

# **Autonomous Flight School**

**CMPE-4371- Senior Design (I, II)**

**Final Report Draft**

**Team Number: 3**

**Group Members:**

**Nathaniel Lozano (I, II)**

**Bradford Scrivener (I, II)**

**Gilberto Rios Andrade (I)**

**Advisors:**

**Dr. Emmett Tomai**

**Dr. Wenjie Dong**

**Instructor:**

**Dr. Heinrich Foltz**

## **General Abstract**

Drones are being used more frequently, this is consistent with an increase in commercial production, military development, delivery systems, racing/stunting and film production.

However, with more research being put into machine learning, image processing, and artificial intelligence; there is an evolution in the method for controlling drones. That evolution is the change from manned aircrafts to unmanned aircrafts.

We are going to be researching the possibilities and limitations of self-directing drones, specifically quadcopters. We will start this research off by using a commercial drone and then build a drone with sensors, this will allow new possibilities. The commercial drone will be used by the team members to develop a safe tracking system using facial/body recognition. Alongside that will be an object avoidance system, that will dodge the objects it detects. The team will add onto the object avoidance system with a spatial visual awareness system. The system will allow the drone to move freely in a space, then avoid and keep track of objects directly in front of its path. The drone being built will have the simple task of flying and hovering.

## Technical Abstract

In the beginning of our research we tested the possibilities and limitations of the commercial drone we chose. Since some possibilities were discovered by a previous group [4], we set out to base our research off their findings. With that in mind, we found multiple topics that could be developed, as a program, for the commercial drone. Those topics included, an object avoidance system, OpenPose body tracking, body recognition cascade, spatial visual awareness system, and reference point system. Since our project is meant to work autonomously, there is a concern for the ethics surrounding the drone.

The object avoidance system will detect a yellow object using HSV (hue, saturation, and value) edging. Once the object is detected, an avoidance algorithm will be used to avoid the object that is in front of the drone. It will be programed in python language and in the visual studio code environment, then run with the drone and its camera.

The body tracking will use an already trained haar cascade, it uses height, width and distance. Using this cascade, we will then write the code integrating its existing functionalities to make the algorithm work correctly on the drone, once it detects a body, the tracking algorithm will proceed to track it. This will be programmed in Python language in visual studio code and then run it with the drone and its camera.

The spatial visual awareness system will visualize objects in a similar manner as the avoidance system but will be using a ratio involving both width of object and distance between drone and object. The ratio will allow the system to determine if the object is directly in front the drone. The system will have a predefined space for the drone to travel in. In that space the

system will allow the drone to avoid objects it detects and keep track of them for any future avoidance. It will be programmed in python language and in the visual studio code environment, then run with the drone and its camera.

The reference point system will be used in the spatial visual awareness system to allow for more accurate movements inside the space. It will use the same object detection method as the spatial visual awareness system. The objects detected by the drone will be used to determine the drones inaccurate positioning in the space. Then make small adjustments to correctly orient the drone. It will be programmed in python language and in the visual studio code environment, then run with the drone and its camera.

The limitations of the commercial drone become apparent due to the drone only having one camera. So, this led us to research the possibilities of a home-built drone. In our research we were able to discover different frames for drones and learn about the different hardware that can be used on a drone. As of now we were able to build a drone with specified hardware, such as CC3D flight controller, EMX motors, LiPo battery 4s, ESCs, and power distribution board. The drone is built to be capable of flight with a Remote Controller. The goal that we are aiming for is to get the drone to fly and hover. We plan to further our research & development in the future, with an auto-leveling system, and an autonomous racing/stunting system.

## Figure List

Figure 1.2.1 Tello drone App GUI.....	2
Figure 1.4.1 FAA reports on commercial drones in United States.....	5
Figure 2.2.3.1 CV2 errors.....	9
Figure 2.2.3.2 CV2 fix.....	10
Figure 2.2.3.3 More CV2 errors .....	10
Figure 2.4.3.1 Current state of drone build .....	14
Figure 3.2.1 Gantt chart part 1 .....	25
Figure 3.2.2 Gantt chart part 2 .....	26
Figure 3.2.3 Gantt chart part 3 .....	27
Figure 3.2.3 Gantt chart part 4 .....	28
Figure 3.2.3 Gantt chart part 5 .....	28
Figure 3.2.3 Gantt chart part 6 .....	39
Figure 4.1.1 Block Diagram .....	31
Figure 4.2.3.1 Yellow RGB Range .....	34
Figure 4.2.3.2 Threshold camera feed .....	34
Figure 4.2.3.3 RGB camera feed .....	35
Figure 4.2.3.4 Libraries used to build OAS .....	36
Figure 4.2.3.1.1 Avoidance Flowchart.....	37
Figure 4.2.5.1 7 by 7 matrix used by system .....	38
Figure 4.2.5.2 Equations for calculating distance between object and drone.....	39
Figure 4.2.5.1.1 Free roam system algorithm .....	41

Figure 4.2.5.1.1 Pathfinding system algorithm.....	42
Figure 4.3.1 Drone Frame and propellers.....	43
Figure 4.3.2 CC3D Flight controller .....	44
Figure 4.3.3 Lipo 4s battery .....	45
Figure 4.3.3.1 Charger and AC power supply.....	46
Figure 4.3.3.2 Battery Specifications .....	46
Figure 4.3.3.3 Charger to battery connection.....	47
Figure 4.3.4.1 CC3D to receiver connection.....	48
Figure 4.3.4.2 CC3D receiver output wires.....	48
Figure 4.3.4.3 ESCs' connections to CC3D.....	49
Figure 4.3.4.4 Controller with transmitter.....	50
Figure 4.3.5.1 Quadcopter (PDB, ESC, Motor, excluding transmitter & receiver).....	51
Figure 4.3.6.1 mAh of our battery .....	53
Figure 4.3.7.1 ESC wizard.....	54
Figure 4.3.7.2 Orientation of propellers.....	54
Figure 4.3.7.3 Flight orientations of drone .....	55
Figure 4.3.7.4 Librepilot failsafe .....	56
Figure 6.3.1 When Configuring is Done .....	67
Figure 6.3.2 Flight orientations of drone.....	68
Figure 6.5.1 Imported libraries.....	70

## Table List

Table 2.5.3.1 Path movement forward for both first and second OAs .....	16
Table 2.5.3.2 First Avoidance dodge distance trails .....	16
Table 2.5.3.3 Second Avoidance dodge distance trails .....	16
Table 2.5.3.4 Second Avoidance re-center distance trails .....	16
Table 2.5.3.5 Second Avoidance forward distance trails.....	16
Table 2.5.3.6 Second Avoidance first movement delay time trails .....	17
Table 2.5.3.7 Second Avoidance second movement delay time trails.....	17
Table 2.7.3.1 Drone movement from cell to cell in the grid.....	20
Table 2.7.3.2 Dimensions of real-life grid.....	20
Table 2.7.3.3 Width of obstacle to be dodged .....	21
Table 2.7.3.4 Time delay for each iteration of movement.....	21
Table 3.3.1 Parts List .....	30
Table 4.3.6.1 Drone Weight.....	52
Table 4.3.6.2 Thrust Tables .....	53
Table 5.1.1 Avoidance 1.0 Test plan .....	56
Table 5.1.2 Avoidance 2.0 Test plan .....	57
Table 5.2.1 Awareness Ver. Free Roam Test plan .....	59
Table 5.2.2 Awareness Ver. Pathfinding Test plan .....	59

## Table of Contents

General Abstract .....	i
Technical Abstract .....	ii
Figure List .....	iv
Table List .....	vi
1. Introduction and Background .....	1
1.1 Introduction .....	1
1.2 Requirements .....	1
1.3 Project Objectives & Specifications .....	3
1.4 Background .....	4
2. Missions .....	6
2.1 Mission 1 .....	6
2.1.1 Objective .....	6
2.1.2 Mission Success .....	6
2.1.3 Results .....	7
2.2 Mission 2 .....	7
2.2.1 Objective .....	8
2.2.2 Mission Success .....	8



2.2.3 Results.....	9
2.3 Mission 3.....	10
2.3.1 Objective .....	11
2.3.2 Mission Success .....	11
2.3.3 Results.....	12
2.4 Mission 4.....	12
2.4.1 Objective .....	13
2.4.2 Mission Success .....	13
2.4.3 Results.....	14
2.5 Mission 5.....	15
2.5.1 Objective .....	15
2.5.2 Mission Success .....	15
2.5.3 Trails .....	15
2.5.4 Results.....	17
2.6 Mission 6.....	17
2.6.1 Objective .....	18
2.6.2 Mission Success .....	18
2.6.3 Results.....	19

2.7 Mission 7 .....	19
2.7.1 Objective .....	19
2.7.2 Mission Success .....	20
2.7.3 Trails .....	20
2.7.4 Results.....	22
2.8 Mission 8.....	22
2.8.1 Objective.....	23
2.8.2 Mission Success .....	23
2.8.3 Results.....	23
3. Team Management.....	24
3.1 Workload.....	24
3.2 Gantt Chart.....	25
3.3 Parts Cost .....	29
4. Design .....	31
4.1 Block Diagram .....	31
4.2 Software .....	31
4.2.1 Overview .....	32
4.2.2 Body tracking OpenPose.....	33

4.2.3 Object Avoidance.....	33
4.2.3.1 Flowchart .....	36
4.2.4 Body Recognition with MobileNet SSD.....	38
4.2.5 Spatial Visual Awareness .....	38
4.2.5.1 Flowcharts.....	40
4.3 Hardware.....	43
4.3.1 Overview .....	43
4.3.2 Components .....	45
4.3.3 Batteries .....	46
4.3.4 Flight Controller, Receiver, & Transmitter.....	47
4.3.5 Frame, Motors, ESCs, PDB, & Propellers.....	50
4.3.6 Calculations.....	51
4.3.7 Flight Controller Software .....	53
5. Testing.....	56
5.1 Object Avoidance System Testing.....	55
5.1.1 Test Plan Object Avoidance System Version 1.0 .....	56
5.1.2 Test Plan Object Avoidance System Version 2.0 .....	56
5.1.3 Results.....	56

5.2 Spatial Visual Awareness System Testing.....	58
5.2.1 Test Plan Spatial Visual Awareness System Version Free Roam .....	59
5.2.2 Test Plan Spatial Visual Awareness System Version Pathfinding .....	59
5.2.3 Results.....	60
6. Setup Guides .....	60
6.1 Mission 1 Guide.....	60
6.2 Mission 2 Guide.....	63
6.3 Body Tracking OpenPose .....	65
6.4 Object Avoidance Guide.....	68
6.5 Body Tracking MobileNet SSD Guide.....	69
6.6. Spatial Visual Awareness Guide.....	70
7. ABET Criteria .....	72
7.1 Engineering Standards .....	72
7.2 Environment.....	73
7.3 Political, Ethical, & Social issues .....	74
7.4 Health & Safety.....	75
7.5 Sustainability.....	76
8. Conclusion & Future Work.....	76
8.1 Conclusion .....	76

8.2 Future Work .....	77
8.2.1 Object Avoidance System Additions .....	77
8.2.2 Spatial Visual Awareness Additions.....	78
9. References .....	79
10. Appendix .....	81

# **1. Introduction and Background**

## **1.1. Introduction**

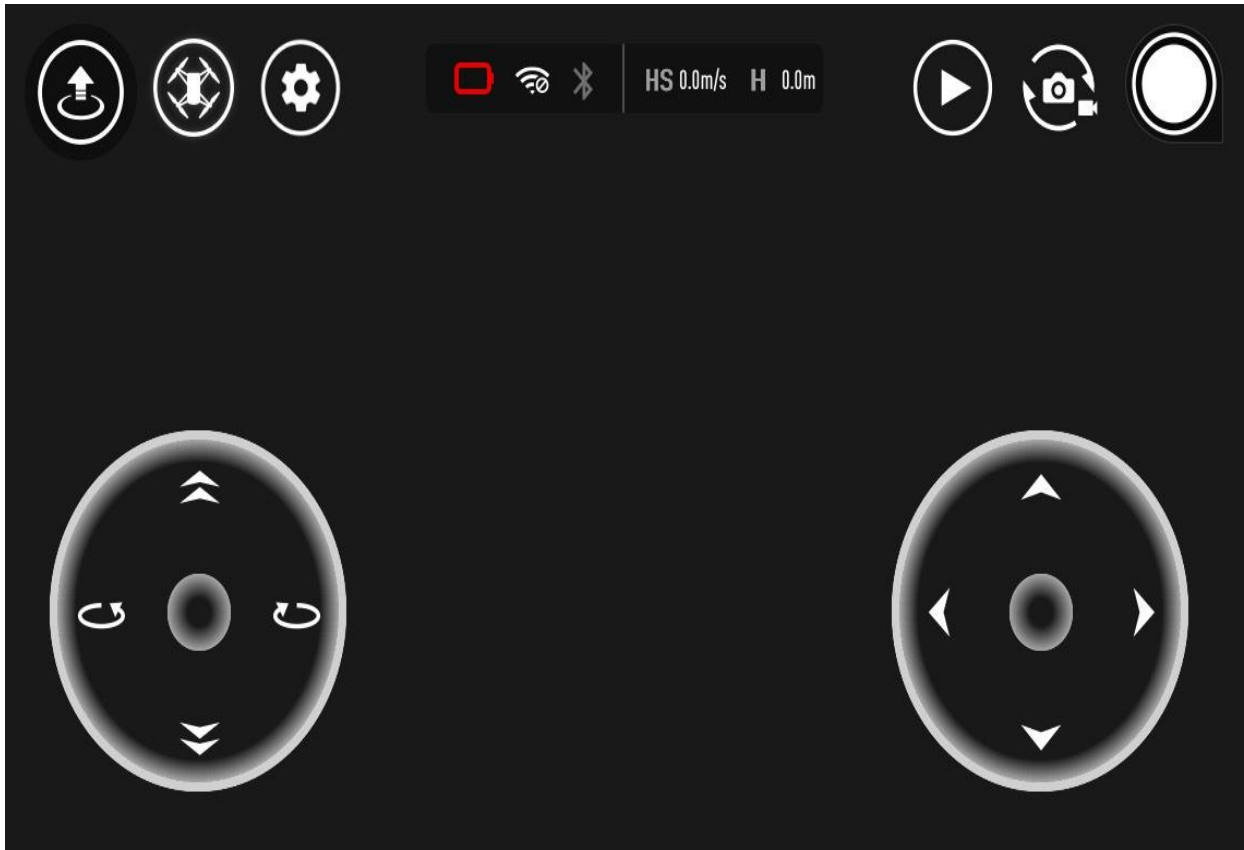
This project will be based on drone technology, it will allow the group members to go over different types of computer engineering aspects. Some of those aspects include digital image processing, embedded systems, and artificial intelligence.

The group members will research different methods for facial recognition and then apply those methods to a drone. This will allow the team to assemble an unmanned aircraft system (UAS). The team will divide the project into multiple portions. The goal is to leave the project in a state that an amateur drone enthusiast could go through the project and pick up where we left off.

## **1.2. Requirements**

The project will consist of building/testing a drone capable of flight and developing software for the drone such as object detection. As for the key function of the drone itself, that will include autonomous drone flight. The drone's autonomous functions will be simple functionalities. Those functionalities include being able to detect a person with a camera, using two different body tracking programs; and detecting an object based on a color system. The autonomous drone should be able to maneuver around the objects it detects, maneuverability includes being able to avoid hitting the object and will use a developed avoidance algorithm to correctly move around a stationary object. The methods used by the drone will be discussed in section 3.3. Once the drone can detect faces and bodies, we will try to develop a way for the drone to track the person that was detected. In the case of a person being lost during tracking; the

program will then switch to the searching algorithm, in order to find the person. The specific parts that will be required for this portion of the project includes the Tello drone.



*Figure 1.2.1 Tello drone App GUI*

The Tello drone will be used for the first half of this project so the group members could gain the basic skills of programming a drone and understand the Tello drone app GUI, seen in figure 2.2.1. Once the group members have exhausted all the capabilities of the Tello drone, they will then start the process of building a drone with multiple electronics rather than a single camera. The parts for the build include microprocessor, flight controller, transmitter, motors, electronic speed controllers, frame, and propellers. The process of building the drone and its specifications will allow for the computer engineering students to cover the hardware portion; since the building process will include testing the electronics, wiring the electronic parts together properly and using the drone to connect to separate devices.

In order to give a final understanding of this project. There will be multiple sections of each individual mission for this project.

### **1.3 Project Objectives & Specifications**

The project objectives will be listed below:

- Get the drone to take flight.
- Create a working flight python program.
- Get the drone to recognize faces.
- Get the drone to detect objects.
- Get OpenPose to detect a person's body
- Get the drone to detect a person's body with OpenPose
- Get the drone to move accordingly to a person's body
- Get the drone to follow a person's body
- Get the drone to power on.
- Get the drone to fly in cardinal directions
- Get the drone to fly forward toward goal.
- Get the drone to avoid object in its path.
- Get the drone to reach the goal.
- Get the drone to reach the goal, while avoiding objects.
- Get the drone to detect a person's body with a working algorithm.
- Get the drone to follow a person successfully.
- Get the drone to freely move in a predefined space.
- Keep track of objects drone encounters.



- Have the drone follow a path from two points in a grid.

The Specifications for the Dji Tello Edu will be listed below:

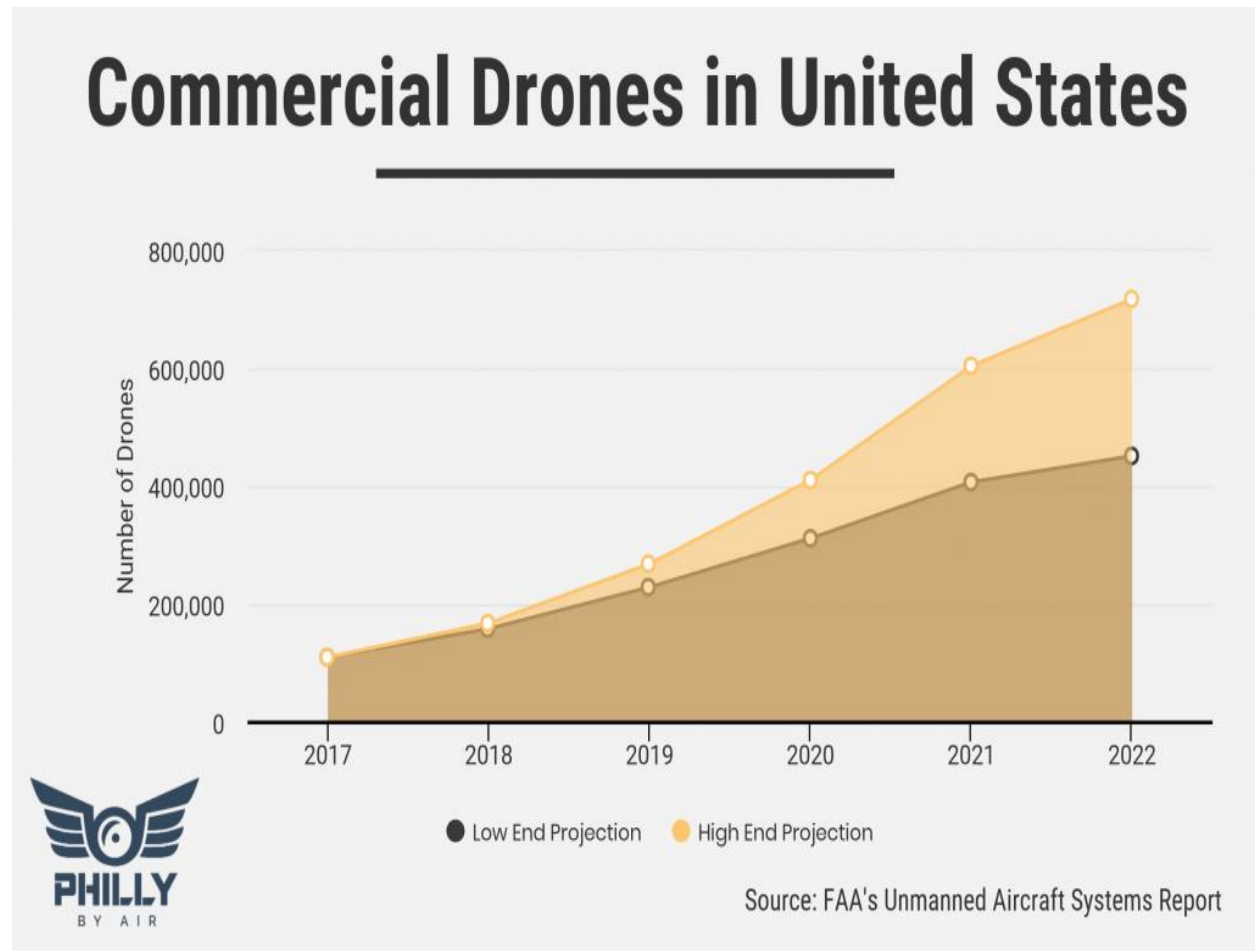
- 1.1Ah, 3.8 V detachable battery
- WIFI 802.11n 2.4G, 720P Live View
- 30 Fps video
- 80 g
- 3 in propellers
- FOV: 82.6°
- Max Flight Time: 13min

The Specifications for the project drone build will be listed below:

- Readytosky 250mm frame
- EMAX RS2205-2300KV 3-4S
- nidi BLHeli-32 30A ESC 3-4S
- 5045 props
- 4S 1500mAh 100C 183g XT60
- T-plug connectors
- SKYRC iMAX B6 charger
- CC3D Revo stm32f4 w/ OPLink
- Flysky FS-i6X + FS-iA6B RX

## **1.4. Background**

The purpose of this project derives from the increase in research on the computer engineering concepts listed in section 1.1. Research in artificial intelligence (AI) seems to be growing the fastest out of all the concepts this project covers [9]. Most of Unmanned aerial



*Figure 1.4.1 FAA reports on commercial drones in United States*

systems (UASs) depend on a pilot to control the primary functions of the aircraft. However, UASs build a base for developing AI. The reason for this stems from the convenience of just pressing a button, then a request will be filled. In figure 1.4.1, it is shown that commercial drones are becoming more accessible to citizens in the United States. This trend is arising from the increase of freelance videographers. Drones help film production tremendously [19], allowing

for camera angles that were unheard of. So, a videographer has the job of capturing video for a production and a drone that can be obtained commercially will make their job easier. Our research and development will make their job much easier. By allowing them to not worry about missing any video opportunities, since the drone will be taking video alongside them.

