, but not affect accuracy significantly.

To evaluate the performance of the proposed single-axis gimbal landing system, the experimental and simulation results are compared directly against the design criteria defined in chapter 6. This comparison is summarized in Table 10.2.

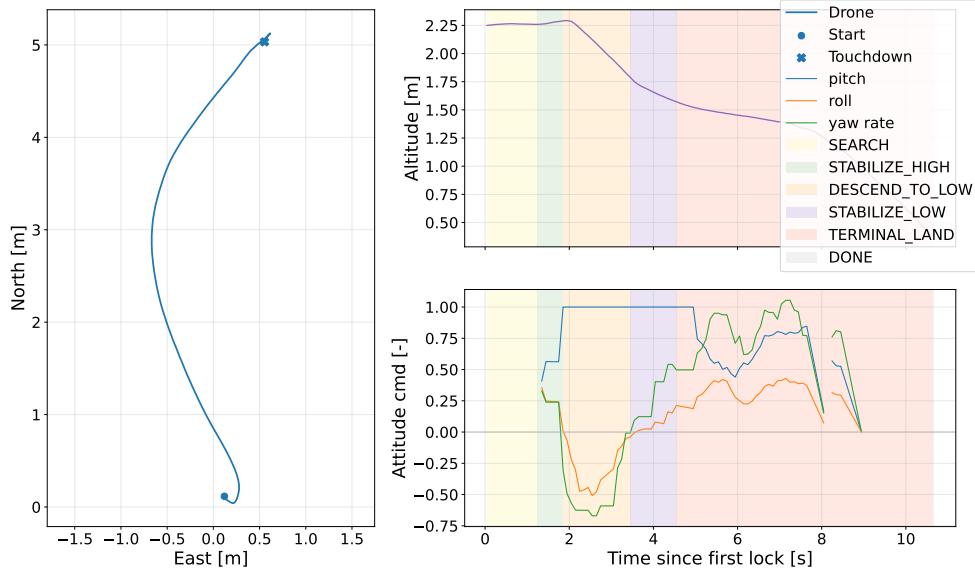


Figure 10.7: Single physical flight dynamic target landing with horizontal global coordinate traces on the left and time series altitude and commands on the right.

Table 10.2: Requirements verification of gimbal camera results.

Scenario	Linear Error	Rotation Error	Duration
Requirement	< 6.6 cm	< 5.0°	< 10.0 s
Simulation – Static	2.9 cm	0.04°	9.0 s
Simulation – Dynamic	20.1 cm	4.4°	10.3 s
Physical – Static	22.8 cm	1.1°	17.6 s
Physical – Dynamic	62.1 cm	1.8°	10.6 s

The performance of the proposed gimbal camera solution satisfies the requirements for static landings in simulation, while physical flight was less accurate and took more time to perform the landing. The accuracy deviation in real flight is caused by the drone drifting with the wind during the last 50 cm of descending, where no attitude commands are sent. The duration of physical flight landing can be reduced significantly by easing safety limits for tilt commands and descent velocity.

Dynamic Landings in simulation were approximately at the upper bound of rotation error and duration requirements, while the accuracy was 20.1 cm compared to the 6.6 cm requirement. The physical flight test only satisfied the ro-

tation error requirement. However, the performance was negatively affected by altitude hold problems which reduces duration and increases the linear error to a disproportionate degree. Without the altitude hold problem, I expect the results to match the static landing performance, with a linear error still outside
5 the required range.

To fulfill the accuracy requirement and in extension provide a solution to the technical problem, the following proposals are made: Either the accuracy requirement may be eased by increasing the tolerance of the mechanical attachment on the vehicle, allowing for a 30% side length accuracy as opposed to 10%.
10 Otherwise, a nested marker solution might allow the down-facing camera to continue detection when the large marker exceeds the camera FOV and thus allow attitude commands during the very last centimeters of descent.

11 Discussion

The contribution of this work is a single-axis gimbal vision-based landing algorithm for a non-cooperating ground target, evaluated against a static-camera baseline. Given the deliberately narrow technical scope established in chapter 5, the focus of this discussion is restricted to: how the methodological choices constrain the interpretation of the results presented in chapter 10, and how those results relate to comparable vision-based landing works in the literature.

11.1 Methodological Limitations

Disturbances in simulation. The PX4-SITL and Gazebo environment does not apply aerodynamic wind forces and the synthetic camera disturbance was chosen as a periodic white-out every ten seconds. The dynamic target motion consists of sinusoidal forward and lateral components, restricted to a maximum combined speed of approximately 0.41 m/s and to turning angles of $\pm 14^\circ$. The dynamic landing simulation results therefore reflect this restricted disturbance and motion model. The real flight tests provide partial coverage of these gaps, although the real-world lighting and wind conditions were not quantified due to a lack of suitable measurement equipment.

Conservative real-flight tuning. A single drone was available to the project, which made aggressive tuning in physical flight risky. The descent profile and the search proportional gain were therefore reduced relative to the simulation configuration. The 17.7 s engagement duration of the static real flight tests, compared with 9 s in simulation, reflects this conservatism rather than a limit of the algorithm in physical flight.

