Final Project Report

ECE 492/592 - (A2) True Love

Jennifer Jimenez, Kaurwaki Babu, Michael Felt

Table of Contents:

I. Abstract

For ECE 492/592 Introduction to Autonomous Drones, we—the TrueLove team—designed an Unmanned Aerial Vehicle (UAV) that uses image processing to locate, pick up, and drop off a letter from one destination to another. The UAV consisted of a SAM4 drone and a payload outfitted with a 3D printing mechanism for pick-up and drop-off, a companion computer (rPi3), lidar, and a camera. Our goal was for the drone to take off, scan the field for a red object (i.e., the letter), center on and advance toward the letter using lidar and color processing, pick up, and then drop off at a separate location marked by a different color or the same color. Due to difficulties with Gazebo and ROS, and windy conditions that narrowed down available field testing days, we could only get our drone to center on a red object, move toward the letter, and drop off the letter separately, however, we struggled with integrating these elements in the real world.

II. Introduction

For this course, the *problem* was outlined in the project goals: to design a UAV that uses image processing to locate, pick up, and drop off a letter from one destination to another.

Outside the scope of this course, advancing and refining image processing and lidar detection capabilities for UAVs could revolutionize 2D mapping, disaster response, search and rescue, and resource delivery. If, for example, individuals are trapped in a remote location following a natural disaster, a UAV similar to ours could identify people and drop off vital resources such as food, water, first aid, and more. More information on the utilization of drones with capabilities similar to those we implemented in our project is located below in *III. Related Works*.

Contributions of Team Members:

- ➔ **Jennifer Jimenez** (Team Leader)**:** Gazebo/ROS testing, Progress Reports, Code (Movement) Testing
- ➔ **Kaurwaki Babu:** Code (Image Processing, lidar, Movement), Progress Reports, Testing
- ➔ **Michael Felt:** Payload design and fabrication, Code (Set up of Companion Computer, Image Processing, Movement), Testing

III. Related Works

While no team in previous years had tackled the TrueLove project, we did find many past groups that implemented functionalities (i.e., image processing) that were critical to achieving our end goal. The Mothership Team (circa Spring 2022), in particular, was a crucial source of inspiration for our payload design. In their project, their carrying mechanism located and picked up a

smaller drone using GPS communication between the larger drone and the smaller drone. While we did not use GPS communication to narrow down the location of our letter, the Mothership Team's pickup mechanism was a great foundation for the fine-tuning of our own pickup/drop-off mechanism.

When brainstorming how our code should be structured, and how the lidar and Camera could be used in tandem for locating the pick-up and drop-off locations, the research paper *Realtime Object's Size Measurement from Distance using OpenCV and lidar* helped us understand the potentialities of image processing with OpenCV and lidar. In the aforementioned paper, researchers proposed using a lidar sensor, a Camera, and the OpenCV Algorithm to estimate the size of an object from a distance. Initially, we hypothesized that this methodology could be used to detect and move toward the object until it was close enough for pickup. While we didn't implement this functionality in our final code, the paper provided helpful insight into the capabilities of OpenCV and showed us how OpenCV functions and classes can be used to achieve our project goals. [1]

A Motion Detection System in Python and OpenCV was more helpful in understanding how background and foreground data could be distinguished using OpenCV, and how these differences help with motion detection and overall image processing. While this paper focused more on tracking a moving object, it provided information regarding camera frame processing and the 'threshold framework' that were integral in our code development. [2]

IV. Approach

A. Payload Design

The first step in our approach was to brainstorm how the project could be implemented and what hardware would be needed to implement said approach. In our initial approach, we planned on putting the letter inside a 3D-printed box and placing the box on a colored tarp that could be identified by a camera. Upon identification, the drone would navigate to the tarp, pick up the box, and drop off the box in a separate location also marked by a colored tarp (same or different color). Identifying the key components of our project deliverables as image processing, pick-up/drop-off, and dynamic movements, we determined we needed to order a camera and lidar as peripherals to a companion computer and develop a pick-up/drop-off mechanism.