## **2. Missions**

### **2.1 Mission 1**

This will be the first mission of the project. The mission will require all group members, Nathaniel, Gilberto, and Bradford to get the drone to take flight. The Tello drone comes with an app that will be used to test the default system given with the Tello drone. However, the actual goal for this mission is to get the drone to take flight using a python program.

As far as specifications go, the team members will get the python program running on their personal computers using Visual Studio Code. When describing flight, we mean for the drone to hover and move in cardinal directions.

#### **2.1.1 Objective**

- Get the drone to take flight.
- Create a working flight python program.
- Develop our skills and knowledge in python.
- Learn the limitations of Tello and try to problem solve.
- Learn difference with our team's software and Tello software.

#### **2.1.2 Mission Success**

For this mission to be a success all team members will be required to get the drone flying on their personal computers. Assuming the drone can fly, it will still be a successful mission even if the drone cannot detect faces, detect objects, or count faces. The success of this mission should not be too difficult to achieve.

### **2.1.3 Results**

Through our research and development, we were able to have a successful mission. While researching we were able to discover an open-source program that was used as a base for this mission [3]. Using this program and libraries, we were able to get the drone to take flight.

We were attempting to build a system that rivaled the Tello Drone's software, that required us to build a program that accessed the camera on the drone. So, the system that was built can fly in all cardinal directions while viewing a live video stream from the camera. The camera view is displayed in a window. The way this works is through a transmission control protocol and Internet Protocol that will establish a connection with the drone and computer that the program is being run on.

## **2.2 Mission 2**

This will be our second mission of our senior design 1 project. The mission will require that all group members will need a basic understanding of coding in python using visual studio. Another requirement for this mission is for the group members to understand GitHub's functionalities. The reason for these requirements is because the goal of this mission is to apply facial recognition and object detection functional. Now the facial recognition for the Tello drone will try to apply a shape. This shape is a green rectangle that will outline the face of a person, saved into the code's library to establish that this is the person (or object) that the drone is

searching for. The drone be able to detect another person's however if it's not the correct face the drone will then highlight a red rectangle around the person's face detecting the face but realizing that this is not the face (or object) the drone is looking for and will continue until the correct face is found.

As for specifications, the group members will be using a GitHub repository to share and manage files. They will also be using GitHub to clone open source code already available for the Tello drone. Multiple packages will be used for the python programs.

### **2.2.1 Objective**

- Get the drone to recognize faces.
- Get the drone to detect objects.
- Get camera on personal computer to recognize faces.
- Get camera on personal computer to detect objects.
- Further develop skills and knowledge in python.

### **2.2.2 Mission Success**

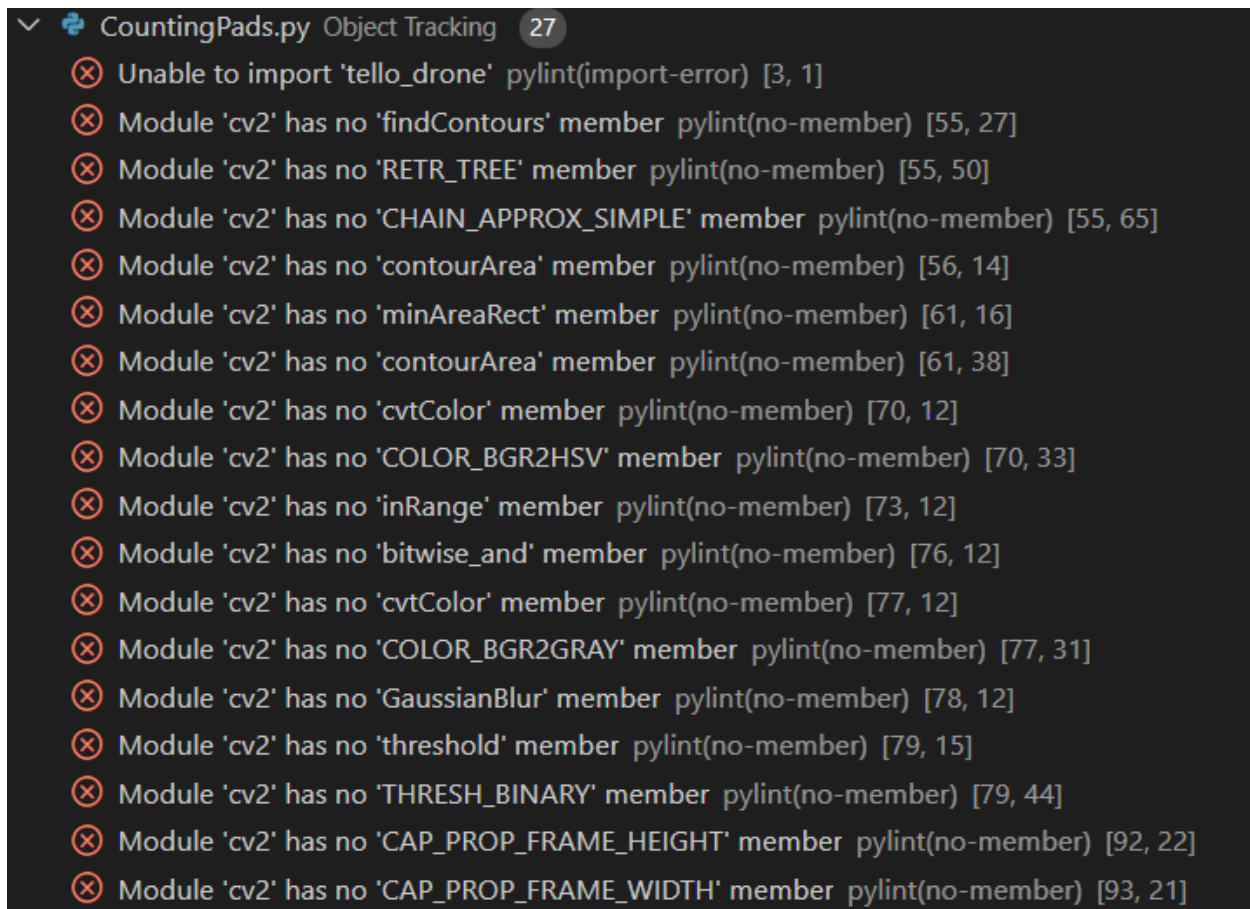
In order for the mission to be a complete success, all team members will need to have the code working on their personal computers. All group members must have a branch in the GitHub repository and understand the GitHub commands listed below:

- Git clone
- Git merge
- Git push
- Git commit

- Git add .

### 2.2.3 Results

Nathaniel started with cloning a GitHub repository [4] that has the python programs with the code that the first team of this project worked on. Once Nathaniel cloned the GitHub repository he then proceeded to install all the packages required for the code to run. Nathaniel then encountered similar errors to the ones from the flight programs. But rather than the errors being caused by pygame, they were caused by cv2 that can be seen in Figure (number of figure) and



```

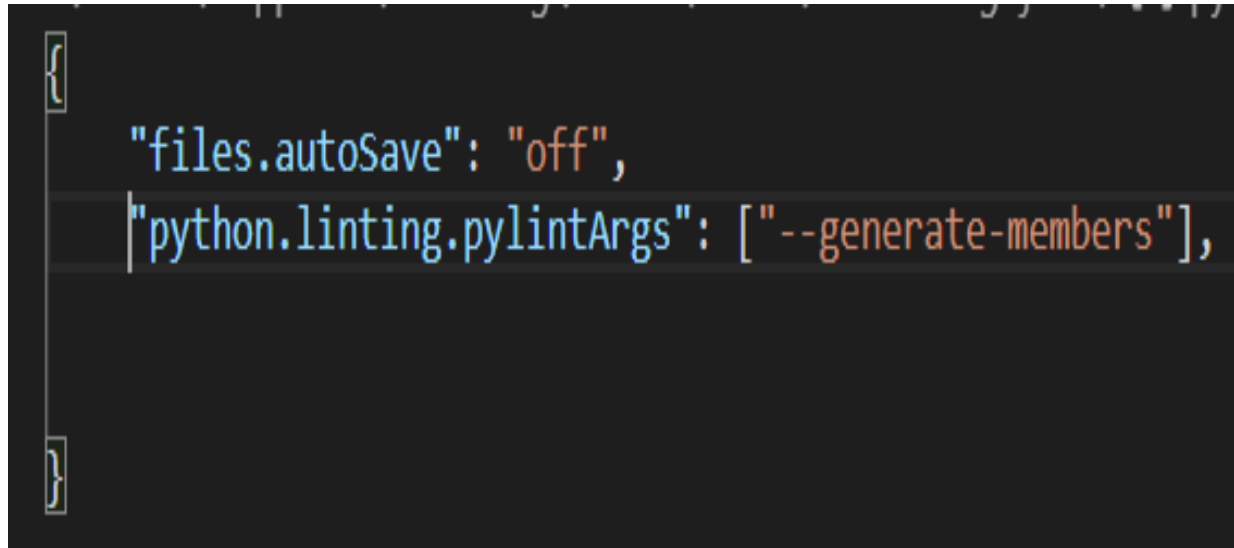
CountingPads.py Object Tracking 27
✗ Unable to import 'tello_drone' pylint(import-error) [3, 1]
✗ Module 'cv2' has no 'findContours' member pylint(no-member) [55, 27]
✗ Module 'cv2' has no 'RETR_TREE' member pylint(no-member) [55, 50]
✗ Module 'cv2' has no 'CHAIN_APPROX_SIMPLE' member pylint(no-member) [55, 65]
✗ Module 'cv2' has no 'contourArea' member pylint(no-member) [56, 14]
✗ Module 'cv2' has no 'minAreaRect' member pylint(no-member) [61, 16]
✗ Module 'cv2' has no 'contourArea' member pylint(no-member) [61, 38]
✗ Module 'cv2' has no 'cvtColor' member pylint(no-member) [70, 12]
✗ Module 'cv2' has no 'COLOR_BGR2HSV' member pylint(no-member) [70, 33]
✗ Module 'cv2' has no 'inRange' member pylint(no-member) [73, 12]
✗ Module 'cv2' has no 'bitwise_and' member pylint(no-member) [76, 12]
✗ Module 'cv2' has no 'cvtColor' member pylint(no-member) [77, 12]
✗ Module 'cv2' has no 'COLOR_BGR2GRAY' member pylint(no-member) [77, 31]
✗ Module 'cv2' has no 'GaussianBlur' member pylint(no-member) [78, 12]
✗ Module 'cv2' has no 'threshold' member pylint(no-member) [79, 15]
✗ Module 'cv2' has no 'THRESH_BINARY' member pylint(no-member) [79, 44]
✗ Module 'cv2' has no 'CAP_PROP_FRAME_HEIGHT' member pylint(no-member) [92, 22]
✗ Module 'cv2' has no 'CAP_PROP_FRAME_WIDTH' member pylint(no-member) [93, 21]

```

*Figure 2.2.3.1 CV2 errors*

some files names were incorrectly imported so that needed a fix. Nonetheless the fix that was used by all group members in the flight program, also worked in as a fix for the errors in the

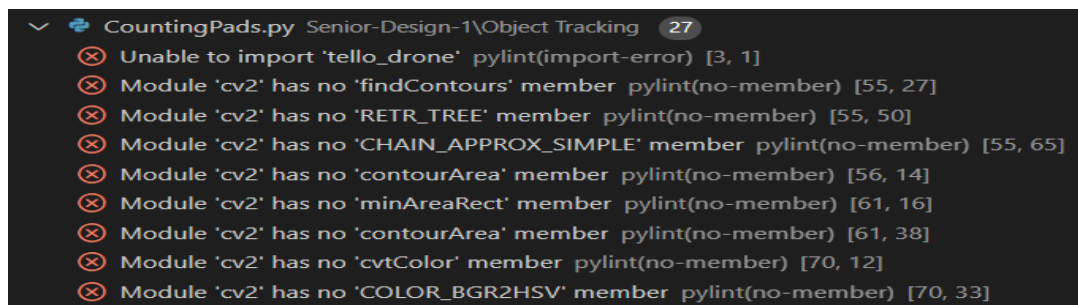
object detection and facial recognition programs, the fix can be seen in Figure (number of figure).



```
{  
    "files.autoSave": "off",  
    "python.linting.pylintArgs": ["--generate-members"],  
}
```

*Figure 2.2.3.2 CV2 fix*

Bradford Scrivner has also cloned the same GitHub repository that Nathaniel has spoken about before. The same error that was discovered from the flight program has appeared again which is shown in figure 3 with a module called “cv2” not being detected and thus causing errors. The fix that Nathaniel had supplied was thought to be it however, the fix did not result in any changes. Nathaniel suspects that it may be a file directory error that is causing this trouble.



```
CountingPads.py Senior-Design-1\Object Tracking 27  
⊗ Unable to import 'tello_drone' pylint(import-error) [3, 1]  
⊗ Module 'cv2' has no 'findContours' member pylint(no-member) [55, 27]  
⊗ Module 'cv2' has no 'RETR_TREE' member pylint(no-member) [55, 50]  
⊗ Module 'cv2' has no 'CHAIN_APPROX_SIMPLE' member pylint(no-member) [55, 65]  
⊗ Module 'cv2' has no 'contourArea' member pylint(no-member) [56, 14]  
⊗ Module 'cv2' has no 'minAreaRect' member pylint(no-member) [61, 16]  
⊗ Module 'cv2' has no 'contourArea' member pylint(no-member) [61, 38]  
⊗ Module 'cv2' has no 'cvtColor' member pylint(no-member) [70, 12]  
⊗ Module 'cv2' has no 'COLOR_BGR2HSV' member pylint(no-member) [70, 33]
```

*Figure 2.2.3.3 More CV2 errors*

## 2.3 Mission 3

Mission 3 is all about establishing a functioning body tracking script using python for the Tello drone. Bradford had discovered a library that would be most beneficial to the making of such script called ‘OpenPose’. OpenPose is a body detection library that is the first open-source real-time system for multi-person 2D pose detection, including body, foot, hand, and facial key points. At first Bradford will test OpenPose to determine if it is the right library to use for this script in areas such as robustness to movement, occlusion, and update frequency. Once this is determined OpenPose will be used in the creation of the required python script. The mission itself will involve a person’s body being able to be fully tracked. The drone itself will move according to where the person has positioned his body in front of the drone’s camera. If the person moves left or right the drone follows accordingly. If the drone moves forward the drone goes back and vice versa in the other direction. All the input that the drone’s camera will be receiving will be displayed on a computer as the drone is airborne.

Bradford will be working on this type of body tracking using his own body while Gilberto will be looking into a different library and method for body tracking in the meantime.

### **2.3.1 Objectives**

- Check functionality and stability of OpenPose
- Get OpenPose to detect a person’s body
- Get the drone to detect a person’s body with OpenPose
- Further develop skills and knowledge needed in Python
- Get the drone to move accordingly to a person’s body
- Get the drone to follow a person’s body
- Determine what happens when more than one body is detected in drone’s path.

### **2.3.2 Mission Success**



For this mission to be deemed successful the drone needs to be able to detect a person's body. The drone then needs to be able to move and follow the person's body accordingly. The mission is not a failure if the drone hits an object following a body or cannot determine what body to follow if more than body is detected.

### **2.3.3 Results**

After creating the Python code needed to establish mission 3 a bug was found. When trying to install the OpenPose library a problem was encountered when configuring the library to work with Python. In order to convert the library a GUI called 'CMake' is required in order to make the library usable to complete the final step. But while CMake configures the library it gets stuck in a section where it tries to download the program 'OpenCV'. OpenCV is a library of programming functions mainly aimed at real-time computer vision. This is needed for the project to work but is already installed from previous missions. Troubleshooting is required to fix this problem. In the time being facial tracking was pursued with the use of HAAR Cascade a machine learning object detection algorithm used to identify objects in an image or video. With this new option facial tracking proved to be very successful and with this success body tracking is now being perused as an optional alternative to OpenPose incase issues are not resolved.

## **2.4 Mission 4**

This mission will be the first mission that will not require the group members to use Tello drone. That is because this mission will involve a drone that the group will build. The group member, Nathaniel, will be working on a single drone. The goal of this mission is to build a drone and get it to fly.

As far as specifications go, Nathaniel will be working on a single drone. The drone will be developed for speed and racing.

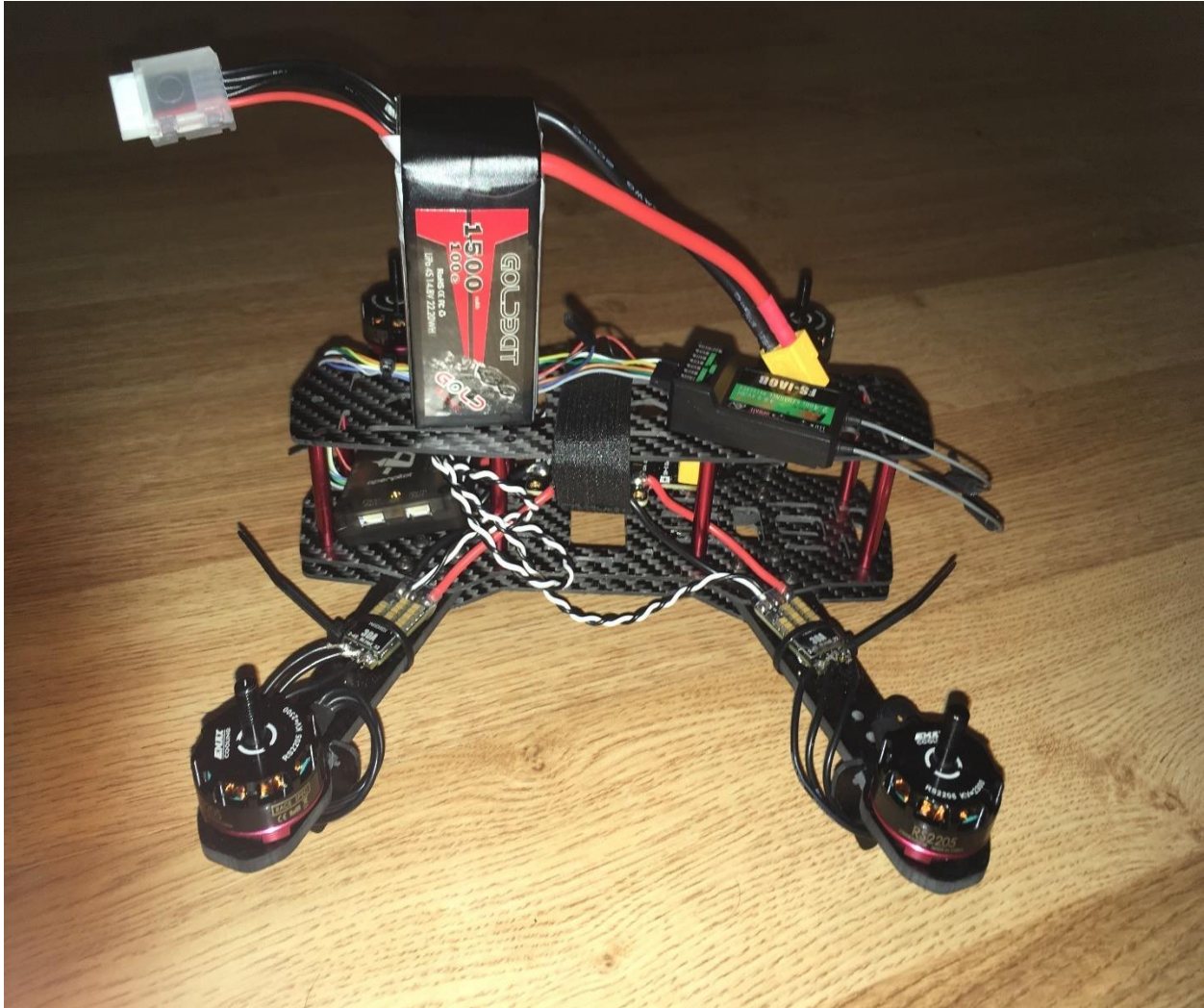
### **2.4.1 Objective**

- Get the drone to power on.
- Get the drone to fly in cardinal directions.
- Develop our skills and knowledge with electronic parts.
- Get drone to land.
- Get drone controller to work.
- Get drone to hover.

### **2.4.2 Mission Success**

In order for this mission to be a success the drone should be able to be flown using a controller. This mission will not be a failure if the drone does not have autonomous functions because this mission is just to build a drone capable of flight. The success of this mission will depend on how well the group can understand the electronic components that are required to get the drone to be capable of flight.

### 2.4.3 Results



*Figure 2.4.3.1 Current state of drone build*

Due to the unforeseen pandemic, known as coronavirus or COVID-19, hardware was put to a halt and was switched to mission 5. But after further development on this mission, Nathaniel was able to wire up all the components properly and Bradford was able to use the LibrePilot software to get the drone to fly off the ground.

### 2.5 Mission 5

This mission was not part of the original plan. However, due to unforeseen effect of COVID-19 this mission was created. This mission is the idea of object avoidance. The mission will consist of the drone flying straight toward a goal with objects in its path to avoid. If the drone is able to not hit an object in its path, then the team will move to get the drone to avoid objects that surround it rather than simply in its forward path.

As far as specifications go, Nathaniel will be working on an object avoidance system.

### **2.5.1 Objective**

- Get the drone to fly forward toward goal.
- Get the drone to avoid object in its path.
- Further develop our skills and knowledge in Python.
- Get the drone to reach the goal.
- Get the drone to reach the goal, while avoiding objects.

### **2.5.2 Mission Success**

In order for this mission to be a success the drone should be able to reach the goal and not hit a single object. This mission will not be a failure if the drone hits a wall or bumps an object it has detected once. The success of this mission will depend on the capabilities of the drone and the mathematics running the avoidance.

### **2.5.3 Trails**

All the different values for distance of a movement command in centimeters and the amount of time for a delay in seconds with their reasons for being chosen or not chosen, are shown in tables 2.5.3.1, 2.5.3.2, 2.5.3.3, 2.5.3.4, 2.5.3.5, 2.5.3.6, 2.5.3.7.

<b>Path forward (first_OA/second_OA)</b>	
70 cm	Allow for the object detection to run smoothly along with the avoidance.
<70 cm	Worked but the time for it to reach the end foal took a long time.
>70 cm	Worked but did not give enough time to detect an object when moving along the path.

*Table 2.5.3.1 Path movement forward for both first and second OAs*

60 cm	Best value in room size 32 x 27 sq. ft.
<60 cm	Works but needs to run dodge movement more than twice, causing unnecessary movement.
>60 cm	Collides with wall on multiple dodge attempts in room size 32 x 27 sq. ft.

*Table 2.5.3.2 First Avoidance dodge distance trails*

<b>dodge (second_OA)</b>	
90 cm	Best value to aid re-center & forward in room size 32 x 27 sq. ft.
<90 cm	Works but does not allow a predefined re center distance to work, causing the drone to be off the original path.
>90 cm	Collides with wall on multiple dodge attempts in room size 32 x27 sq. ft. Does not work with a re-center.