Static-camera baseline only simulated. The comparison between the proposed single-axis gimbal and the static-camera baseline is available only in simulation. The static-camera baseline was not flight-tested, as that would have required developing an estimator to support the blind, linear-extrapolation landing. In simulation, the true target state was passed to the algorithm in place of such an estimator, and the static-camera baseline can therefore not be used in physical flight as implemented here. The gimbal-versus-baseline statements in chapter 10

11.2. Comparison with Similar Published Works

therefore rest on the simulation results, while the physical flight serves as a validation of the algorithm under conditions not present in the simulation, such as limited computing power, sensor noise, imperfect wireless communication links and wind disturbances.

5

Physical flight failure mode. The altitude-hold failure observed in the dynamic target real flight tests, where the magnitude grew as battery state of charge decreased, is independent of the vision-based landing algorithm. Without this failure mode, the observed lateral and yaw tracking suggests that the dynamic landing results would approach the static landing results, possibly with a small added forward error in the final descent.

10

Marker-out-of-frame limitation. Loss of marker visibility in the final centimeters of descent is a geometric limitation common to single-marker detector solutions at the scale used here. Solutions to this problem are available in the literature, such as nesting a smaller marker inside the main one [23] or extrapolating the relative target state through a Kalman filter [17], but neither was implemented within the scope of this work.

15

20 11.2 Comparison with Similar Published Works

The 2.9 cm simulated and 23 cm real-flight average linear errors achieved on a static target are consistent with the centimeter-range results reported for ArUco-based visual landing in section 4.3. The dynamic-target results are positioned against works addressing landing on moving platforms with similar sensing strategies. Keller and Ben-Moshe [15] also use a single-axis tilting camera. Their implementation differs significantly as they used a commercial sub-250 g DJI drone with their algorithm utilizing a software development kit on top of DJI's closed autopilot stack. They achieved 1 cm accuracy on moving but cooperative targets. The accuracy improvement over a static-camera baseline observed in this work is consistent with the argument that sustaining the visual lock from intercepting to landing improves robustness against unpredictable target movement. For landing on static or cooperative dynamic targets, a well-tuned estimator with a static camera may suffice, reducing the mechanical part count at the cost of additional software complexity. The single-axis gimbal algorithm developed in this work is purely geometric and conceptually elegant.

25

30

35

12 Conclusion

This work set out to address the problem statement

5 “How can the service range of delivery drones be increased to an extent that allows rural parcel deliveries?”

and the technical research question derived from it:

10 “To what extent can rapid and robust autonomous drone landing be achieved on a non-cooperating ground target using vision-based control?”

The drone delivery literature review in subsection 2.3.1 identified the short service range of current delivery drones as the principal barrier to rural deployment. Chapter 4 proposed to extend that range by docking with existing public transport vehicles, and chapter 5 identified rapid and robust docking on a non-cooperating ground target as the open technical sub-problem to be solved.

20 A vision-based landing algorithm was developed around a single 4×4 ArUco marker, an OpenCV-based detector, a back-projection from pixel error to horizontal ground error in meters, a proportional control law, and a state machine handling search, descent and terminal landing. A single-axis camera gimbal was proposed as the central contribution, allowing the camera to maintain target lock from interception until the terminal vision-anchored landing. The algorithm was evaluated against a static-camera baseline through a synthetic-camera simulation built on PX4 software-in-the-loop and Gazebo, and validated in physical flight tests on a Holybro X500 v2 quadrotor.

Performance was measured against the three requirement categories specified in chapter 6.

30 **Duration.** The average engagement duration from first marker lock to touch-down remained at or below the ten-second upper bound derived from rural bus dwell times in all scenarios except the static target physical flight test, where a conservative descent profile and a reduced search gain extended the average duration to 17.7s. The duration requirement is therefore met in simulation and achievable in physical flight by relaxing the conservative tuning used in the static

flight tests.

Accuracy. The linear error requirement of 6.6 cm is met by the gimbal camera on a static target in simulation, with an average error of 2.9 cm, but is not met in any of the remaining scenarios. The dynamic simulation result of 20.1 cm is dominated by marker loss in the final centimeters of descent, and the physical flight result of 62 cm on a dynamic target is dominated by the altitude-hold failure documented in chapter 10. The angular error requirement of 5° is met in all simulation scenarios for the gimbal camera and in the physical flight tests. A nested ArUco solution would allow the detector to continue producing an error estimate when the outer marker exceeds the camera frame, which would reduce the dynamic landing error toward the magnitude observed in the static scenarios for both simulations and physical flight.

Robustness. The algorithm tolerated the simulated 200 ms white-outs every 10 s without failure, and the physical flight tests demonstrated tolerance to gusts of up to approximately 7 m/s and changing outdoor lighting.

The comparison between the proposed single-axis gimbal and the static-camera baseline shows that the gimbal solution trades a small increase in engagement duration for substantially better accuracy, particularly on dynamic targets, where the static baseline must extrapolate the target state during a blind terminal-landing phase. In simulation, the gimbal reduces the average linear error from 13.1 cm to 2.9 cm on static targets and from 28.6 cm to 20.1 cm on dynamic targets, while the angular error on the dynamic target drops from 9.5° to 4.4° .