After analysis of how past groups implemented similar pick-up/drop-off behavior, we developed three different approaches for our project:

1. 'Hook through Loop' Implementation: This system would consist of a mounting plate, a 'hook' rod, and the servo motor. The servo motor would act as a rack and pinion mechanism and control

the rod vertically, while the metal rod would slip through loops on the 3D-printed letter box for pick-up and drop-off.

2. Electromagnetic Implementation: This system would include an electromagnetic module controlled by the companion computer (rPi4) that would 'attract' (essentially pick up) the letter. In this case, the letter or the 3D-printer box housing the letter would have a thin sheet of iron placed on top, allowing the electromagnetic module to attach itself. During drop-off, the electromagnetic module would be disengaged.
3. Claw Implementation: This system would be a 3D print of a claw with multiple teeth that would open and close along the handle of the letter carrier. It would have a servo motor open when aligning to the handle and closing on the handle to secure it. For this design to work, the camera would need to be mounted differently, looking straight at the ground rather than in front. The drone would need to hover closely above the letter payload for proper pickup.

Due to time and resource constraints, we determined the 'Hook through Loop' Implementation would be more feasible, and would allow us to get a head start, as we could begin working before the necessary additional hardware was ordered and delivered (barring the servo motor). Figure 4.1 shows the final CAD design for our pick-up mechanism. All components, except for the mounting plate, servo motor, and screws were 3D printed. To make the letter easier for pick-up and drop-off, we decided to tape the letter to a circular metal rod with colored paper rather than place it in a box. This way, the camera could more accurately identify the location of the letter, and the 3D rod could hook onto the letter using the metal rod. A picture of this setup can be found in *IV. Experimental Results and Analysis*.

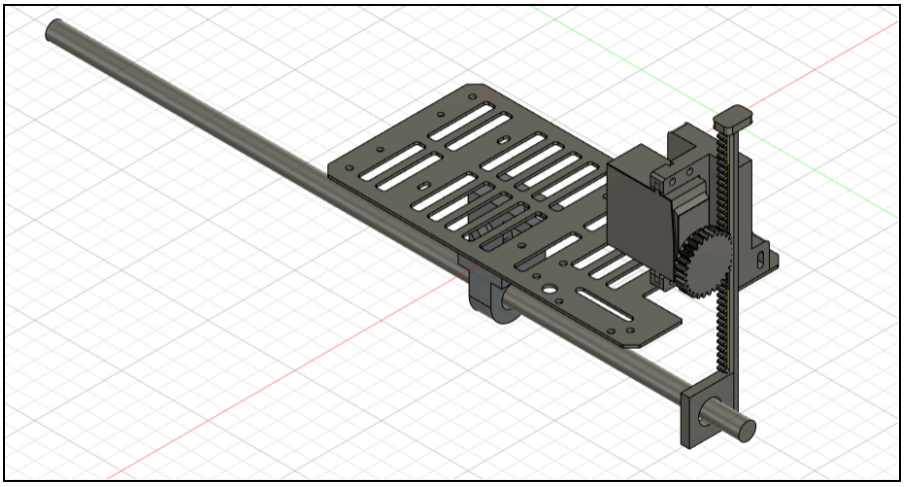

Figure 4.1: CAD Design for Payload

After finalizing the payload mechanism, we were able to submit our final hardware supply list (i.e., rPi4, Logitech camera, TFLuna Luna, Servo Motor). The figure below (Figure 4.2) shows our tentative functional components and their connections. The servo motor and camera are

critical peripherals that ensure that our drone can recognize and pick up the letter; both are connected to the companion computer (Raspberry Pi 3). Additionally, the WiFi Card ensures the UAV can communicate with the GCS, and the lidar enables the drone to better identify its surroundings and more accurately achieve our demo goals. The diagram also shows the voltage needs of these peripherals and how the rPi3 will support them.

