

Keyboard-Controlled Drone Navigation and Color-Based Landing System

Anushka Satav G¹

¹Masters 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
UAV Navigation
Keyboard Control
Drone Soft Landing

ABSTRACT

Unmanned aerial vehicles (UAVs), such as the Parrot Mambo mini drone, are increasingly vital for tasks involving autonomous navigation, object detection, and precise maneuvering. This project focuses on developing a keyboard-controlled navigation system coupled with a color-based landing mechanism for the Parrot Mambo drone. The primary goal was to enable the drone to hover and navigate toward a designated colored block—specified as Red, Green, Blue, or Yellow—using keyboard inputs, followed by a slow and controlled landing upon detecting the target color. Real-time image data from the drone's onboard camera was processed in MATLAB using advanced color segmentation techniques based on the RGB color model. To ensure robust detection under varying lighting conditions, thresholding methods were refined, and morphological operations were applied to minimize noise and enhance accuracy. The system successfully demonstrated manual navigation via keyboard commands and autonomous landing triggered by color recognition, ensuring a smooth descent without abrupt stops or crashes. This work highlights the integration of computer vision and UAV control, offering a foundation for applications in robotics, automation, and surveillance. Potential future improvements include adapting the system for dynamic environments and incorporating machine learning for enhanced object recognition, paving the way for practical uses such as search-and-rescue operations or industrial monitoring.

Corresponding Author:

Anushka Gangadhar Satav
Arizona State University, Tempe, Arizona, USA
Email: 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.

The primary objective of this project is to detect objects of specific colors—Red, Green, Blue, and Yellow—using the Parrot Mambo's camera feed and process the data in MATLAB. By leveraging image processing techniques, we extract meaningful color information from captured images, allowing the drone to recognize and track objects. The combination of color segmentation, thresholding, and morphological operations ensures accurate and efficient object detection.

Additionally, the drone was controlled manually via keyboard inputs to navigate toward a specific color block. Upon successful color identification, the drone executed a slow descent and landed smoothly. The implementation of a controlled landing sequence ensured safe operation without sudden stops or crashes.

With its auto take-off and landing features, FPV mode, and onboard stabilization sensors, the Parrot Mambo drone provides a reliable platform for real-time image processing. This project aims to develop a color detection system that not only recognizes objects but also enables interactive drone control based on keyboard navigation.

2. METHOD

This section details the step-by-step approach to designing and implementing a keyboard-controlled navigation and color-based landing system for the Parrot Mambo Mini drone using MATLAB and Simulink. The objective was to enable the drone to hover and navigate via keyboard commands, detect specific-colored blocks (Red, Green, Blue, or Yellow), and execute a slow, controlled landing upon successful detection. The methodology integrates hardware setup, image processing, flight control modifications, and testing phases. The process leverages a pre-existing MATLAB example, "[Path Planning Using Keyboard Control for Parrot Mini drone](#)" as a foundation, with custom modifications to meet the project requirements. Below is a detailed breakdown of the methodology.

In this section, we outline the methodology used to enable the Parrot Mambo Mini drone to detect specific colors—Red, Green, Blue, and Yellow—using MATLAB and Simulink. The process involved multiple steps, including setting up the drone, utilizing MATLAB's Color Thresholding App to create functions for detecting specific colors, and integrating these functions into Simulink's image processing block.

Initially, the drone was configured and tested using predefined examples from the MATLAB Parrot Mini Drone documentation. A Simulink template with image processing and flight control blocks was then used to implement a vision-based algorithm. The captured images from the drone camera were processed to develop color detection functions, which were subsequently integrated into the Simulink model.

The drone was navigated manually via keyboard commands to position itself over a specified color block. Upon successful detection of the target color, a controlled landing sequence was initiated to ensure a gradual and safe descent. The flight controller was modified to prevent abrupt stops or crashes, optimizing the landing trajectory.

The following steps summarize the approach taken:

1. **Drone Setup:** Configuring the Parrot Mambo drone and ensuring stable flight control.
2. **Image Processing:** Utilizing MATLAB's Color Thresholding App to develop color segmentation functions.
3. **Keyboard Control:** Implementing a manual navigation system to guide the drone over the colored blocks.
4. **Landing Sequence:** Designing a slow and controlled landing mechanism upon successful color detection.