Regarding the technical research question, the work concludes that rapid and robust vision-based landing on a non-cooperating ground target is feasible within the ten-second duration bound set by rural bus dwell times. Accuracy within the proposed mechanical tolerance is achieved in simulation on a static target and approached in the other scenarios.

Regarding the problem statement, this project contributes with a method for extending the service range of delivery drones by utilizing public transport to limit flight distance. Specifically, the developed gimbal camera algorithm allows rapid and robust docking with a non-cooperative ground target. With the increased service range, delivery drones can play a supplementary role in future rural last-mile delivery focused on low-volume single-parcel routes, complementing rather than replacing consolidated van delivery.

13 Future Work

The scope of this project was deliberately narrowed to the autonomous docking sub-problem identified in chapter 5, leaving the open problems summarized below.

5

Routing and path planning. Real-time integration of No-Fly Zones, coordination of multiple delivery drones and multi-modal routing under probabilistic timetables all remain open.

10 **Remaining docking phases.** Five of the six phases illustrated in Figure 7.3 are out of scope for this work and remain open. In particular, computer-vision vehicle recognition, secure mechanical attachment to the vehicle roof and journey monitoring during transit are each substantial sub-problems.

15 **Vision and disturbance robustness.** A nested-marker or Kalman-filtered solution to the close-range marker-out-of-frame failure mode, robustness to a broader range of lighting and weather conditions, and addition of aerodynamic wind forces to the PX4-SITL airframe model are all relevant extensions of the present work.

20

Hardware tuning and reliability. The conservative descent profile and reduced search gain used in the real flight tests reflect the practical constraint of limited time and spare parts, meaning less risk was tolerated. Closing the gap to the simulation configuration with more aggressive gains and solving the altitude-hold failure observed in the dynamic real flight tests, would bring the real flight performance closer to the simulation results.

25

Regulatory and operational assessment. A BVLOS operating certificate and the interaction between the proposed docking maneuver and existing public transport safety regulations remain to be addressed. A broader operational and economic assessment of whether the proposed system is viable at scale falls outside the technical focus of this report and is therefore also left for future work.

30

Bibliography

- [1] PostNord. *E-commerce in the Nordics Spring_2025*. <https://www.postnord.com/insights/reports/e-commerce-in-the-nordics-2025-spring/>. Last reviewed: 09-04-2026. 2025.
- 5 [2] PostNord in association with HUI Research. *E-barometern_AR_2025_ENG*. https://www.postnord.com/siteassets/PDF/E-barometern_AR_2025_ENG.pdf. Last reviewed: 25-03-2026. 2026.
- [3] Effigy Consulting. *THE NORDICS CEP MARKET EVOLUTION*. <https://www.effigy-consulting.com/the-nordics-parcel-market-2023-preliminary-results/>. Last reviewed: 09-04-2026. 2024.
- 10 [4] United Nations. *Targets and Indicators*. https://sdgs.un.org/goals/goal12#targets_and_indicators. Last reviewed: 25-03-2026. 2025.
- [5] Abhijeet Kumar, V.R. Prybutok, and Vikas Sangana. "Environmental Implications of Drone-Based Delivery Systems: A Structured Literature Review". In: *Clean Technologies* 7 (Mar. 2025), p. 24. DOI: 10.3390/cleantechnol7010024.
- 15 [6] Global Market Insights. *Last Mile Delivery Vehicle Market Size Share 2025 - 2034*. 2025. URL: <https://www.gminsights.com/industry-analysis/last-mile-delivery-vehicle-market>.
- [7] Alessandro Giordano, Paul Fischbeck, and H. Scott Matthews. "Environmental and economic comparison of diesel and battery electric delivery vans to inform city logistics fleet replacement strategies". In: *Transportation Research Part D: Transport and Environment* 64 (2018). The contribution of electric vehicles to environmental challenges in transport. WCTRS conference in summer, pp. 216–229. ISSN: 1361-9209. DOI: <https://doi.org/10.1016/j.trd.2017.10.003>. URL: <https://www.sciencedirect.com/science/article/pii/S1361920916308525>.
- 20 [8] Małgorzata Markowska and Jakub Marcinkowski. "Rural E-Customers' Preferences for Last Mile Delivery: Evidence from Poland". In: *Energies* 15 (Nov. 2022). DOI: 10.3390/en15228351.
- 25 [9] Jiyeon Park, Solhee Kim, and Kyo Suh. "A Comparative Analysis of the Environmental Benefits of Drone-Based Delivery Services in Urban and Rural Areas". In: *Sustainability* 10 (Mar. 2018), p. 888. DOI: 10.3390/su10030888.
- 30

