Autonomous Target Scoring Drone

Authors: Bert Yu, *CPE,* Bibek Adhikari, *CPE,* Eric Rodriguez, *CPE,* Sukhmanjit Kaur, *EE*

Advisors: Zakher M. Kassas, *Professor*, Mahdi Maaref

***Abstract*—Fall quarter was focused on getting the base of the drone up and running. We secured all the necessary hardware to build the drone and image processing, and assembled each separately. We tested flight with the drone using different code and algorithms. Then we tested image processing on a Raspberry Pi to make a basic version of the scoring algorithm. Winter quarter was focused on developing the autonomous drone, adding onto the image processing, and running scoring tests with the whole system.**

I. Introduction

OUR purpose in this project is to make an automated target scoring drone that would be able to score mortar tests. It was originally proposed to us as being partnered with the Navy, which we later confirmed in order to work it. The purpose of the project is to get the current target scoring more efficient. Our current system is currently automated for one waypoint. After the user inputs coordinates on the GUI, it hovers over to the target and then takes an image and processes it. The data will then be sent back to the GUI where it will be displayed for the user.

II. Challenges

In terms of actually making the scoring system, the most difficult part was figuring out how to automate it and how to make it compatible with the Raspberry Pi. Although we could make the drone fly fairly simply manually, we had to be able to control it through a receiver and transmitter; hence we had to research multiple examples [1.], to get an idea of how other autonomous drones had been programmed. Additionally, the motors, and parts of the frame occasionally broke from testing, thus a lot of time would need to be allocated to waiting for parts before any other tests could be performed.

In terms of the image processing, getting the program to work for a digital painted picture was easy. However, once you factor in real life situations, it becomes much more difficult. The main problem is that there are too many colors in real life and it gets mixed in with what the program is looking for. This is especially true when shadows make a circular shape as that is what the program is looking for. Currently, the program searches for colored circles, and with all the possible shadows in the wild, it becomes difficult for the program to discern the target.

III. Hardware and Software Used

In order to fabricate the drone, the following was purchased:

- Drone Frame
- 4 Electronic Speed Controllers
- 4 Brushless Motors/Propellers
- Raspberry Pi
- Navio Flight Controller
- Raspberry Pi GPS Module
- Raspberry Pi Camera
- LiPo Battery
- 1 Flight Controller
- Portable Battery for the Pi
- Receiver and Transmitter
- Telemetry Module

The image processing and the GUI module were made using Python. The drone flight code was also made in Python.

A. Methods for Fabrication of Drone:

After considering the autonomous system that Rabah implemented with the Raspberry Pi, and how cheap of an option it was, we decided to have the Raspberry Pi act as the microcontroller [1.] Fabrication was done by utilizing the materials listed above and connecting each component. The first step was to assemble the frame. We used M3 screws and standoffs to connect the base plate onto other components. As soon as a foundation was established, the motors that were attached to them were then soldered onto the power distribution area. Next, the flight controller was attached above the power distribution area, and the ESCs were attached to the motors. Then, the top plate was connected to the standoffs attached to the base plate and the LiPo battery was velcroed on the bottom plate and connected to the power distribution. In order to deliver power to the motors, the battery was connected to a LiPo battery which also powered the flight controller.

Fig 1: Shows the drone

B. Methods of Drone Flight

For communication we chose to use telemetry between the Raspberry Pi flight controller and the base station. [2.][3.] For our current position in the project, long range flying is out of the question. So until we hit that point, Telemetry communication is enough. It allows for us to have a private connection between the drone and our homebase. Telemetry also does not need a network in order to connect, only a sending signal and a receiving signal. The ideal environment for our drone will be in the middle of nowhere at places where the military is going to test. These may be places in the Middle East with no network connectivity whatsoever, so it will need another way for data to transmit.

The receiver on the drone will be able to detect any commands that the transmitter sends to the drone and moves accordingly. With autonomous flight, the Python script will be called through Mavlink, and the drone will run until the receiver receives transmitted commands. The script allows the drone to takeoff and fly to the specified waypoint that the user inputs.

C. Methods of GUI:

The GUI is done in Python and Pycharm IDE. It consists of a simple window where a user can enter how many locations they can go. Then the GUI will prompt the user on entering the longitude and latitude for the locations they wish to go to. For example, they can enter: Longitude: +40.689060 , Latitude: -74.044636 or Latitude: 40 degrees, 42 minutes, 51 seconds N ,Longitude: 74 degrees, 0 minutes, 21 seconds W. If users wish to change the coordinates, they can click on the reset button to restart the inputs. After entering the longitude and latitude, the user can press enter to go to those locations. On the bottom of the screen, the user can view a live simulation of the drone's flight. Example: Update: Drone is on the base, Enter longitude and latitude to go to (number of) locations. Going to the first location. Also on the GUI there is a current status on the screen where

every time a drone has made it to a certain location, it will show the checkmark.

Then, there is a live update of a drone going from one location to another location. When the drone gets to a certain location, image processing will take place. The image that is captured by the drone is then shown on the GUI screen. Also, the GUI will read the text file received from image processing and place the score on the screen.