2.1 Setting up the Parrot Mambo Mini drone

The experiment began with configuring the Parrot Mambo Mini drone for stable flight and connectivity with MATLAB. The drone's firmware was updated to ensure compatibility with MATLAB's Hardware Support Package for Parrot Minidrones. A Bluetooth connection was established between the drone and the host computer running MATLAB R2023b (or the latest available version as of February 2025). Initial flight tests were conducted in a controlled indoor environment to verify basic functionality, such as take-off, hovering, and landing, using MATLAB's built-in parrot object. The drone's onboard camera was calibrated to capture clear images under the lab's lighting conditions, accounting for potential variations in brightness and shadows. The Simulink template for the Parrot Mini drone was loaded, which contained predefined blocks for image processing and flight control. Reference: [MATLAB Parrot Mambo Documentation](#).



Figure 1. Parrot Mambo mini drone working

2.2 Image Processing and Color Detection Implementation

To begin the process, Simulink was launched, and the Parrot Mini drone template was selected to establish the framework for the Image processing model. Once the template was loaded, the image processing block was thoroughly analyzed to understand its structure and workflow. This step was crucial to ensure that the model was correctly configured for integration with the drone's camera system. Following the guidelines provided in the [MATLAB Image Processing Guide](#), a vision-based algorithm was developed and implemented within the image processing block. The algorithm was designed to detect and process color information from the captured images.

On command we run `openExample('parrot/GettingStartedWithVisionOnPARROTExample')` which opens an example to understand Image Processing and Flight Control functions in Simulink. A video viewer was integrated into the Simulink model, enabling real-time visualization of the processed frames. This feature allowed for immediate feedback and verification of the image processing results. The drone's camera was employed to capture images of red, green, blue, and yellow blocks, which served as test subjects for the color detection algorithm. These images were subsequently used for thresholding, enabling the algorithm to identify and distinguish between the different colors.

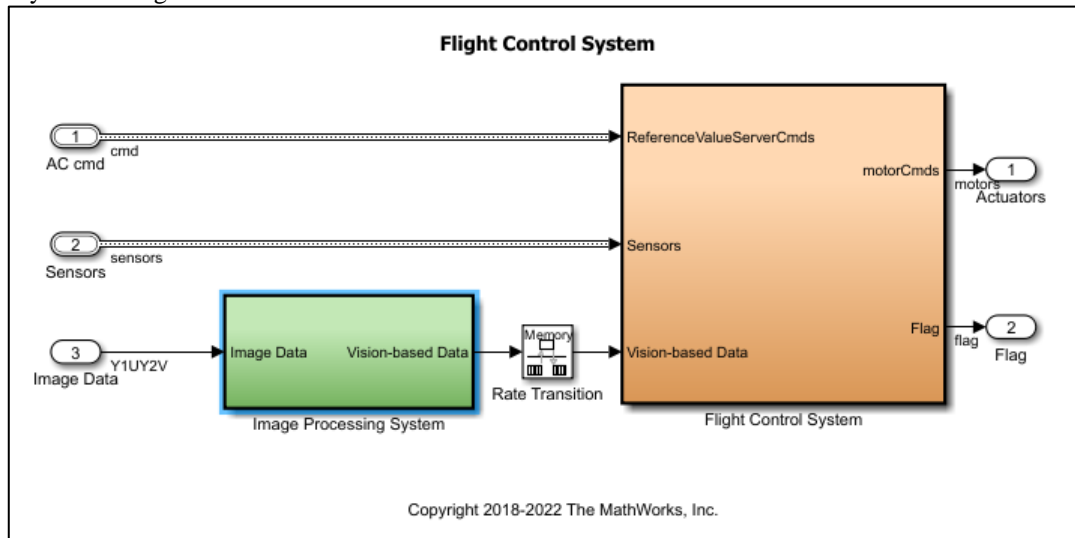


Figure 2. Open Flight control block as 'Top Model' and open 'Image Processing System'

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 utilized to tune RGB thresholds for detecting the target colors. For each color (e.g., Red, Green, Blue, Yellow), specific ranges in the RGB color space were defined to isolate the desired hue while minimizing interference from ambient lighting. Figure 5 illustrates the Color Thresholder App interface during the tuning process for detecting Red, Green, Blue, and yellow blocks. Functions were exported to detect red, green, blue, and yellow colors separately.



