Vision-Based Landing of a Parrot Mambo Drone on a Moving Line-Follower Robot

Gourishankar Mahadeo Bansode
School of Computing and Augmented Intelligence
Arizona State University
Tempe, Arizona, USA
gourishankar@asu.edu

***Abstract*—This report describes the Vision-Based Landing project from RAS 546: Robotics Systems II. The project combined a Parrot Mambo Minidrone with a small line-follower robot that carried a colored landing platform. The goal was to make the drone detect the platform with its downward camera, stay aligned while the robot was moving, and land near the end of the track. The system used MATLAB/Simulink, color thresholding, centroid and area measurement, a PI alignment controller, and a simple state machine for takeoff, tracking, descent, and landing. In testing, the platform was tracked at speeds up to 0.2 m/s, detection reliability was above 95%, average tracking error stayed below 8 pixels, and landing accuracy was 80% within 5 cm over 10 trials. This portfolio version uses simple language and includes a contribution statement, what was learned, and a short reference list.**

I. Introduction

Drone landing on a moving platform is a useful robotics problem because it requires perception, control, and timing at the same time. A real system must see the target, estimate where it is, stay aligned with it, and begin descent at the right moment. MATLAB/Simulink and the Parrot Minidrone support package were a practical starting point for this project because they provide camera access, deployment tools, and example models that are well suited for fast prototyping and indoor flight experiments [1], [2], [5].

This project focused on a small indoor version of that problem. Instead of landing on a fixed pad, the drone had to land on a platform carried by a line-follower robot.

The main idea was simple: let the drone see the moving platform, keep it near the center of the image, and start landing only when the timing looked safe. Even though the idea sounds straightforward, it became a strong systems project. The camera pipeline, the controller, the robot motion, and the landing logic all had to work together for the landing to succeed.

II. How the System Works

The setup used three main parts: a Parrot Mambo Minidrone, a small line-follower robot, and a lightweight landing platform mounted on the robot. The platform was designed to be easy to detect from the drone's downward camera, and the ground robot was tested first to make sure it could follow its track at a repeatable speed. Figure 1 shows the main hardware used in the project.

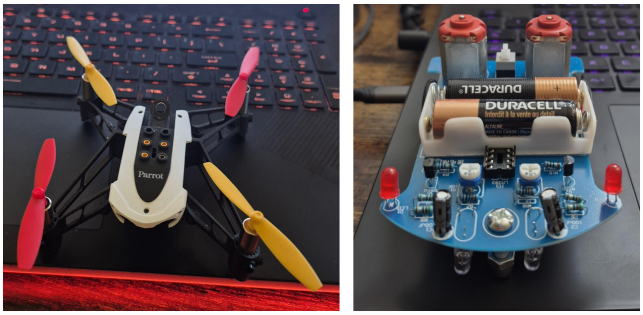

Fig. 1. Main hardware used in the project: the Parrot Mambo Minidrone and the line-follower robot carrying the landing platform.

The software was built in Simulink. This made it possible to read the camera image, process the platform region, generate control commands, and deploy the logic to the drone in one environment [1], [5]. To detect the platform, the image from the downward camera was converted into a form that was easier to threshold, and a binary mask was created for the platform color [1], [2]. From that mask, the system extracted two simple image features: the centroid and the segmented area. The centroid showed where the platform was in the image, and the area gave a rough cue about distance and descent readiness.

Once the platform features were available, the controller used the centroid error to correct lateral motion and yaw. In simple terms, if the platform moved away from the center of the image, the controller pushed the drone back toward it. This follows the main idea of image-based visual servoing, where image features are used directly to guide motion [4].

Landing required a second layer of logic. The system used a small state machine with four stages: takeoff, tracking, descent, and landing. The drone stayed in tracking mode until the platform was close enough to the center of the image and large enough in the frame. A lead-distance estimate based on robot speed and drone descent rate was then used to begin the landing sequence near the end of the track. Before full moving-platform trials, stationary landing tests were used to tune thresholds, reduce false detections, and check that the descent trigger was not starting too early. The full loop was: see the platform, measure its image features, keep it centered, and descend when the timing was right. Figure 2 summarizes that pipeline.