- [10] Vipul Garg et al. "Drones in last-mile delivery: A systematic review on Efficiency, Accessibility, and Sustainability". In: *Transportation Research Part D: Transport and Environment* 123 (2023), p. 103831. ISSN: 1361-9209. DOI: <https://doi.org/10.1016/j.trd.2023.103831>. URL: <https://www.sciencedirect.com/science/article/pii/S1361920923002286>.
5
- [11] Zipline. *zipline-surpasses-2-million-deliveries*. <https://www.zipline.com/newsroom/news/announcements/zipline-surpasses-2-million-deliveries-raises-more-than-600m-to-power-next-phase-of-growth-and-expands-operations-to-houston-and-phoenix>. Last reviewed: 25-03-2026. 2026.
10
- [12] Wing. *solutions*. <https://wing.com/solutions>. Last reviewed: 10-04-2026. 2026.
- [13] Jakob Ege Gru Madsen et al. *Modelling and Analysis of a Quadcopter for Last-Mile Package Delivery: Analysis of a Mechatronic System*. https://kdbk-aub.primo.exlibrisgroup.com/permalink/45KBDK_AUB/3b147k/alma9922350598505762. Last reviewed: 25-03-2026. 2025.
15
- [14] Victor R. F. Miranda et al. "Autonomous Navigation System for a Delivery Drone". In: *CoRR* abs/2106.08878 (2021). arXiv: 2106.08878. URL: <https://arxiv.org/abs/2106.08878>.
- [15] Assaf Keller and Boaz Ben-Moshe. "A Robust and Accurate Landing Methodology for Drones on Moving Targets". In: *Drones* 6.4 (2022). ISSN: 2504-446X. DOI: 10.3390/drones6040098. URL: <https://www.mdpi.com/2504-446X/6/4/98>.
20
- [16] Davide Falanga et al. "Vision-based autonomous quadrotor landing on a moving platform". In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, pp. 200–207. DOI: 10.1109/SSRR.2017.8088164.
25
- [17] Jaemin Kim et al. "Vision-Based Geolocation of Moving Ground Targets Using Kalman Filtering with a Gimbal Camera on Board a UAV". In: *Aerospace* 12.12 (2025). ISSN: 2226-4310. DOI: 10.3390/aerospace12121065. URL: <https://www.mdpi.com/2226-4310/12/12/1065>.
30
- [18] Hailong Huang, Andrey Savkin, and Chao Huang. "Drone Routing in a Time-Dependent Network: Toward Low-Cost and Large-Range Parcel Delivery". In: *IEEE Transactions on Industrial Informatics* (July 2020), pp. 1–1. DOI: 10.1109/TII.2020.3012162.
35

- [19] Ahmad Merei et al. "A Survey on Obstacle Detection and Avoidance Methods for UAVs". In: *Drones* 9.3 (2025). ISSN: 2504-446X. DOI: 10.3390/drones9030203. URL: <https://www.mdpi.com/2504-446X/9/3/203>.
- 5 [20] Jon D. Fricker. *Bus dwell time analysis using on-board video*. 2010. URL: https://engineering.purdue.edu/~ce560/HW/Bus_Dwell_Time_jdf.pdf.
- [21] S. Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion". In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292. ISSN: 0031-3203. DOI: <https://doi.org/10.1016/j.patcog.2014.01.005>. URL: <https://www.sciencedirect.com/science/article/pii/S0031320314000235>.
- 10 [22] Karisma Trinanda Putra et al. "Computer Vision-Driven Precision Landing in Quadcopter Drones Using ArUco Markers". In: *2024 4th International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*. 2024, pp. 236–240. DOI: 10.1109/ICE3IS62977.2024.10775518.
- 15 [23] Artur Khazetdinov et al. "Embedded ArUco: a novel approach for high precision UAV landing". In: *2021 International Siberian Conference on Control and Communications (SIBCON)*. 2021, pp. 1–6. DOI: 10.1109/SIBCON50419.2021.9438855.

A Static Landing Algorithm

Algorithm 3: Landing on static target

Input: ground error (e_x, e_y) and lock flag from the ArUco detector;
relative yaw ψ ; altitude h ; camera mode $\in \{\text{gimbal}, \text{static}\}$

```
/* State machine */
switch state do
  case idle do
    | after a start-up delay, send arm;  $\rightarrow$  arming
  case arming do
    | once armed, command take-off to takeoff_alt;  $\rightarrow$  take-off
  case take-off do
    | when  $h \approx \text{takeoff\_alt}$ , enable autopilot hover with roll, pitch, yaw
    | and thrust commands from mission script;  $\rightarrow$  search
  case search do
    | align yaw with direction to the known landing pad coordinates
    | and pitch forward
    | when the marker is locked and the ground error is below  $\epsilon_{\text{search}}$ ,
    |  $\rightarrow$  descend
  case descend do
    | minimize ground error and relative yaw while descending
    | when  $h \leq h_{\text{low}}$ ,  $\rightarrow$  stabilize-low
  case stabilize-low do
    | minimize ground error and relative yaw while holding altitude at
    | descend_alt
    | after a dwell of  $T_{\text{low}}$ ,  $\rightarrow$  terminal-landing
  case terminal-landing do
    | if camera mode is gimbal then sweep the gimbal from  $45^\circ$  to
    | straight down while minimizing ground error and relative yaw.
    | After the sweep, thrust is reduced and the drone lands
    | else descend blind at target extrapolation of true target state upon
    | switch to terminal-landing (simulation of static camera baseline)
  case not visually locked do
    | if visual lock returns, resume the stored state; otherwise hover for
    |  $T_{\text{hold}}$ , then land
```