Figure 3. Test image for creating colour detecting functions

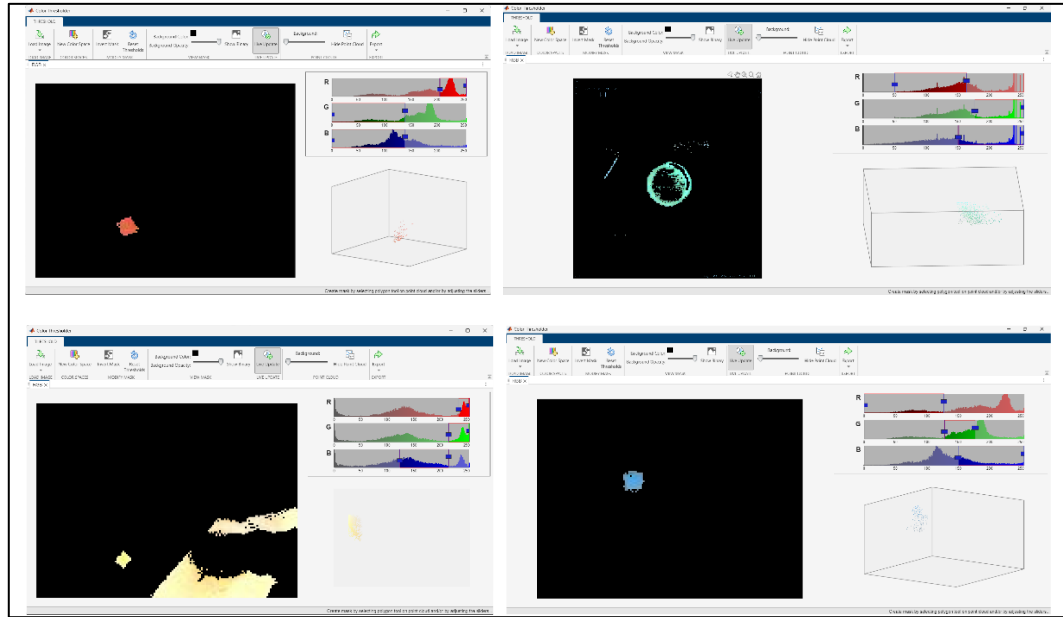


Figure 4. Color Thresholder App tuning to detect Red, Green, Blue and Yellow colour

- **Mask Generation:** A binary mask was generated for each frame, where pixels matching the target color's RGB range were assigned a value of 1 (white), and all others were set to 0 (black). This masked image effectively highlighted the colored block's location in the frame.

