

# Autonomous Line Following and Precision Landing of a Parrot Mambo Minidrone Using Simulink

Anushka Satav G<sup>1</sup>

<sup>1</sup>Masters Student in Robotics and Autonomous Systems (AI), Arizona State University, Tempe, AZ, USA

---

## Keywords:

Parrot Mambo  
MATLAB  
Color Threshold  
Color Detection  
Real-time Image Processing  
Autonomous UAV Navigation  
Line Following Algorithm  
Path Tracking  
Computer Vision  
RGB Line Detection  
Flight Control System  
Drone Soft Landing  
Matlab Mini-drone competition

---

## ABSTRACT

Unmanned Aerial Vehicles (UAVs), such as the Parrot Mambo Minidrone, play a crucial role in autonomous navigation and real-time image processing applications. This project focuses on developing an autonomous line-following and landing system for the Parrot Mambo drone using MATLAB and Simulink. The objective is to enable the **drone to take off autonomously, track a predefined Red, Green, or Blue (R/G/B) line, and land precisely at the designated endpoint**. Real-time image data from the drone's onboard camera is processed using color detection and thresholding techniques to accurately follow the target line. To ensure robust tracking under varying lighting conditions, adaptive thresholding and filtering operations are applied to minimize noise and enhance detection accuracy. The system integrates computer vision with UAV flight control logic, enabling smooth and stable navigation along the path. This work demonstrates the effectiveness of Simulink-based UAV control in real-world applications, offering potential for further advancements such as adaptive path planning and reinforcement learning-based navigation for more complex environments.

---

## Corresponding Author:

Anushka Gangadhar Satav  
Arizona State University, Tempe, Arizona, USA  
Email: [anushka.satav@asu.edu](mailto:anushka.satav@asu.edu)

---

## 1. INTRODUCTION

Drones have gained significant attention in recent years due to their versatility and applications in various fields, including aerial photography, surveillance, and automation. The Parrot Mambo, a compact and beginner-friendly drone, serves as an excellent platform for learning and experimentation. With its downward-facing camera, onboard sensors, and stable flight capabilities, it provides a foundation for developing computer vision algorithms. This project focuses on utilizing the Parrot Mambo drone for color-based object detection and controlled landing, demonstrating its potential for vision-based tasks.

This project focuses on programming the Parrot Mambo drone to autonomously follow a predefined Red, Green, or Blue (R/G/B) line and land precisely at the designated endpoint which is a circle. Using MATLAB and Simulink, real-time image processing techniques are applied to detect and track the target line while ensuring stable flight control. The integration of color segmentation, thresholding, and morphological operations enhances detection accuracy under varying lighting conditions.

Building upon the logic developed in previous lab exercises, the drone autonomously takes off, processes camera data to identify and follow the path, and executes a controlled landing sequence. This approach eliminates the need for manual intervention, making it a step toward fully autonomous UAV operations. The project demonstrates the effectiveness of Simulink-based UAV control and real-time computer vision techniques in robotics applications, providing a foundation for further advancements in autonomous aerial navigation.

## 2. METHOD

This section details the systematic approach followed to design and implement an autonomous line-following and landing system for the Parrot Mambo Minidrone using MATLAB and Simulink. The goal was to enable the drone to autonomously take off, detect and follow a predefined Red, Green, or Blue (R/G/B)

line, and execute a controlled landing at the endpoint. The methodology incorporated hardware setup, image processing using real-time data, flight control logic enhancements, and rigorous simulation and field testing.

The foundation of the system was a pre-existing Simulink template from the MATLAB Parrot Minidrone support package. It was extended with custom logic for color detection, visual navigation, and autonomous state-based decision-making using Stateflow. The following steps describe the workflow:

