Precision Landing on a Moving Platform

Rishabh Bhandari[*] and Luke Brown[†]
University of Illinois at Urbana-Champaign, Champaign, IL, 61820, United States

Andrew Myers[‡]
University of Illinois at Urbana-Champaign, Champaign, IL, 61820, United States

Matthew Smith[§]
University of Illinois at Urbana-Champaign, Champaign, IL, 61820, United States

The goal of this project is to achieve autonomous precision landing on a small moving platform. We have developed a tracking system that enables the drone to follow the path of a moving object and land while the object is still in motion. This project is motivated by the local need for disaster response drones and precision, autonomous delivery drones. We implemented this precision landing on a 3D-printed platform moving in the x–y plane with a 100% success rate across 10 moving tests.

I. Nomenclature

Roman Symbols

A, B	=	Matrices used to describe the state-space system model
a_z	=	Measured acceleration in the z direction [m/s^2]
C, D	=	Matrices used to describe the state-space observer model
f	=	Function that computes the derivative of the controller state
f_z	=	Force in the z direction [N]
g	=	Gravitational acceleration [m/s^2]
h	=	Function that computes the derivative of the observer state
J_x, J_y, J_z	=	Moment of inertia about the x, y, and z axis, respectively [kg·m^2]
K, L	=	Controller and observer gain matrices, respectively
m	=	Mass [kg]
o	=	Outputs vector for the observer
p_x, p_y, p_z	=	Position in the x, y, and z direction, respectively [m]
Q, R	=	LQR weight matrices
s, i, p	=	State, input, and constant parameters vectors, respectively
u	=	Input vector with respect to an equilibrium point
v_x, v_y, v_z	=	Velocity in the x, y, and z direction, respectively [m/s]
x	=	State vector with respect to an equilibrium point
$\hat{x}$	=	State estimate vector
y	=	Outputs vector with respect to an equilibrium point

Greek Symbols

β	=	Gain value for tracking platform velocity
θ	=	Pitch angle [rad]

[*]Undergraduate Student, Aerospace Engineering
[†]Undergraduate Student, Aerospace Engineering
[‡]Undergraduate Student, Aerospace Engineering
[§]Undergraduate Student, Aerospace Engineering

τ_x, τ_y, τ_z	=	Torque about the x, y, and z axis, respectively [N·m]
ϕ	=	Roll angle [rad]
ψ	=	Yaw angle [rad]
$\omega_x, \omega_y, \omega_z$	=	Angular velocity about the x, y, and z axis, respectively [rad/s]

Subscripts

$diag$	=	Diagonal matrix
eq	=	Evaluated at the equilibrium point
o	=	Related to the observer

II. Introduction

Autonomous unmanned aerial vehicles (UAVs) are increasingly used in applications such as search and rescue, inspection, and precision delivery. One of the most demanding valuable capabilities is autonomous precision landing on moving or unstable surfaces. Achieving this requires accurate state estimation, robust control, and reliable real-time feedback. Recent research has shown significant progress in vision-based autonomous landing systems for UAVs, with approaches that use onboard cameras, external sensors, or motion capture systems to improve landing accuracy [1–3].

The goal of this project is to design, implement, and test a drone guidance system capable of autonomously landing on a small moving platform. The system will use the Qualisys motion capture network to provide position feedback for both the drone and the target. A custom tracking algorithm with integral action will adjust the drone's trajectory to match the motion of the platform. The target will carry an active marker deck to ensure position feedback, and the drone will execute a smooth descent using closed-loop control. The platform will initially move in the x–y plane, with the goal of extending this to include vertical z-axis motion.

This builds on past laboratory experiences from AE 483 but extends them to include relative motion between the vehicle and the landing surface. Previous studies have shown that precision landings can be achieved using computer vision or relative estimation even without GPS, particularly through systems that combine external motion capture with model-based controllers [2, 4]. Piponidis et al. demonstrated an approach that achieved stable touchdown on moving targets using onboard sensing and adaptive control [5]. Our project aims to implement and validate a comparable ability with drone lab motion capture, focusing on tuning the controller, controlling latency, and success rate between different target speeds.

The benefits of this technology are considerable. In disaster relief operations, an autonomous drone could deliver supplies or sensors to otherwise inaccessible locations. In industrial inspection, precision landing capability could enable automated servicing or data collection on moving machinery, improving worker safety. However, as with all autonomous technologies, such systems also pose ethical and security concerns if reused for surveillance or unauthorized access. Careful consideration of these risks is important to ensure that autonomous systems advance public safety and welfare.

The rest of this report details the design, implementation, and testing of the autonomous precision landing system. Section III describes the control architecture and algorithm design, Section IV outlines the experimental setup and testing procedure, and Section V presents results and analysis. The project ultimately determines to demonstrate repeatable precision landings on a moving platform, achieving successful touchdowns in at least 70% of flight trials.

III. Theory

A. Controller Design

A custom controller was designed to obtain better performance than the default controller. Equations of motion for the quadcopter are derived using python code that are too complex to share here. They can be found in our code **??**. They are related through a function, f, that takes the states (s), the inputs (i), constant parameters (p), and take the general form $\dot{s} = f(s, i, p)$, where

$$s = \begin{bmatrix} p_x, & p_y, & p_z, & \psi, & \theta, & \phi, & v_x, & v_y, & v_z, & \omega_x, & \omega_y, & \omega_z \end{bmatrix}^T, \quad (1)$$

$$i = \begin{bmatrix} \tau_x, & \tau_y, & \tau_z, & f_z \end{bmatrix}^T, \tag{2}$$

and

$$p = \begin{bmatrix} m, & J_x, & J_y, & J_z, & g \end{bmatrix}^T. \tag{3}$$

The nonlinear system is then linearized about the equilibrium point

$$s_{eq} = \begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \end{bmatrix}^T, \tag{4}$$

and

$$i_{eq} = \begin{bmatrix} 0, & 0, & 0, & m \cdot g \end{bmatrix}^T, \tag{5}$$

which assumes that the drone is hovering in place by setting the force generated in the z direction equal to the force due to gravity acting on the drone.