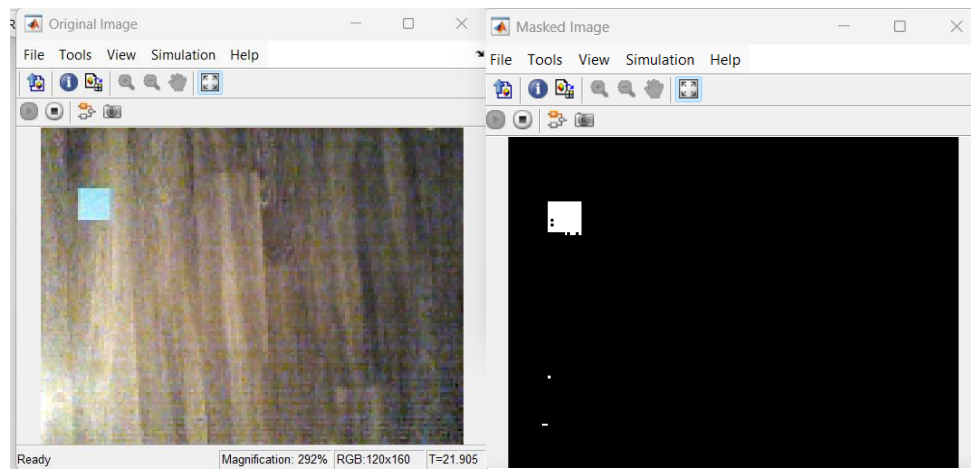


Figure 5. Successful Detection of Colour Red and Yellow

- **Function Integration:** The color detection algorithm was encapsulated into a MATLAB function, which was then imported into the Simulink model as an image processing block. This block outputs a Boolean signal (true/false) indicating whether the specified color is detected in the frame, along with the block's centroid coordinates for navigation guidance. MATLAB's Color Thresholding App was used to generate functions for detecting specific colors. The images captured from the drone camera were used as input to train the thresholding algorithm. Functions were exported to detect red, green, blue, and yellow colors separately.