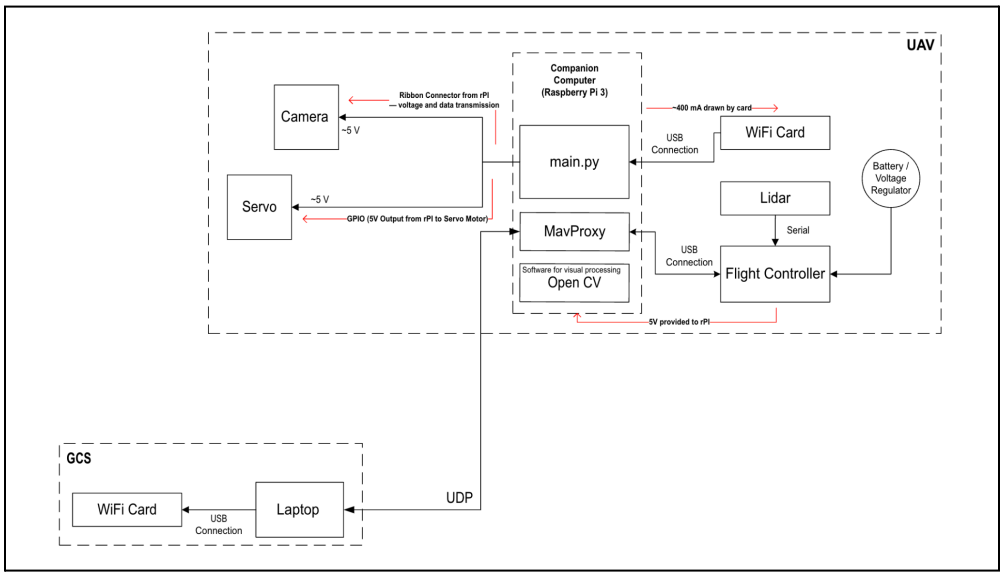

Figure 4.2: Hardware Connections

The final version of our payload is shown below in Figure 4.3.

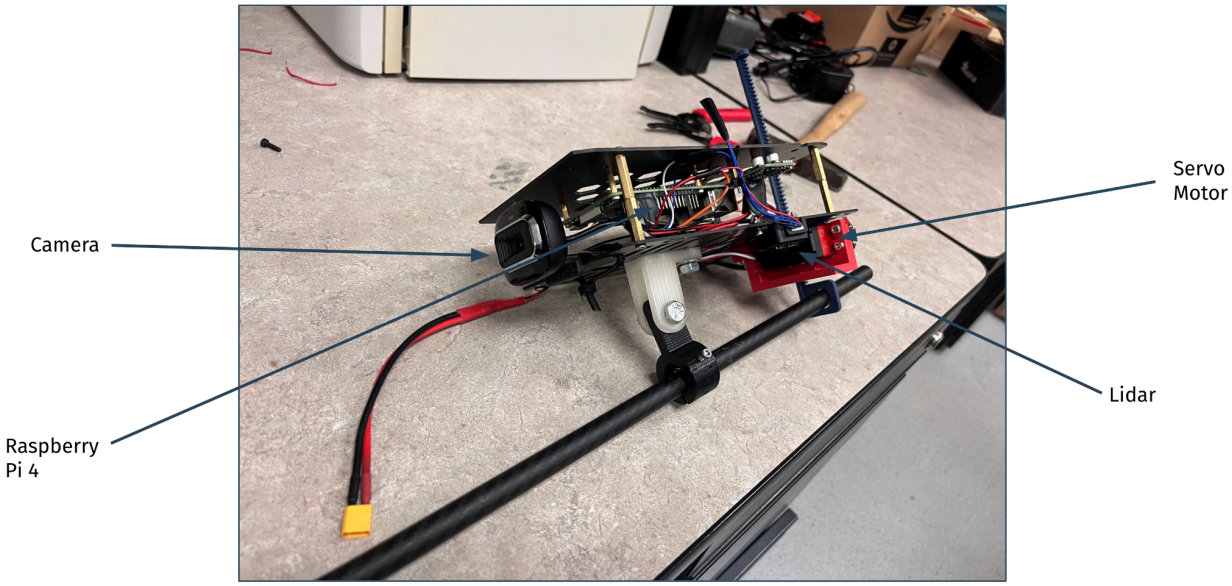

Figure 4.3: Labeled Payload

The next critical step in our approach was code development and virtual testing (using ROS and Gazebo.