To linearize, the Jacobian of f is taken with respect to both the states and the inputs, which can be mathematically shown as

$$A = \left.\frac{\partial f}{\partial s}\right|_{(s,i,p)=(s_{\text{eq}},i_{\text{eq}},p_{\text{eq}})} \quad \text{and} \quad B = \left.\frac{\partial f}{\partial i}\right|_{(s,i,p)=(s_{\text{eq}},i_{\text{eq}},p_{\text{eq}})}. \tag{6}$$

These coefficient matrices, A and B are used in a state-space system described by

$$\dot{x} = Ax + Bu, \tag{7}$$

where

$$x = s - s_{\text{eq}}, \tag{8}$$

and

$$u = i - i_{\text{eq}}. \tag{9}$$

From here, the LQR method [6] was used to solve for a gain matrix, K, where $u = -Kx$. After an iterative process to design a controller that performs better than the default, we arrived at Q and R values of

$$Q_{diag} = \begin{bmatrix} 1000, & 1000, & 200, & 400, & 1, & 1, & 1, & 1, & 1, & 100, & 100, & 25, & 1, & 1, & 1, & 1, & 1 \end{bmatrix}, \tag{10}$$

and

$$R_{diag} = \begin{bmatrix} 1.606\text{e}6, & 1.606\text{e}6, & 1.825\text{e}8, & 2.419\text{e}3 \end{bmatrix}, \tag{11}$$

B. Observer Design

A custom observer is also designed for this project. The observer uses a similar state matrix to the controller, but omits p_x, p_y, and ψ so that the system is observable. The observer state vector is defined as

$$s_o = \begin{bmatrix} p_z, & \theta, & \phi, & v_x, & v_y, & v_z, & \omega_x, & \omega_y, & \omega_z \end{bmatrix}^T, \tag{12}$$

and the observer input, i_o, is defined as

$$i_o = \begin{bmatrix} \omega_x, & \omega_y, & \omega_z, & a_z \end{bmatrix}^T. \tag{13}$$

The output, o, is related to the observer state, observer input, and parameters by a function h such that $o = h(s_o, i_o, p)$. This model is then linearized about the chosen equilibrium point such that

$$C = \left.\frac{\partial h}{\partial s_o}\right|_{(s_o,i_o,p)=(s_{\text{eq}},i_{\text{eq}},p_{\text{eq}})} \quad \text{and} \quad D = \left.\frac{\partial h}{\partial i_o}\right|_{(s_o,i_o,p)=(s_{\text{eq}},i_{\text{eq}},p_{\text{eq}})}. \tag{14}$$

These coefficient matrices, C and D, are used in a state-space system described by

$$y = Cx + Du, \tag{15}$$

where

$$x = s_o - s_{\mathrm{eq}}, \tag{16}$$

$$u = i_o - i_{\mathrm{eq}}, \tag{17}$$

and

$$y = o - h(s_{eq}, i_{eq}, p_{eq}). \tag{18}$$

The observer is now defined as

$$\dot{\hat{x}} = A\hat{x} + Bu - L(C\hat{x} + Du - y), \tag{19}$$

where $\hat{x}$ is the state estimate, and L is a gain matrix found through LQR, using A^T, C^T, R_o^{-1}, and Q_o^{-1}. Q_o and R_o are defined as

$$Q_{o_{diag}} = \begin{bmatrix} 1.527e{-1}, & 1.039e{-1}, & 1.111e05 \end{bmatrix}, \tag{20}$$

and

$$R_{o_{diag}} = \begin{bmatrix} 355.999, & 307.787, & 152.416, & 11.491, & 8.500, & 8.500 \end{bmatrix}. \tag{21}$$

C. Platform Design

Due to the nature of the project and the goal of landing on a moving platform, a suitable landing platform was needed for testing. To determine the platform concept that was to be used, all members of the team were tasked with brainstorming ideas and generating a pros and cons list, as shonw in Table 4.

Table 4 Landing Platform Ideas

#	Name	Description	Pros	Cons
1	Sliding Platform	Flat board on smooth rods or drawer slides; move it manually left–right.	Easy to make.	Not as consistent movement from test-to-test.
2	Drone Box	Use the box the drone came in and push it with your hands/a ruler across the carpet.	Simple, and we already have the box.	Very small landing area; movement will likely be inconsistent
3	Motorized Platform	Mount a motor to the center wheel under the platform, similar to idea 1, so it moves consistently on its own.	Removes potential for human error and variance in platform speed; consistent, replicable movement.	Would require spending time on the engineering of the platform; not necessarily feasible within our time constraints.
4	3D Printed Platform	A combination of a dedicated 3D printed platform, using strings to pull and provide translation.	Control over platform size; easy to implement.	Motion relies on someone manually moving the platform.

Upon close examination by the group, design number 4 was chosen due to the combination of being easy to create and simple to implement, allowing the group to move to the testing stage quickly. While the 3D platform must be manually moved, the trade-off for time and simplicity outweighs the benefits of spending hours creating a motorized platform.

The final platform design is shown in Fig. 1. The platform has side lengths of 18 cm, and includes four nubs underneath to allow the platform to move across the drone lab carpet. Wheels were initially considered, but nubs were ultimately chosen as they were less complex and achieved the same performance. Four indented circles are located on top of the platform to ensure accurate placement of the four passive markers. Finally, circular cutouts are included on all four edges as tie points for string that can be used to pull the platform from a distanc without interfering with the flight path or mocap coverage. This design was 3D printed using PLA in a Bambu Lab P1S printer.