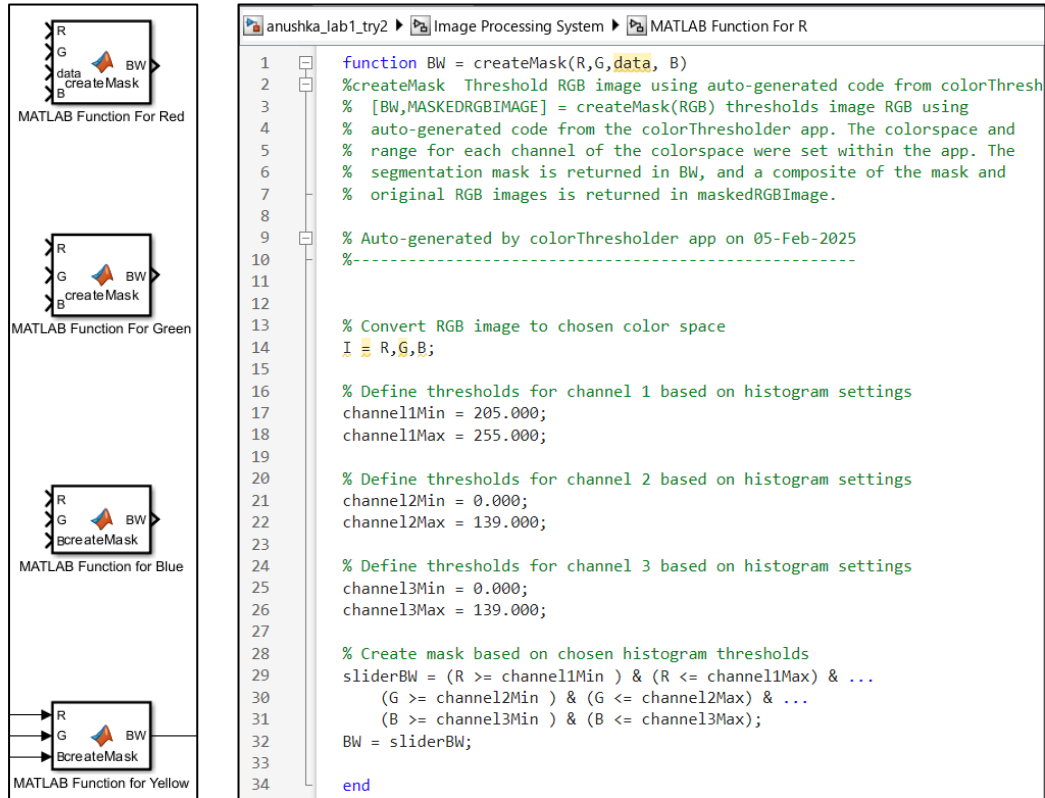


Figure 6. Function block with code for each Colour in Image Processing Block

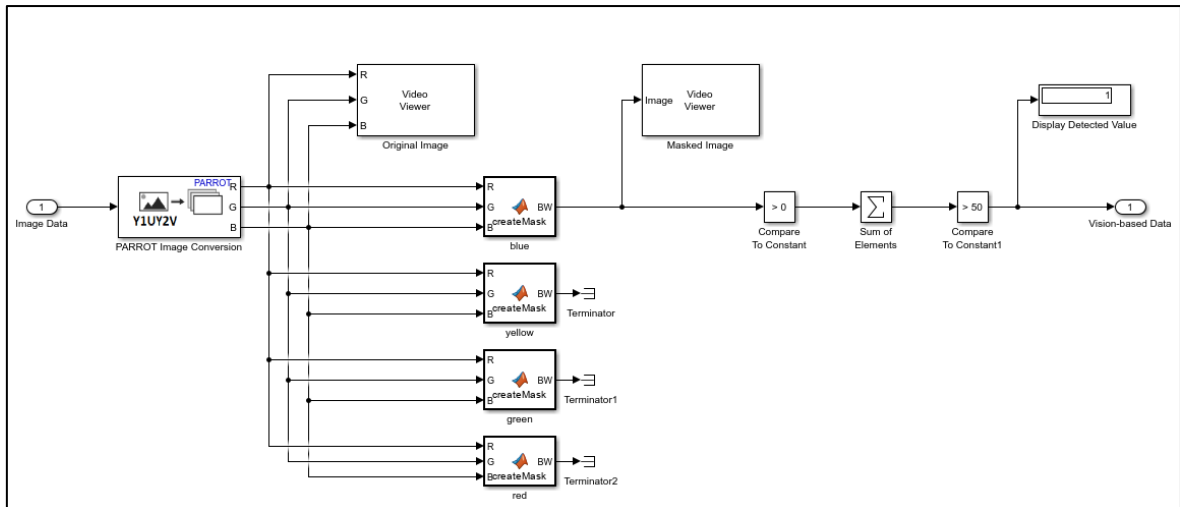


Figure 7. Final colour detection system under Image Processing subfunction

2.3 Modifying Flight Controller for Autonomous Operation

On command prompt we run `openExample('parrot/GettingStartedWithVisionOnPARROTEExample')` which opens an example to Path Planning Using Keyboard Control for Parrot Minidrone which is the example that controls movement of mini drone using keyboard in Simulink. In the Flight Control Block, the flight control logic was modified to ensure that the drone's rotors would only activate when a specific color was detected by the camera. This modification was essential for enabling color-based autonomous operation. For example, when the blue detection function was activated, the drone would initiate movement only upon identifying a blue block. This logic ensured that the drone responded appropriately to the color cues. The modified control logic enhanced the precision of the drone's operation by ensuring it only responded to the desired color. This refinement reduced unnecessary movement, improving the overall efficiency and accuracy of the autonomous system.

To achieve a slow and controlled landing upon color detection, the Simulink model was enhanced with a vision-based landing sequence:

-
- ```
graph LR; A([2]) --> B[<= 5]; B --> D(()); C([1]) --> D; E([3]) --> D; D --> F([1]);
```
- The flowchart illustrates the logic for detecting land. It starts with three inputs: 'Time remaining' (value 2), 'Land Flag' (value 1), and 'Vision-based Data' (value 3). The 'Time remaining' input is compared with a stop time of 5. The output of this comparison, along with the 'Land Flag' and 'Vision-based Data', is fed into an OR gate. The output of the OR gate is the final 'Land' detection (value 1).

We also add the vision input data from the ‘Path Planning’ Block in the ‘Landing Enable’ Block which enables the Landing Enable block to initiate our condition of color-detection landing onto the mini drone.