B. Code Development

Using a variety of online resources (including the GitHub repositories of previous groups), we split our code development into three main groups: image processing, pick-up/drop-off implementation, and MAVLINK integration.

In an ideal world (with more time and field testing), lidar could have been used for dynamic altitude control during flight; essentially, the drone would maintain altitude using lidar measurements rather than the barometer, which would have significantly improved altitude accuracy and flight path resistance to wind. Furthermore, using the research mentioned in *III. Related Works*, we could have estimated distance from the letter algorithmically, rather than the more brute-force method used during our demo.

However, due to a variety of reasons, our final implementation was much different than the ideal version.

a. Pick-up and Drop-off

For the pick-up and drop-off testing, we focused on testing the servo motor movement to have it in the correct rod position for the different phases of the workflow. We changed the duty cycle of the GPIO controlling the servo to three different levels to simulate the neutral position, pickup position, and dropoff position of the rod that the servo motor controlled. We manually determined the duty cycle values that would work for the three positions that would avoid the rack of the rack and pinion system from hitting the drone during flight.

As shown in Figure 4.4, before the motor moved into the pick-up position, it would use the lidar to approximately get in the correct vertical position to hook our package. Using the VRone, we tested that the lidar would read about a meter or more heading to the package and read less than 0.5 meters once it was above the table and in front of the package. Through manual testing, we determined that 0.5 meters or less above the table would guarantee that the rod would go through the metal ring of our package without the package hitting the drone's propellers.

b. Image Processing

To test image processing, we used the VRone to identify the red flag on the metal loop inside the lab and outside in the sun. We used the OpenCV blob tracking implementation to check our HSV values for the color red and to manage our color thresholds for testing. The HSV values used in the code were manually measured using the "manual_threshold_measurement.py" script. This

script would output the camera feed, a mask feed, the resulting camera/mask feed, and an adjustable bar. Moving the bar would alter the Hmin, Smin, Vmin, Hmax, Smax, and Vmax, which would change which colors were masked out. After some manual adjustments, we were able to get an accurate representation of the color red in HSV values. We then used a mask that marked the center of the largest red object. With the center identified, we wrote functions that would send commands to center the red blob vertically and horizontally. Unfortunately, through our VRone testing, only the horizontal provided correct values.

Figure 4.4 (flowchart) shows the control flow used during the final demo.