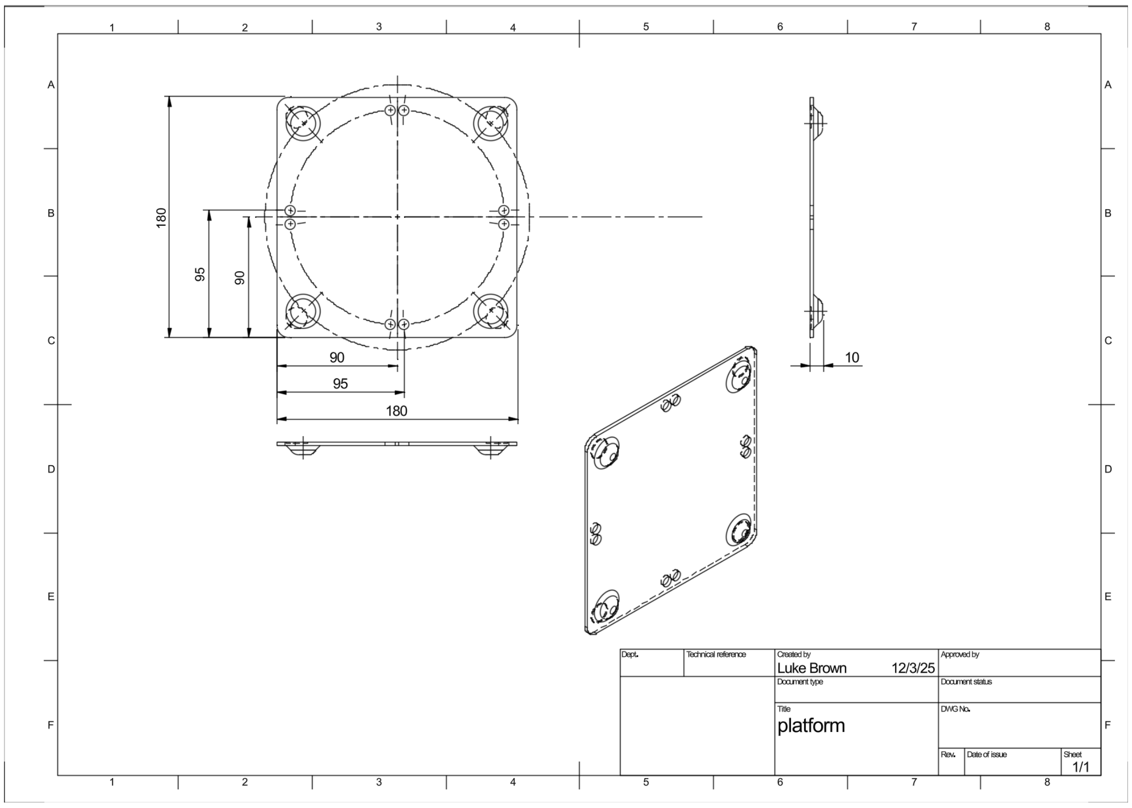

Fig. 1 The final landing platform schematic shows the landing area, as well as the indented passive marker locations.

D. Tracking

A tracking algorithm is needed to take the drone and platform position as inputs, and output move commands to the drone. In order to consider a wide range of ideas, each member of the team was again responsible for researching and proposing a method. Within each proposal, as seen in Table 5, is a rough description of how implementing the method might work, as well as pros and cons for the method in comparison to other choices. Each team member has the opportunity to make a case for their method during a design meeting, and the final method choice was chosen unanimously by the group after discussion.

The result of this discussion was the choice of method 1, an $\alpha-\beta$ filter, as the preferred method due to the simplicity of the model. This model works well for estimating the velocity of the platform, which we can use to update the guess of the platform's position at the next time step.

The $\alpha - \beta$ filter is a tracking method that works by smoothing out measurement values to converge on a state estimate. Two gains, α and β, are introduced to weight the influence of the measurements on the estimation of the platform's position and velocity, respectively. Due to the high accuracy of the mocap measurements, we can take the mocap position measurement as the absolute truth, and the α part of the filter is not needed. The gain β is multiplied by the difference between the current and previous positions of the platform divided by the time difference. This gives an updated estimate for the velocity of the platform, weighted by a gain β. This value can be multiplied by the time difference and added to the current platform position to estimate the position at the next time step, which is sent to the drone. By programming the drone to fly to where the platform will be at the next time step, the drone will not continuously lag behind the platform.

Table 5 Tracking Algorithm Ideas

#	Name	Description/Implementation	Pros	Cons
1	$\alpha - \beta$ Filter [7]	Use gain α to smooth out measurement noise in position. Use gain β to smooth out calculated noise in velocity. Update drone position and velocity based on observed position of target.	Very simple to comprehend and implement; only two gain values to adjust.	Behaves poorly for non-constant velocity.
2	Artificial Potential Field [8], [9]	Create a function that generates a gradient with a minima located above the center of the target. This function is then used to determined desired acceleration.	We already have experience with this method from AE 353, and therefore should be able to implement it quicker than other methods.	This method could potentially saturate the quadcopter's motor commands and lead to irratic behavior.
3	Nonlinear / Relative-State MPC for Quadrotor Landing [10, 11]	Solve a finite-horizon optimal control problem at each step. The MPC predicts the platform motion (from Qualisys) and commands position/velocity setpoints that drive the relative error to zero. Integral action can be added by augmenting the state with the accumulated position error.	Handles moving targets directly; enforces input/state limits; naturally produces smooth trajectories.	Requires more computation and model setup than simple filters; performance depends on motion-capture update quality.
4	Proportional Navigation (PN) Tracking [12]	Implements a guidance law that commands acceleration proportional to the line-of-sight (LOS) rate between the drone and the moving platform. Using position data from the Qualisys system, the LOS angle λ and its rate $\dot{\lambda}$ are computed each cycle. The commanded lateral acceleration is $a_{\perp} = NV_c\dot{\lambda}$, where N is a navigation constant (typically 2–5) and V_c is the closing speed. This acceleration is converted into velocity setpoints for the drone's controller, and descent in z begins only when the lateral position error $\|r\|$ is below a chosen threshold.	Simple, lightweight, and real-time friendly; few tunable parameters; robust to moderate noise and latency; easy to integrate with existing velocity-command architecture.	Assumes approximately constant-velocity target; requires filtered LOS-rate estimation; not globally optimal like MPC; performance decreases for highly nonlinear or abrupt target motion.