1. **Drone Setup and Configuration** – The Parrot Mambo Minidrone was configured using the MathWorks Simulink Support Package. Initial manual tests ensured stable wireless communication, proper take-off and landing, and image capture functionality. The drone's basic control behavior was validated before proceeding to automation.
2. **Image Processing for Line Detection** – The drone's downward-facing camera captured real-time image frames that were processed to identify colored lines using the Color Thresholder App in MATLAB. Threshold values for Red, Green, and Blue were extracted and implemented in Simulink as MATLAB Function blocks. The processed image was split into left and right regions, and region-wise color density was analyzed using matrix operations. Morphological filtering and blob detection blocks enhanced noise immunity. A logic signal (LAND\_LOGIC) was also generated based on bounding box symmetry, indicating readiness to land.
3. **Autonomous Line Tracking** – The output from the image processing block—indicating whether the path veered left, right, or was centered—was used to generate flight adjustments. A dedicated Stateflow chart (Chart1) translated these signals into steering commands, with logic to veer left, right, or continue forward. This enabled continuous trajectory correction, ensuring the drone remained aligned with the line throughout the flight.
4. **Autonomous Landing Mechanism** – Landing was initiated upon detecting a predefined stopping condition—either a secondary color (e.g., red) or bounding box symmetry implying the drone had reached the line's endpoint. A higher-level Stateflow chart governed this behavior. When triggered, the drone entered the LAND state, where vertical descent (zout) was gradually reduced for a soft landing. Transitions between Hover → Cruise → Rotation → LAND ensured stability at every phase.
5. **Simulink Model Integration and Testing** – The final system was assembled in Simulink by integrating image processing, flight control, Stateflow logic, and path planning subsystems. The model was tested in simulation using 3D animation blocks and later deployed on the actual hardware. Parameters were fine-tuned based on observations to improve trajectory following, descent rate, and real-time responsiveness.

## 2.1 Drone Setup and Configuration

The experiment began with the setup and configuration of the Parrot Mambo Minidrone to ensure stable flight and reliable communication with MATLAB and Simulink. The process started by connecting the Parrot Mambo Minidrone to MATLAB through the Parrot Minidrone Support Package, establishing a seamless interface for control and data exchange. Initial flight control tests were then conducted in a controlled indoor environment to verify the drone's basic functionality, including take-off, hovering, and landing. These tests confirmed that the drone operated as expected under manual commands before advancing to autonomous operations. Next, the Simulink template for the Parrot Minidrone was loaded, providing a foundational framework for integrating image processing and flight control algorithms. To prepare for vision-based navigation, the drone's downward-facing camera was calibrated to optimize image capture conditions, adjusting for factors such as lighting variations to ensure consistent performance during operation.



Figure 1. Parrot Mambo Minidrone used in this project

## 2.2 Image Processing for Line Detection

The development of robust color-based navigation relied on real-time image processing of frames captured by the Parrot Mambo drone's downward-facing camera. MATLAB Simulink-Image Processing System block was used to extract the position and orientation of the target R/G/B path. A customized image processing pipeline was developed using the MATLAB Color Thresholder App and integrated into Simulink as a functional subsystem (see Figure. 2).

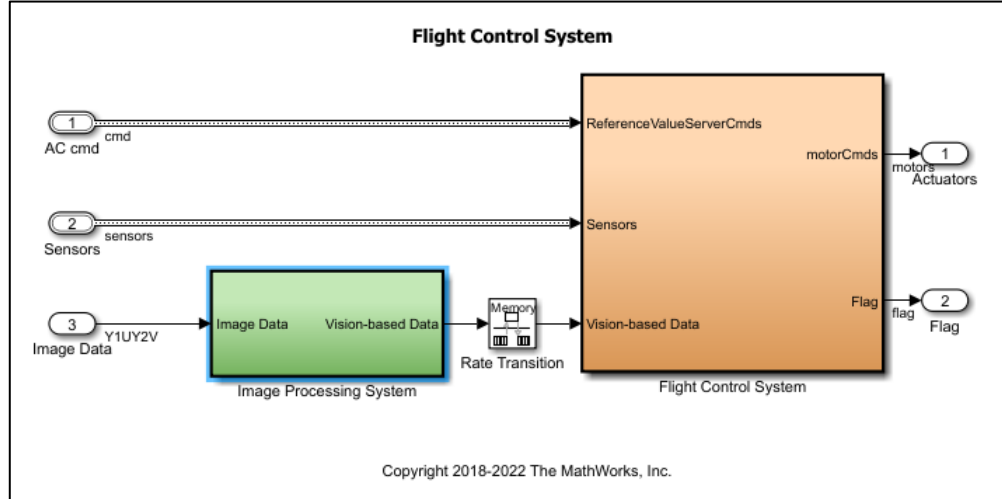


Figure 2. Simulink Top-Level Model and Image Processing System Block

To enable the drone to identify colored blocks (Red, Green, Blue, and Yellow), an image processing pipeline was developed in MATLAB. Real-time video frames from the drone's camera were captured and processed using the following steps:

- **Color Segmentation:** The MATLAB Color Thresholder App was used to define threshold ranges in the RGB color space for Red and Green. Each range was tuned manually to ensure consistent detection under indoor lighting. Corresponding MATLAB functions were exported and embedded into Simulink blocks to detect these colors in real-time (see Figure.3 and 4).