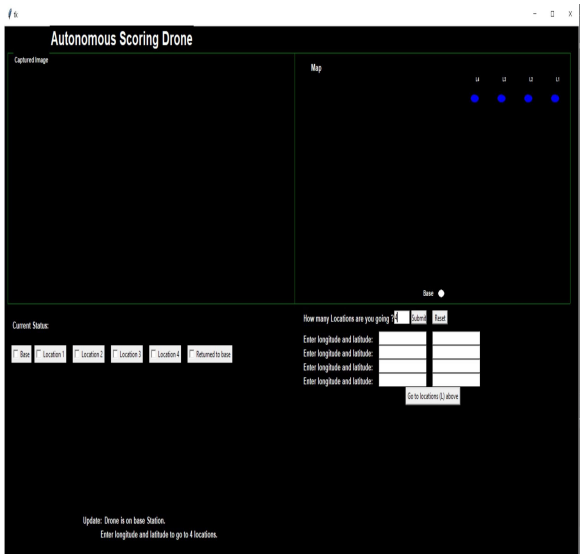

Fig 2: Simple GUI window

D. *Methods of Image Processing*

Fig 3: Expected output

The image processing has changed a lot since the first conception. Originally, the image processing would look at the colors of the image and simply pick out the target and the shot based on that. This worked incredibly well for a computer painted image, but translated very poorly to real images. This is because there are various opportunities for colors to be messed up including shadows, weather conditions, and other environmental variables.

The current version of the image processing uses OpenCV functions such as HoughCircles and Canny . These functions do edge detection and try to recognize circles on whatever image is inputted, like how TS Huang emphasizes in his paper [4]. Modifying the parameters allows it to recognize only well defined circles rather than what it assumes to be a circle. With our test images, it worked correctly about 90% of the time.

D. *Results:*

After assembling all necessary components, the drone is able to fly manually without any problem. Autonomous flight to waypoints specified by the user is able to function as well. However, multiple waypoint

flight and having the drone adapt to weather conditions is something that has to be worked on.

The GUI allows the user to input multiple coordinates. These coordinates are saved and sent over telemetry to the Raspberry Pi acting as the flight controller. However, the system currently only sends over one set of coordinates.

The image processing is essentially finished, but it could be refined more. The image can be taken via the Raspberry Pi camera and it can also be processed through the custom functions. It will also output the processed image file and the score text file in order to be sent back to the GUI. Although, it only works about 90% of the time as sometimes it recognizes circles where there are not any.

E. Performance:

Overall, the drone works as intended but only travels to one point. Other than that, the individual components perform to our expectations. The drone is stable when flying in the air manually, and the drone is able to fly to one waypoint autonomously. It is even able to take a picture while hovering over the target thanks to scripts and functions. Finally, it is able to transfer the data to the base in order to be displayed on the GUI.

IV. Conclusion

The past two quarters were primarily focused on fabricating the drone, enabling manual and autonomous flight, image processing and construction of a GUI. It was decided to build the drone from scratch rather than buying one and hacking it because it would be easier to run the imaging processing and the GUI functions in Python. The user would be able to choose between flying the drone manually or autonomously. If flown autonomously, the drone would read the waypoints the user inputted in the GUI and fly to those hovering above the destination for a while and then take a picture. This image would then get sent back to the base station, which was a laptop, where the Python image processing algorithm would run and return a score depending on how well the user's shot was to the intended target.

Other than combining all of the components that were completed, other improvements could also be made. The image processing works 90% of the time because it recognizes shadows or sometimes random parts of the image as circles. The best way to improve this would be through machine learning, where the program explicitly recognizes the targets and shots that it is looking for. Additionally, having the image processing and autonomous flight work well regardless of weather conditions was another future milestone. The GUI could also be modified so that notifications or alerts pop up when signal is lost or the drone has crashed.

V. Acknowledgements

This section serves as a brief acknowledgment to all the members of Team Yark: Bert Yu and Sukhmanjit Kaur who worked on the fabrication of the drone and achieving basic flight; Eric Rodriguez, who developed the image processing algorithm; and Bibek Adhikari who created a graphical user interface. Also thanks to our advisors, Mahdi Maaref and Zakher M. Kassas for providing advice and setting weekly goals for us to achieve. Also thanks to the EECS 159 course and UROP, for reimbursing and providing us with the necessary equipment in order to complete this project.

References

[1.] Rabah, M., Rohan, A., Talha, M. et al. Int. J. Control Autom. Syst. (2018) 16: 3013. https://doi.org/10.1007/s12555-018-0017-x

[2.] Kan M, Okamoto S, Lee JH. Development of drone capable of autonomous flight using GPS. In: Proceedings of the international multi conference of engineers and computer scientists, vol. 2, 2018.

[3.] Patrik, A., Utama, G., Gunawan, A.A.S. *et al.* GNSS-based navigation systems of autonomous drone for delivering items. *J Big Data* 6, 53 (2019) doi:10.1186/s40537-019-0214-3

[4.] T. S. Huang, W. F. Schreiber and O. J. Tretiak, "Image processing," in *Proceedings of the IEEE*, vol. 59, no. 11, pp. 1586-1609, Nov. 1971. doi: 10.1109/PROC.1971.8491 keywords: {Image processing;Image enhancement;Optical sensors;Pattern recognition;Image coding;Optical computing;Image quality;Optical imaging;Degradation;Holographic optical components}, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1450421&isnumber=31150