IV. Experimental Methods

A. Platform Tracking Testing

In order to test the drone system, the first step was to confirm that the landing platform was properly set up and to send motion capture data to our Python script. Specifically, our script shall accurately measure the location of the center of the platform over time in the x, y, and z positions such that the estimates stay within the bounds of the nine-square-meter carpeted test area in the Talbot Drone Lab. Additionally, the system shall maintain over 90 percent mocap coverage for the duration of platform tracking in each given test.

To verify that our Python data handling script meets the first requirement, we will conduct 11 tests in which the platform is moved in random patterns across the carpeted area in the drone lab. We will then examine the collected motion capture data and identify that the platform data stayed within the carpeted area and that the motion capture coverage was greater than 90 percent for each test.

B. Stationary Platform Testing

For preliminary tracking algorithm testing, the drone first needs to be able to land on the platform while the platform is stationary. Specifically, the drone shall land on the stationary platform over 70 percent of the time when the drone and platform start at random locations on the drone lab carpet.

Success is defined by landing on the platform with any part of the drone. A complete success is defined by landing fully on the platform, with all four drone spars in contact with the platform. A partial success is defined by the drone center landing close enough to the platform that at least one of the spars makes contact with its surface. Finally, a failure is defined as no parts of the drone making contact with the platform when landing. This definition sufficiently categorizes the drone's success due to the size of our platform, which is just larger than two drone widths across. We consider both partial and complete success to count towards the success rate.

To verify that the tracking algorithm is able to meet this requirement, seven flights will be conducted, running our tracking algorithm. The initial starting locations of both the drone and the landing platform will be chosen at random. The drone will start with zero yaw in the motion capture world frame. A success rate will be calculated from these seven flights based on visual inspection of the drone and platform. The goal of each trial is to ensure that, regardless of the drone's starting point and the target's location, the drone will be able to track the platform and land on it.

C. Moving Platform Testing

Once it is verified that the tracking algorithm is able to track and land on a stationary platform, the final step is to test tracking a moving platform. In order to be considered successful, our drone shall land on the moving platform in at least 70 percent of test flights. Success will be categorized in the same manner as stationary platform testing. For each flight, the drone and platform started at random initial positions, with a team member pulling the platform from a distance in a random direction at a roughly constant velocity. A minimum of 10 flight tests will be performed.

Throughout this testing process, the controller gains will be updated to better track the platform. The final gains of our controller and observer are listed in Section V. Additionally, an example flight path for motion in a line, circle, and triangle is presented in Section V.

V. Results and Discussion

This section summarizes how well our precision-landing system performed in different testing conditions. The results include (1) tracking a moving platform using mocap, (2) tracking a stationary platform, and (3) landing on a platform moving in the x–y plane.

A. Moving Platform Mocap Tracking

In order for the drone to track the moving platform and land on its center, the drone needs real-time data of the position of the platform. Thus, 11 tests were performed, focused solely on determining if the platform can be successfully tracked by the mocap system while moving in various patterns. Among the patterns tested are straight lines in the x, y, and z directions, diagonal lines, squares, circles, and triangles.

Figure 2 showcases a particular result from one such test when moving the platform in a straight line in the $-z$ direction. As seen in the figure, the platform's position was tracked 100% of the time, which is above the 90% tracking goal. The results for all 11 tests are listed in Table 6.

Across the board, the platform was tracked by the mocap system at a 100% success rate. This fulfills the >90% tracking requirement, giving the team confidence to move on to the next project goal.

B. Static-Platform Tracking Performance

When the platform was still, the drone consistently moved towards it and settled over the landing area. Figure 3 shows a top-down view of the third stationary platform test, showing that the drone can track the position of the platform and land very close to the center. However, the drone travels past the platform before backtracking to the center, indicating that tuning was needed for the tracking of p_x and p_y. Furthermore, Fig. 4 shows the three-dimensional trajectory of the drone in the same stationary test, in which the drone continued to ascend past the hover height before descending to the platform. This additionally indicated that the tracking of p_z needed tuning.

Three main quantities were tuned to produce better results. The most influential change was decreasing the frequency of the move commands from 10 Hz to 1.333 Hz, eliminating much of the drift in all three position values. Additionally, the maximum speed of the drone was decreased from 2 m/s to 0.5 m/s, causing smoother movement and less overshooting.

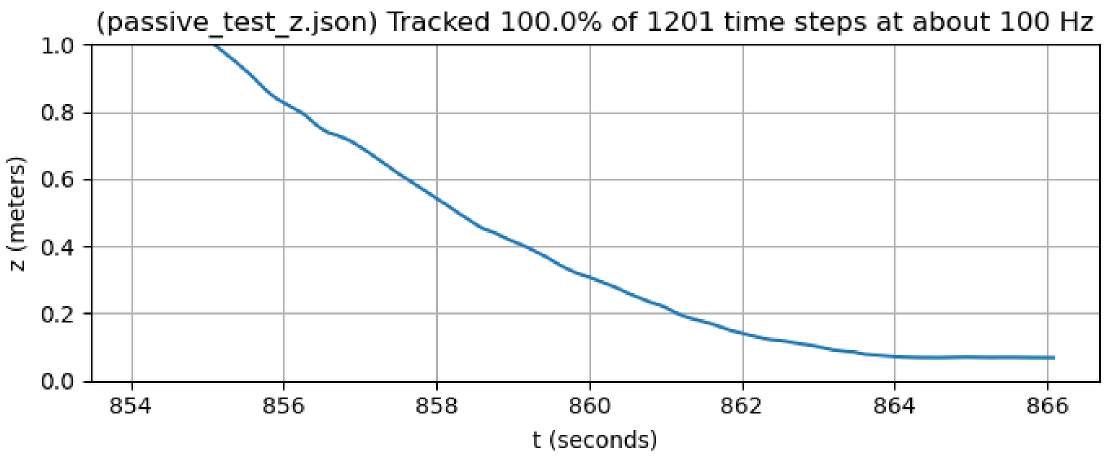

Fig. 2 Testing the mocap tracking of the platform in the z-direction results in 100% tracking over the duration of the test.