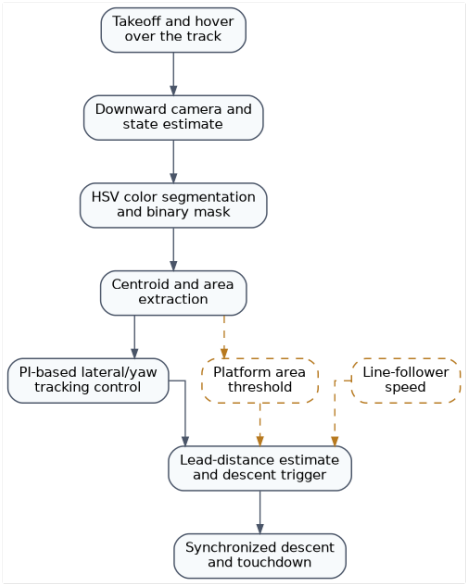

Fig. 2. System diagram for the landing system. The pipeline moves from the moving platform and camera input to platform detection, feature extraction, alignment control, and timed landing.

III. Results

The prototype worked reliably in a controlled indoor setting. In moving trials, the drone kept the platform near the image center while the robot moved at speeds up to 0.2 m/s. Across 10 runs, the system achieved more than 95% platform detection reliability, average tracking error below 8 pixels, and 80% landing accuracy within 5 cm of the target endpoint.

These results showed that the strongest part of the system was the vision pipeline. The platform was detected consistently in most runs, which meant the color-thresholding approach was good enough for the project conditions. The harder part was the final landing timing. It was not enough to see the platform correctly; the drone also had to begin descent at the right moment while the target was still moving.

The project also showed several practical limits. Lighting changes could disturb the mask, especially when reflections appeared on the floor or the platform. Small controller changes could also affect hover stability and landing accuracy. When the gains became too aggressive, the drone started to oscillate more and the final touchdown became less repeatable. Even with those limits, the project achieved its main goal: it demonstrated that a small drone could use simple vision and control logic to land on a moving ground robot.

Another useful result was the testing process itself. The final landing system did not appear all at once. The project moved step by step from color detection, to stationary landing, to moving-platform tracking, and then to full timed landing.

IV. Future Improvements

If the project continues, the first upgrade would be better sensing and estimation. Adaptive thresholding could make the color mask more stable under lighting changes, and online speed estimation could improve the landing trigger when the robot speed changes during the run. A more predictive controller could also help reduce delay during tracking.

A second upgrade would be a stronger landing target. This version mainly used color as the visual cue, which worked well indoors but can become less reliable when the scene change Direct communication between the drone and the ground robot would also improve coordination because the landing decision would not depend only on what the camera sees.

V. Contribution and What Was Learned

This was a **solo project** completed by Gourishankar Bansode. He assembled and tested the line-follower robot, mounted the landing platform, built the image-processing pipeline, extracted centroid and area information, tuned the PI controller, created the landing state machine, deployed the Simulink model, and ran the experiments. Because the project was completed individually, the full system from hardware setup to final evaluation was his responsibility.

One of the most useful parts of this project was systems integration. It was not enough to make one block work in isolation. The camera, color detection, controller, state machine, and robot motion all had to work together in a live loop. That meant the project required careful testing, gain tuning, and repeated debugging to find whether a bad landing came from vision, timing, or control.

This project taught several useful skills. First, it gave practical experience with vision-based robotics, especially color segmentation, threshold tuning, and using image features for control [6].

VI. Conclusion

In this project, a Parrot Mambo drone tracked and landed on a moving platform carried by a line-follower robot. The system combined color-based detection, centroid tracking, PI control, and timed descent logic in one working prototype. The final results were strong enough to show that the idea works, and the failure cases gave a clear path for future improvement. For the portfolio, this project shows both technical implementation and the lessons learned from building a complete vision-based robotics system.

References

[1] MathWorks, "Getting Started with Image Processing Algorithms for Parrot Minidrone," [Online]. Available: https://www.mathworks.com/help/simulink/supportpkg/parrot_ref/getting-started-with-parrot-minidrone-vision.html

[2] MathWorks, "Fly a Parrot Minidrone and Detect Objects," [Online]. Available: https://www.mathworks.com/help/simulink/supportpkg/parrot_ref/color-detection-and-landing-parrot-example.html

[3] MathWorks, "Path Planning Using Keyboard Control for Parrot Minidrone," [Online]. Available: https://www.mathworks.com/help/simulink/supportpkg/parrot_ref/path-planning-keyboard-example.html

[4] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, Dec. 2006, doi: 10.1109/MRA.2006.250573.

[5] MathWorks, "Parrot Minidrones - Hardware Support," [Online]. Available: https://www.mathworks.com/hardware-support/parrot-minidrones.html

[6] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Cham, Switzerland: Springer, 2022.