B Static Camera Baseline Simulation Results and Graphs

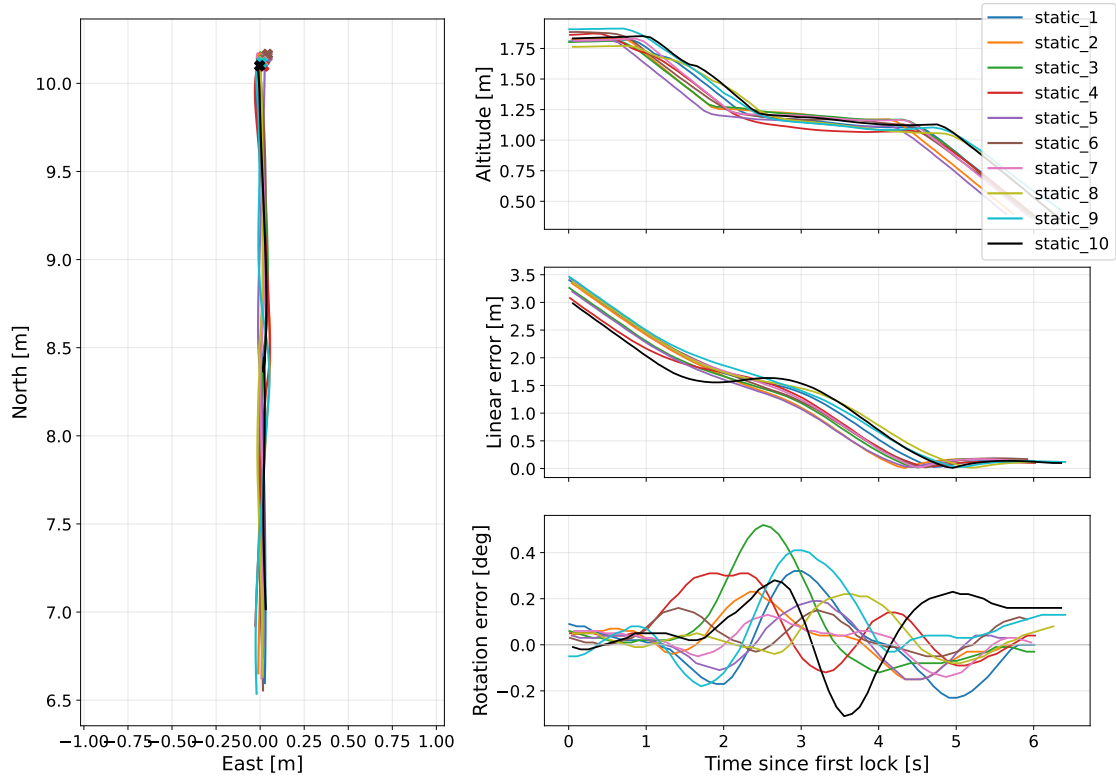


Figure B.1: Ten runs of static camera static landings divided into four plots: Horizontal drone coordinate traces and timeseries plots of altitude, linear error and rotation error