Table 6 Platform Mocap Tracking Tests

Test Number	Platform Movement	Mocap Tracking Percentage	Success?
1	Stationary	100%	Success
2	$\pm y$ direction	100%	Success
3	$\pm x$ direction	100%	Success
4	$+x$ direction	100%	Success
5	$-z$ direction	100%	Success
6	$+x, -y$ diagonal	100%	Success
7	Square	100%	Success
8	Triangle	100%	Success
9	Circle	100%	Success
10	$+x, -y, -z$ 3D diagonal	100%	Success
11	$+y$ diraction	100%	Success

Finally, the landing tolerance was reduced to ±8 cm from the center of the platform in p_x and p_y and within 12 cm above the platform.

These changes were implemented following the third stationary test. Figure 5 demonstrates a top-down view of the seventh stationary platform test. The drone flies directly towards the platform, including only minor backtracking to land within the more strict landing tolerances. A three-dimensional view of this test is shown in Fig. 6. The drone no longer drifts above the hover height, starting to descend towards the platform at the first move command. Near the platform, the drone had minor issues landing, likely due to ground effects and the precision needed for the landing tolerances. Regardless of this trajectory, the drone still landed completely on the platform.

Table 7 compiles all seven tests involving a stationary platform. After having inconsistent results in the first three tests, the variables were tuned before the fourth trial. After the tuning was completed, the drone landing became much more consistent, aside from test 5, which featured minor issues with mocap tracking. Throughout all seven tests, the success rate was 71.4%, which is above the 70% threshold desired to consider the project a success. Furthermore, when looking at the results after tuning, the drone was able to track the platform at a 75% success rate, giving the team confidence to proceed to tracking a moving plaform.

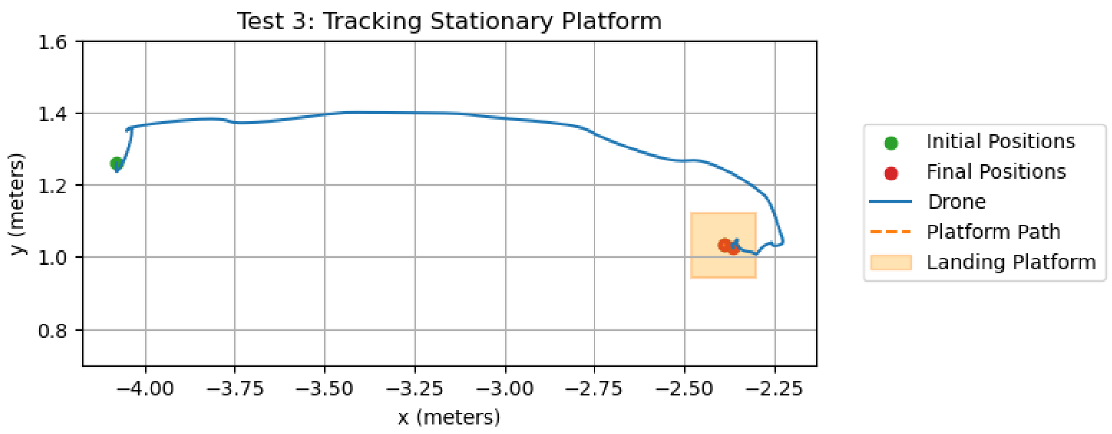

Fig. 3 In the third stationary platform test, the drone tracks and lands on the platform, but overshoots the platform before landing.

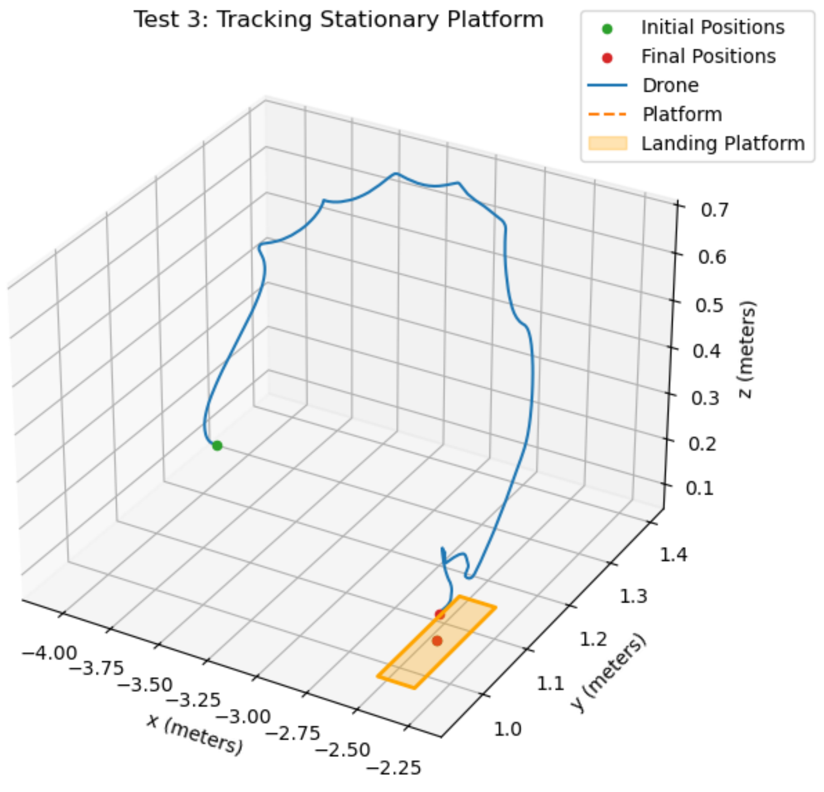

Fig. 4 The trajectory for the third stationary platform test is plotted in three dimensions.

C. Tracking a Platform Moving in the x–y Plane

When the platform moved in straight lines, the drone was generally able to follow it. Ten moving platform tests were performed, with the platform moving in either the x, y, or a planar diagonal direction. One example is the third moving test, in which the drone tracked the platform moving in the x direction. Figure 7 showcases the trajectory of the drone from a top-down view, and Fig. 8 shows the trajectories of the drone and platform in three dimensions. Through these views, it can be clearly seen that the drone landed nearly directly on the center of the platform, making this test a complete success.