*Table 2.5.3.3 Second Avoidance dodge distance trails*

<b>re-center (second_OA)</b>	
90 cm	Matches the best value for dodge, so it allows the drone to re-center onto the original path.
90*n cm	The n variable would be the amount of times the drone dodged but would run a larger value than supposed to.

*Table 2.5.3.4 Second Avoidance re-center distance trails*

<b>avoid forward (second_OA)</b>	
120 cm	Best value to avoid collision with an object and wall in a room 32 x 27 sq. ft.
<120 cm	Will not fully avoid an object, and collide with the object upon re-centering.

>120 cm	Unnecessary extra distance that collides with the wall in a room 32 x 27 sq. ft.
---------	--

*Table 2.5.3.5 Second Avoidance forward distance trails*

<b>first movement delay (second_OA)</b>	
5 seconds	Works best in allowing the algorithm to execute the first & second movements.
4 seconds	Executed the first (dodge) and third(re-center) movements and excludes the second(forward) movement.
6 seconds	Works, but adds more delay time that is necessary. Slowing down the system

*Table 2.5.3.6 Second Avoidance first movement delay time trails*

<b>second movement delay (second_OA)</b>	
4 seconds	Works best in allowing the algorithm to execute the second & third movements.
3 seconds	Executed the first (dodge) and second(forward) movements and excludes the third(re-center) movement.
5 seconds	Works, but adds more delay time that is necessary. Slowing down the system.

*Table 2.5.3.7 Second Avoidance second movement delay time trails*

## 2.5.4 Results

This mission ended in success and with plans for future work. The OAs currently flies in a forward path then if a yellow object enters its path, it will dodge then re-center. This allows it to fully move around the object while staying on its path. Right now, the drone's start and stop system, is starting when the command "python main.py" is executed then stops when the 'q' key is entered. Multiple tests were run to determine its consistency and functionality.

## 2.6 Mission 6

This mission will be about body recognition and tracking using deep learning neural networks, we will use a trained network called MobileNet SSD. SSD is designed for object

detection in real-time, it uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. MobileNet is the neural network that is used for classification and recognition whereas the SSD is a framework that is used to realize the multi-box detector.

For this project we will not be training a Neural Network ourselves from scratch since that would take a lot of time, instead we will be using an already trained network called MobileNet SSD, using this network we will then implement the code to successfully detect a person's body, then once a body is detected, we will work on tracking it. The challenge for this will be creating a code using an algorithm that doesn't lose track of the person's body so easily, for this the team will have to explore the different algorithms and decide which one will be the best to use.

### **2.6.1 Objective**

- Explore the different algorithms for body recognition.
- Get a working algorithm to detect a person's body.
- First detect a body using the laptop webcam.
- Get the drone to detect a person's body with this algorithm.
- Be able to follow a person successfully.
- Determine what happens when more than one body is detected in drone's path.
- Do not lose track of a body easily when its moving at a higher speed.

### **2.6.2 Mission success**

For this mission to succeed the drone needs to be able to detect a person's body and track it. The drone then needs to be able to move and follow the person's body accordingly. The

mission is not a failure if it loses track of a person or body when more than one person/body is detected.

### **2.6.3 Results**

After doing the research needed to start this mission, the python code that we wrote to detect bodies is working successfully from the laptop webcam, the next step is almost done, which is writing a working algorithm to track the body. Testing on the drone will begin soon, this will determine if this mission was successful.

## **2.7 Mission 7**

This mission was brought to attention by Dr. Emmett Tomai. Nathaniel Lozano was the one who thought up the idea of a grid solution to this system. This mission is the idea of a spatial visual awareness system. The mission will consist of the drone free flying in a predefined space and flying in a path from two arbitrary points in a space. In both types of drone movement, the system will be allowing the drone to detect, avoid and keep track of objects. For flying on a path, a pathfinding algorithm will be necessary

As far as specifications go, Nathaniel will be working on the Spatial Visual Awareness system.

### **2.7.1 Objective**

- Successfully fly freely in a predefined space.
- Successfully fly in a path from point a to point b, in a predefined space.
- Get system to detect, avoid and keep track of objects.
- Successfully use a pathfinding algorithm



## 2.7.2 Mission Success

For this mission to be a success the drone should be able to move freely and on a path from point a to point b in a predefined space, while detecting, avoiding, and keeping track of objects. This mission will not be a failure if drone has inaccurate movement inside the predefined space. The success of this mission will depend on the capabilities of the drone and the mathematics running the predefined space and tracking.

## 2.7.3 Trails

All the different values for distance of a movement in the grid in centimeters, dimensions of the grid in real-life, the amount of time for a delay in seconds for iteration, and width of obstacles to be dodged with their reasons for being chosen or not chosen, are shown in tables 2.7.3.1, 2.7.3.2, 2.7.3.3, 2.7.3.4.

Drone movement in grid	
20 cm	Allow for the object detection to run smoothly but with the drone moving from cell to cell having a bit of inaccuracy. In comparison it has the least inaccuracy. Allowed real-life grid to be 1 meter by 1 meter.
50 cm	Worked but movement from cell to cell caused inaccuracy. Caused real-life grid to be 2.5 meters by 2.5 meters.
100 cm	Worked but the inaccuracy for cell to cell movement was too large. Caused real-life grid to be 5 meters by 5 meters.

*Table 2.7.3.1 Drone movement from cell to cell in the grid*

Dimensions of grid	
1 by 1 meter	Allow for the object detection to run smoothly but with the drone moving from

	cell to cell having a bit of inaccuracy. . In comparison it has the least inaccuracy. Allowed drone movement to be 20 cm
2.5 by 2.5 meters	Worked but movement from cell to cell caused inaccuracy. Caused drone movement to be 50 cm
5 by 5 meters	Worked but the inaccuracy for cell to cell movement was too large. Caused drone movement to be 100 cm

*Table 2.7.3.2 Dimensions of real-life grid*

<b>Width of obstacle</b>	
3.5 in	Allows enough space in the 20 cm by 20 cm cells for the drone to avoid.
5 in	Camera has problems detecting the object when approaching.
7 in	Takes up too much space in the 20 cm by 20 cm cells for the drone to avoid.

*Table 2.7.3.3 Width of obstacle to be dodged*

<b>Time delay (movements)</b>	
5 seconds	Works best in allowing the algorithm to execute the movement, detection/avoidance, and pathfinding
4 seconds	Causes some stuttering issues for when the drone is trying to move
6 seconds	Works, but adds more delay time that is unnecessary. Slowing down the system rather than improving the speed at which the drone can avoid the obstacle that is located in front of it.

*Table 2.7.3.4 Time delay for each iteration of movement*

## 2.7.4 Results

This mission ended in success and with plans for future work. The system currently flies in a forward path from point a to point b then if a yellow object enters its path, it will dodge then calculate a new path using the same a star pathfinding algorithm that was used to calculate the original path from point a to point b. This allows it to fully move around the object while staying on a path that leads to point b. The system also has working free roam movement. It allows the drone to move in any direction and dodge an obstacle in front of the drone. The only problem with this mission is the inaccuracy of the drone's movement. The drone's movement match that of the system, but in the distance moved is inaccurate for most cases. This does not cause this mission to be a failure, it simply provides more evidence that the reference point system is needed. Right now, the drone's start and stop systems include, starting when either command "python free\_roam\_grid.py" or command "python pathfinder\_grid.py" is executed. For the free roam grid, it will stop when the q key is pressed. For the pathfinder grid, it will stop when it has reached the end of the path (i.e., point b) Multiple tests were run to determine its consistency and functionality.

## 2.8 Mission 8

The mission will be about using object recognition and tracking using OpenCV in order to create a guidepost that would enable a Tello drone to more accurately achieve a marked distance. If you did not know OpenCV is a library of programming functions mainly aimed at real time computer vision. Due to prior testing before the creation of this mission it was determined that the actual distance marked when encoding is not entirely accurate. Due to this realization it was decided

that a guidepost of sorts would be created to ensure that whatever distance the drone needed to travel was achieved with upmost perfect accuracy in mind.

### **2.8.1 Objective**

- Drone detecting specific objects
- Drone will mark distance before and after transit to determine distance accuracy
- Drone moves towards objects to exactly where they lay

### **2.8.2 Mission Success**

For this mission to succeed the drone needs to be able to detect an object guidepost and fly towards it while correcting its own flight if needed. It is to also mark its distance before and after flight to object to determine accuracy of flight. The mission is not a failure if an object is suddenly taken away or moved while drone is in flight.

### **2.8.3 Results**

This mission was successful and with plans for future work. The system currently flies in a forward path from point a (starting point) to point b (reference point) with a yellow object being the reference point, it will begin by taking off from its starting point and then mark the focal length distance from the drone's camera to the yellow object (reference point). The only problem with this mission is still the inaccuracy of the drone's movement. Accuracy of the drone has indeed improved from numerous tests by using an object as a reference point. However, the drone is still inaccurate while traveling to the reference point and arriving at the reference point. This does not cause this mission to be a failure, with the increased accuracy obtained by using a reference point it provide the necessary accuracy for the pathfinding grid. Right now, the drone's

start and stop systems include, starting with command “python Waypoint.py” . For the waypoint, it will stop when the q key is pressed. Multiple tests were run to determine its consistency and functionality.

### **3. Team Management**

#### **3.1 Workload**

As we talked to our advisors, we were able to decide the amount of work we will be able to complete for this project. That is how we settled on the missions. But with the amount of the missions we had, we then had to divide up the work based on how much time each member could put in the project. Along with the missions we had all the documentation that we had to write up. The assigned work that each member worked on will be listed below:

- Final Report – Nathaniel, Gilberto, & Bradford
- Setup Guides - Nathaniel, Bradford, & Gilberto
- Mission 1 – Nathaniel, Bradford, & Gilberto
- Mission 2 – Nathaniel, Bradford, & Gilberto
- Body tracking – Gilberto & Bradford
- Drone build - Nathaniel
- Object Avoidance- Nathaniel
- Midterm Poster – Nathaniel, Bradford, & Gilberto
- Weekly reports - Nathaniel, Bradford, & Gilberto
- Gantt Chart – Nathaniel
- Block Diagram – Nathaniel

- Flow Charts – Nathaniel
- Final Report – Nathaniel, Bradford
- Spatial Visual Awareness – Nathaniel
- Reference Point (Waypoint) - Bradford

## 3.2 Gantt Chart

The full Gantt chart will be shown in figure 3.2.1, 3.2.2, and 3.2.3.

Drone Flight	15d	01/27/20	02/14/20		Nathaniel, Bradford, Gilberto
Acquire Open-Source Code	5d	01/27/20	01/31/20		Nathaniel, Bradford, Gilberto
Testing Code	5d	02/03/20	02/07/20	2	Nathaniel, Bradford, Gilberto
Debugging any errors	5d	02/10/20	02/14/20	3	Nathaniel, Bradford, Gilberto
Object Detection & Facial Recognition	10d	02/17/20	02/28/20	1	Nathaniel, Bradford, Gilberto
Acquire Open-Source Code	2d	02/17/20	02/18/20	4	Nathaniel, Bradford, Gilberto
Testing Code	3d	02/19/20	02/21/20	6	Nathaniel, Bradford, Gilberto
Debugging any errors	5d	02/24/20	02/28/20	7	Nathaniel, Bradford, Gilberto
Improvements	15d	03/02/20	03/20/20	5	Nathaniel, Bradford, Gilberto
Object Maneuvering	5d	03/02/20	03/06/20	8	Bradford
Body Tracking	10d	03/09/20	03/20/20	10	Gilberto
Build Drone & Fly Drone	30d	03/02/20	04/10/20	5	Nathaniel, Bradford, Gilberto
Assemble/solder Electronics	5d	03/02/20	03/06/20	8	 Nathaniel Lozano
Test Electronics	5d	03/09/20	03/13/20	13	Nathaniel
Fix any misconnected electronics	5d	03/16/20	03/20/20	14	Nathaniel
Testing flight code	5d	03/23/20	03/27/20	15	Nathaniel
Debugging any errors	10d	03/30/20	04/10/20	16	Nathaniel

*Figure 3.2.1 Gantt chart part 1*

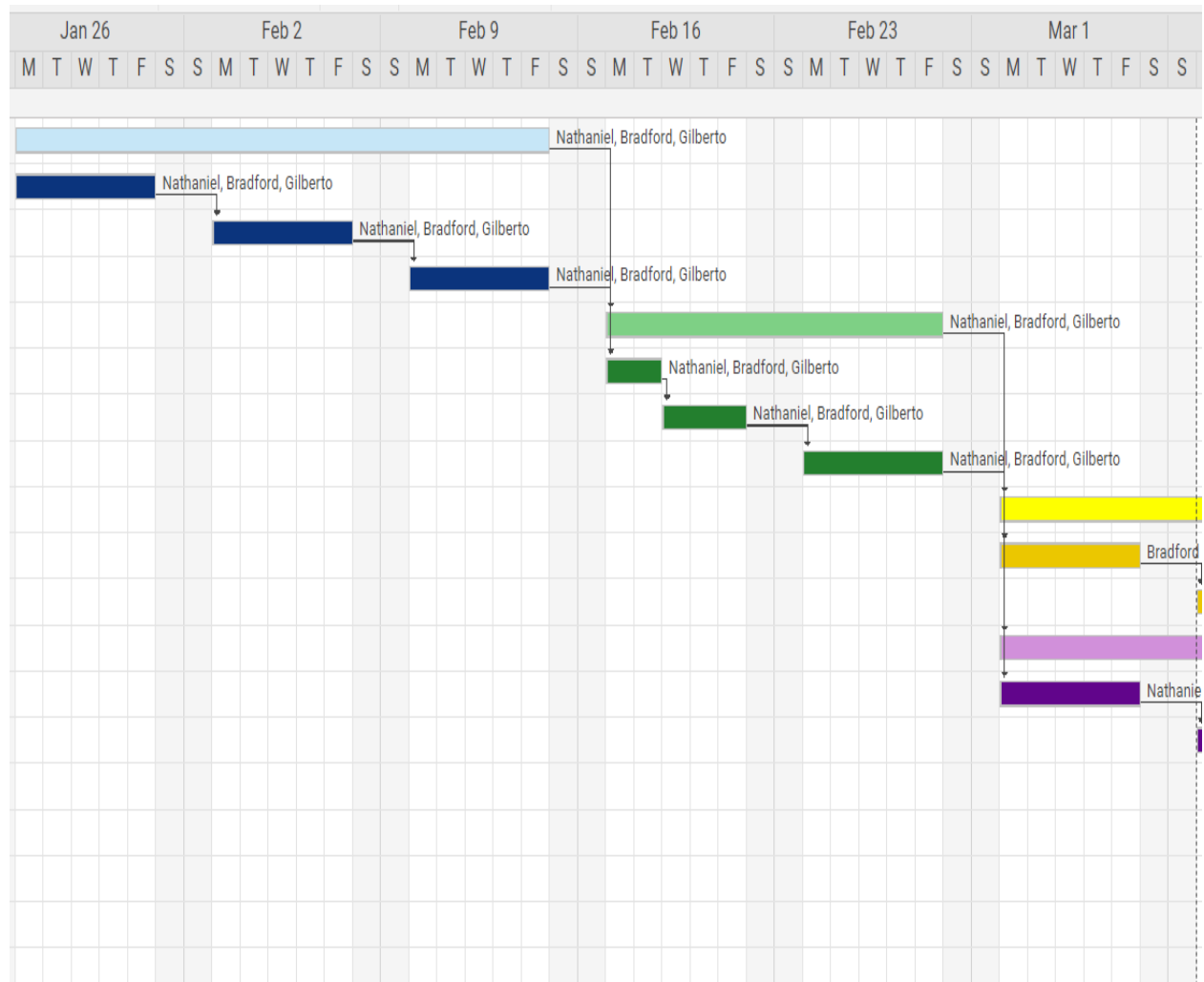


Figure 3.2.2 Gantt chart part 2

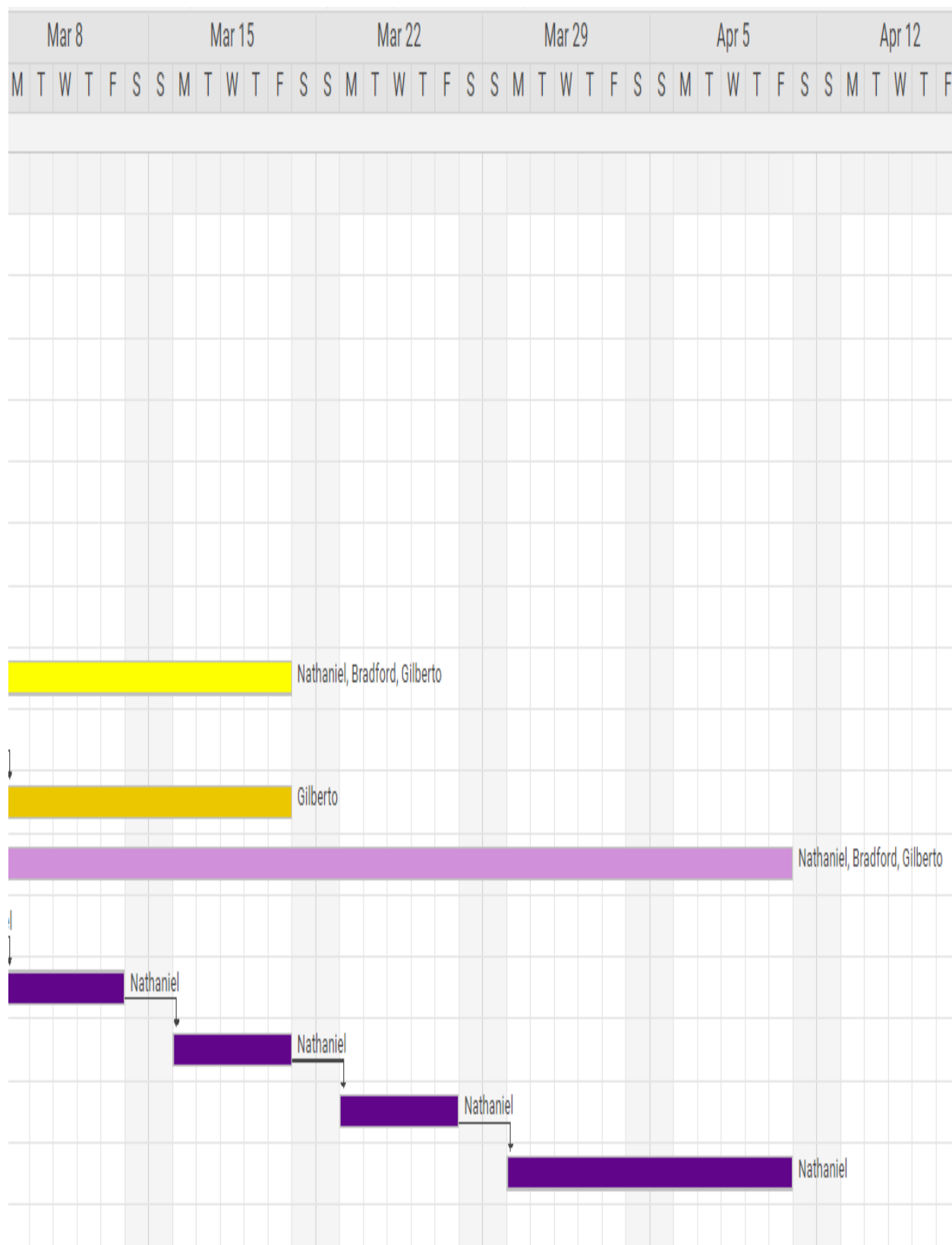


Figure 3.2.3 Gantt Chart part 3



Brainstorm new systems	1d	08/27/20	08/27/20		Group
Spatial Visual Awareness	67d	08/28/20	11/30/20	1	Nathaniel
Solution for new system	10d	08/28/20	09/10/20	1	Nathaniel
Work on calculating distance	5d	09/11/20	09/17/20	3	Nathaniel
Build grid for system	5d	09/18/20	09/24/20	4	Nathaniel
Test and debug free roam movement in grid	5d	09/25/20	10/01/20	5	Nathaniel
Build demo for free roam grid	10d	10/02/20	10/15/20	6	Nathaniel
Test and debug pathfinding movement	10d	10/16/20	10/29/20	7	Nathaniel
Wire up drone build	5d	10/22/20	10/28/20		Nathaniel
Cleanup documentation	15d	10/29/20	11/18/20		Nathaniel
Develop reference points for movement inaccuracy	32d	10/08/20	11/20/20		Bradford
Get drone build to fly using librepiplot	10d	11/05/20	11/18/20		Bradford