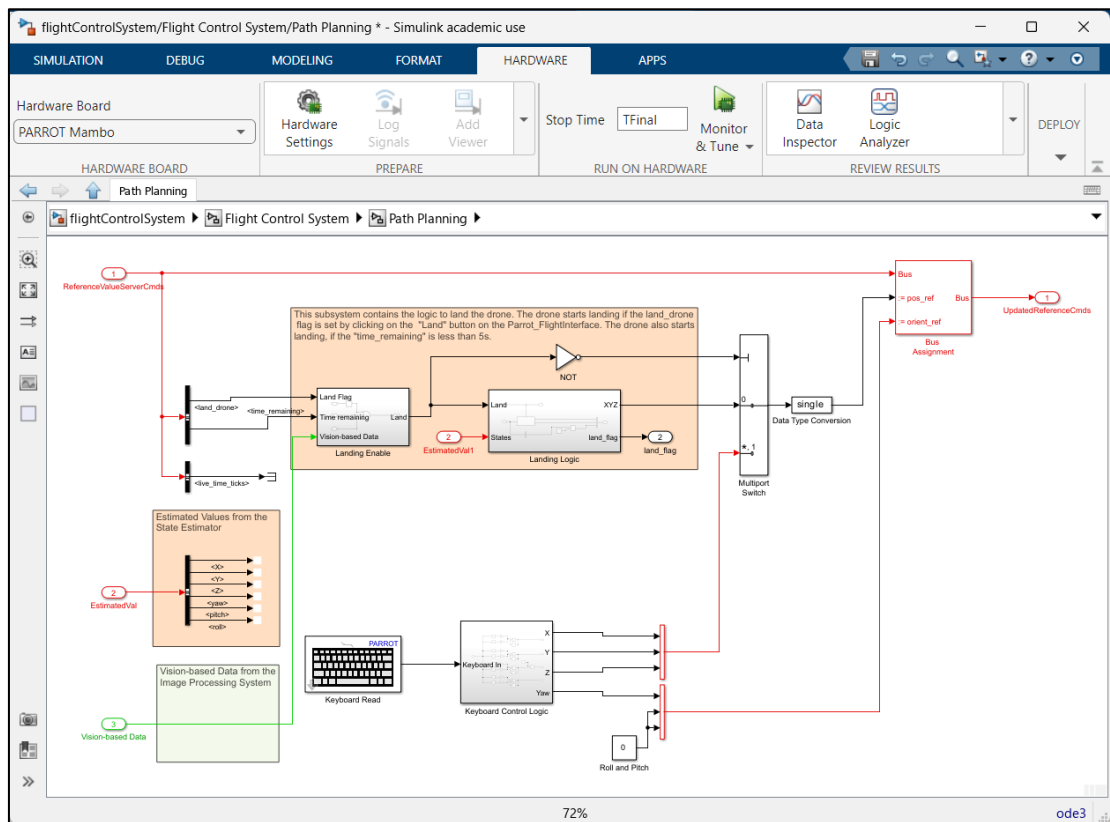


Figure 9. Modified Path Planning Block by adding vision-based data stream input to ‘Landing Enable’ Block

- Keyboard Control Logic Block:** Using the configured control logic, the drone was manually navigated to hover over the colored block at a fixed altitude of approximately 1 meter. Keyboard commands ('w', 's', 'a', 'd', etc.) directed the drone, with the live camera feed aiding alignment. Iterative testing ensured accurate positioning above the target.