The landing success rate is summarized in Table 8. Our goal was at least a 70% success rate, and we ended up

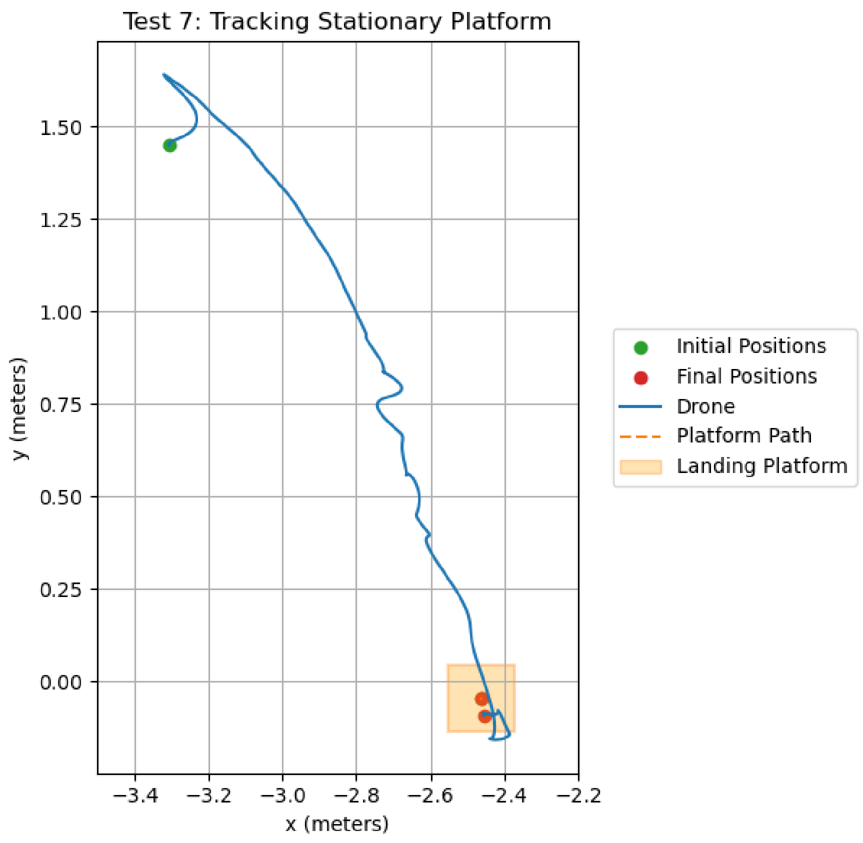

Fig. 5 In the seventh stationary platform test, the drone tracks and lands on the platform with minimal overshooting.

Table 7 Stationary Platform Tracking Tests

Test Number	Distance from Platform Center (cm)	Landing	Success?
1	7.132	Edge of platform	Partial
2	12.19	Off platform	Failure
3	2.635	On platform	Complete
4	6.869	Edge of platform	Partial
5	17.91	Off platform	Failure
6	5.884	Edge of platform	Partial
7	4.707	On platform	Complete

exceeding that goal with a 100% success rate.

It is important to note that the landing data for the second moving test has been discarded, as the data from the platform did not fully track during the drone landing sequence. The group watched the drone land completely on the platform, yet the final position of the platform is not underneath the drone in the corresponding JSON file. This trial is still treated as a complete success due to the drone still landing on the platform, but the actual landing distance is not included. The fact that data did not fully track is important to keep in mind for the remaining trials.

As a whole, the data in Table 8 indicates that the drone performed exceptionally well in tracking the platform and landing on its center. It is important to distinguish that the drone performed worse when moving in a diagonal, with

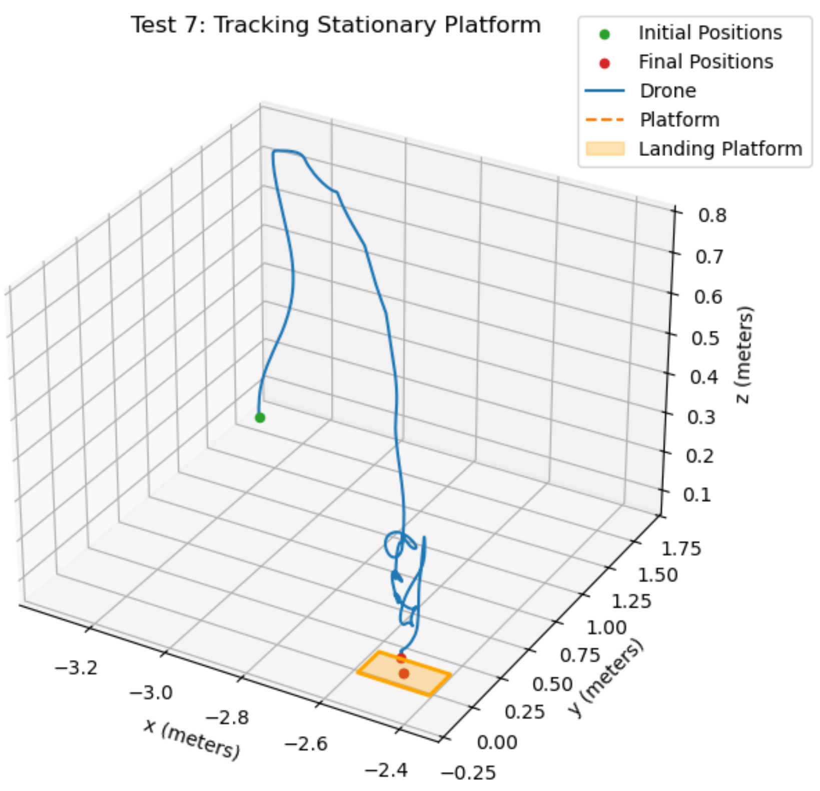

Fig. 6 The trajectory for the seventh stationary platform test is plotted in three dimensions.