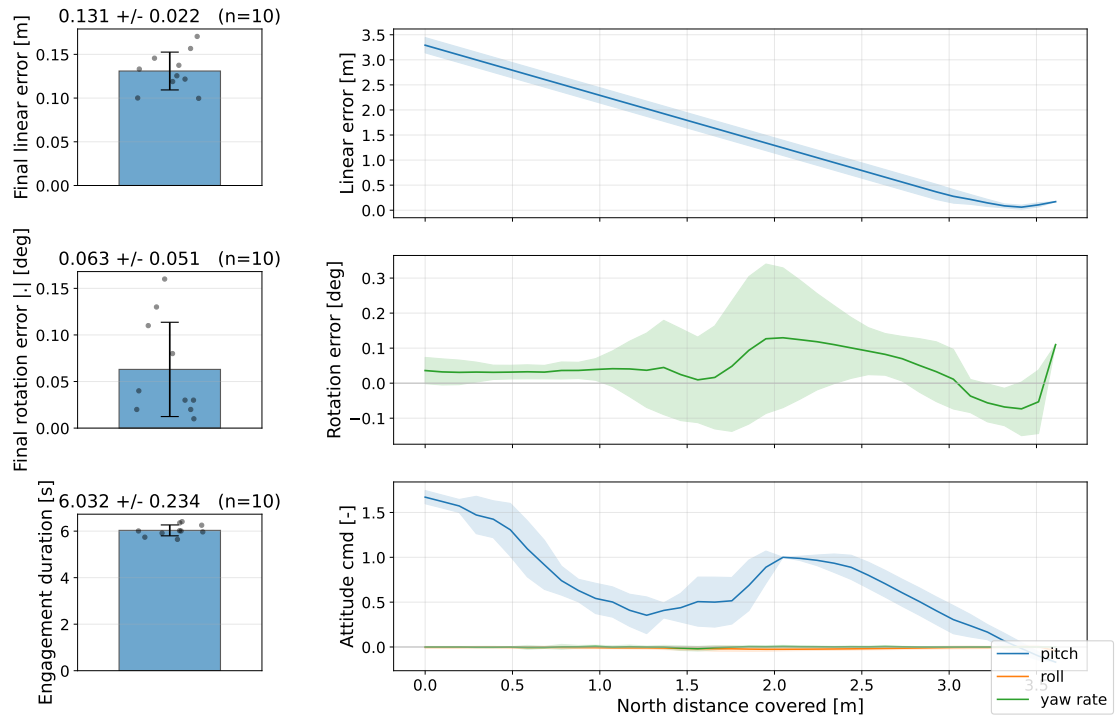


Figure B.2: Ten runs of static camera static landings represented as an average line with one standard deviation bands around it.

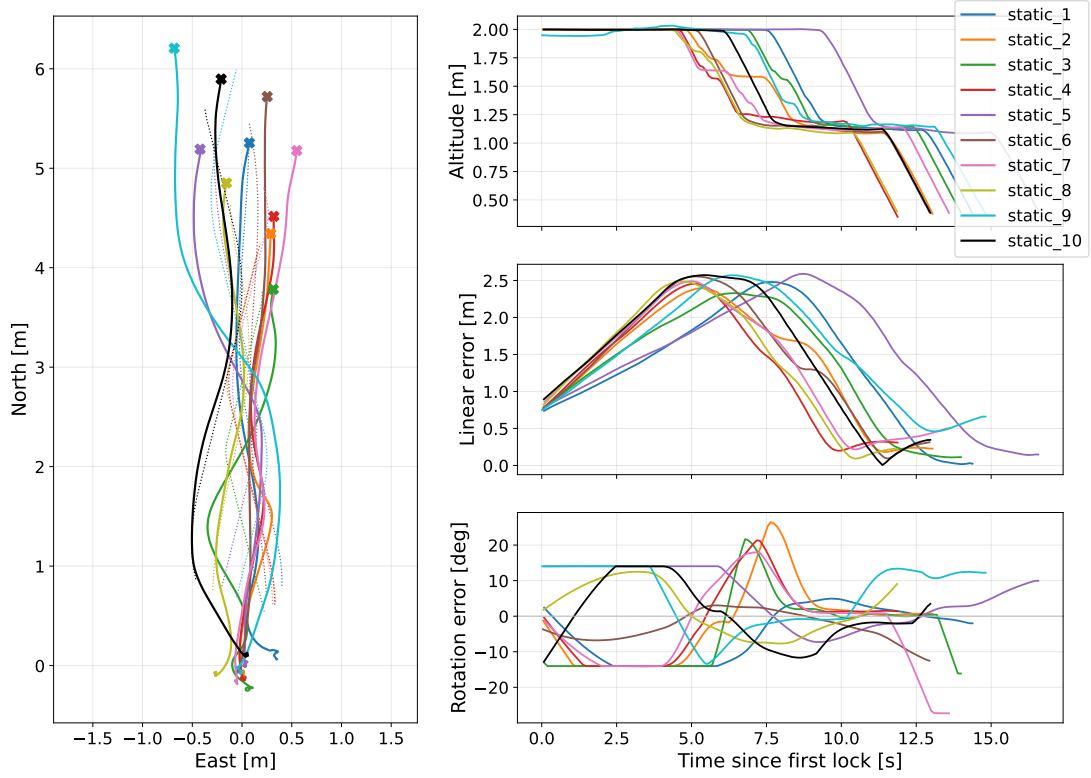


Figure B.3: Ten runs of static camera dynamic landings overlayed in four plots: Horizontal global coordinate traces of target (dotted lines) and drone (full lines) with an x indicating the final landing position. The time series plots show altitude, linear error and rotation error from first detection to touchdown