Figure 3.2.4 Gantt Chart part 4

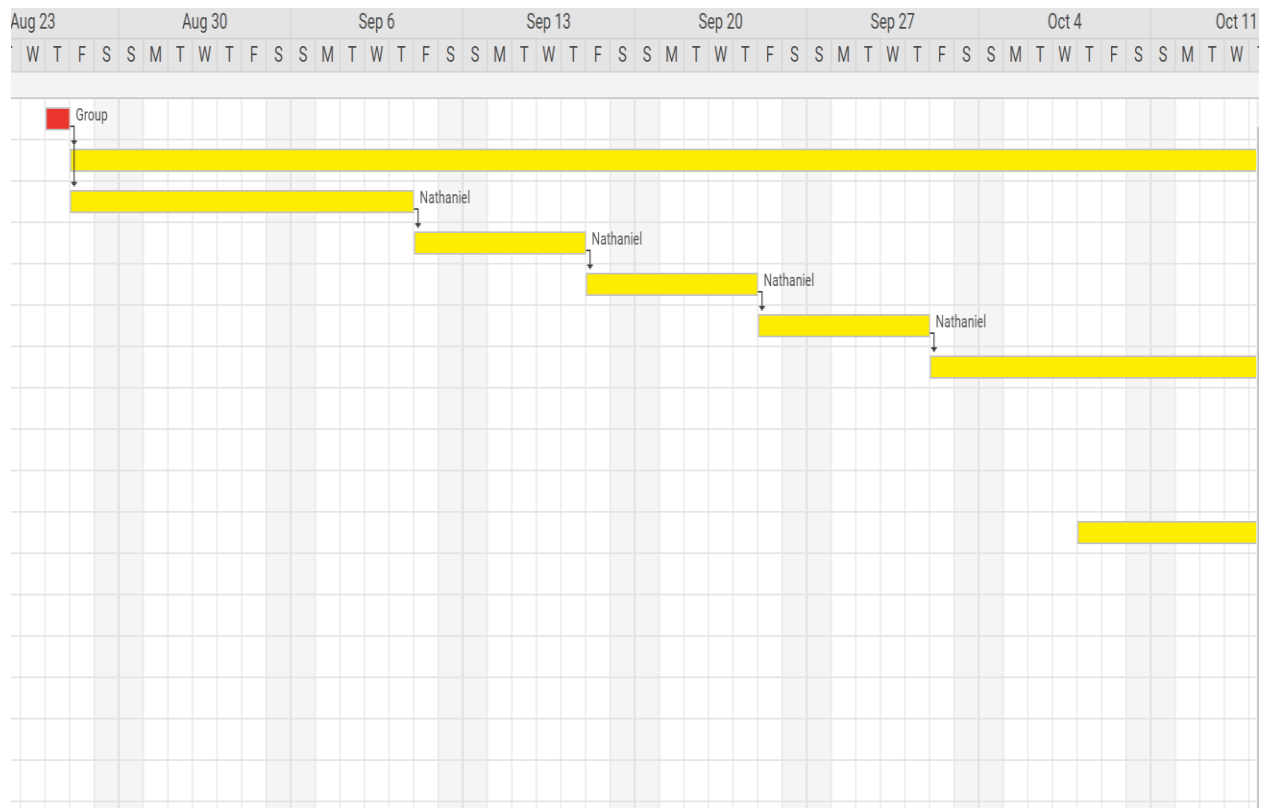
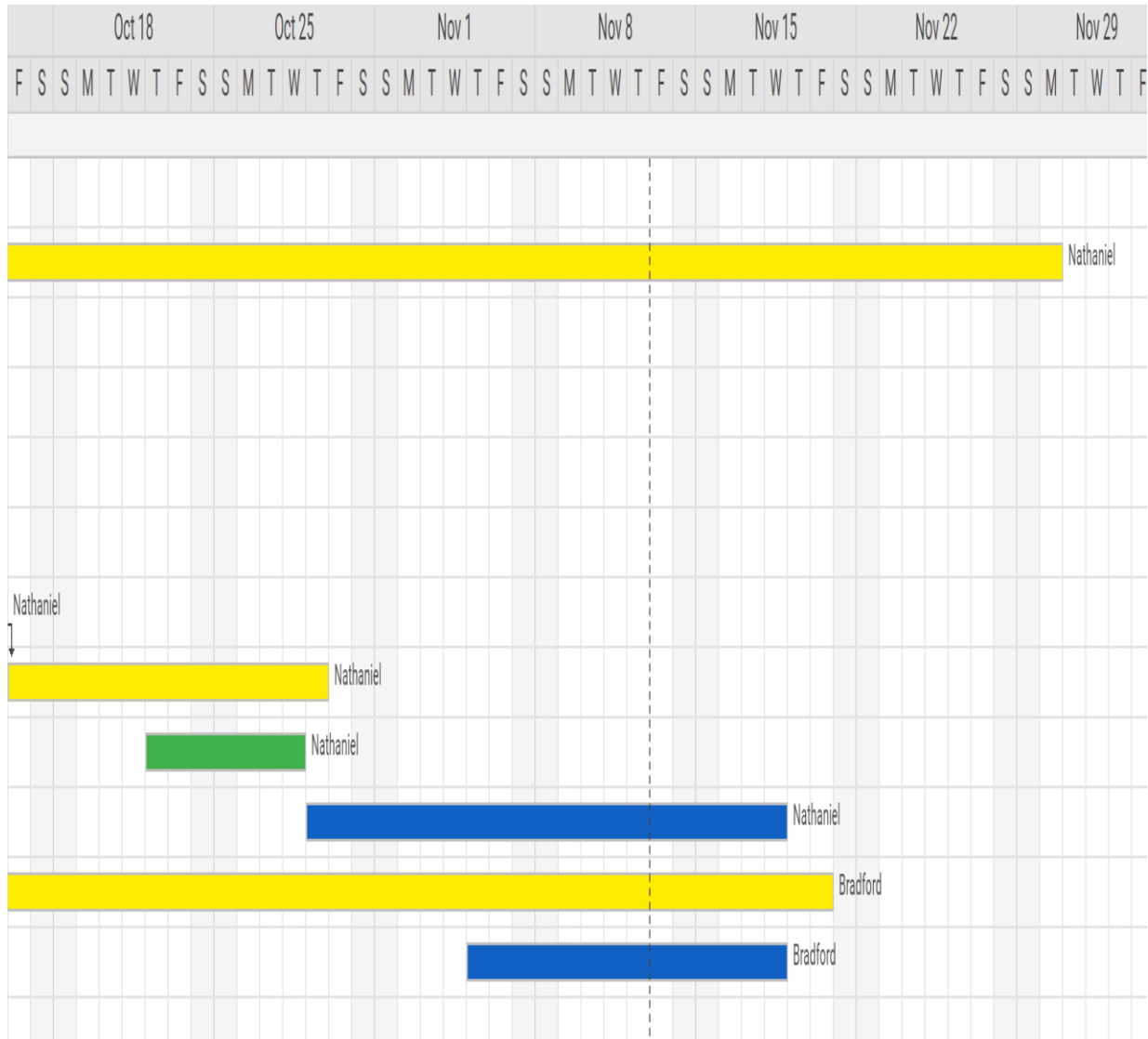


Figure 3.2.5 Gantt Chart part 5



*Figure 3.2.6 Gantt Chart part 6*

### 3.3 Parts Costs

The project is being funded by the computer science department so the parts that are truly necessary to build the drone did not exceed our budget. Any costs that were required came from the hardware portion of the project. The software portion had free compilers and editors so that required no costs, just computers. Since all group members had computers to work on, there were no issues. Nonetheless we still decided to include a table to show the costs of parts, since

part of the project is meant for an amateur drone enthusiast to make developments on this project. The parts and cost can be seen in table 3.4.1.

<b>Parts List</b>	
<b>Name</b>	<b>Cost(\$)</b>
Readytosky 250mm frame	19.99
EMAX RS2205-2300KV 3-4S	57.99
midici BLHeli-32 30A ESC (4pk) 3-4S	41.99
5045 (x16)	14.99
4S 1500mAh 100C 183g (2pk) XT60	37.99
Storage bag	12.99
T-plug connectors (3 pair pk)	8.9
SKYRC iMAX B6 charger	39.66
AC power supply	20.8
CC3D Revo stm32f4 w/ OPLink	50.22
Flysky FS-i6X + FS-iA6B RX	49
Matek PDB-XT60 w/BEC (5V and 12V)	7.99
<b>Total Cost: \$ 362.51</b>	

*Table 3.4.1 Parts List*

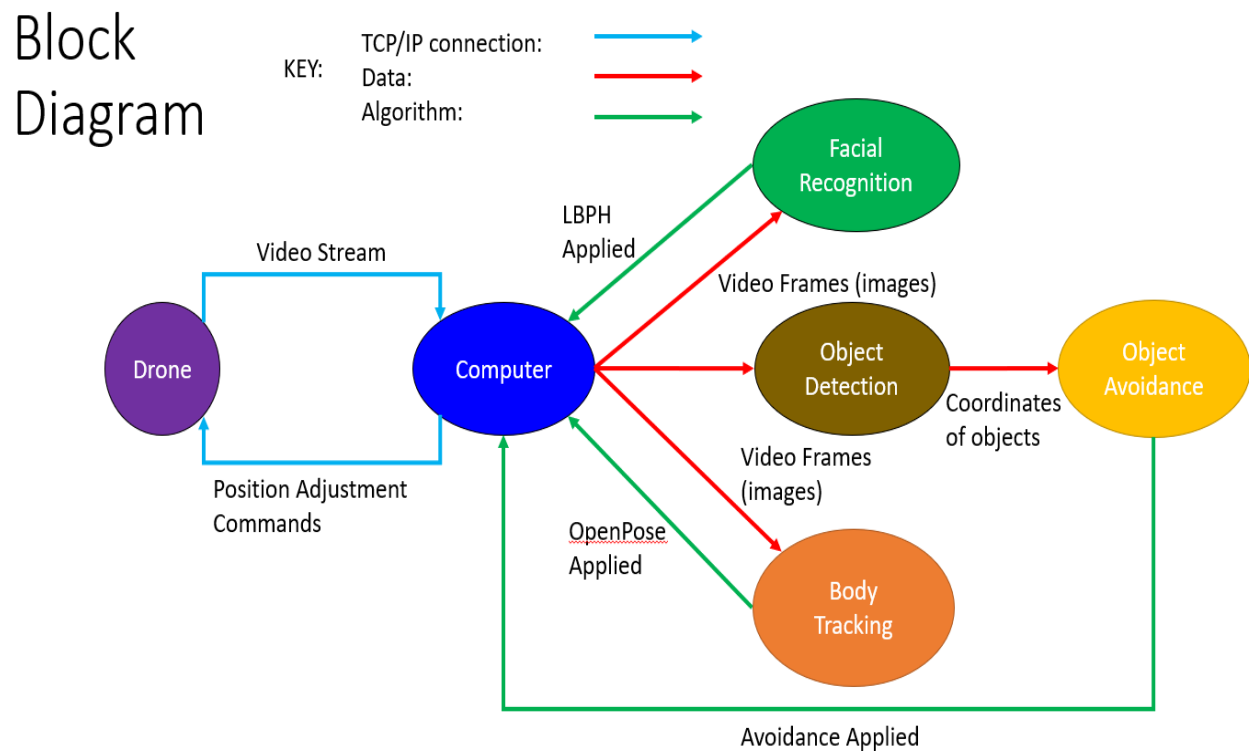
While some of the parts used may be expensive, when looking at the parts from an industry perspective the price can be brought down substantially. For instance, an electronic speed controller (ESC) can be mass produced and the price will be dropped substantially. The flight controller used could be designed to exclude the extra ports that are not being used in this

project. The PCB will be much cheaper to mass produce due to the fabrication process. To reiterate a previous statement about funds, there is no need to worry. Thanks to the generosity being put into this project by Dr. Tomai, we were able to have substantial funds.

## **4. Design**

### **4.1 Block Diagram**

The block diagram is shown in figure 4.1.1. All blocks in the diagram will be analyzed in another section.



*Figure 4.1.1 Block Diagram*

### **4.2. Software**

The software will include multiple types of systems that revolve around image processing technology on drones.

#### **4.2.1 Overview**

Nathaniel will be working on the object avoidance software (OAS). This portion of the software came to fruition, because COVID-19 halting Nathaniel's progress with the hardware build for this project. OAS is centralized around a color object detection system. Once the system determines a colored object is in the path of the drone, it will dodge the object. After the object is dodged, the drone will continue moving in a forward direction. Once the base level goal is achieved, Nathaniel will attempt to further the OAS by making additions to improve performance.

Nathaniel has also taken on the spatial visual awareness system (SVAS). This system came to

Bradford will be working on the software portion of the drone needed for body tracking a topic that the team decided to pursue. In order to accomplish this Bradford discovered a library of code called OpenPose. This will allow a person's body to be fully tracked. The drone itself will move according to where the person has positioned his body in front of the drone's camera. If the person moves left or right the drone follows accordingly. If a person were to move forward the drone moves back and vice versa if a person were to move back. All the input that the drone's camera will be receiving will be displayed on a computer as the drone is airborne. In the end drone will be able to follow a person's body and video live stream will be displayed on a computer.

Gilberto will be working on another version of body recognition and tracking using a deep learning Neural Network, for this part we will not be training a Neural Network ourselves from scratch since that would take a lot of time, instead we will be using an already trained network to detect body called MobileNet SSD, using this trained network he will have to write the code to successfully detect a person's body, then once a body is detected, he will work on tracking it. This approach will first detect a body, then proceed to track it, the end goal is to have it follow a person even if its for a small amount of time.

#### **4.2.2 Body tracking OpenPose**

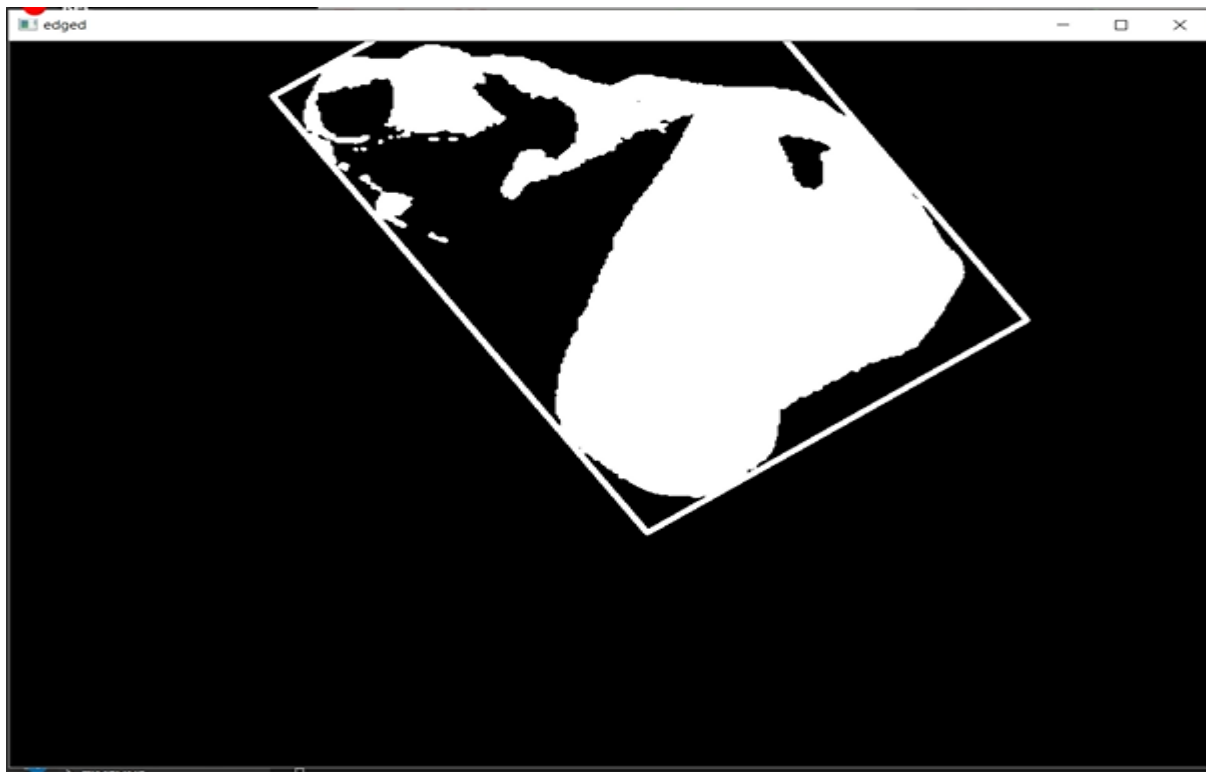
OpenPose represents the first real-time multi-person system to jointly detect human body, hand, facial, and foot key points (in total 135 key points) on single images. It works with Ubuntu (14, 16), Windows (8, 10), Mac OSX, and Nvidia TX2 operating systems and is compatible with Python and C++. With this existing library it makes it very possible to track an entire person's body. It can recognize multiple people at a time however, performance will be depending on how many people are in the frame of the camera of the drone. To use this library, it requires the user's computer to have a decent high-end GPU or CPU. It can be used with a lower end GPU or CPU, but the frames given for the live feed of information that will be shown on the computer will be poor.

#### **4.2.3 Object Avoidance**

Since the OAS revolves around the color system, there were predetermined RGB values that needed to be predefined in the program, shown in figure 4.3.1. The values are predefined to

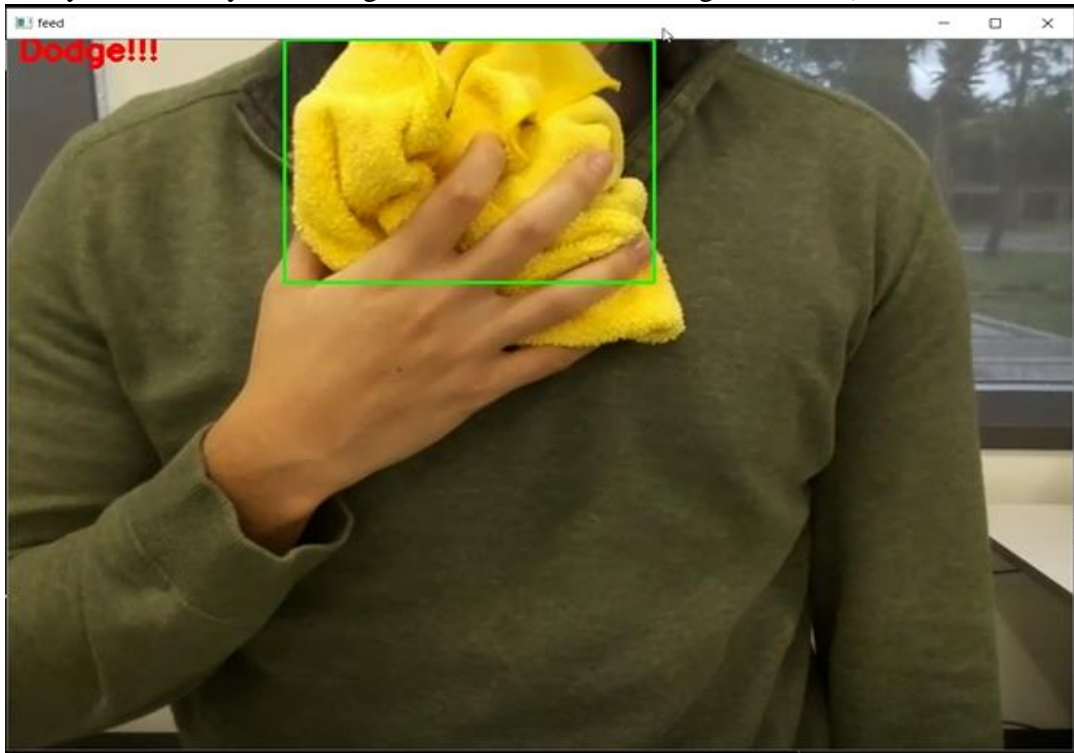
```
#yellow color detection range  
l_b = np.array([22,153,62])  
u_b = np.array([80,255,255])
```

*Figure 4.2.3.1 Yellow RGB Range*



*Figure 4.2.3.2 Threshold camera feed*

detect yellow. The yellow range is used to allow the usage of HSV (hue, saturation, and



*Figure 4.2.2.3 RGB Camera feed*

value) edge detection. HSV edge detection thresholds the video stream from the drone, based on the yellow range and is shown in figure 4.2.3.2. Then a box will now be able drawn on the yellow object and that in return is used to calculate an area, this is shown in figure 4.2.2.3. The area will be used to determine if an object is in the path of the drone.

When the OAS determines that the object is in the path of the drone, it will run the dodge function. The dodge function will move the drone away from the object and then once it has determined if the object is not still in its path, it will continue moving forward. All this is done will the program is using the camera feed from the drone. Too make this program run more smoothly and to accomplish the goal that was setup for OAS, multiple libraries were used and are shown in figure 4.3.2.



```
import numpy as np
import cv2
import imutils
import tello_drone as tello
import time
```

*Figure 4.2.3.4 Libraries used to build OAS*

#### **4.2.3.1 Flowchart**

The flow chart for the OAS will be shown in figure 4.2.3.1.1 This flow chart will show the OAS from start to finish, with both the HSV edging and avoidance algorithm.

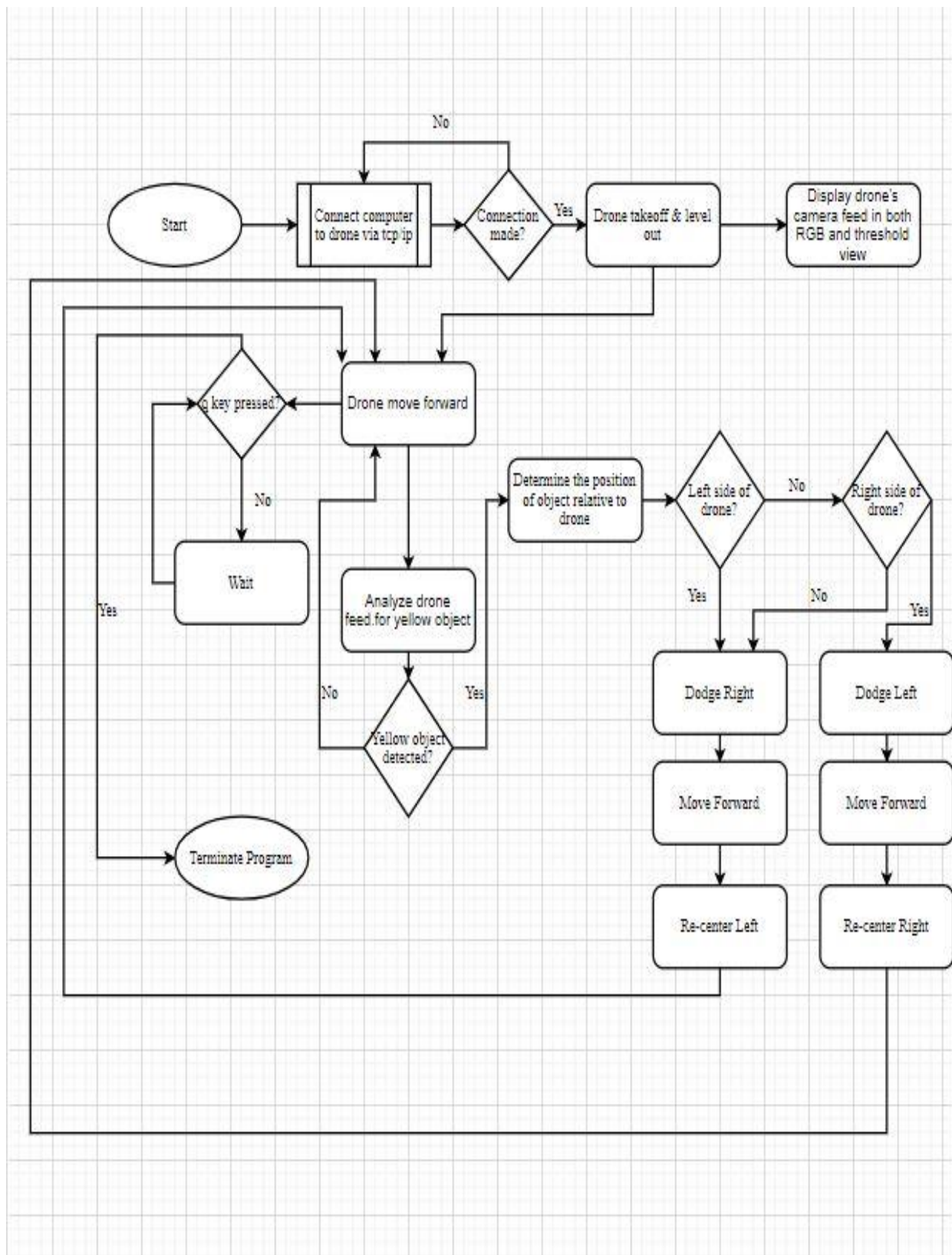


Figure 4.2.3.1.1 Avoidance Flowchart

## 4.2.4 Body Recognition with MobileNet SSD

Single Shot Detectors (SSD) and MobileNets. When combined together these methods can be used for super-fast, real-time object detection on resource constrained devices, from there we'll discover how to use OpenCV's DNN (Deep Neural Networks) module to load a pre-trained object detection network.