Table 8 Moving Platform Tracking Tests

Test Number	**Platform Movement**	**Distance from Platform Center (cm)**	**Landing**	**Success?**
1	*y* direction	4.209	On platform	Complete
2	*x* direction	N/A	On platform	Complete
3	*x* direction	0.4437	On platform	Complete
4	*x* direction	3.961	On platform	Complete
5	*y* direction	7.725	Edge of platform	Partial
6	*xy* diagonal	8.255	Edge of platform	Partial
7	*xy* diagonal	10.43	Edge of platform	Partial
8	*y* direction	6.528	Edge of platform	Partial
9	*y* direction	4.149	On platform	Complete
10	*x* direction	4.344	On platform	Complete

both trials ending with a partial success. In future iterations of the project, we would test more complex 2D paths, such as squares, circles, and triangles, to see how the drone performs. The current successes of the drone point towards the success rate remaining well above the 70% threshold.

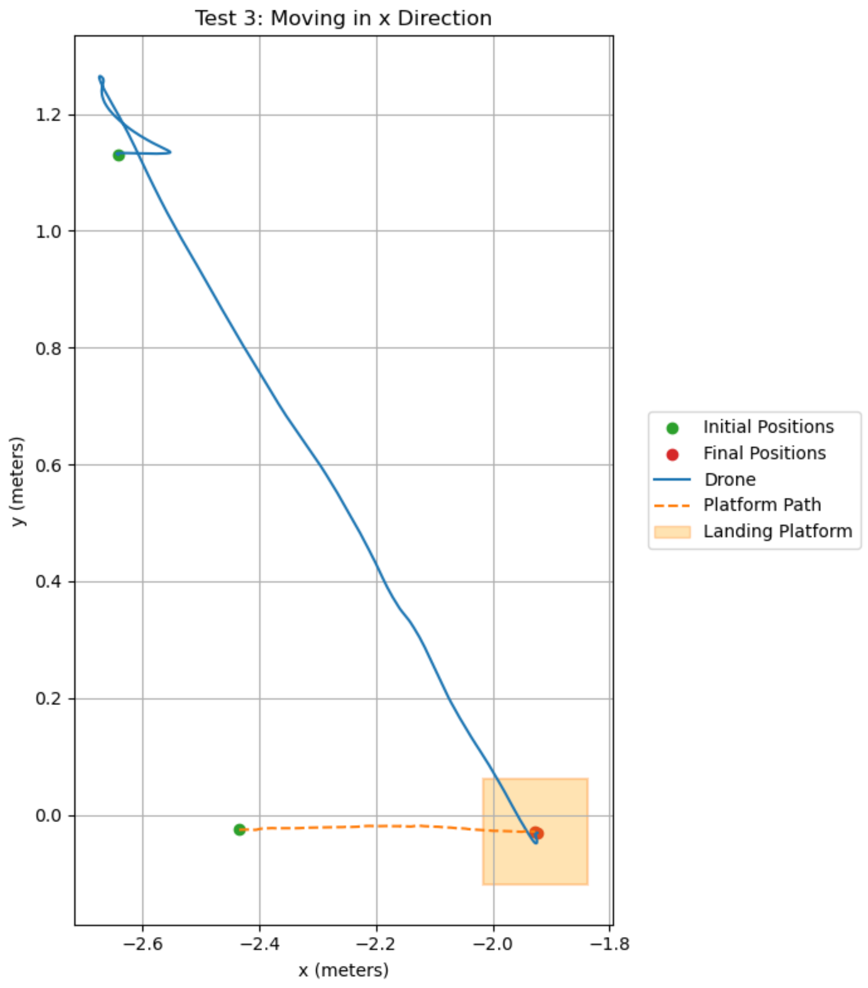

Fig. 7 The drone linearly tracks the position of the platform, landing nearly on the exact center.

D. Unexpected Behaviors

A few behaviors that appeared repeatedly that were not expected inlcude (i) side drift during final descent: sometimes the drone drifted sideways right before touching down. This may have been caused by motion-capture latency or less reliable marker detection when the drone was close to the ground. (ii) Oscillations during quick platform motions: when the platform suddenly changed speed or direction, the drone sometimes overcorrected. This fits what we observed from the potential field method, which can react poorly to abrupt changes. (iii) Slower vertical response: in moving-platform tests, height corrections were sometimes slower than expected. This may have been influenced by ground effect or limits in how quickly the drone adjusts its thrust.

E. Sources of Error and Uncertainty

Throughout testing, several factors likely affected our results: delay in motion-capture data reaching the drone, sensor noise from the IMU and Qualisys system, simplifications in our linear model compared to real drone dynamics, sensitivity of the potential field method to tuning choices, variations in how the platform was moved by hand, and changes in battery level affecting thrust output. These uncertainties will be quantified where possible using RMSE values and trial-to-trial variability when the data are complete.

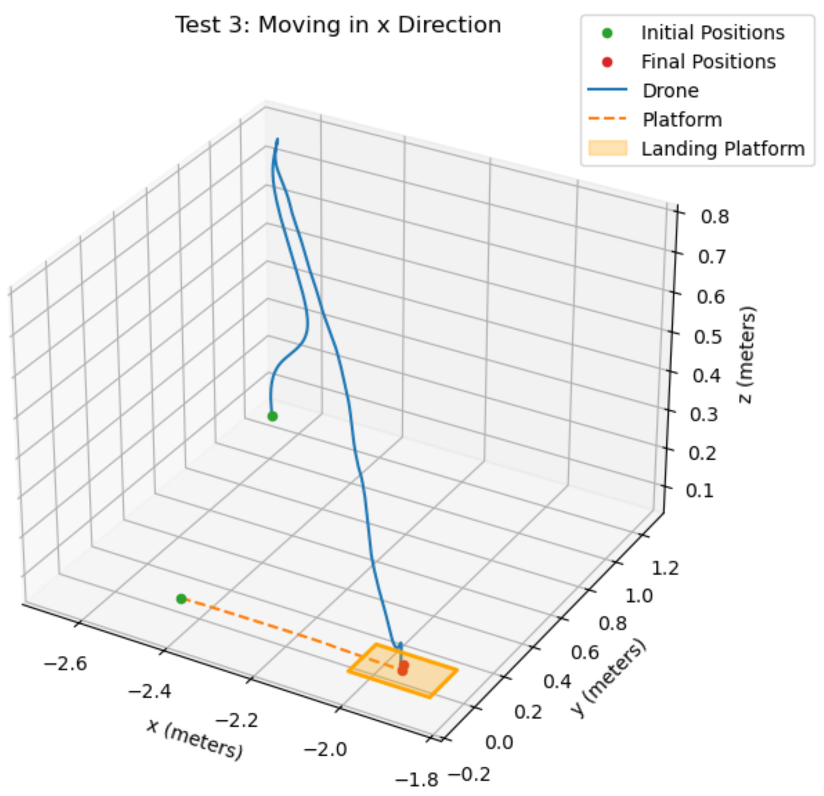

Fig. 8 This 3D view of the third moving test further shows the drone landing on the platform.