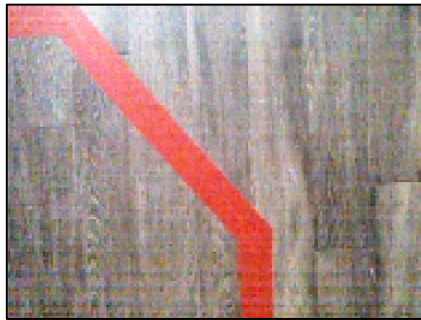


Figure 3. Test image for creating colour detecting functions

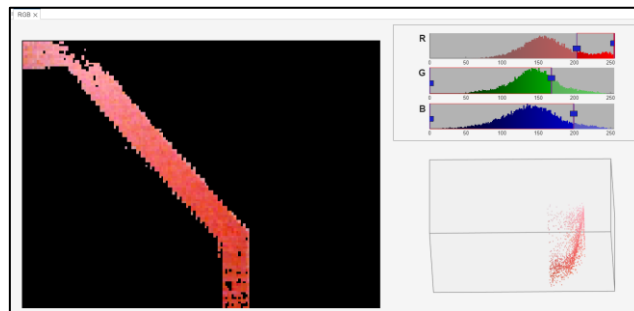


Figure 4. Color Thresholder App for Red Path Detection

- **Mask Generation and Filtering:** Each input video frame was converted into a binary mask where pixels matching the color threshold were marked white (1) and all others black (0). The resulting binary image was passed through a **median filter** to eliminate noise, and **blob analysis** was used to extract **bounding box coordinates** and the **number of blobs** of the detected region (see Figure. 5).

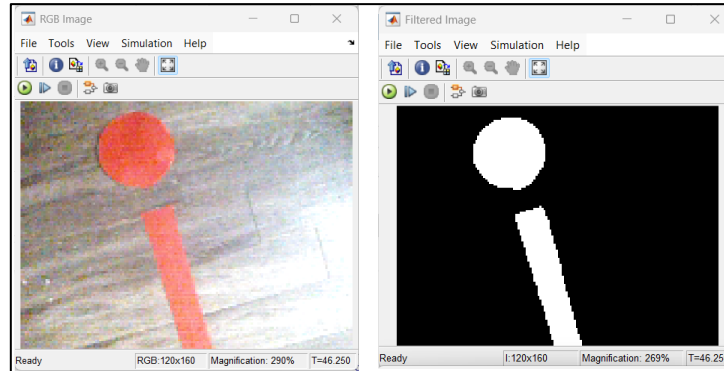


Figure 5. Original RGB Image and Median-Filtered Binary Mask

- **Left/Right Region Comparison:** The binary mask was divided into left and right halves. Matrix sum operations on each side calculated the density of the detected color. This comparison generated directional cues for the drone—indicating whether the line was veering left, right, or remained centered.
- **Landing Condition Logic:** For landing detection, bounding box symmetry was used. If the width of blobs on both sides was approximately equal (difference < threshold), it indicated that the drone was centered above a colored region. This condition was used to set a Boolean flag (LAND\_LOGIC), which served as a trigger for the autonomous landing state in the control system.
- **Simulink Integration:** The entire pipeline was encapsulated in an **Image Processing System** block in Simulink (see Figure. 7). The output from this block included:
  1. A Boolean value indicating color detection.
  2. Centroid data to support navigation alignment.
  3. The LAND\_LOGIC flag used to transition to the landing phase.

This setup ensured that the drone could reliably follow color-coded paths while dynamically identifying when to initiate the descent sequence.