Like previously mentioned, if we combine both the MobileNet architecture and the Single Shot Detector (SSD) framework, we arrive at a fast, efficient deep learning-based method to object detection. The end result is a deep learning-based object detector that can process approximately 6-8 FPS (depending on the speed of your system, of course).

## 4.2.5 Spatial Visual Awareness

This system will need a space to work with, the space will be made as a grid shown in figure 4.2.5.1. The grid is a 7 by 7 matrix, with a border of 1 values and innards of 0 values. The

```
1111111
1000001
1000001
1000001
1000001
1000001
1111111
```

*Figure 4.2.5.1 7 by 7 matrix used by system*

system will keep track of the drone's position inside the grid by changing the x and y coordinates when the drone moves. Also, the drone will not always be facing the direction it started facing. So, the system will keep track of the direction by updating the direction value every time the

drone makes a movement that will alter its real-life direction (i.e., rotating clockwise). The system will recognize the drone is being blocked by an obstacle, when the front position is a 1 value. The system will allow the drone an option of free roam movement, that will allow the drone to move onto any cell in the matrix that is not blocked. Another option is a path driven movement. This will allow the drone to move from two This system uses the same HSV edge detection to detect objects. However, this system uses a different method for determining when to avoid an obstacle, and it also keeps track of objects. The drone will dodge an obstacle when placed directly in front of the drone. The way this system knows if an obstacle is directly in front of it, is by calculating the distance between itself and the obstacle. The equations in figure 4.2.5.1, shows just how this distance is calculated. The variables are defined as follows. Variable  $r$  is the ratio of  $W_{po}$ ,  $D_p$ , and  $W_{ro}$ .  $W_{po}$  is the pixel width of the obstacle grabbed from the camera feed.  $D_p$  is the predetermined distance from the drone to the object.  $W_{ro}$  is the real object width and is equal to  $W_p$ , that is the predefined width of the object. Laslty  $D_b$  is the the distance between the drone and obstacle

$$r = (W_{po} * D_p) / W_{ro}$$

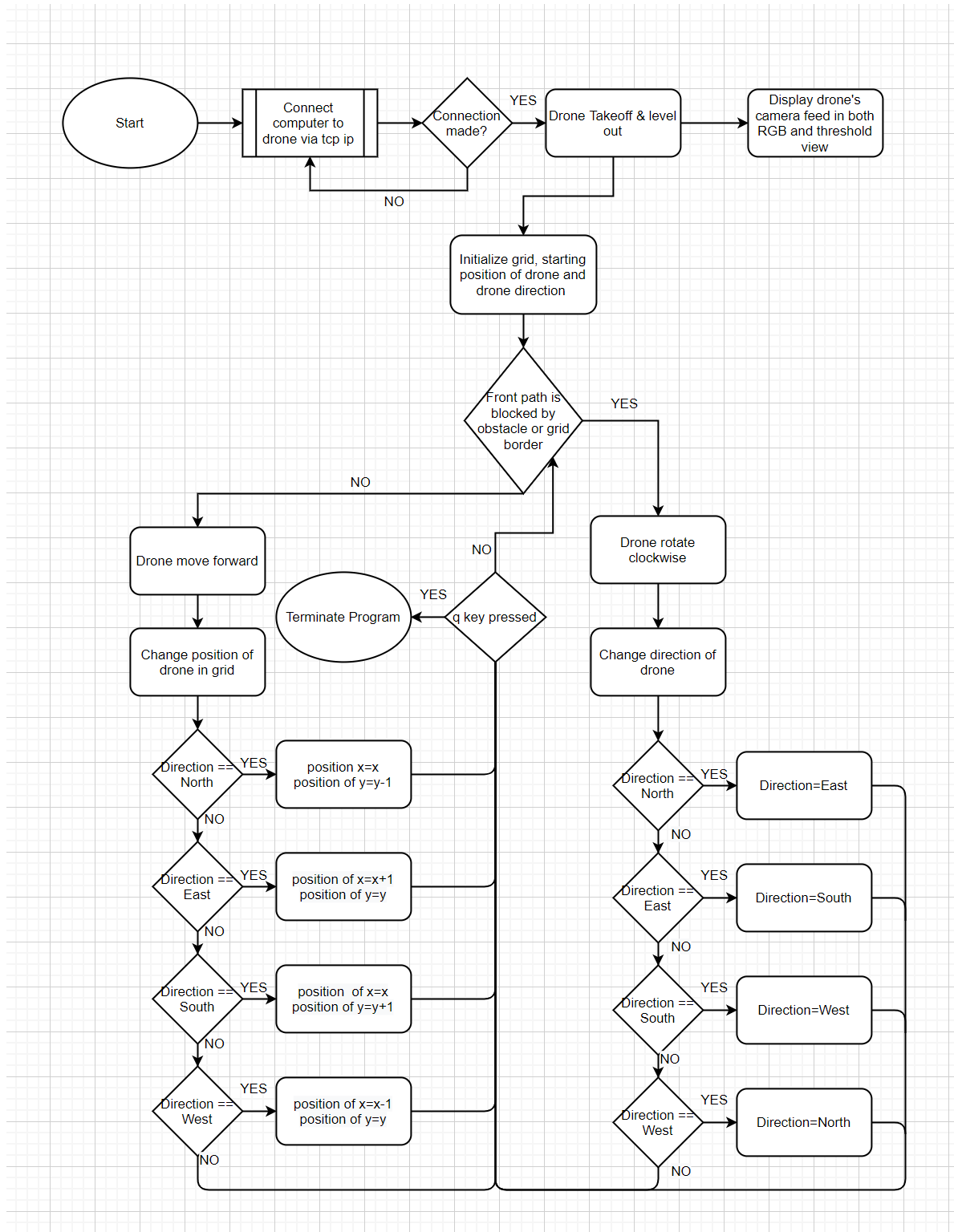
$$D_b = (W_p * r) / W_{po}$$

*Figure 4.2.5.2 Equations for calculating distance between object and drone*

that is calculated every iteration of drone movement. The purpose of  $r$  is so that it can be used as a constant for the  $D_b$  equation. With both constants  $W_p$  and  $r$ , along with  $W_{po}$  that grows in value when the drone moves closer towards object and shrinks in value when the drone moves farther from the object; we are able to determine when  $D_b$  is equal or not equal to  $D_p$ . When  $D_b$  is not equal to  $D_p$ , the drone will continue in a forward direction for free roam and for pathfinding it will follow the current path. Once  $D_b$  is equal to  $D_p$ , the system is able to tell the drone to rotate clockwise to avoid hitting the obstacle then the drone will continue a forward path for free roam. But for pathfinding the system will calculate a new path and the drone will follow it. Along with making those movements when an object is found, the system will also store a 1 value in the front position on the grid. That will allow the system to know the obstacle's position relative to the grid (i.e.,  $x$  and  $y$  values) and allow the drone to avoid the obstacle if it ends up returning to that position.

#### **4.2.5.1 Flowcharts**

The flowchart in figure 4.2.5.1.1, shows the system when running the drone with free roam movement algorithm. The flowchart in figure 4.2.5.1.2, shows the system when running the drone with pathfinding movement algorithm.



*Figure 4.2.5.1.1 Free roam system algorithm*

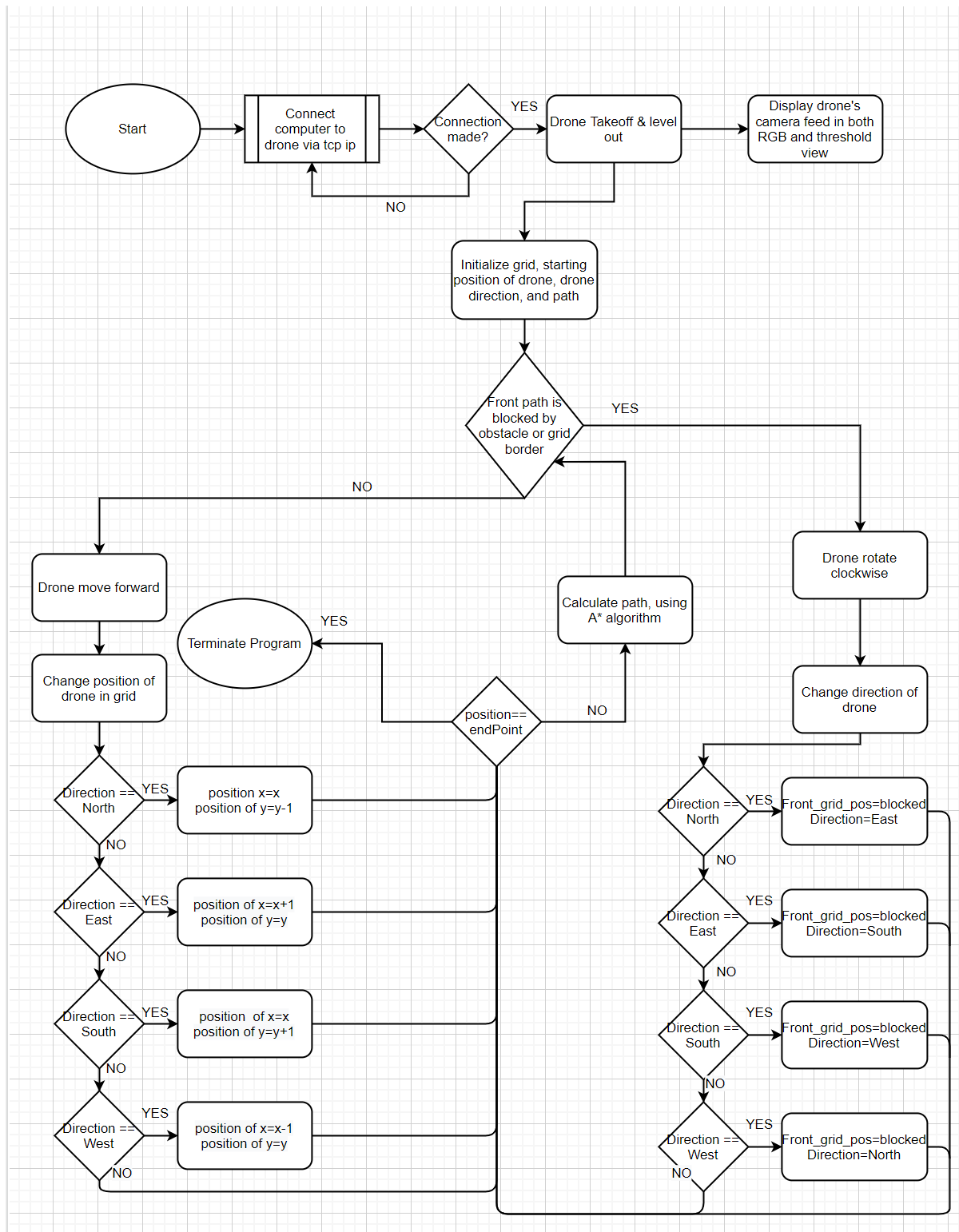


Figure 4.2.5.1.1 Pathfinding system algorithm

## 4.3. Hardware

The hardware will all be used for the purpose of getting the drone to fly. Due to time constraints there will be no further additions from this team other than flight.

### 4.3.1 Overview

With the Tello Drone, there are some limitations to it. The primary cause of this comes

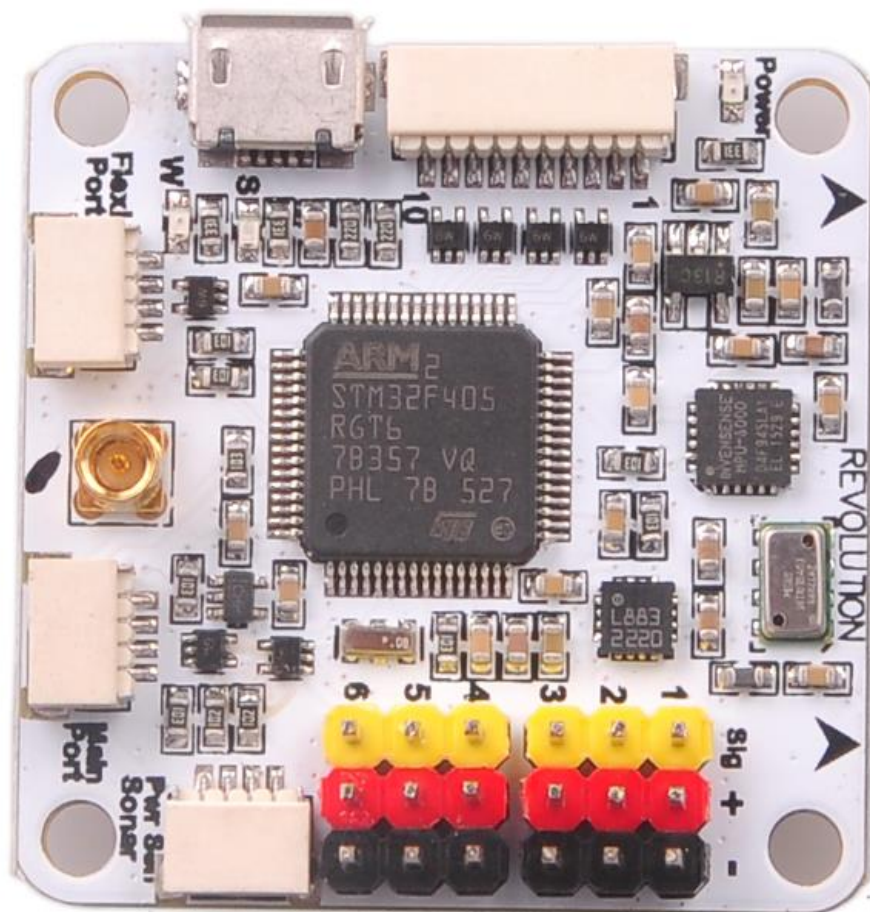


*Figure 4.3.1.1 Drone Frame and propellers*

from the fact that it only has one camera to define its environment. While the camera allows many possibilities, it also adds a large amount of difficulty and restrictions to the drone itself. So, the plan for the drone build is to give us an opportunity to work with more electronics that will extend our researching capabilities.



Before we could get started on the build, we needed to do some research on the parts necessary for the drone build. With the assistance of our advisors and another group, we were able to finalize the parts we needed. For the purpose of building a drone capable of flight using a



*Figure 4.3.1.2 CC3D Flight controller*

controller, we decided to go with frame shown in figure 4.3.1.1, CC3D Revolution flight controller shown in figure 4.3.1.2, and GOLDBAT 4S 1500mAh 100C 14.8V LiPo Battery Pack in figure 4.3.1.3. With the parts previously listed, we will be able to assembly a drone capable of flight.



*Figure 4.3.1.3 Lipo 4s battery*

## **4.3.2 Components**

This project is intended to allow our team to dive into the research surrounding drone technology. But alongside this intention, we also want this project to allow a drone enthusiast to build a drone and further our research. So, with those intentions we picked our components for this drone build.

### 4.3.3 Batteries



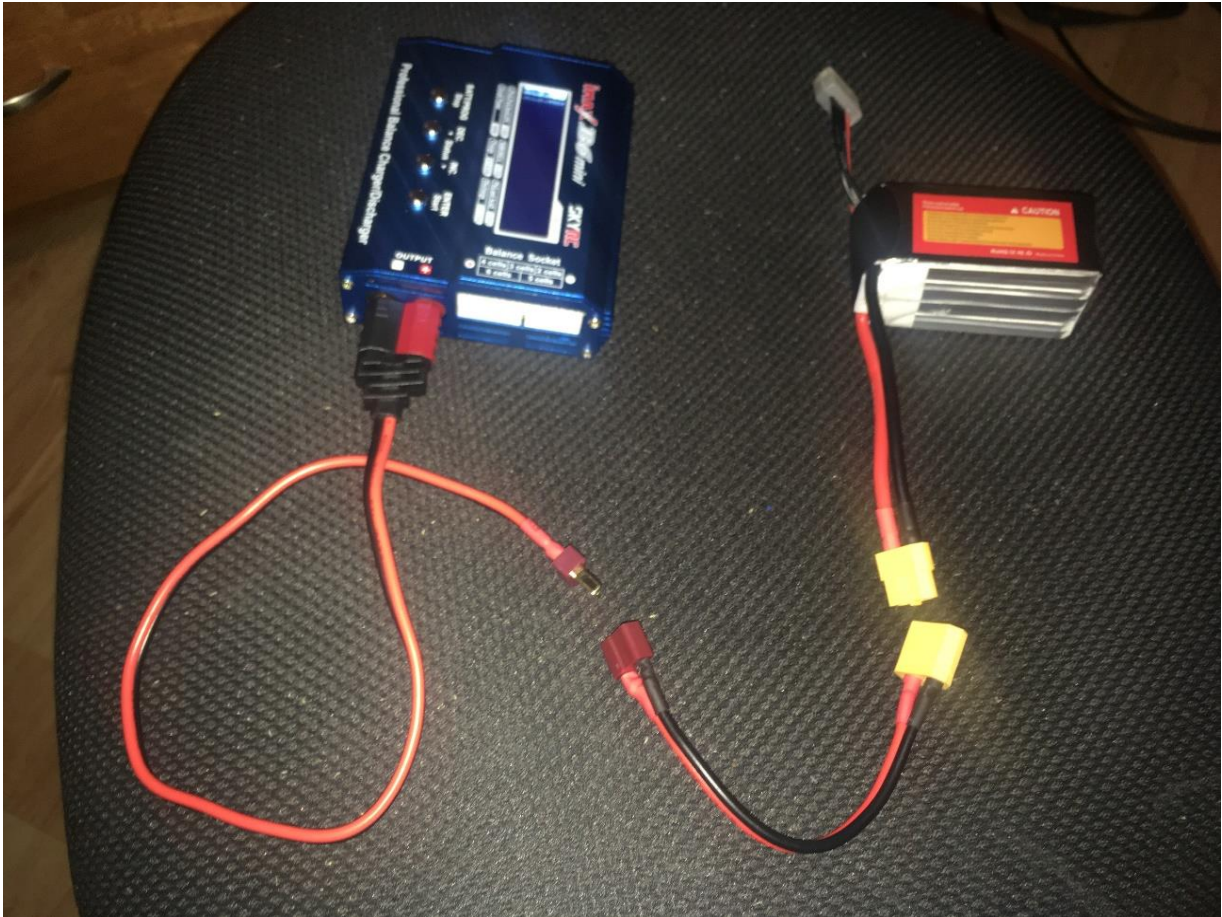
*Figure 4.3.3.1 Charger and AC power supply*

The batteries we will be using for this drone build are 4S 1500mAh 100C 183g (2pk) XT60. The key characteristic of this battery will be shown in figure 4.3.3.2. Now since we need to recharge this battery, we used SKYRC iMAX B6 charger and ac adapter. The two were used as seen in figure 4.3.3.1. It was not just a straight connection from battery cables to port in charger. There were some required connectors that allowed the battery to connect to charger and



*Figure 4.3.3.2 Battery Specifications*

refill its charge. The connection made will be shown in figure 4.3.3.3. The battery has a JST-XHR, yellow piece coming from battery cable, and the charger has a T plug, red piece coming from charger. These two cables cannot connect together so a T plug connector was required. The T plug connector is shown in figure 4.3.3.3. The battery will charge in a storage bag that will contain any explosion if it occurs and it will stay in there until it charges to a voltage value 14.8 V.