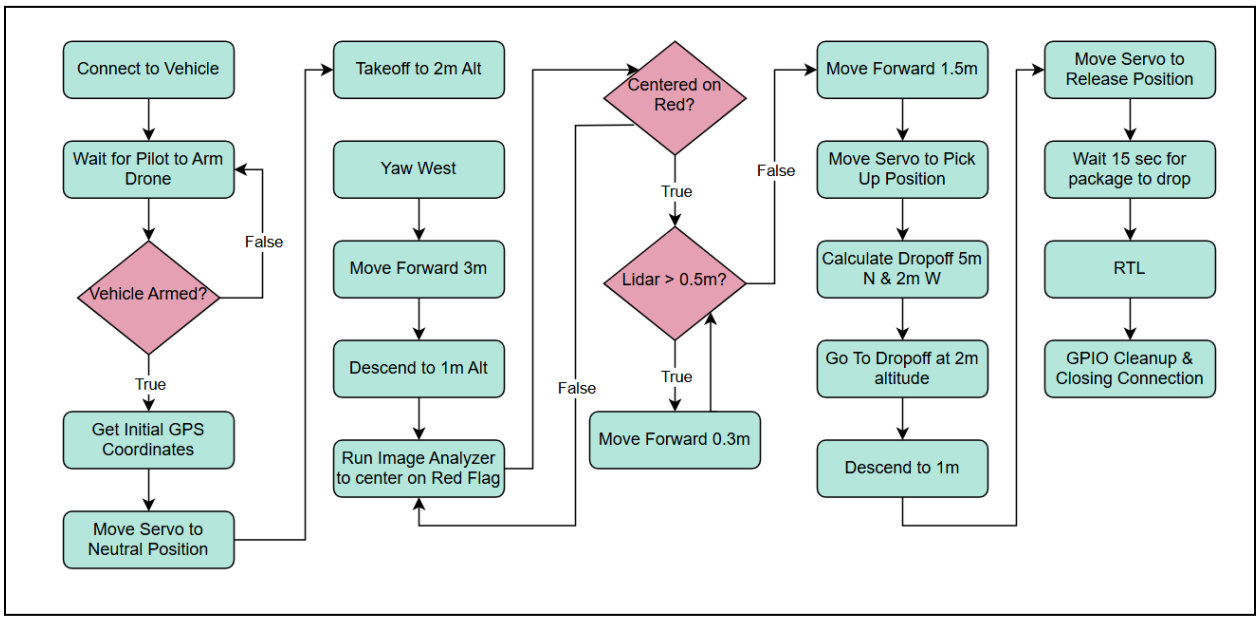

Figure 4.4: Demonstration Control Flow

C. Virtual Testing

a. Image Processing

For testing the camera feed in Gazebo simulation, Gazebo must be bridged to ROS and ROS to OpenCV, however, we had significant problems in both downloading these applications and bridging them together. We tried bridging Gazebo Harmonic to ROS 2 Humble and later on to ROS Jazzy, but did not have success before our final demonstration. Therefore, we were unable to check our code's centering functionality before going to the field, as the VRone couldn't use drone movement commands.

With only the horizontal providing positive movement values, we had a centering script that only moved the yaw of the drone to center on the red flag. It wasn't until after our final demonstration that we learned the negative values we were receiving from the vertical centering were correct, since vertical drone movement up is along the negative z-axis.

b. MAVLINK implementation

To test the MAVLINK integration, we used Gazebo to view the simulated flight path with SITL. Our script had the drone save its GPS coordinates for later, lifting off at 2m, flying west for 3m, dropping altitude to 1m, and continuing west for 3 more meters to get roughly near where our letter and the platform were located, as shown in the figure below. As we had difficulties getting lidar and image frames from Gazebo, we couldn't simulate the centering of the object. So, we then simulated the drone moving to the drop-off location that was 5m North and 2m West of its starting point and RTL after.

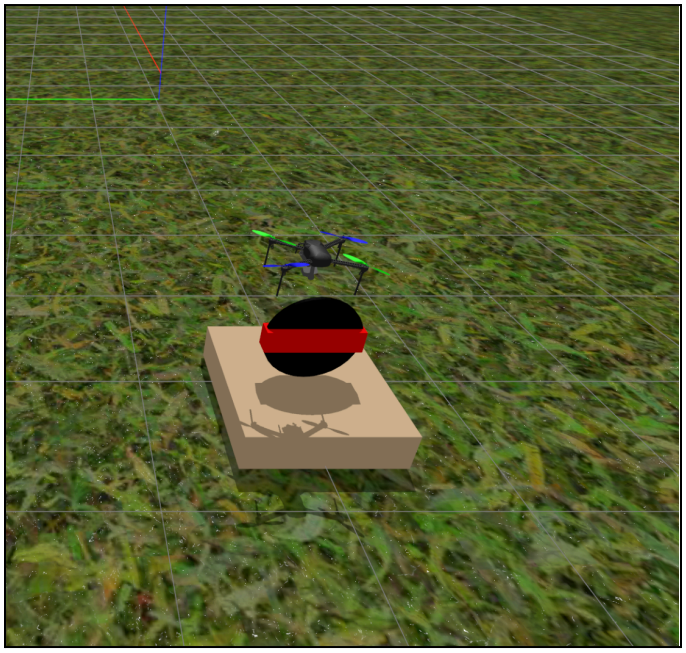

Figure 4.5: Gazebo Flight Simulation

V. Experimental Results and Analysis

Although we were not able to have a full integrated demonstration of our intended workflow, we had the individual parts working of the whole implementation. The three different parts were approaching the target and pickup, aligning with to target, and dropoff.