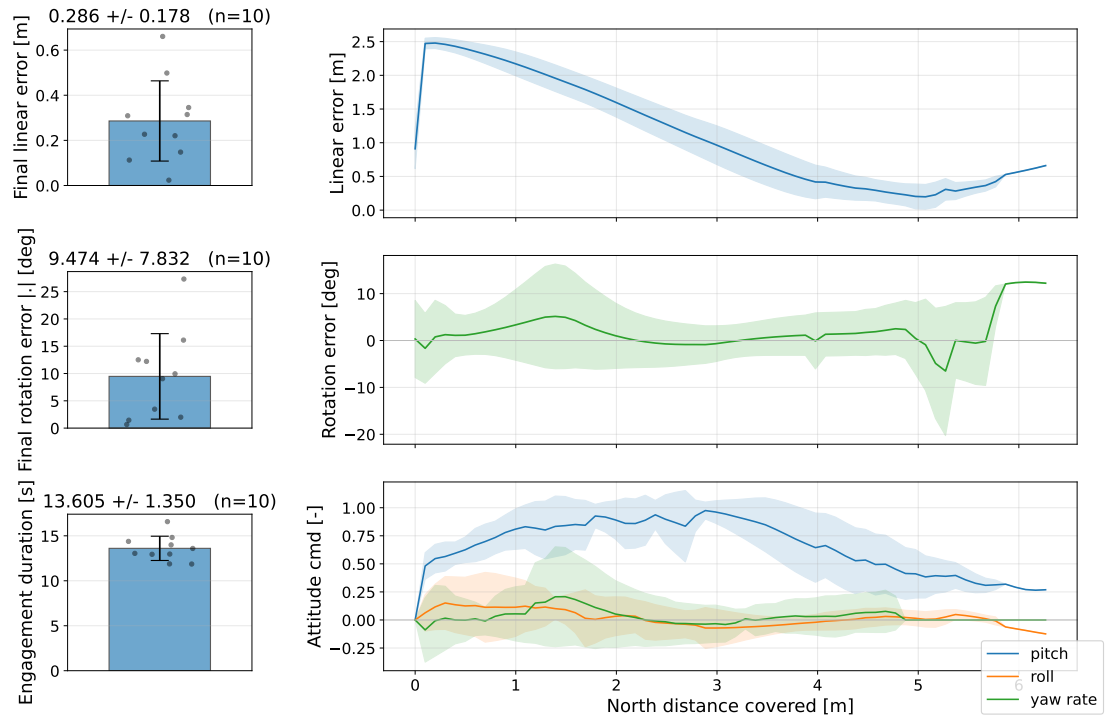


Figure B.4: Ten runs of static camera dynamic landings visualized as an average line with one standard deviation bands around it

C ROS 2 Computation Graph Visualized using rqt_graph

The active ROS 2 node architecture and communication topics during the visual landing mission are shown in Figure C.1.

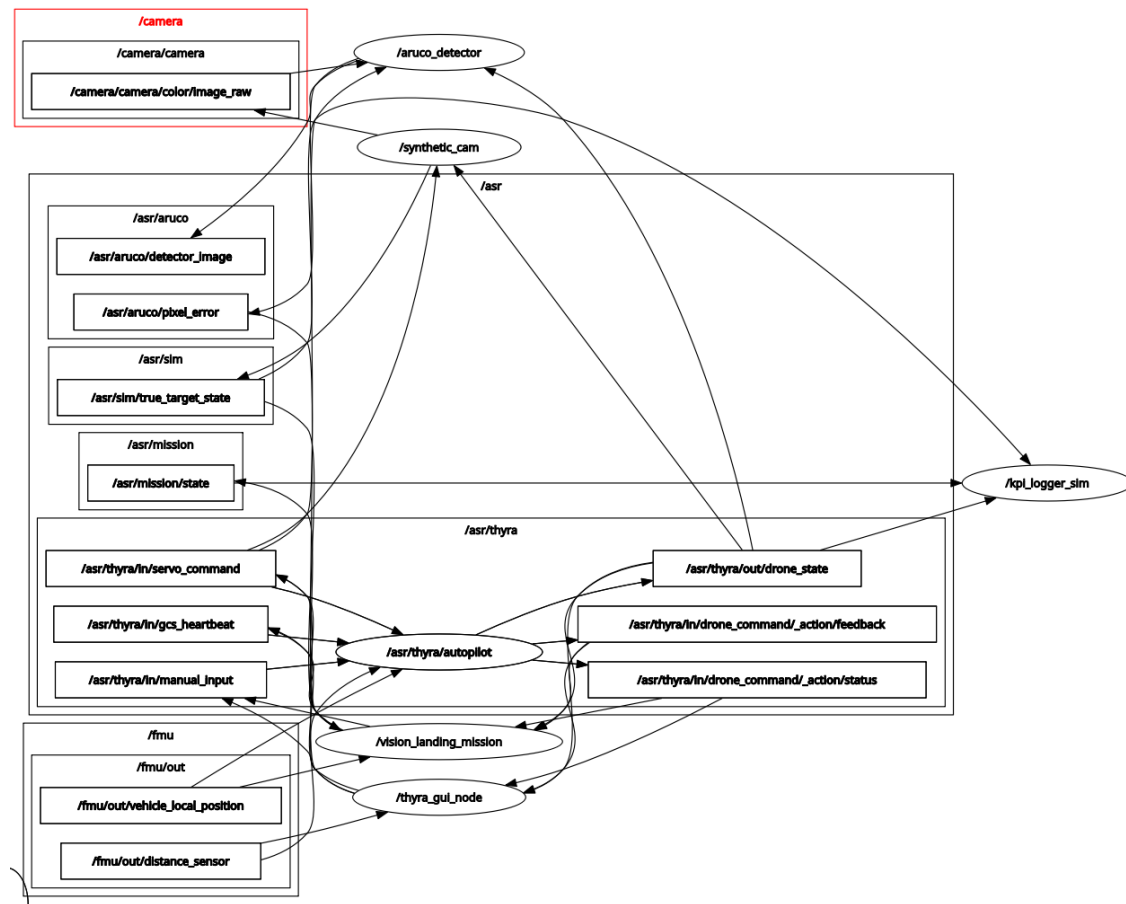


Figure C.1: ROS 2 computation graph screenshot from rqt_graph, displaying the active nodes and topic channels.