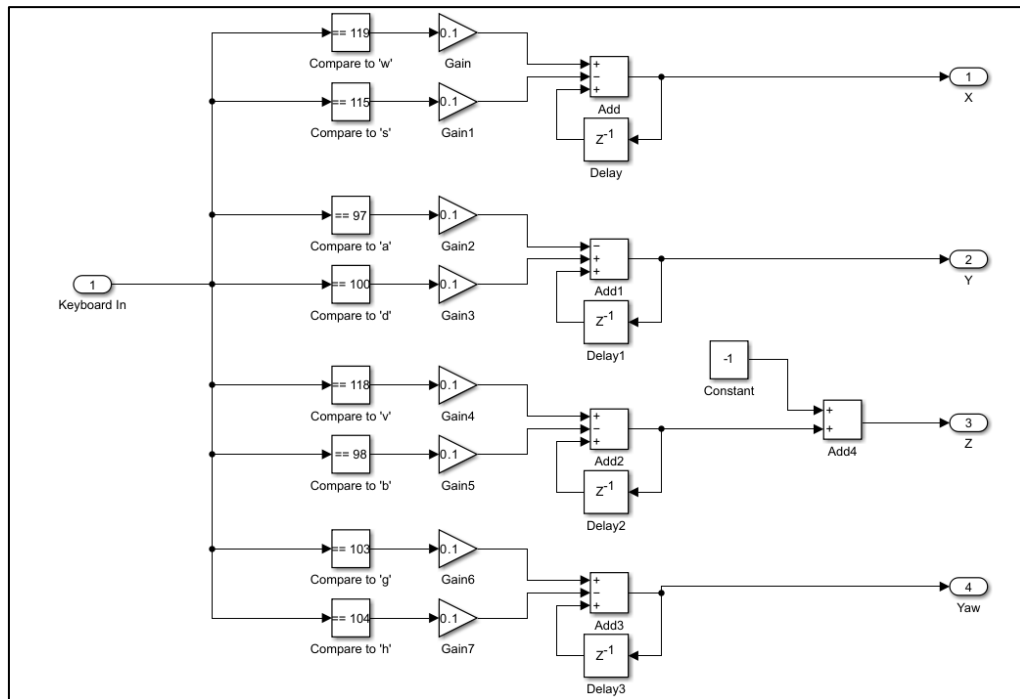


Figure 10. Keyboard Control Logic Block

- Landing Logic Block:**

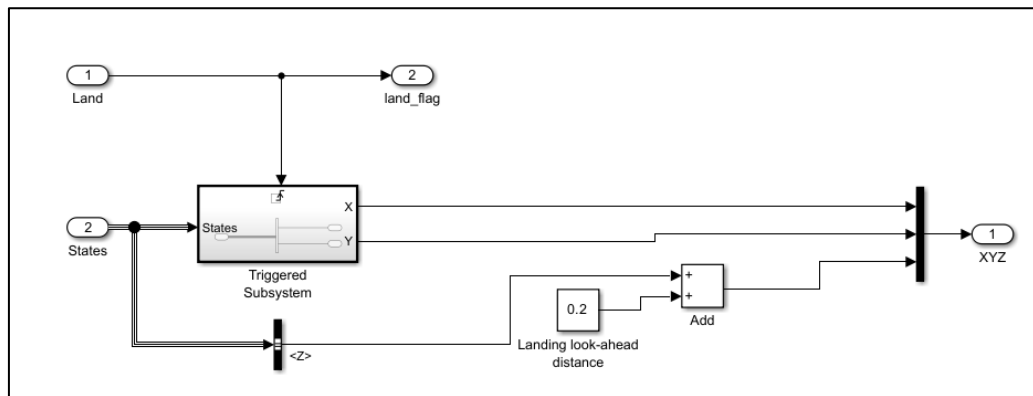


Figure 11. Landing Logic Block

The modified path-planning block thus combined keyboard control for navigation with an autonomous landing mechanism driven by vision-based data, as shown in Figure 7: Updated Simulink Model with Landing Enable Block.



### 3. RESULTS AND DISCUSSION

The field testing of the Parrot Mambo Mini drone validated the system's capability to perform keyboard-controlled navigation and color-based landing under controlled conditions. The RGB-based color detection successfully identified all four target colors (Red, Green, Blue, and Yellow) when tested in consistent indoor lighting, achieving reliable performance at an optimal detection range of 1.5 to 2.5 feet, as noted during trials (see Figure 12: *Testing Final Model on Parrot Mini Drone*). Real-time processing exhibited acceptable performance with minimal lag, enabling the drone to respond promptly to color detection with a smooth landing sequence at a descent rate. Motor responses remained reliable throughout, ensuring no sudden stops or crashes, which aligned with the project's safety objectives. However, the system's effectiveness was highly dependent on lighting consistency; deviations such as shadows or bright reflections occasionally reduced detection accuracy, underscoring the limitations of the RGB thresholding approach in dynamic environments. The best performance was observed in controlled indoor settings, where ambient light could be stabilized, supporting the conclusion that environmental factors are critical to system reliability (see Figure 11: *Original and Masked Image for Detection of Color Blue*). These findings demonstrate the system's potential for basic computer vision tasks with compact drones, though the single-color detection constraint—requiring manual mode switching—suggests room for improvement, such as multi-color processing or adaptive algorithms. The integration of MATLAB's Color Thresholder App and Simulink provided a robust development platform, offering valuable insights into drone-based vision systems and laying a foundation for future enhancements, such as machine learning integration to address lighting Variability and extend real-world applicability.