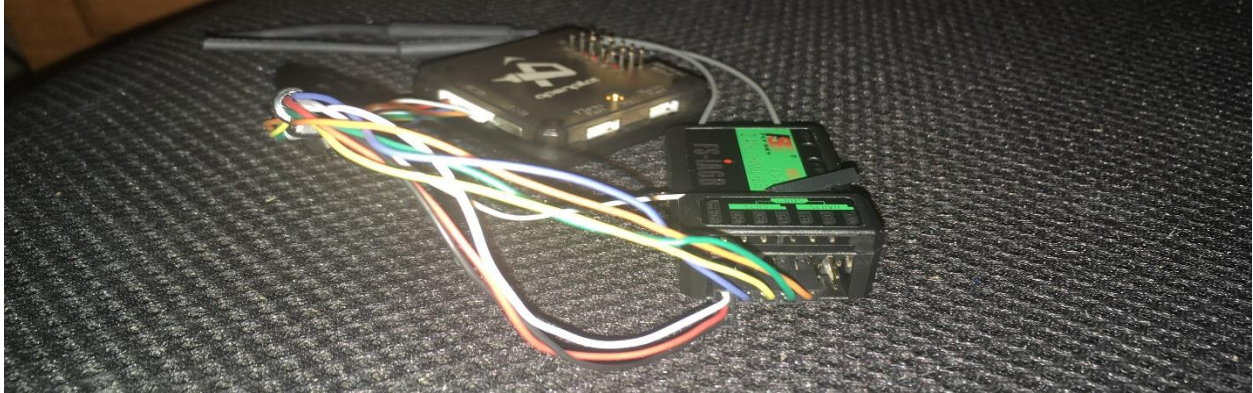
*Figure 4.3.3.3 Charger to battery connection*

#### **4.3.4 Flight Controller, Receiver, & Transmitter**

The CC3D flight controller will be the component that we use to calibrate motors, and the

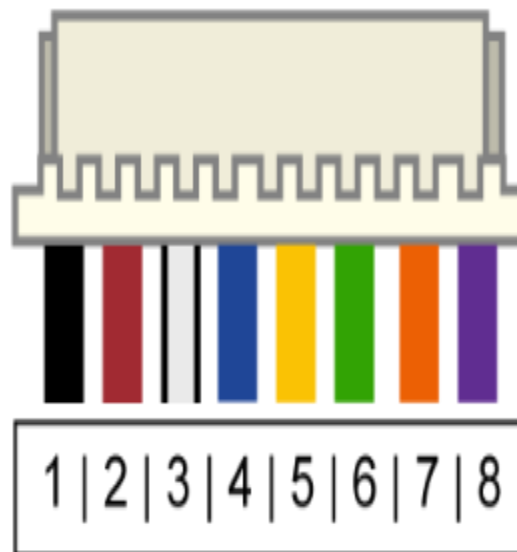


receiver to transmitter. The CC3D will be placed in the middle platform of the drone frame and receiver is placed on the top platform of the drone. The connection that the CC3D needs is shown in figure 4.3.4.1. The red, white, and black wires go into ch1 on the receiver. Red is +5 V, black



*Figure 4.3.4.1 CC3D to receiver connection*

is ground, and white is throttle signal. Blue wires go into ch2 on the receiver and is Roll signal. Yellow wires go into ch3 on the receiver and is pitch signal. Green wires go into ch4 on the receiver and is yaw signal. Orange wires go into ch5 on the receiver and is flight

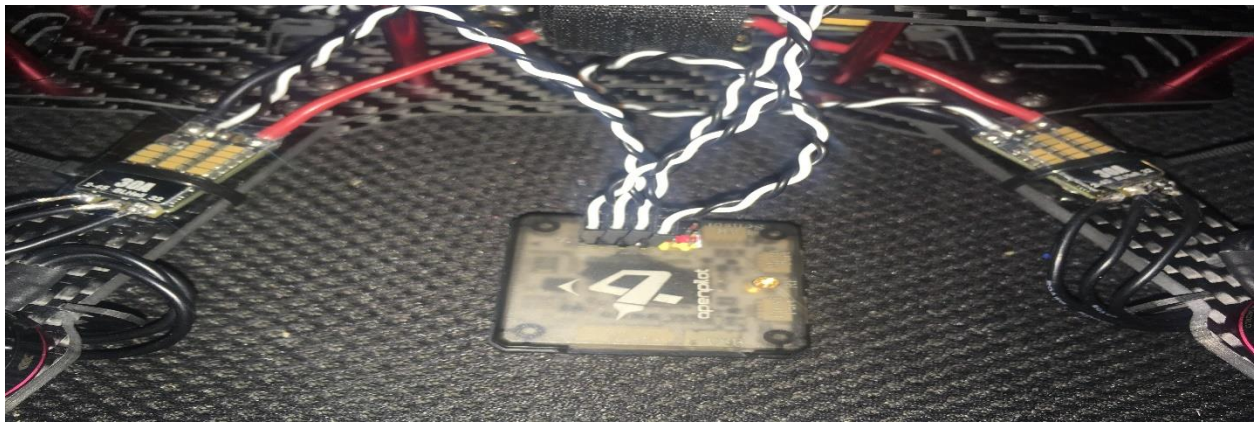


*Figure 4.3.4.2 CC3D receiver output wires*

mode signal. All signals will be put in the signal pins on the receiver. Now in other cases the color of cables may differ, so in that case just follow from left to right based on figure 4.3.4.2.

Channel 8 of the CC3D receiver will not be used for the quadcopter build.

The CC3D also makes a connection with the ESCs in order to calibrate the motors. The connection needed is on four channels of the CC3D, shown in figure 4.3.4.3. The ESC connection is a white signal cable and a black ground cable that go to a red pin and yellow pin on the CC3D, respectively. Again, there may be a case of colored wires that differ from this build. In that case, it is correct to connect the signal cable to signal pin and ground cable to ground pin.



*Figure 4.3.4.3 ESCs' connections to CC3D*

It is necessary to bind the transmitter, in the controller, to the receiver used. First putting a bind cable in the receiver and then powering the receiver. Once that is complete an LED on the receiver will start flashing and then the bind key button on the controller must be held for 3 to 5 seconds. After that turn the controller on and it will begin binding. To know it has completed the process, the screen on the controller will notify status and LED on receiver will stop flashing. The controller can be seen in figure 4.3.4.4.



*Figure 4.3.4.4 Controller with transmitter*

### **4.3.5 Frame, Motors, ESCs, PDB, & Propellers**

The components in this section were put together for the base design of racing drones. The frame needed to be light and sturdy since it is meant for racing. The frame we chose is made of carbon fiber and some aluminum rods in the main body to hold two layers for components. The motors for this setup will be connected to the electronic speed controllers (ESCs) that are then connected to the flight controller and power distribution board (PDB). The schematic is shown in figure 4.3.5.1. The components needed to be soldered together. Motor wires were soldered to the ESCs and then the ESCs were soldered to the PDB. With the combination of 5045 propellers and motors we will be able to get the required amount of thrust to lift the drone. The required amount of thrust and weight of the drone will be calculated in the section below this one. The PDB will serve the purpose of distributing the charge from the 4s battery. The PDB has an XT60 connection that we soldered to the board. The XT60 connection will allow us to power the PCB with 4s battery. Like stated before, the PDB will distribute charge. Specifically, it will distribute a continuous current of 25 Amperes to the four ESCs we are using.

With the guidance and advice of another team, we were able to avoid destroying any ESCs with improper connections and incorrect battery supplies.

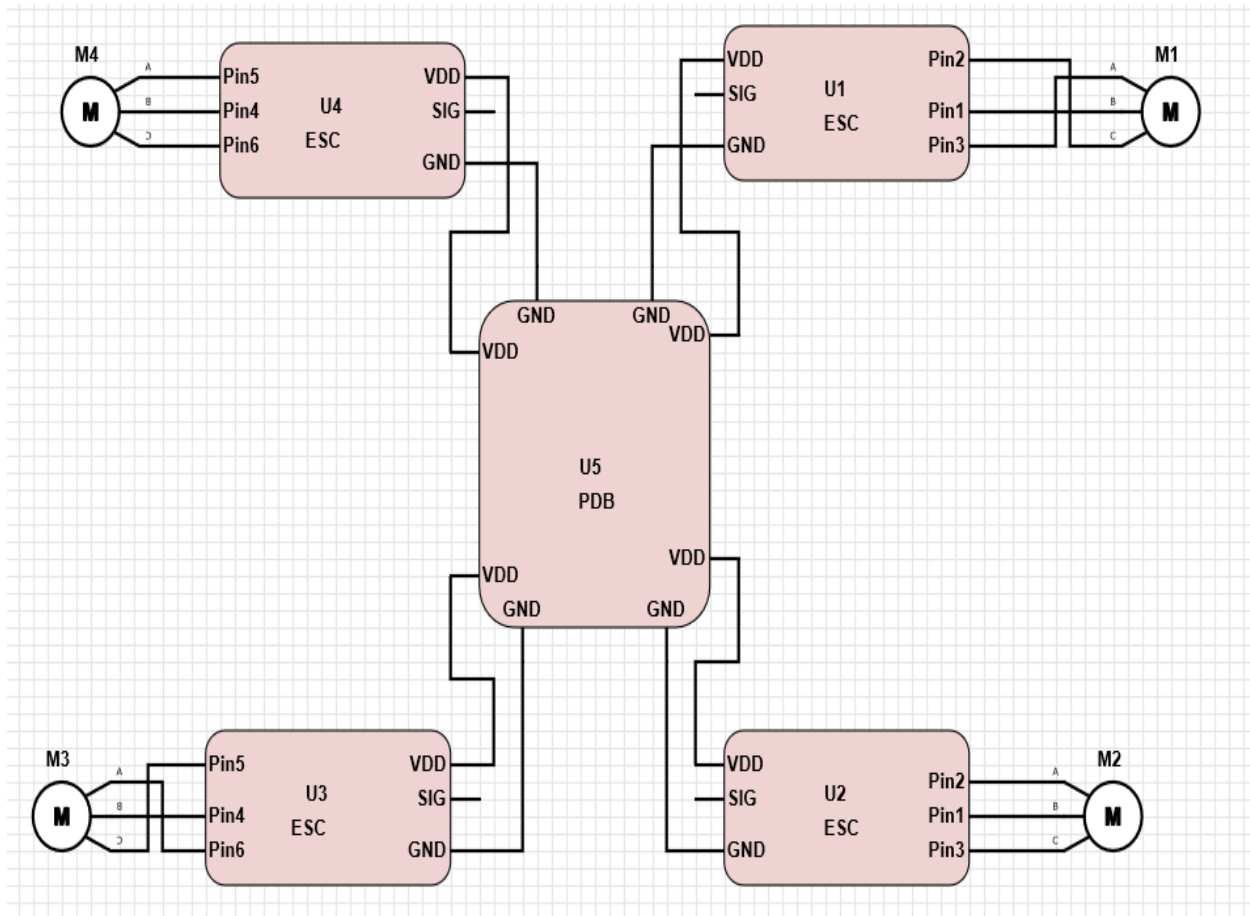


Figure 4.3.5.1 Quadcopter (PDB, ESC, Motor, excluding transmitter & receiver)

### 4.3.6 Calculations

For the drone build we needed to make a few calculations before the drone was capable of flight. That included thrust to weight ratio and current consumption of our system. The ratio we will be looking for is a 4:1 ratio, since we are building a racing drone. The ratio is large because racing drones must be both agile and acrobatic. But before 4:1 ratio is achieved, a 1:1 ratio is necessary for the drone be able to lift off a surface.



The weight we estimated can be seen in table 4.3.6.1. Since we did not have the resources to measure the weight of the drone components, we had to resort to estimations given by product websites and device manuals. Device manuals proved to be difficult since in some cases the product was Chinese, so the manual was in Mandarin. Nonetheless we were able to acquire the total weight.

<b>Name</b>	<b>Weight (g, approximation)</b>
Readytosky 250mm frame	135.1
EMAX RS2205-2300KV 3-4S	120
midici BLHeli-32 30A ESC (4pk) 3-4S	20
5045 props	6
4S 1500mAh 100C 183g (2pk) XT60	6.45
CC3D Revo stm32f4 w/ OPLink	60.5
FS-iA6B RX	14.9
Matek PDB-XT60 w/BEC (5V and 12V)	11
<b>Total Weight: 373.95g ~ 374g</b>	

*Table 4.3.6.1 Drone Weight*

Now with the total weight estimated to 374g, we needed a thrust of 1.5kg of thrust to maintain a 4:1 thrust to weight ratio. In table 4.3.6.1, that was estimated with values found in an individual's research [10]. At half functionality, a motor will draw 4.6 amperes and have 342 grams of thrust. With four motors it will total 1.4kg of thrust. That would only be a 100g less than needed and a 6.67 % difference from the weight we need. So, our thrust is not that far off from a 4:1 ratio.

Paddle Size	Percentage (%)	Thrust (g)	Current (A)	Power (W)	Efficiency (g/W)
5045	50	342	4.6	74.5	4.59
5045	100	938	21.6	324.9	2.48

*Table 4.3.6.2 Thrust Tables*

Now with the thrust to weight ratio, we needed to determine the amount of current that will be consumed. Since we are going to use the motors at 50 percent functionality, that means we will look at table 4.3.6.2. In the table it shows that a motor will consume 4.6 A so with four motors it is 18 A. Now we use the equation  $\text{amps} = \text{mAh}/(\text{hours} \times 1000)$ , to determine mAh for our battery. Now substituting with our values for five minutes we get the equation shown in figure 4.3.6.1, we estimate that the value we want is 1500 mAh. Our battery is at a value of 1500mAh, this will perfectly fit the time that we want. The reason we are going for five minutes is due to drone races lasting an average of 3 minutes.

$$X = \frac{18}{12} * 1000 = 1500$$

*Figure 4.3.6.1 mAh of our battery*

### **4.3.7 Flight Controller Software**

We decided to go with the open source software Librepilot. Librepilot comes with many different setup wizards. The two that were our target, included the ESC calibration and RC controller connectivity. The ESC wizard can be seen in figure 4.3.7.1. The ESC calibration

allowed us to test the motors. The way it did this was by giving us the ability to manually tune the motors. As we tuned the motor, we were able to determine which motors rotated and the directions they rotated. Since we needed

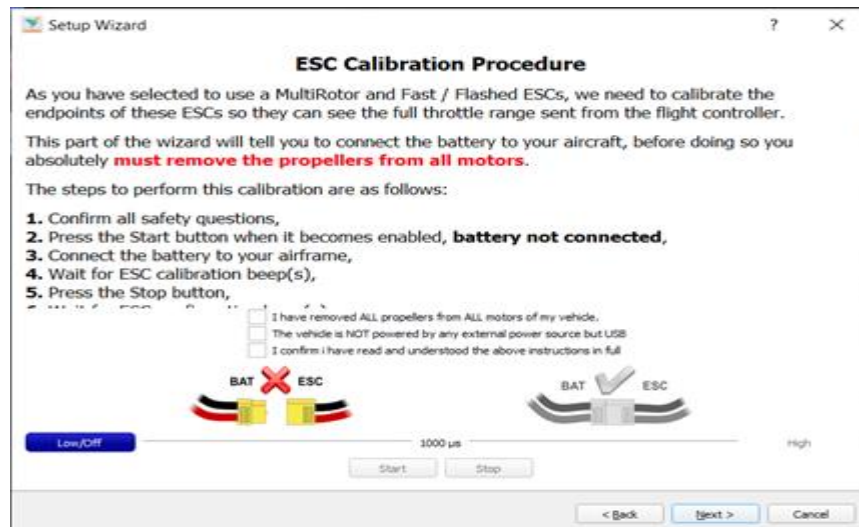


Figure 4.3.7.1 ESC wizard

to make sure that the motors were rotated in the direction shown in figure 4.3.7.2. If any of them were, then we would have to resolder the wires in the correct orientation. Once we had tuned all

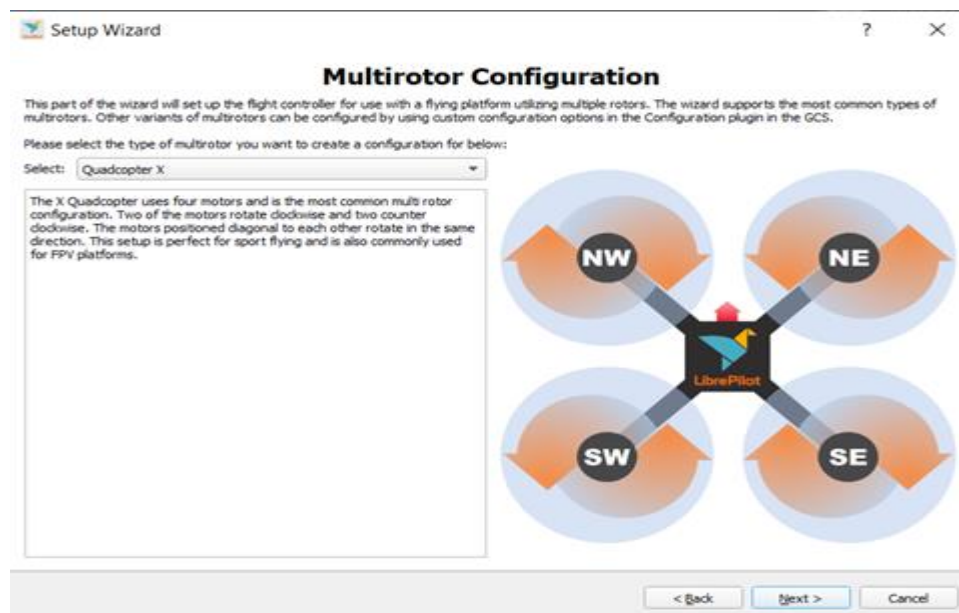
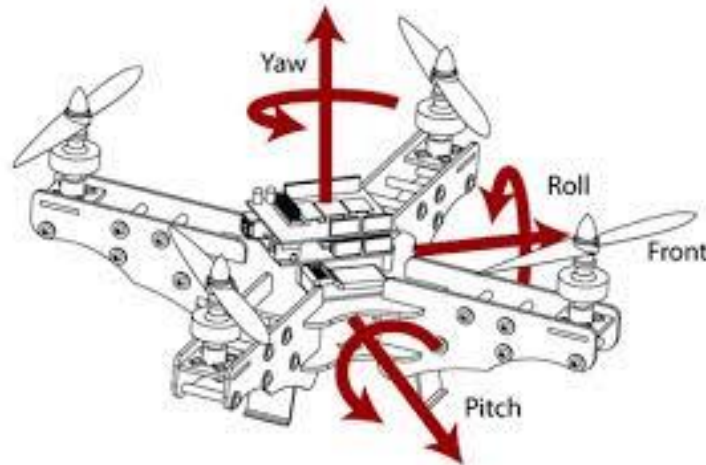


Figure 4.3.7.2 Orientation of propellers

the motors to our specifications, we were given the option of preset calibrations. Whether we picked it or not, the result would still be saving the tune to the flight controller.

Once we finished the calibration, we needed to setup the flight controller to read the inputs from the RC controller. Since the drone operates on flight movements, it benefitted us to do research on that. In our research we found figure 4.3.7.3, that shows the orientations of flight.



*Figure 4.3.7.3 Flight orientations of drone*

The wizard for setting up the RC controller started with allowing us to test the actions in real time with the GUI. That also allowed us to make certain that the controls were not inverted. Once the controls were configured, we had to setup a failsafe. This failsafe is put in place in case the RC controller loses connection with the drone. When the failsafe is initiated, the drone will shut off all power and crash land. This may cause damage to our drone build but it is beneficial for us to deal with broken components rather than completely lose the drone. To setup this

failsafe, we had to set the power off when connectivity is lost. This can be seen in figure 4.3.7.4. That would be the last step to get this drone flying.

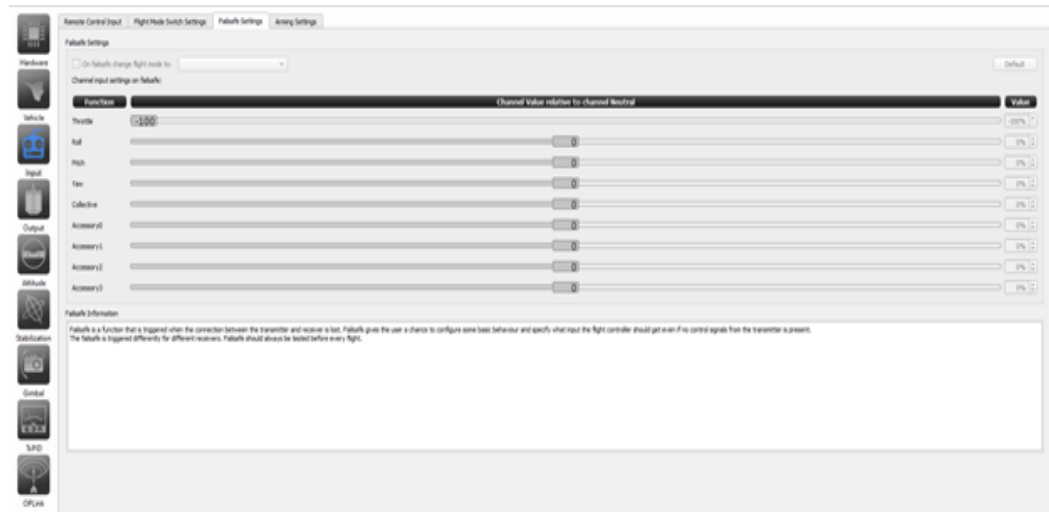


Figure 4.3.7.4 Librepilot failsafe

## 5. Testing

### 5.1 Object Avoidance System Testing

The testing for this system will consist primarily of the bug fixes that were developed for the program to run properly and the accuracy of the working product. In some instances, a huge portion had to be deleted or edited. For that case it seemed feasible to simply name it the newer version.

#### 5.1.1 Test Plan Object Avoidance System Version 1.0

The test plan can be seen in the table below:

Title	Avoidance Ver. 1.0
File method	Object Avoidance (first_OA.py)
Test Conditions	1) Dji Tello EDU Drone. 2) Any yellow object in the following range of rgb values lower [22,153,62] upper [80,255,255], of any shape and not larger than an average person's hand. 3) White background, as to avoid color noise in the video feed. 4) Single yellow object 5) Object should appear in the path of the drone at any moment during runtime, then not move until

	completely out of the drone's view. 6) 'q' key will need to be pressed to end program.
Test Cases	a) 10 different positions in the path of the drone will be tested. b) Each position should have a certain avoidance direction that will be initiated, either left or right.
Test Procedure	Once the program starts up, the tester should record the number of test and position. Then once the drone starts moving in a forward direction, the tester will then decide to move the object into the desired position in the drone's path and hold steady. Then record one of the following marks, avoided, collision, or wrong avoidance direction.
Acceptable Test Results	100% of positions will have 100% avoided. Since drone collisions will cause substantial damage and risk a bystanders' injury.

*Table 5.1.1 Avoidance 1.0 Test plan*

### 5.1.2 Test Plan Object Avoidance System Version 2.0

The test plan can be seen in the table below:

Title	Avoidance Ver. 2.0
File method	Object Avoidance (second_OA.py)
Test Conditions	1) Dji Tello EDU Drone. 2) Any yellow object in the following range of rgb values lower [22,153,62] upper [80,255,255], of any shape and not larger than an average person's hand. 3) White background, as to avoid color noise in the video feed. 4) Single yellow object 5) Object should appear in the path of the drone at any moment during runtime, then not move until completely out of the drone's view. 6) 'q' key will need to be pressed to end program.
Test Cases	a) 10 different positions in the path of the drone will be tested. b) Each position should have a certain avoidance direction that will be initiated, either left or right. Then move forward and re-center, either left or right.
Test Procedure	Once the program starts up, the tester should record the number of test and position. Then once the drone starts moving in a forward direction, the tester will then decide to move the object into the desired position in the drone's path and hold steady. Then record one of the following marks, avoided/re-center, collision, wrong avoidance/re-center.
Acceptable Test Results	100% of positions will have 100% avoided/re-center. Since drone collisions will cause substantial damage and risk a bystanders' injury.

*Table 5.1.2 Avoidance 2.0 Test plan*

### 5.1.3 Results

We were able to get the results we wanted for the OAS testing. While we initially had some problems with the program crashing when the drone lost view of the object, that came before these tests. So, we were able fix it. The test for version 1.0 went perfectly, all positions tested went perfectly. Now for one test run, our conditions for the background were not completely white so some noise did cause the program to misread directions. However, this was not a problem with coding failure but involved an incorrect controlled environment. Version 2.0 came because some limitations were discovered during version 1.0 testing. Version 1.0 would not allow a re-center method in the avoidance. So, its forward path would not be the same path it originated from. And although it passed the test to avoid collision, it failed an objective of getting to follow a path. Version 2.0 testing was successful. However, in order to get the drone to read the movement command properly, there had to be time delays add. The time delay would temporarily stop code from being executed. It allowed for a working dodge and re-center but, it caused some delay in the camera feed. So, this means an object could enter its path during this delay and cause collision. That collision could occur but, in the controlled environment we must have the object stay in the same position until it has been completely avoided. As the for the delay issue, it will more than likely be handled next semester as part of the future work. Also, we have succeeded the mission of object avoidance for this semester with the test plan for version 1 and 2.