D Additional Physical Flight Plots

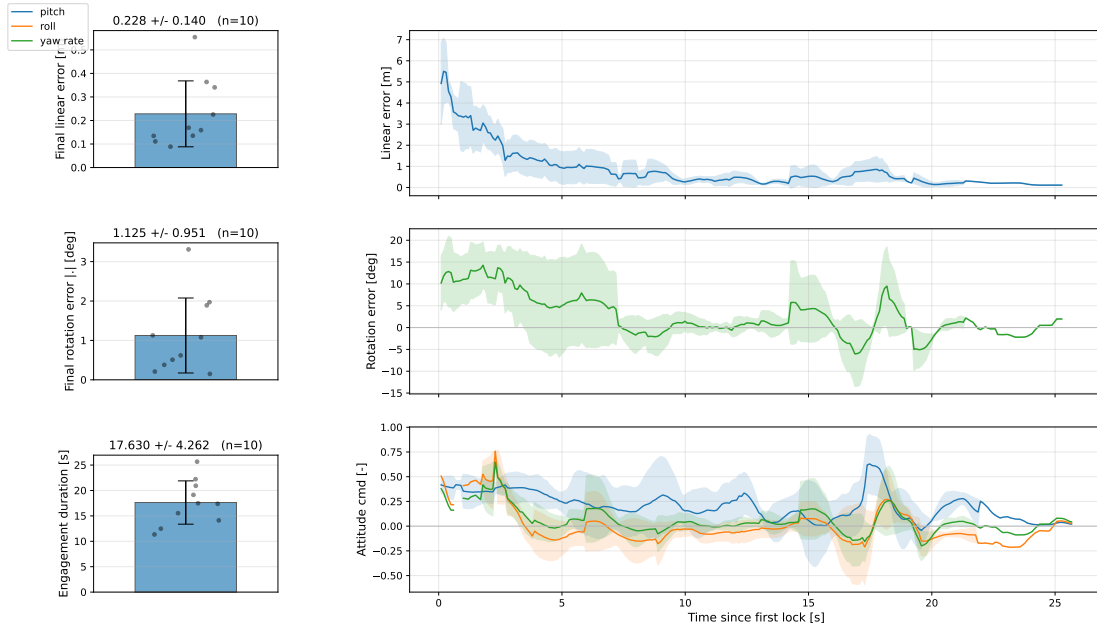


Figure D.1: Ten runs of physical flight static landings visualized as an average line with one standard deviation bands around it

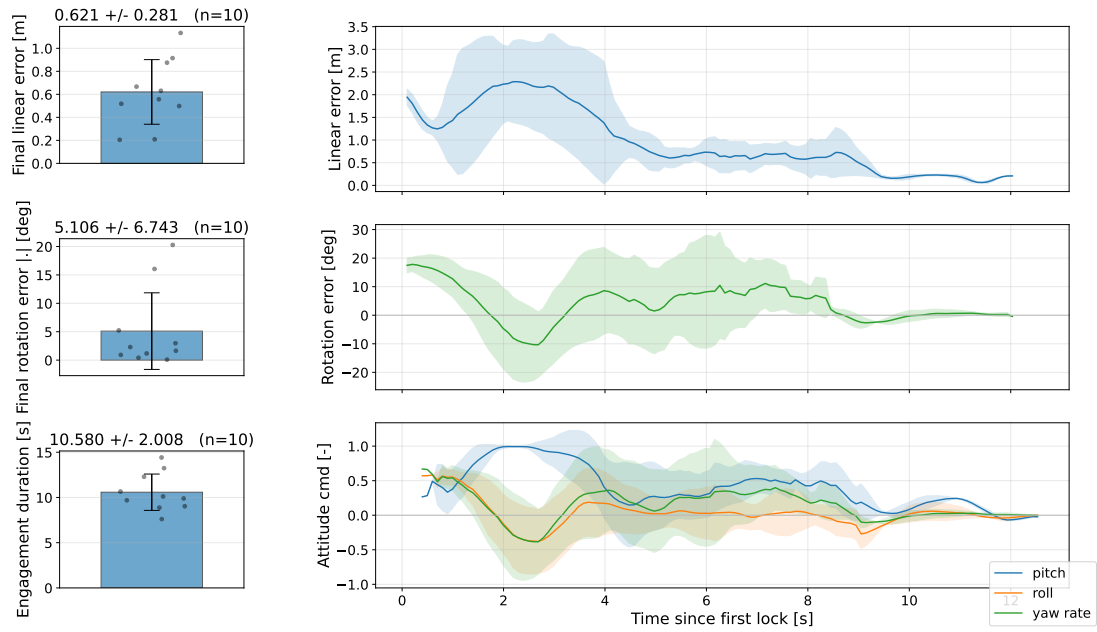


Figure D.2: Ten runs of physical flight dynamic landings visualized as an average line with one standard deviation bands around it