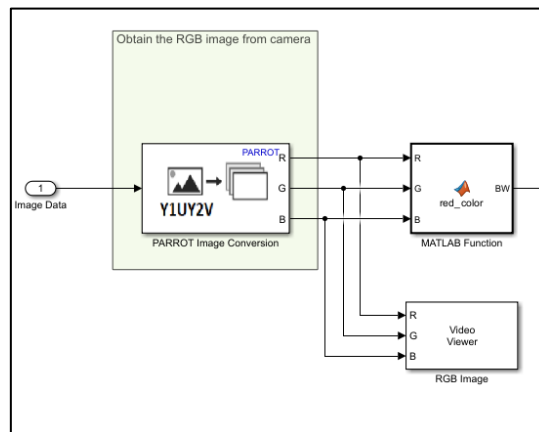


Figure 6. Color Detection Logic Blocks in Simulink

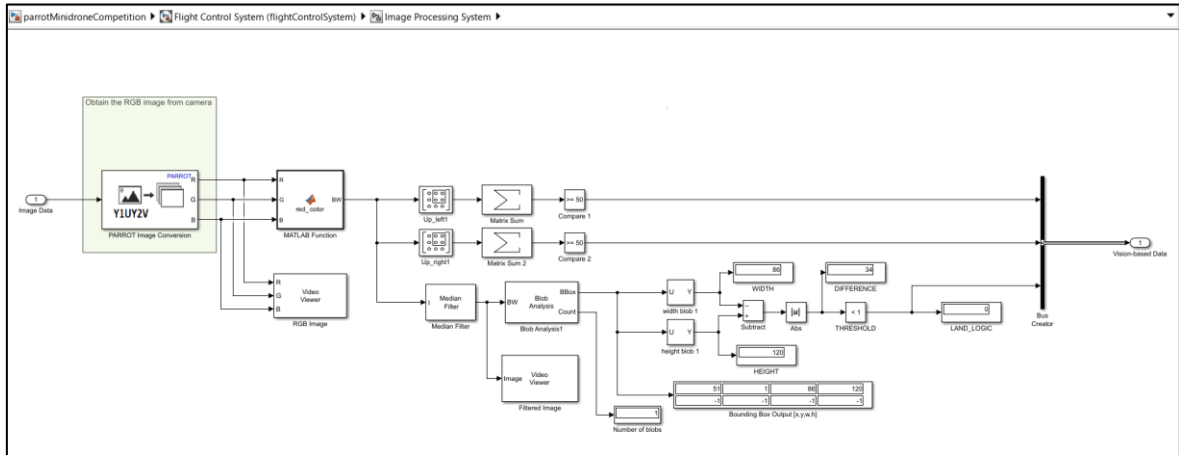


Figure 7. Image Processing Subsystem with Color Segmentation and Logic

### 2.3 Autonomous Line Tracking

Once the drone took off and detected the colored line using its downward-facing camera, the next task was to maintain its trajectory along the path. To accomplish this, a closed-loop visual servoing strategy was implemented using **real-time image feedback and Stateflow-based steering logic**. The binary mask generated by the image processing block was divided into left and right regions. A matrix sum operation calculated the number of white pixels (indicating the detected line) in each half. These sums were compared against a predefined threshold to determine if the line appeared more to the left or right of the center. This directional information was passed into a **Stateflow chart** (see Figure. 8) that acted as the drone's steering controller.

The logic within the chart followed this structure:

```

if Left == true
    OUT = -1; % Turn Left
elseif Right == true
    OUT = 1; % Turn Right
else
    OUT = 0; % Go Straight
end

```

The output (OUT) from this logic was used to modify the reference yaw input in the path planning block. This ensured that the drone continuously adjusted its heading to remain centered on the detected path. The implementation allowed the drone to make smooth directional adjustments in response to deviations in the path while maintaining a stable forward velocity. This setup formed the core of the line-following behavior, allowing the system to autonomously navigate along red, green, or blue lines of varying orientations.

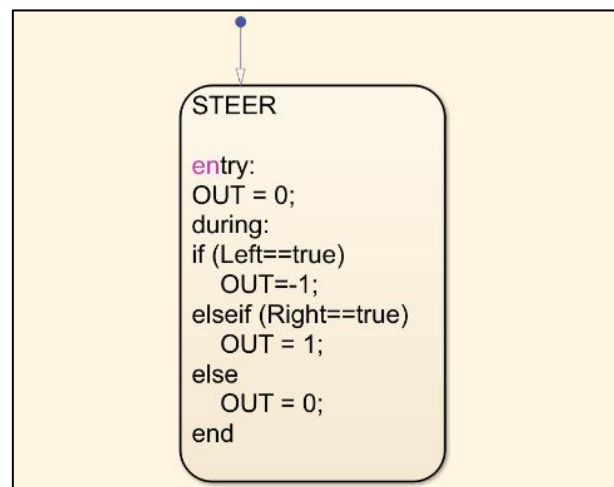


Figure 8. Stateflow Chart for Line Following

## 2.4 Autonomous Landing Mechanism

The final phase of the mission involved designing a fully autonomous and precise landing sequence. To achieve this, a vision-based strategy was developed using image feedback and a dedicated Stateflow controller.

### Landing Trigger Logic:

Landing was initiated based on visual confirmation of a predefined condition: symmetry in the bounding box width of the detected colored path. The assumption was that when the line appears centered directly beneath the drone, the left and right image regions would contain approximately equal color densities. The logic was implemented as follows:

- The difference between the widths of the detected blobs on the left and right halves of the image was continuously monitored.
- When this difference remained below a threshold for a consistent duration, a Boolean flag LAND\_LOGIC was activated.
- This signal was then passed into the navigation system to trigger descent.

This method proved effective in detecting the landing zone without relying on explicit secondary colors or markers.