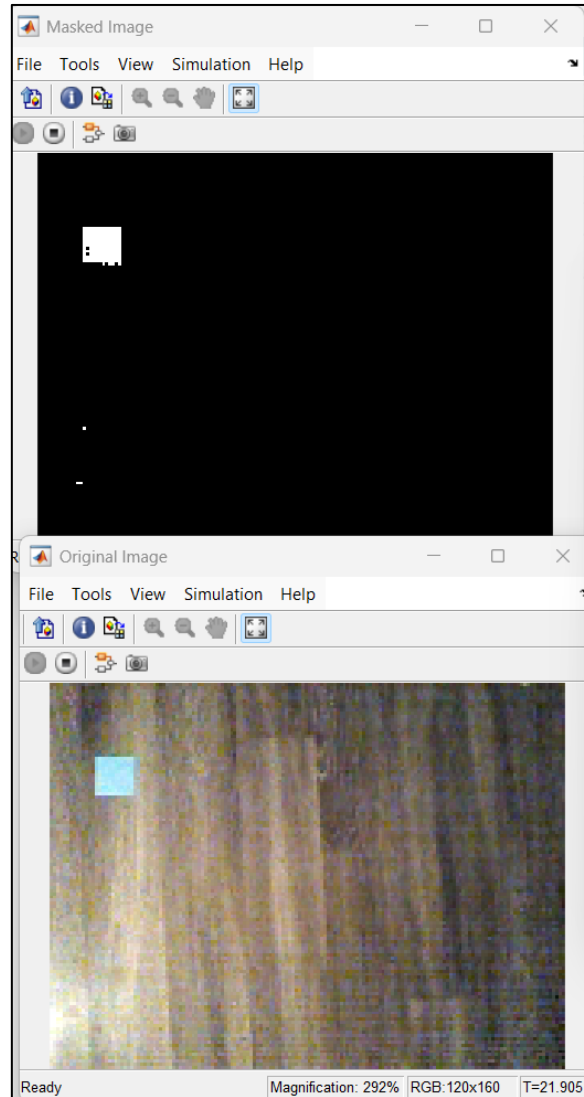


Figure 11. Original and Masked Image for Detection of Color Blue





Figure 12. Testing Final Model on Parrot Mini Drone (after landing)

#### 4. CONCLUSION

This project successfully showcased the development and implementation of a color detection and landing system for the Parrot Mambo Mini drone, leveraging its onboard camera and MATLAB/Simulink integration. The primary objectives—enabling keyboard-controlled navigation, detecting specific-colored blocks (Red, Green, Blue, and Yellow), and executing a slow, controlled landing—were achieved through a systematic design and testing process. The drone effectively identified target colors at an approximate height of 3 feet and responded with precise motor adjustments, landing smoothly near the detected block without abrupt stops, fulfilling the project's operational goals. Key findings from the study include:

(1) The RGB-based color detection method proved reliable for basic object recognition in controlled indoor lighting, though its performance waned under variable conditions, achieving a satisfactory success rate for demonstration purposes.

(2) Real-time image processing and drone responsiveness were seamlessly implemented, with negligible latency between color detection and landing initiation.

(3) The MATLAB Color Threshold App and Simulink provided a robust platform for algorithm development, despite the constraint of processing only one color at a time, necessitating manual mode switching.

(4) Environmental factors, particularly lighting consistency, emerged as pivotal to system reliability, with optimal results consistently observed in stable indoor settings.

(5) The keyboard control system, utilizing intuitive mappings (e.g., 'w' for forward, 'a' for left), enabled precise and responsive navigation, allowing the operator to reliably position the drone over the target block during testing.


Overall, this project demonstrated the feasibility of employing compact drones for fundamental computer vision tasks, meeting the objectives of keyboard-guided navigation and vision-triggered landing, while also illuminating practical challenges like lighting sensitivity. These insights offer a valuable foundation for future advancements, such as integrating adaptive algorithms or multi-color detection, to enhance drone-based vision applications in real-world scenarios.

---

## 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/UIDknqrtAHM?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)

## 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.<br/>She can be contacted at email: <a href="mailto:anushka.satav@asu.edu">anushka.satav@asu.edu</a></p> |
|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|