## **5.2 Spatial Visual Awareness System Testing**

The testing for this system will consist primarily of the bug fixes that were developed for the programs to run properly and the accuracy of the working product. The two programs include the free roam grid and pathfinding grid. Both have separate test that were used in a similar manner.

### 5.2.1 Test Plan Spatial Visual Awareness System Version Free Roam

The test plan can be seen in the table below:

Title	Awareness Ver. Free Roam
File method	Free roam movement (free_roam_grid.py)
Test Conditions	1) Dji Tello EDU Drone. 2) Any yellow object in the following range of rgb values lower [22,153,62] upper [80,255,255], of cylindrical shape and width 3.5 in. 3) White background, as to avoid color noise in the video feed. 4) Single yellow object 5) Object should appear in the path of the drone at any moment during runtime, then not move until completely out of the drone's view. 6) 'q' key will need to be pressed to end program.
Test Cases	a) 10 different positions in the path of the drone will be tested. b) Each position should have different x and y coordinate values
Test Procedure	Before the program starts, the tester should place the drone in a starting position (x=5, y=5) and then place an object in any location in the grid. Once the program starts up, the tester should record the number of test and position and let the drone move. Then record one of the following marks, avoided, collision, or wrong avoidance direction.
Acceptable Test Results	90% of positions will have avoided. Since drone collisions will cause substantial damage and risk a bystanders' injury.

*Table 5.2.1 Awareness Ver. Free Roam Test plan*

### 5.2.2 Test Plan Spatial Visual Awareness System Version Pathfinding

The test plan can be seen in the table below:

Title	Awareness Ver. Pathfinding
File method	Pathfinding movement (pathfinding_grid.py)
Test Conditions	1) Dji Tello EDU Drone. 2) Any yellow object in the following range of rgb values lower [22,153,62] upper [80,255,255], of cylindrical shape and width 3.5 in. 3) White background, as to avoid color noise in the video feed. 4) Single yellow object 5) Object should appear in the path of the drone at any moment during runtime, then not move until completely out of the drone's view. 6) the program will end when the drone has reached the end point of the path, or if the 'q' key is pressed.
Test Cases	a) 10 different positions in the path of the drone will be tested. b) Each position should have different x and y coordinate values
Test Procedure	Before the program starts, the tester should place the drone in a starting position (x=5, y=5) and then place an object in any location in the grid that would block the path from the starting point to ending point. Once the program starts up, the tester should record the number of test and position



	and let the drone move. Then record one of the following marks, avoided, collision, bug, or wrong avoidance direction.
Acceptable Test Results	90% of positions will have avoided. Since drone collisions will cause substantial damage and risk a bystanders' injury.

*Table 5.2.2 Awareness Ver. PathfindingTest plan*

### **5.2.3 Results**

For the free roam system, the tests were able to produce the results that were wanted. There were some issues at the beginning of testing that stemmed from the movement commands not allowing the detection to operate, but with some quick debugging the system was all good to go. The testing environment Nathaniel worked in was well enough that the noise from the camera feed did not affect the system in a negative manner. Now for the pathfinding system, the test were able to produce the results that were wanted. However, there was a bug that halted testing for longer duration than anticipated. The bug involved the python library (opencv), that Nathaniel used to process the image data. After a debugging session, Nathaniel was able to get the system to full operation. The system worked well with handling a free roam movement system, and a pathfinding movement system. With the test being done in a controlled environment there were some variables that could be affected by a different environment. Some of those variables include, width of object, size of grid in system/real-life, and saturation of image. Most of those variables could vary well cause the systems to be dysfunctional. However as of right now through the test documented, the systems work as intended.

## **6. Setup Guides**

### **6.1 Mission 1 Guide**

Installing Mission 1 for Flight Test

## Requirements-

- 1.) Create an account at <https://github.com/>
- 2.) Install Visual Studio Code <https://code.visualstudio.com/download>
  - 2a.) Once Visual Studio Code is installed download a Python Extension from the Extensions tab
- 3.) Install Python 3.8.2 <https://www.python.org/downloads/>

## Instructions-

1. Create a folder in Visual Studio Code and name it what you like. For an example “Drone Project”.
2. While in the folder open a terminal and make sure the terminal directory matches with the folder being used
3. In the terminal type “ pip install –upgrade pip”, Hit Enter
4. Then next type “git clone <https://github.com/damiafuentes/TelloSDKPy.git>” into the terminal, Hit Enter
5. Then type “cd TelloSDKPy” into the terminal as well, hit Enter.
6. Lastly type “pip install -r requirements.txt”, hit Enter. This will load a text file showing names and versions of other required files that are also needed. These would be files “numpy==1.15.4”, “opencv-python==3.4.3.18”, and “pygame==1.9.4”.
7. Next open a sperate command prompt terminal outside of Visual Studio Code (Make sure when opening the command prompt to make sure your accessing it as an administrator).

8. Type “python -m pip install numpy” in the command prompt, then hit Enter (you should see “successfully installed” afterwards in the cmd, this will appear after installing opencv and pygame as well).
9. Then type “python -m pip install opencv-python” in the cmd, Hit Enter.
10. Lastly type “python -m pip install pygame” in cmd, Hit Enter.
11. Back in Visual Studio Code, in the .vscode folder located in the folder created in step 1, create json file called “settings.json”.
12. In this settings.json file type:

```
{  
  
    "python.pythonPath": "C:\\Users\\Owner\\AppData\\Local\\Programs\\Python\\Python  
38-32\\python.exe",  
  
    "python.linting.pylintArgs": ["--generate-members"],  
  
}
```

Note that the path for “python.pythonPath” may be different depending where you installed python.

13. Save all work in Visual Studio Code and exit the application.
14. Then reopen Visual Studio Code and open the folder where your project is located.
15. On The Tello drone itself press the power button for about 2 seconds.
16. In your computer desktop access your wifi list and select “TELLO-FDXXXX”. For an Example it could appear as “TELLO-FD9363”.
17. Open a terminal from “DJITelloPy” and type “python example.py”.

18. An window will appear labeled “Tello video stream” showing video feed from the drone’s camera.

19. The controls are:

- T: Takeoff
- L: Land
- Arrow keys: Forward, backward, left and right.
- A and D: Counter clockwise and clockwise rotations
- W and S: Up and down.

20. To exit the video stream window simply click the close button in the window.

## 6.2 Mission 2 Guide

Installing Mission 2 for Facial Recognition

Requirements- (Not Needed if Mission 1 is Completed)

- 1.) Create an account at <https://github.com/>
- 2.) Install Visual Studio Code <https://code.visualstudio.com/download>
  - 2a.) Once Visual Studio Code is installed download a Python Extension from the Extensions tab
- 3.) Install Python 3.8.2 <https://www.python.org/downloads/>

Instructions- (Assuming Mission 1 is Completed)

1. Create a folder in Visual Studio Code and name it what you like. For an example “ Facial Drone Project”.
2. While in the folder open a terminal and make sure the terminal directory matches with the folder being used
3. Then next type “git clone <https://github.com/DrexelLagare/Senior-Design-1.git>”, hit ‘Enter’.
4. Type “bash” and then “pip install -r requirements.txt”, hit Enter. (This will download required dependencies)
5. Back in Visual Studio Code, in the .vscode folder located in the folder created in step 1, create json file called “settings.json”.
6. In this settings.json file type:

```
{  
  
    "python.pythonPath": "C:\\Users\\Owner\\AppData\\Local\\Programs\\Python\\Python  
38-32\\python.exe",  
  
    "python.linting.pylintArgs": ["--generate-members"],  
  
}
```

Note that the path for “python.pythonPath” may be different depending where you installed python.

7. Save all work in Visual Studio Code and exit the application.
8. Then reopen Visual Studio Code and open the folder where your project is located.
9. Now in the images folder create a folder and name that folder the person you want the drone to recognize.

10. Inside the person's folder add images and number them 1, 2, 3, etc.
11. In order to train on the images, make sure you are in the right directory and in the terminal type “bash” followed by “python traindata.py”. Training is complete when you get a message: “Done training data...”.
12. Make sure the person to find is in the room and data was trained for that person.
13. On the Tello drone itself press the power button for about 2 seconds.
14. In your computer desktop access, your Wi-Fi list and select “TELLO-FDXXXX”. For an Example it could appear as “TELLO-FD9363”.
15. Open a terminal from “face\_recognition” and type “bash” followed by “python main.py”.
16. An window will appear labeled “Tello video stream” showing video feed from the drone’s camera, the drone will then launch and starts searching.
17. If the confidence rating is greater than or equal to 85, there will be a green rectangle on the face with the person's name on top, lock on to the person and start following. (NOTE: The person must face the drone to be followed)
18. If the confidence rating is below 85 than there will be a red rectangle with the name of "Unknown" on the face and the drowm will keep rotating to search for the person.
19. To exit the video stream window simply click the close button in the window.

## 6.3 Installing OpenPose Guide

- **Requirements** for the default configuration:
  - Download Visual Studio 2019 at “<https://visualstudio.microsoft.com/downloads/>”.
  - Download the GUI CMake “cmake-3.17.1-win64-x64.msi” from “<https://cmake.org/download/>”

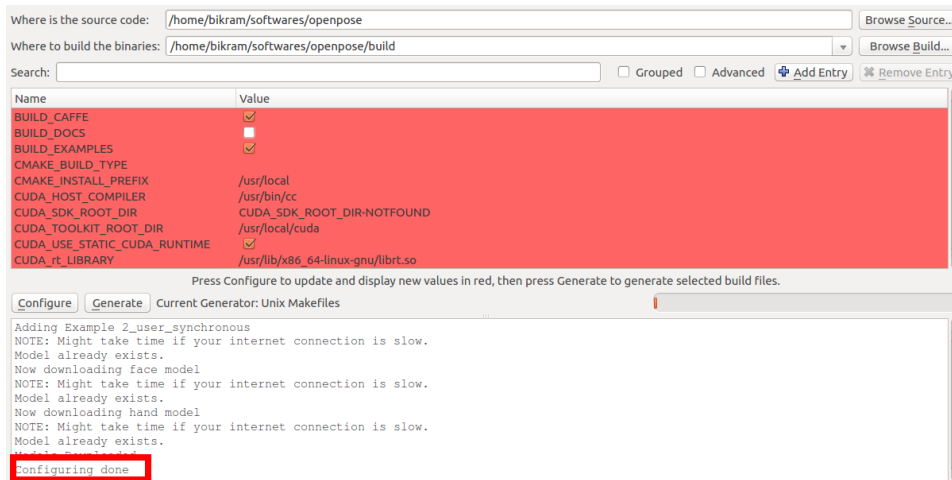
- Download CUDA (Nvidia GPU) version at “<https://developer.nvidia.com/cuda-75-downloads-archive>”:
  - NVIDIA graphics card with at least 1.6 GB available (the `nvidia-smi` command checks the available GPU memory in Ubuntu).
  - At least 2.5 GB of free RAM memory for BODY\_25 model or 2 GB for COCO model (assuming cuDNN installed).
  - Highly recommended: cuDNN.

If the Computer Does Not have a Nvidia GPU:

- OpenCL (AMD GPU) version windows x64
  - <https://developer.nvidia.com/opencl>:
  - Vega series graphics card
  - At least 2 GB of free RAM memory.
- CPU-only (no GPU) version:
  - Around 8GB of free RAM memory.
- Highly recommended: a CPU with at least 8 cores.
- **Dependencies:**
  - OpenCV (all 2.X and 3.X versions are compatible).
  - Caffe and all its dependencies.
- 1. Go to “<https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/installation.md>” and download the .zip file for OpenPose.
- 2. Open CMake GUI and select the OpenPose directory as project source directory, and a non-existing or empty sub-directory (e.g., build) where the Makefile files (Ubuntu) or

Visual Studio solution (Windows) will be generated. If build does not exist, it will ask you whether to create it. Press Yes.

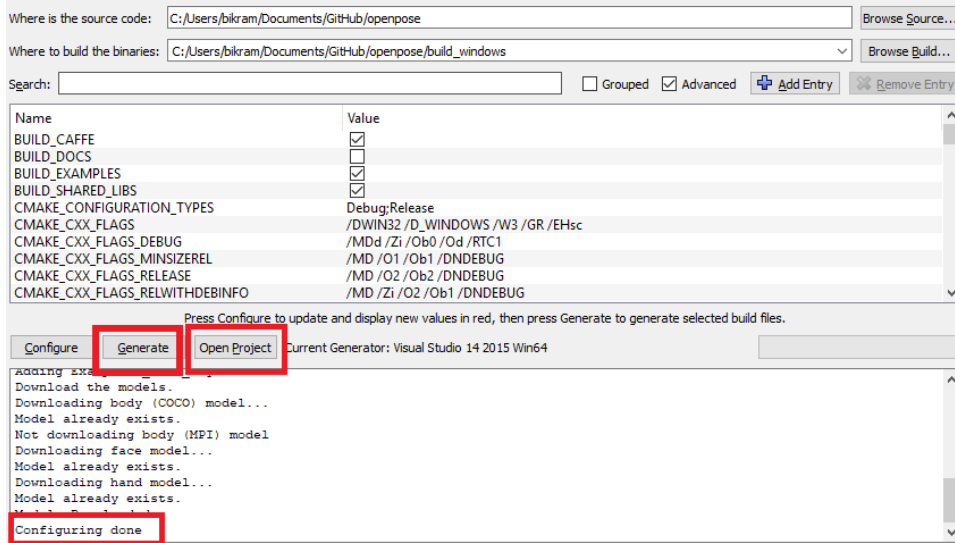
3. Press the Configure button. Once you do a new window will appear, in this window select only the *Visual Studio 2019* generator and then manually choose *x64* for the optional platform for generator.
4. If this step is successful, the Configuring done text will appear in the bottom box in the last line. Otherwise, some red text will appear in that same bottom box.



*Figure 6.3.1 When Configuring is Done*

5. In the red box of figure 6.3.1 there should be a check box called “BUILD\_PYTHON”. Select that box.





*Figure 6.3.2 Flight orientations of drone*

6. Finally hit the *Generate* button in Figure 6.3.2 and you may now proceed to OpenPose building.

## 6.4 Object Avoidance Guide

1. Code should be acquired via cloning Github repository.
  2. Open folder in Visual Studio code.
  3. Install libraries via pip.
  4. Starts with scip.spatial, numpy, and others. The others may be excluded, but I kept for possible future work.
- `python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose`

5. Next up is opencv.

- `pip install --user opencv-contrib-python`

6. If any missing import errors start popping up, check libraries of pip installs with command below.

- `pip list`

7. Now comes running programs. And that starts with main.py. Run the command below in the terminal:

`Python main.py`

8. To end main.py the q key must be pressed.

9. Other files are for main purposes or testing.

## **6.5 Body Recognition w/ MobileNet SSD Guide**

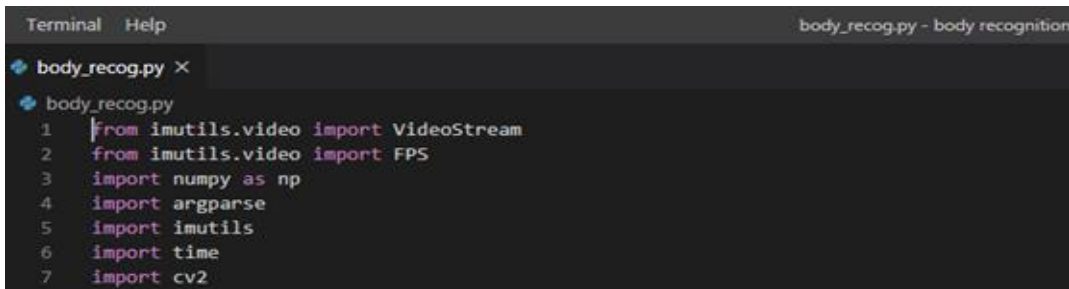
1. Create an account at <https://github.com/>

2. Install Visual Studio Code <https://code.visualstudio.com/download>

2. A) Once Visual Studio Code is installed download a Python Extension from the Extensions tab

3. Install Python's latest version <https://www.python.org/downloads/>

4. Import packages shown in figure 6.5.1

A screenshot of a terminal window with a dark background. The title bar at the top says "Terminal Help" on the left and "body\_recog.py - body recognition" on the right. Below the title bar, there's a tab labeled "body\_recog.py" with a close button. The main area shows the code for "body\_recog.py" with line numbers 1 through 7. The code imports VideoStream and FPS from imutils.video, numpy as np, argparse, imutils, time, and cv2.

```
body_recog.py
1 from imutils.video import VideoStream
2 from imutils.video import FPS
3 import numpy as np
4 import argparse
5 import imutils
6 import time
7 import cv2
```

Figure 6.5.1 Imported libraries

5. For this guide, you will need [imutils and OpenCV 3.3](#).

- Installing OpenCV using pip, Python's package manager, is *by far* the fastest and easiest method to install OpenCV on your system.

6. Go to [https://github.com/UTRGV-CS-Projects/202020-spring-2020-projects-afs-2/tree/gilberto\\_branch/body%20recognition%202](https://github.com/UTRGV-CS-Projects/202020-spring-2020-projects-afs-2/tree/gilberto_branch/body%20recognition%202) and download the zip file, you will need the two files named MobileNetSSD.

7. You can now start working on your own recognition algorithm. MobileNet SSD has been trained and has the capabilities to detect up to 20 different objects, you just need to come up with your code and the desired thing you want to detect.

## 6.6 Spatial Visual Awareness Guide

1. Code should be acquired via cloning Github repository.
2. Open folder in Visual Studio code.
3. Install libraries via pip.

4. Starts with scip.spatial, numpy, and others. The others may be excluded, but I kept for possible future work.

- `python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose`

5. Next up is opencv.

- `pip install --user opencv-contrib-python`

6. Next is pathfinding

- `pip install pathfinding`

7. If any missing import errors start popping up, check libraries of pip install with command below.

- `pip list`

8. Now comes running programs. And that starts with `distance_focalLength_calculator.py`. First change the variables that the commented sections in code say to change, based on your conditions. Run the command below in the terminal (will allow you to set parameters in next files):

- `Python distance_focalLength_calculator.py`

9. Too end `distance_focalLength_calculator.py` the q key must be pressed.

10. Now comes adding the values you got from step 8 to files `free_roam_grid.py` and `pathfinder_grid.py`. Variables to swap are called, `focal_length` and `obj_width`.

11. Then run each individually with the commands below:

- `Python free_roam_grid.py`

- Python pathfinder\_grid.py

12. Other files are for quick demos or testing.

## **7. ABET Criteria**

### **7.1 Engineering Standards**

Since this project relies on an Unmanned Aircraft System (UAS), we have to take a look at the rules set in place Federal Aviation Administration (FAA). The part that takes precedence for our project is part 107 [2]. In a summary, this part sets requirements on the drone and responsibilities of the pilot in command of the drone. The most outstanding requirements include, UAS must remain within visual line-of-sight (VLOS) of the pilot in command of the drone. In a similar fashion, the drone must remain close enough to the pilot that they can see it without the aid of any device other than corrective lenses. Another important requirement is Air Traffic Control (ATC) permission. The requirement of ATC permission relies on the class of the airspace. University of Texas Rio Grande Valley's airspace, class G, does not have any extra restrictions besides the basic restrictions. Following the drone's requirements are the responsibilities of the pilot. The pilot has the responsibility of acquiring a remote pilot airman certificate with a small UAS rating. The pilot must also supervise those pilots that are not certified. We will be trying our best to follow the laws set in place by the FAA and avoid breaking the law.

Since this is a simple senior design project and not a development project for a company, we will not be following the International Organization for standards (ISO) 29119 standards. Nonetheless most companies follow ISO 29119 standards, so we did some research. The

standards are set up in published parts of 1 through 5 [11], that set software testing standards. We will be discussing those standards in order from first released to last.

ISO 29119 standards starts with part 1, part 1 was released on September 2013. Part 1 is the base of the standards. By the way it explains the vocabulary being used, gives examples of the standards in practice, and how to apply both the vocabulary and concepts to other standards.

ISO 29119 standards follows with part 2, was released on September 2013. Part 2 sets a generic test model for companies to follow in their software testing. The model gives descriptions of software testing at the project management level, dynamic test level, and organizational level. This part could still be used in conjunction with other development models.

ISO 29119 standards continues with part 3, was released on September 2013. Part 3 concerns the documentation for software testing. Similar to part 2, it gives examples and covers project management level, dynamic test level, and organizational level. Some documents include, test policy, test plan, test design specification, test results, etc.

ISO 29119 standards then goes with part 4, was released on December 2015. Part 4 is meant to add techniques to be used with part 2. The techniques split into three main categories, structure-, specification-, and experience-based test design.

ISO 29119 standards ends with part 5, was released on November 2016. Part 5 is meant to cover keyword-driven testing. This is used for companies that are going to use test that are automated for testing. Meaning a software will test the software, rather than a tester do the testing.

## **7.2 Environment**

At a quick glance it may seem like our project may have no environmental impacts, but that would be wrong. Both the commercial drone we are using to test software and the drone we are building, use Lithium batteries.

Lithium batteries have been growing in popularity and will continue to grow in popularity. While this will help decrease the use of fossil fuels as a source energy, there is the underlying problem of lithium being a natural resource. Lithium is mined primarily from caves located Bolivia, Chile, Argentina, and Australia [12]. The lithium mining process takes quite a toll on the carbon footprint. Although lithium batteries are making a negative impact on the environment, it is not noticed because lithium batteries are promoted as a rechargeable battery to reduce carbon emissions. However, this may turn out to be a good thing since it will promote more improvements to battery technology [13]. With improvements to battery technology, there may be a possibility that lithium batteries are replaced with much more efficient batteries.

### **7.3 Political, Ethical, & Social issues**

It is a popular trend to put a camera on drones. Whether it be a drone hobbyist buying a commercial drone with a camera or police department using a drone on a chase rather than a helicopter. With this comes societies concern for their privacy [15]. The concern arises from the question, who is controlling the drone? Is it simply someone doing their job or a criminal trying to spy on people? During Obama's presidency, he pushed for privacy policies toward drones [14]. The Departments of Defense, the Interior, Justice, and Transportation, and the National Aeronautics and Space Administration (NASA), listened to Obama and made privacy policies to be in place for agencies to maintain standards regarding citizens privacy.