### Stateflow-Based Landing Control:

A dedicated Stateflow chart (Chart) was used to manage flight states. The high-level control behavior included:

- **Hover:** Default idle state during take-off and initialization.
- **Cruise:** Active forward motion and yaw adjustment during line following.
- **Rotation:** Corrective orientation if the drone strays from the path.
- **LAND:** Triggered once LAND\_LOGIC becomes true.

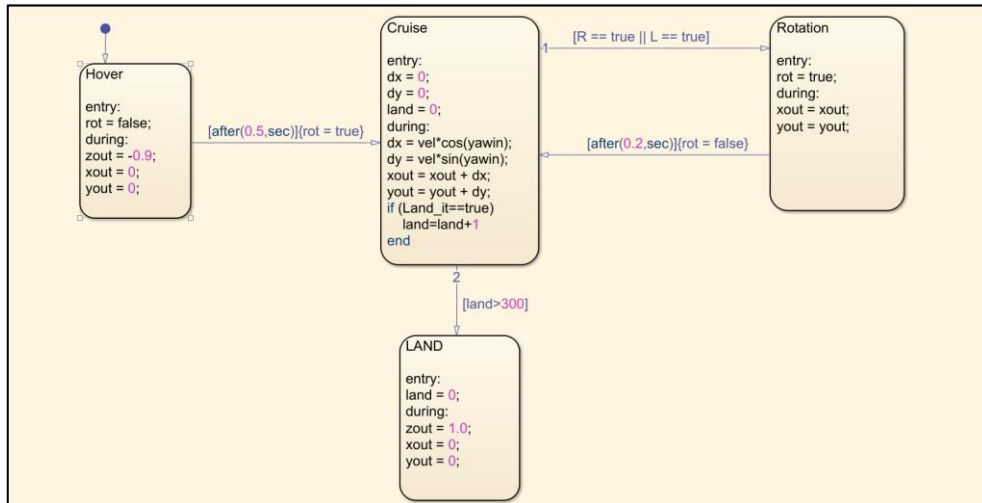


Figure 9. High-Level Stateflow Controller for Navigation and Landing

Upon entering the LAND state, the drone's vertical velocity (zout) was gradually reduced in a controlled descent. The chart ensured smooth transitions between states and prevented erratic switching by introducing hysteresis and counters for condition persistence.

### Simulink Integration:

The landing logic was tightly integrated into the path planning subsystem (see Figure. 9). The LAND\_LOGIC output from the image processing block was routed to the Stateflow chart, which coordinated all transitions. The chart then issued velocity commands to the motor control block to execute the descent. This approach resulted in consistently smooth landings, with minimal bouncing or overshoot, validating the effectiveness of vision-driven autonomy in a constrained environment.