VI. Conclusion

In this project we developed and tested a system that allows a Crazyflie quadrotor to land on a small moving platform. The system relied on motion-capture measurements from Qualisys and a custom control stack running onboard the drone. This stack included a linear-quadratic regulator designed about hover and an observer that estimated unmeasured states using IMU data.

To handle platform motion, we implemented a simple prediction layer in the Python client. Platform velocity was estimated using a β-filter and used to predict the platform position one step ahead. The drone was commanded toward this predicted position using rate-limited position setpoints. With appropriate tuning of command rate, maximum speed, hover offset, and landing tolerances, the drone reliably aligned with the platform before descending.

Tests on a stationary platform showed consistent convergence and touchdown within the specified tolerances. When the platform moved along straight paths in the x, y, and diagonal directions, the drone was able to follow the motion and land successfully in all trials, with some edge landings occurring as platform speed increased. These results are consistent with the use of a constant-velocity prediction model and a low update rate, which favor smooth, straight-line motion.

The main limitations observed were small position errors near touchdown and increased edge landings during faster motion. These effects are likely due to motion-capture latency, discrete step commands, and the simplicity of the velocity prediction model. Future improvements could include higher-rate setpoint updates, more advanced state estimation for the platform motion, and guidance expressed directly in the platform reference frame. These changes would allow the system to handle faster motion and provide a stronger foundation for future experiments involving more complex trajectories.

Appendix: Data, Code, and 3D Printing Files

Data and code were submitted as an attachment to this report and are available upon request.

"Flight.py" is the flight code that contains the tracking algorithm. When run with mocap enabled, the drone will automatically take off, maneuver to the platform, and land.

"controller_ae483.c" is the controller file with custom controller and observer implemented.

"milestone-3.ipynb" is a jupyter notebook that plots data from the moving platform tracking tests.

"moving_platform_tests.ipynb" is a jupyter notebook that continues plotting data from the moving platform tracking tests.

"milestone-results.ipynb" is a jupyter notebook that compiles results from our three testing milestones into one notebook.

"animations.ipynb" is a jupyter notebook that creates animated videos of the 3d plotted flight paths.

"platform v2.3mf" is a 3D printer file that can be used to print the landing platform used in our tests.

Appendix: Link to Final Video

A link to a video of our drone performance can be found here: link to final video.

Acknowledgments

We would like to thank our TA, Anmol Harshana, for his continued support throughout the design of our project. Additionally, we would like to thank Professor Timothy Bretl and Professor Wayne Chang for their guidance and for the template code used for our drone design.

References

[1] Xin, L., Tang, Z., Gai, W., and Liu, H., "Vision-Based Autonomous Landing for the UAV: A Review," *Aerospace*, Vol. 9, No. 11, 2022, p. 634. https://doi.org/10.3390/aerospace9110634, URL https://www.mdpi.com/2226-4310/9/11/634.

[2] Alarcón, F., García, M., Maza, I., Viguria, A., and Ollero, A., "A Precise and GNSS-Free Landing System on Moving Platforms for Rotary-Wing UAVs," *Sensors*, Vol. 19, No. 4, 2019, p. 886. https://doi.org/10.3390/s19040886, URL https://www.mdpi.com/1424-8220/19/4/886.

[3] Sconfienza, A., "Autonomous Landing of a UAV on a Moving Platform," Ph.D. thesis, Politecnico di Torino, 2021. URL https://webthesis.biblio.polito.it/secure/20395/1/tesi.pdf.

[4] Wang, P., Wang, C., Wang, J., and Meng, M., "Quadrotor Autonomous Landing on Moving Platforms," *Procedia Computer Science*, Vol. 209, 2022, pp. 40–49. https://doi.org/10.1016/j.procs.2022.10.097, URL https://www.sciencedirect.com/science/article/pii/S1877050922015460.

[5] Piponidis, M., Aristodemou, P., and Theocharides, T., "Towards a Fully Autonomous UAV Controller for Moving Platform Detection and Landing," *arXiv preprint arXiv:2210.08120*, 2022. URL https://arxiv.org/abs/2210.08120.

[6] Bretl, T., "Optimization and Optimal Control," , ???? URL https://aero.refpages.org/control/optimal-controllers/#lqr, accessed: November 12, 2025.

[7] Becker, A., "The $\alpha - \beta - \gamma$ Filter," , 2025. URL https://kalmanfilter.net/alphabeta.html, accessed: November 10, 2025.

[8] Kownacki, C., "Artificial Potential Field Based Trajectory Tracking for Quadcopter UAV Moving Targets," , 2024. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC10893262/pdf/sensors-24-01343.pdf, accessed: November 11, 2025.

[9] Bretl, T., "Collision Avoidance," , 2025. URL https://tbretl.github.io/ae353-sp25/notes/20250423-collision-avoidance.pdf, accessed: November 11, 2025.

[10] Alexis, K., Nikolakopoulos, G., and Tzes, A., "Model predictive control of a quadrotor for autonomous takeoff and landing," *2011 IEEE International Conference on Control Applications (CCA)*, IEEE, 2011, pp. 2247–2252.

[11] Faessler, M., Achtelik, M., Siegwart, R., et al., “Autonomous, vision-based landing and pattern detection for MAVs,” *Journal of Field Robotics*, 2017.

[12] Zarchan, P., *Tactical and Strategic Missile Guidance*, 6^{th} ed., AIAA, 2012.