A. Approaching the Target and Pickup

We ran code that would run through the control flow shown in Figure 4.4, except RTL would be initiated after moving the servo motor into pick-up position. Unfortunately, while the drone was able to identify and approach the letter, the wind from the propellers on the drone blew the letter away. That said, the servo motor still engaged in the pick-up position and moved toward the drop-off location. At the drop-off location, it moved the motors to the drop-off position before RTL.

B. Field Centering

To demonstrate field centering, we ran code that would get the drone to adjust its yaw as the target moved back and forth. This showed our drone's ability to shift horizontally to center on an object. Images 5.1 and 5.2 below show the yaw adjustments based on the movement of the red paper.

Images 5.1: Centering

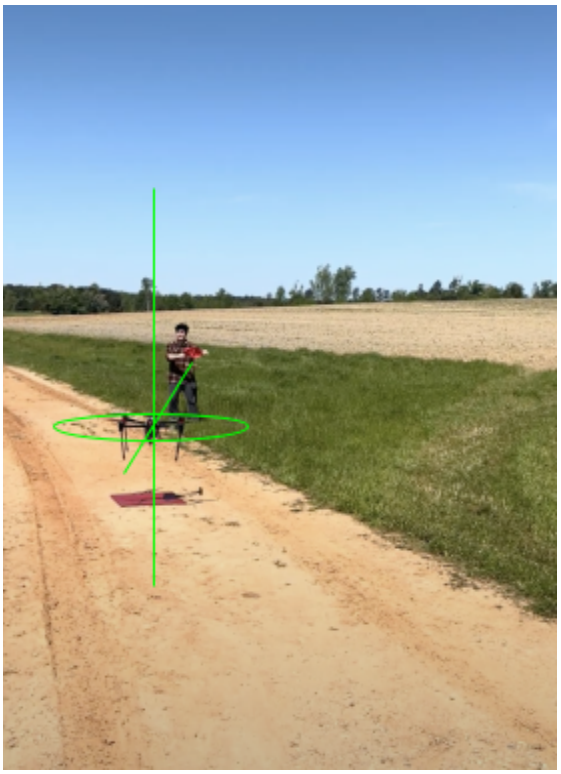

Image 5.2: Centering

C. Dropoff

For the dropoff routine, we began with the letter already secured on the hook so that the flight script would focus solely on the navigation and release of the payload. The drone would initially lock the letter, then travel to the drop-off location, drop the letter, and then return to the original starting location. The demonstration was a success, and we were able to show a successful dropoff of the letter.

VI. Conclusion

The goal of the TrueLove project was to program a drone to identify, pick up, and drop off a letter autonomously. While the drone was unable to perform all of these objectives in one run, the underlying milestones associated with these broader objectives were achieved, including

accurate color identification and centering using a camera, a pick-up and drop-off mechanism capable of handling the letter, and overall flight movement. In achieving these, we expanded our knowledge of drone simulation environments, the integration of multiple peripherals to a rPi3, and the capabilities of Python packages. One issue we had throughout the project was setting up and using Gazebo and ROS to test the functionality of our payload mechanism and our overall code. Ultimately, we were able to test basic drone movement in Gazebo, but were unable to model camera and lidar functionality, which impacted our performance during field testing.

Due to weather conditions and schedule constraints, we were unable to test in the field before the final demo day, but with additional time and fine-tuning based on our analysis, the payload mechanism and code would provide a great foundation for achieving all project objectives in one run by future teams.

VII. References

[1] Dr. B. R. P. Et. al., “Realtime Object’s Size Measurement from Distance using OpenCV and LiDAR,” *Turkish Journal of Computer and Mathematics Education (TURCOMAT),* vol. 12, no. 4, pp. 1044–1047, Apr. 2021, doi: https://doi.org/10.17762/turcomat.v12i4.599.

[2] S. Parveen and J. Shah, "A Motion Detection System in Python and OpenCV," *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, Tirunelveli, India, 2021, pp. 1378-1382, doi: 10.1109/ICICV50876.2021.9388404. keywords: {Visualization;Webcams;Detectors;Software systems;Motion detection;Security;Python;Motion Detection;OpenCV;Python;Surveillance;Bokeh},