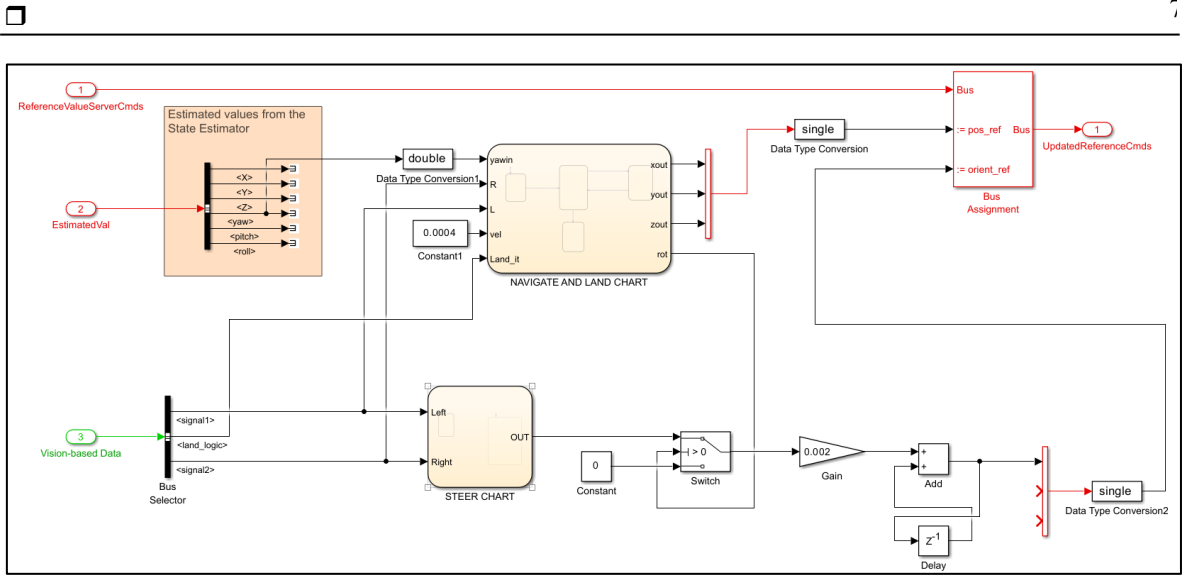


Figure 10. Path Planning Subsystem with Landing Trigger Integration

## 2.5 Simulink Model Integration and Testing

To validate the performance and reliability of the autonomous system, all individual components were integrated into a unified Simulink model. This included the image processing subsystem, path planning logic, flight control algorithms, and Stateflow-based state machines for navigation and landing. The final model featured a modular structure, making it easy to debug and update individual blocks during iterative development. Each subsystem communicated through structured data lines, ensuring synchronization between perception and actuation layers.

### Simulation Testing:

Before deploying the model on the physical drone, extensive testing was performed using the Simulink 3D Animation environment. This allowed for validation of navigation and descent logic without hardware risks. Simulated test cases included:

- Tracking different colored paths (R/G/B)
- Verifying turn logic and line centering
- Ensuring smooth state transitions in the landing state machine

Tuning of controller gains, thresholds, and image segmentation values was done iteratively during this phase.

### Hardware Deployment:

After successful simulation, the model was deployed onto the Parrot Mambo Minidrone using the auto-code generation feature in Simulink. Field testing was performed in a controlled indoor environment with printed colored lines.

- The drone exhibited stable trajectory tracking with real-time vision feedback.
- The landing sequence was initiated correctly when the end condition was visually detected.
- Performance was robust across different lighting conditions with minor tuning.

Overall, the system consistently achieved smooth line following and autonomous soft landing, validating the design approach and demonstrating the effectiveness of the Simulink-based development pipeline.



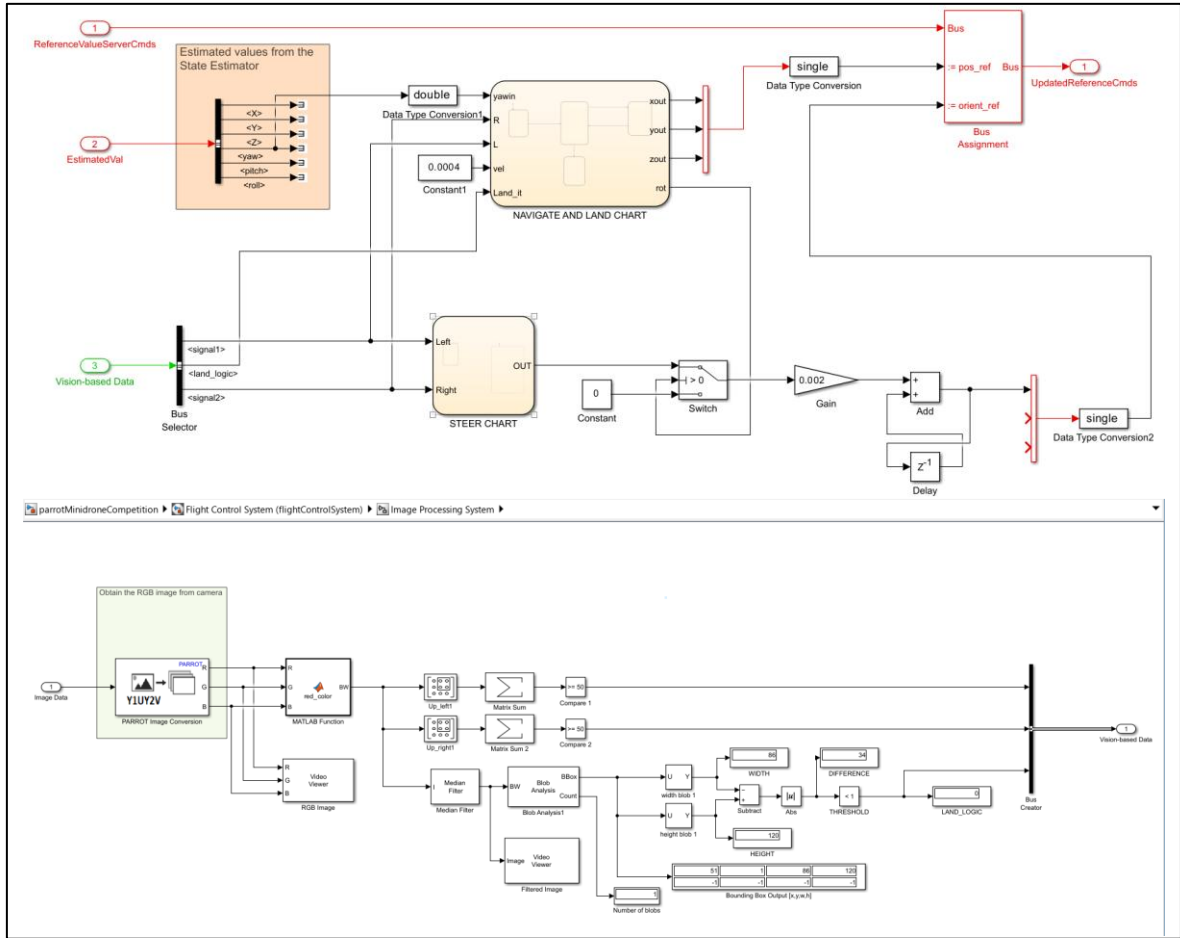


Figure 11. Final Integrated Simulink Model showing Image Processing and Navigation Algorithm

### 3. RESULTS AND DISCUSSION

This section presents the outcomes of the autonomous line-following and landing system tested on the Parrot Mambo Minidrone. Both simulation and real-world experiments were conducted to evaluate the system's effectiveness in color detection, path tracking, stability during navigation, and precision landing. The results highlight the performance of individual subsystems and the overall integration of vision-based control within a model-based Simulink framework.

#### 3.1. Line Detection and Tracking Performance

The drone's onboard camera successfully detected and followed Red, Green, and Blue (R/G/B) lines using the color segmentation functions developed in MATLAB. Under stable indoor lighting, the binary mask generation and centroid extraction performed with high accuracy, allowing for precise line localization.

The split-region analysis (left vs. right) was effective in generating steering cues. False detections were minimal and were further suppressed by applying median filtering and blob analysis to clean up the binary mask. The modular design of the image processing block enabled quick tuning for different color paths during testing.

Minor performance degradation was observed in dimly lit conditions or when shadows appeared on the path. However, these were mitigated by threshold recalibration and histogram-based contrast adjustments.



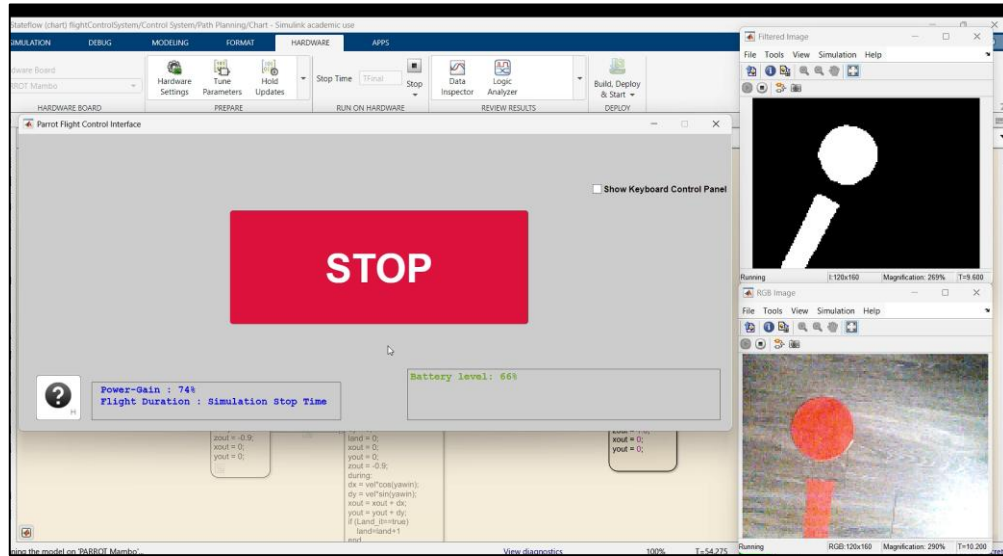


Figure 12. Comparison of Original RGB Frame and Corresponding Binary Mask for Red Line Detection

### 3.2. Flight Stability and Control

The drone demonstrated stable and responsive behavior while following the path. The visual servoing logic—implemented through the Stateflow chart (Chart1)—allowed the drone to continuously correct its heading in response to deviations from the centerline.

The turning decisions based on left/right region analysis were smooth and timely. When veering off-track, the drone would autonomously rotate to realign itself before proceeding forward. PID-based velocity control ensured that turns were not too aggressive, contributing to stable flight.

In simulation as well as in real-world trials, the drone was able to maintain consistent alignment with the colored path, even when curves or directional shifts were present.

### 3.3. Autonomous Landing Accuracy

The autonomous landing system was triggered successfully using the LAND\_LOGIC condition derived from bounding box symmetry. When the path appeared centered beneath the drone, a smooth descent was initiated.

The Stateflow-based high-level controller transitioned the drone from cruising to landing mode without abrupt changes in altitude. Controlled reduction of the vertical velocity ( $z_{out}$ ) ensured soft touchdowns with minimal rebound or instability.

Multiple real-world tests confirmed the reliability of the landing logic, with the drone consistently landing within proximity of the intended endpoint. The system demonstrated robustness and repeatability across different path lengths and lighting variations.

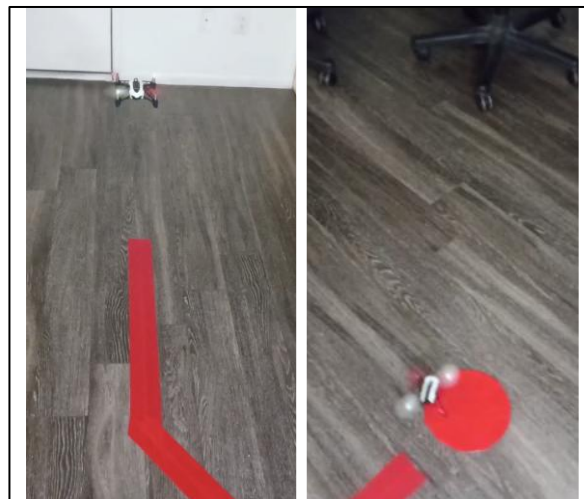


Figure 1. Testing Final Model on Parrot Mini Drone (after landing)

#### 4. CONCLUSION

This project successfully demonstrated the design and implementation of an autonomous line-following and landing system for the Parrot Mambo Minidrone using MATLAB Simulink. By building upon foundational elements from earlier lab exercises, the system was enhanced to achieve fully autonomous navigation tasks, including take-off, R/G/B line tracking, and precision landing.

Real-time color detection was implemented using MATLAB's Color Thresholder App, with custom image processing functions integrated into Simulink. A binary mask and centroid-based approach enabled reliable path tracking, while region-wise analysis allowed the drone to interpret directional cues. This facilitated continuous course correction during flight, guided by a vision-based steering logic implemented through Stateflow. Landing was triggered using a robust visual condition based on bounding box symmetry, ensuring the drone was correctly aligned over the endpoint. A high-level Stateflow chart managed transitions between different flight modes, enabling a gradual and stable descent during the landing sequence.


The final integrated system was tested in both simulation and real-world conditions, where it consistently followed the intended path and executed soft landings with high accuracy. The results highlight the effectiveness of combining Simulink's model-based design approach with real-time image processing and state-based control for developing autonomous UAV behaviors.

This lab serves as a strong foundation for advancing into more complex missions, such as multi-target navigation or dynamic obstacle handling, and aligns well with the objectives of the MATLAB Minidrone competition.

#### REFERENCES

- [1] <https://www.mathworks.com/help/simulink/setup-and-configuration-parrot.html>
- [2] [https://www.mathworks.com/help/simulink/supportpkg/parrot\\_ref/color-detection-and-landing-parrot-example.html](https://www.mathworks.com/help/simulink/supportpkg/parrot_ref/color-detection-and-landing-parrot-example.html)
- [3] [https://www.mathworks.com/help/simulink/supportpkg/parrot\\_ref/getting-started-with-parrot-minidrone-vision.html](https://www.mathworks.com/help/simulink/supportpkg/parrot_ref/getting-started-with-parrot-minidrone-vision.html)
- [4] <https://youtu.be/UIDknqrAHM?feature=shared>
- [5] [https://www.mathworks.com/help/simulink/supportpkg/parrot\\_ref/path-planning-keyboard-example.html](https://www.mathworks.com/help/simulink/supportpkg/parrot_ref/path-planning-keyboard-example.html)
- [6] <https://www.mathworks.com/academia/students/competitions/minidrones/global-drone-student-challenge.html>
- [7] <https://www.mathworks.com/help/images/ref/colorthresholder-app.html>
- [8] <https://www.mathworks.com/products/3d-animation.html>
- [9] <https://www.mathworks.com/academia/student-competitions/tutorials-videos.html>

#### BIOGRAPHIES OF AUTHORS

	<p><b>Anushka Satav</b> is a master's student at Arizona State University Pursuing Robotics and Autonomous Systems (AI). She has completed her Bachelor of Technology in Robotics and Automation at MIT World Peace University, Pune, India. She can be contacted at email: <a href="mailto:anushka.satav@asu.edu">anushka.satav@asu.edu</a></p>
---	--