The FAA tends to handle most of the political side of drones [2]. It does this by setting rules and regulations for all types of drone users. In our case we needed to worry about the educational and hobbyist drone users. Both are similar in requirements. Drone pilot certificates are required to operate the drone. Depending on class of airspace, the pilot may need to request permission from traffic control. Also, the drone being used must be registered, in order to be accounted for in case of accidents.

Since we are using multiple open-source libraries we must discuss license agreements. A library that is consistent with most of the software in this project is OPENCV library. Now from a developer's point of view, it would be unethical to simply use the functions from the library and not acknowledge the creators of OPENCV. Nor would it be ethical to acknowledge the copyrights that apply to the library [7]. Now from the point of view of the team that created OPENCV, it would be unethical to claim copyright when no license agreement was never agreed upon. We will be avoiding both these hypotheticals, by referencing and acknowledging OPENCV. We have also made sure that OPENCV creators have made license agreements public knowledge [7]. Another library being used is SCIPY, they also have a license agreement as public knowledge [8].

## **7.4 Health & Safety**

Drones have exposed propellers that could reach speeds of 10000 rpm. With speeds like that, people are more likely to injure themselves. In one case, the drone landed on a triathlete's head [16]. The report says that during a triathlon, a videographer lost control of a drone and the drone nosedived straight onto the top of a triathlete's head. The triathlete was treated on scene and later taken to the hospital. In another case, there was an unmanned drone, found near the US-



Mexico border, that was carrying pounds of drugs [17]. The drone was found by a citizen and reported to the authorities. The drone was described to have bricks of drugs strapped to it with plastic webbing and black tape. This next case concerned the safety of the residents in the white house [18]. It involved a drone landing in the front lawn of the United States white house. This raised many questions of safety and security at the white house. In the report, the drone pilot said connection with the drone was lost and that resulted in the drone pilot not being charged.

## **7.5 Sustainability**

As stated in previous sections, drones are not the safest RC vehicles made. With the amount of cases that keep occurring, there is bound to be a case in which the injury is due to a design flaw made by the company. That would then make the company liable for the injuries, so it is not very sustainable to keep injuring. With the drones being autonomous this may lead to a decrease in injury.

The images the program needs for training the facial recognition is substantial. This is because it is best to give the algorithm a plethora of images that contain different angles of the face. That will allow the algorithm to have a better assumption of the face it is requested to recognize.

## **8. Conclusion & Future Work**

### **8.1 Conclusion**

We all started this project with close to no experience with the topics this project covers. So, as we continued the development for this project, we were able to get a firm grasp on most of the topics this project goes over. Due to the constraints put on us by COVID-19, we ended this

project with most of our development involving the Dji Tello drone. Rather than have both the drone build and tello drone. However, this allowed us to truly understand the positive societal impact a well-developed drone could have. While the systems we developed are far from making a positive impact, they showed hope for any further development.

As mentioned before, there were constraints that did not allow us to further development with our drone build. Nonetheless, we were able to get the drone to a state that allowed for calibrating the motors and that allowed for the drone to fly off the ground.

One thing that was noticed was the fact that we were still able to get functioning algorithms that we had previously thought would only be possible with the drone build. That showed us that we could go even further with the Tello, going into the future. Now that does not take away from the fact that the drone build would allow much further development of algorithms, it simply means we can still make many improvements to current algorithms applied to the Tello and even make new algorithms. But one thing to note is that we did not dive too deep into the possibilities of an Arduino and its sensors. So, we cannot say for certain that the Tello drone is better than our drone build.

## **8.2 Future Work**

The plan for the project going forward is to continue development on the Tello drone. With the current systems set in place, we have plenty of additions that could be made. As it stands now, we have not hit many walls with the Tello capabilities. For the time being, no entirely new systems have not come to mind. But we have left this project to the point that another group could build off our findings and possibly develop newer systems.

### **8.2.1 Object Avoidance System Additions**

The current version of our OAS starts moving forward when program begins, then continues moving forward until an object enters its path then makes an attempt to dodge and re-center. Once a successful avoidance is complete, the system will continue moving forward until an object enters again or 'q' key is pressed to end the program. The main addition moving forward is getting the system to start moving forward based on a green object entering the drone's view. Then to stop the program it will be the same method but with a red object. This could be a simple addition, but since we have not started development, we are not certain.

A list of the objectives for this addition will be listed below:

- Get the algorithm to start running when the drone views green.
- Get the algorithm to stop running when the drone views red.
- Use the same HSV edging used already developed.

## **8.2.2 Spatial Visual Awareness Additions**

The current SVAS runs the type of movements inside a grid, that include free roam and pathfinding. They both detect objects and keep track of them but the free roam, allows the drone to fly in freely through a grid and avoid/detect/track objects. While the pathfinding, allows the drone to fly from point a to point b and avoid/detect/track objects. Both movements allow to be stopped by the 'q' key, but pathfinding is meant to stop when it reached point b (end point). So, the additions for the free roam is to allow the drone to randomly pick a direction to move in (i.e., North, East, South, West), rather than just move forward until path blocked then turn right. For the pathfinding and addition would be to allow the drone to turn in the direction of the path, rather than rotate until path is not blocked. For both an addition would be to avoid objects before

they are directly in front of the drone. Due to the movements being inaccurate, we could setup the reference point system to increase accuracy.

A list of the objectives for this addition will be listed below:

- Get free roam to generate random direction.
- Get pathfinding to truly understand its path.
- Allow SVAS to avoid objects not directly in front of drone.
- Use reference points to increase accuracy.

## **9. References**

- [1] “Tello - Apps on Google Play,” Google. [Online]. Available: [https://play.google.com/store/apps/details?id=com.ryzerobotics.tello&hl=en\\_US](https://play.google.com/store/apps/details?id=com.ryzerobotics.tello&hl=en_US). [Accessed: 05-Apr-2020].
- [2] “Educational Users,” FAA seal, 06-Jan-2020. [Online]. Available: [http://www.faa.gov/uas/educational\\_users/](http://www.faa.gov/uas/educational_users/). [Accessed: 05-Apr-2020].
- [3] Damiafuentes, “damiafuentes/DJITelloPy,” GitHub. [Online]. Available: <https://github.com/damiafuentes/DJITelloPy>. [Accessed: 05-Apr-2020].
- [4] DrexelLagare, “DrexelLagare/Senior-Design-1,” GitHub. [Online]. Available: <https://github.com/DrexelLagare/Senior-Design-1>. [Accessed: 05-Apr-2020].
- [5] “Spatial algorithms and data structures (scipy.spatial)¶,” Spatial algorithms and data structures (scipy.spatial) - SciPy v1.4.1 Reference Guide. [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/spatial.html>. [Accessed: 05-Apr-2020].
- [6] “Changing Colorspaces¶,” OpenCV. [Online]. Available: [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_imgproc/py\\_colorspaces/py\\_colorspaces.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html). [Accessed: 05-Apr-2020].
- [7] “About,” OpenCV. [Online]. Available: <https://opencv.org/license/>. [Accessed: 05-Apr-2020].
- [8] “SciPy license¶,” SciPy license - SciPy.org. [Online]. Available: <http://www.scipy.org/scipylib/license.html>. [Accessed: 05-Apr-2020].
- [9] D. M. West and J. R. Allen, “How artificial intelligence is transforming the world,” Brookings, 25-Oct-2019. [Online]. Available: <https://www.brookings.edu/research/how-artificial-intelligence-is-transforming-the-world/>. [Accessed: 05-Apr-2020].

- [10] emax-rs2205-2300kv-2600kv-motors. [Online]. Available: <https://oscarliang.com/emax-rs2205-2300kv-2600kv-motors/>. [Accessed: 05-Apr-2020].
- [11] “29119-5-2016 - ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing,” IEEE. [Online]. Available: <https://standards.ieee.org/standard/29119-5-2016.html>. [Accessed: 06-Apr-2020].
- [12] “How Green are Home Batteries? The Environmental Impact of Lithium-Ion,” Solar.com, 01-Jul-2019. [Online]. Available: <https://www.solar.com/learn/how-green-are-home-batteries-the-environmental-impact-of-lithium-ion/>. [Accessed: 06-Apr-2020].
- [13] M. Eberhard, “A Bit About Batteries,” Tesla, Inc, 14-Sep-2010. [Online]. Available: <https://www.tesla.com/blog/bit-about-batteries>. [Accessed: 06-Apr-2020].
- [14] “FACT SHEET: Enabling a New Generation of Aviation Technology,” National Archives and Records Administration. [Online]. Available: <https://obamawhitehouse.archives.gov/the-press-office/2016/06/21/fact-sheet-enabling-new-generation-aviation-technology>. [Accessed: 06-Apr-2020].
- [15] S. Rice, “Eyes In The Sky: The Public Has Privacy Concerns About Drones,” Forbes, 04-Feb-2019. [Online]. Available: <https://www.forbes.com/sites/stephenrice1/2019/02/04/eyes-in-the-sky-the-public-has-privacy-concerns-about-drones/>. [Accessed: 06-Apr-2020].
- [16] “Australian triathlete injured after drone crash,” BBC News, 07-Apr-2014. [Online]. Available: <https://www.bbc.com/news/technology-26921504>. [Accessed: 06-Apr-2020].
- [17] Nbc, “Drone Carrying Meth Crashes Near San Ysidro Port of Entry,” NBC 7 San Diego, 22-Jan-2015. [Online]. Available: <https://www.nbcsandiego.com/news/local/drone-carrying-meth-crashes-near-san-ysidro-port-of-entry/57864/>. [Accessed: 06-Apr-2020].
- [18] J. Pegues, “Drone over White House highlights security concerns,” CBS News, 26-Jan-2015. [Online]. Available: <https://www.cbsnews.com/news/drone-over-white-house-sparks-new-security-concerns/>. [Accessed: 06-Apr-2020].
- [19] S. Zacharek, “Drones Are Revolutionizing the Way Film and TV Is Made,” Time, 31-May-2018. [Online]. Available: <https://time.com/5295594/drones-hollywood-artists/>. [Accessed: 08-Apr-2020].

## **10. Appendix**

### **A.1 Object Avoidance readme document**

# **2020-spring-2020-projects-afs-2**

---

- Object Avoidance System
  - i. Object detection/tracking using HSV.
  - ii. Avoidance Algorithm set it place.

## **Dodging a yellow object while moving forward**

---

1. The drone will move forward in a controlled environment in which a yellow object will move into its path
2. Once a yellow object moves into the drone's path, the hsv\_edge function in main.py will confirm that the object is yellow.
3. Once a yellow object is found, the aviodance algorithm will draw a box around the object.
4. The box will allow the dodge to be performed by calculating the area and position then attempt to dodge but only if the object is in the path of the drone.
5. The objects position in relation to the drone will be based on x and y cordinates on the camera view and area of drawnn box around object.
6. Once all that is confirmed, the drone will commence dodging. Dodging entails moving away from object then continue forward.

## **Current Implementation**

---

This is an implementation of object tracking on the dji Tello drone based on HSV Edging using OpenCV and Python 3.6 on visual studio code.

The current implementation allows a user to:

- Test other dodging methods using 'testing\_OA.py'
- Launch the drone using command in terminal 'python main.py'

- View both the video feed from the drone to the computer and the HSV edging used by the drone.

It allows the drone to:

- Move in a forward path, while detecting objects.
- Detect object of the color yellow.
- If hsv edging detects a yellow object, drone will use avoidance algorithm to dodge objects in its path.

**Note:** Current implementation allows only one yellow object dodge at a time. Also a dodge and recenter movement.

**Warning!!** We are aware of an issue when initially installing imports. The issue revolves around opencv and visual studio code.

1. Open up the 'settings.json' file in visual studio code.
2. The code below should be put in the json file (autosave is just a personal preference).

```
{
  "files.autoSave": "off",
  "python.linting.pylintArgs": ["--generate-members"],
}
```

3. If that fix does not work then there might be an issue with your path, as discovered by a team member.

## Installs via pip

---

1. Starts with scip.spatial, numpy, and others. The others may be excluded, but I kept for possible future work.

```
python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose
```

2. Next up is opencv.

```
pip install --user opencv-contrib-python
```

3 If any missing import errors start popping up check libraries of pip installs with command below.

```
pip list
```

# Avoiding a yellow in direct path of drone

---

1. Requires a solid yellow object.
2. Drone must connect with the computer via wifi.
3. Open terminal and enter command below.

```
python main.py
```

4. Drone will launch and start moving forward detecting objects.
5. If a yellow object is determined to be in the drone's path it will perform a dodge/recenter and on the camera view, red text 'dodge!!!' will display.
6. To stop the drone, the 'q' key must be pressed.

## Have fun testing your own methods

---

1. Same conditions necessary for 'main.py' must be met by 'testing\_OA.py'
2. There will be a 'testing\_OA.py' file for testing any new methods you might have.
3. This will allow you or anyone to build a unique Avoidance Algorithm.
4. Running, will require the command below.

```
python testing_OA.py
```

5. If you manage to getting a better working algorithm add to 'main.py'

## Handling both Avodiance systems

---

1. The file 'first\_OA.py' has no known issues. However, it does only performs a single movement dodge.
2. The file 'second\_OA.py' is the current setup for 'main.py' . The problem with this method is they delay factor that was added to sending the drone commands in a timely manner. The problem with this is that an object could enter frame during this delay an cause collision. However this is a controlled environment so it works accordingly.

### A.2 Object Avoidance main.py

```
from scipy.spatial import distance as dist
import numpy as np
import cv2
```



```

import imutils
import tello_drone as tello
import time

host = ''
port = 9000
local_address = (host, port)

# Pass the is_dummy flag to run on a local camera
drone = tello.Tello(host, port, is_dummy=False)
#drone = tello.Tello(host, port, is_dummy=True) #local camera

frame_read = drone.get_frame_read()

frame1 = frame_read.frame
frame2 = frame_read.frame

cap = drone.get_video_capture()
#functions for dodging
def dodge_right_center(x):
    drone.move_right(x)

def dodge_left_center(x):
    drone.move_left(x)

#function for reaching goal
def goal(x):
    drone.move_forward(x)

#works great with finding an object based on color
def hsv_Edge(frame):
    #yellow color detection range
    l_b = np.array([22,153,62])
    u_b = np.array([80,255,255])

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    mask =cv2.inRange(hsv,l_b,u_b)

    res = cv2.bitwise_and(frame, frame, mask=mask)

    gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
    edge = cv2.GaussianBlur(gray, (5,5), 0)
    _, edge = cv2.threshold(edge, 20, 255, cv2.THRESH_BINARY)

    return edge

```

```

#will keep since it does work for focusing on the object I want
def f_Marker( mark ):

    """
    finds largest contour/object among contours in frame
    ln#74-75
    returns minAreaRect and contourArea of largest object
    """

    contours, hierarchy = cv2.findContours(mark, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    cnts = [(cv2.contourArea(contour), contour) for contour in contours]

    if len(contours)==0:
        return 0, 0, None
    else:
        c = max(cnts, key=lambda x: x[0])[1]
        return cv2.minAreaRect( c ), cv2.contourArea(c), c


moved0=True
moved1=False
moved2=False
left= False
right =False
c=0


while True:
    edged = hsv_Edge(frame1)
    marker, area, cnt = f_Marker( edged )

    height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
    width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    # Calculate frame center
    center_x = int(width/2)
    center_y = int(height/2)

    obj_center_x = center_x
    obj_center_y = center_y
    z_area = 0

    """
    if area from f_Marker is below threshold this case 1000 and found
    continues to be false rotate drone which acts as a search for object.
    """

    if area >= 5000 or moved1==True or moved2==True:

```

```

# draw a bounding box around the image and display it
if moved0 == True or area >= 5000:

    box = cv2.cv.BoxPoints(marker) if imutils.is_cv2() else
cv2.boxPoints(marker)
    box = np.int0(box)
    cv2.drawContours(frame1, [box], -1, (0, 255, 0), 2)

    M = cv2.moments(cnt)

    (startX, startY, endX, endY) = box

    obj_center_x = int((startX[0] + endX[0]) / 2.0)
    obj_center_y = int((startY[1] + endY[1]) / 2.0)
    z_area = (endX[1] + endY[1]) * (endX[0] + endY[0])

# using moments to calculate center ln. 255
o_x = int(M['m10']/M['m00'])
o_y = int(M['m01']/M['m00'])
mz_area = M['m00']

# Calculate recognized face offset from center
# offset_x = obj_center_x - center_x
offset_x = o_x - center_x
# Add 30 so that the drone covers as much of the subject as possible
# offset_y = obj_center_y - center_y - 30
offset_y = o_y - center_y - 30

cv2.putText(frame1, "Dodge!!!", (10, 20),
cv2.FONT_HERSHEY_SIMPLEX,
            1, (0, 0, 255), 3)
cv2.imshow("feed", frame1)
if moved1 == True and area < 5000:
    time.sleep(3)
    moved1 = False
    moved2 = True
    drone.move_forward(120)
    time.sleep(3)
elif moved2 == True and area < 5000:
    if left == True:
        time.sleep(1)
        moved2 = False
        drone.move_left(90)
        time.sleep(1)
    elif right == True:
        time.sleep(1)
        moved2 = False

```

```

        drone.move_right(90)
        time.sleep(1)
    elif moved0==True or area >= 5000 and c<2:
        if not -90 <= offset_x <= 90 and offset_x is not 0:
            if offset_x < 0:
                left=True
                c+=1
                moved0=False
                moved1=True
                dodge_right_center(90)
                time.sleep(2)
            elif offset_x > 0:
                right=True
                c+=1
                moved0=False
                moved1=True
                dodge_left_center(90)
                time.sleep(2)
            else:
                right=True
                c+=1
                moved0=False
                moved1=True
                dodge_left_center(90)
                time.sleep(2)
        else:
            right=True
            c+=1
            moved0=False
            moved1=True
            dodge_left_center(90)
            time.sleep(2)
    else:
        goal(70)
    cv2.imshow("feed", frame1)
    cv2.imshow("edged", edged )

    frame1 = frame2
    frame2 = frame_read.frame

    if cv2.waitKey(1) == ord('q'): # Land the drone once q key is hit
        break

```

```

drone.end()
cv2.destroyAllWindows()

```

### **A.3 Object Avoidance first\_OA.py**

```

from scipy.spatial import distance as dist
import numpy as np
import cv2
import imutils
import tello_drone as tello

```

```

import time

host = ''
port = 9000
local_address = (host, port)

# Pass the is_dummy flag to run on a local camera
drone = tello.Tello(host, port, is_dummy=False)
#drone = tello.Tello(host, port, is_dummy=True) #local camera

frame_read = drone.get_frame_read()

frame1 = frame_read.frame
frame2 = frame_read.frame

cap = drone.get_video_capture()
#functions for dodging
#still need to test but for right now
#only move to right works
def dodge_right_center(x):
    drone.move_right(x)
    #drone.move_forward(x)
    #drone.move_left(x)
#still need to test but for right now
#only move to left works
def dodge_left_center(x):
    drone.move_left(x)
    #drone.move_forward(x)
    #drone.move_right(x)

#function for reaching goal
#still need to test but for right now
#only move forward works
def goal(x):
    drone.move_forward(x)

def adjust_tello_position(offset_x, offset_y, offset_z):
    """
    Adjusts the position of the tello drone based on the offset values given
    from the frame
    :param offset_x: Offset between center and object x coordinates
    :param offset_y: Offset between center and object y coordinates
    will try to get it to center but for now it dodges
    """

    if not -90 <= offset_x <= 90 and offset_x is not 0:

```

```

        if offset_x < 0:
            dodge_right_center(60)
        elif offset_x > 0:
            dodge_left_center(60)

#works great with finding an object based on color
def hsv_Edge(frame):
    #yellow color detection range
    l_b = np.array([22,153,62])
    u_b = np.array([80,255,255])
    #testing other colors
    #when works main works properly
    #may use this for additions to program
    #l_b = np.array([51,51,00])
    #u_b = np.array([255,255,204])

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    mask =cv2.inRange(hsv,l_b,u_b)

    res = cv2.bitwise_and(frame, frame, mask=mask)

    gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
    edge = cv2.GaussianBlur(gray, (5,5), 0)
    _, edge = cv2.threshold(edge, 20, 255, cv2.THRESH_BINARY)

    return edge
#will keep since it does work for focusing on the object I want
def f_Marker( mark ):

    """
    finds largest contour/object among contours in frame
    returns minAreaRect and contourArea of largest object
    """

    contours, hierarchy = cv2.findContours(mark, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    cnts = [(cv2.contourArea(contour), contour) for contour in contours]

    if len(contours)==0:
        return 0, 0, None
    else:
        c = max(cnts, key=lambda x: x[0])[1]
        return cv2.minAreaRect( c ), cv2.contourArea(c), c

```

```

while True:
    edged = hsv_Edge(frame1)
    marker, area, cnt = f_Marker( edged )

    height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
    width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    # Calculate frame center
    center_x = int(width/2)
    center_y = int(height/2)

    obj_center_x = center_x
    obj_center_y = center_y
    z_area = 0

    """
    This if statement will determine whether to keep moving toward a goal
    or move away from the object in front of it
    Right now it will move out of the path to avoid object
    but will not move back into the path
    Will be attempting an update to allow this
    other than that it does reach the goal
    """

    if area >=5000:

        # draw box around object then it will be displayed
        box = cv2.cv.BoxPoints(marker) if imutils.is_cv2() else
cv2.boxPoints(marker)
        box = np.int0(box)
        cv2.drawContours(frame1, [box], -1, (0, 255, 0), 2)

        M = cv2.moments(cnt)

        (startX, startY, endX, endY) = box

        obj_center_x = int((startX[0] + endX[0]) / 2.0)
        obj_center_y = int((startY[1] + endY[1]) / 2.0)
        z_area = (endX[1]+endY[1]) * (endX[0] + endY[0])

        # using moments to calculate center ln. 255
        o_x = int(M['m10']/M['m00'])
        o_y = int(M['m01']/M['m00'])
        mz_area = M['m00']

        # Calculate recognized face offset from center

```

```

# offset_x = obj_center_x - center_x
offset_x = o_x - center_x
# Add 30 so that the drone covers as much of the subject as possible
# offset_y = obj_center_y - center_y - 30
offset_y = o_y - center_y - 30

cv2.putText(frame1, "Dodge!!!", (10, 20), cv2.FONT_HERSHEY_SIMPLEX,
            1, (0, 0, 255), 3)
cv2.imshow("feed", frame1)
adjust_tello_position( offset_x, offset_y, mz_area)
else:
    goal(70)
cv2.imshow("feed", frame1)
cv2.imshow("edged", edged )

frame1 = frame2
frame2 = frame_read.frame

if cv2.waitKey(1) == ord('q'): # Land the drone once q key is hit
    break

```

```

drone.end()
cv2.destroyAllWindows()

```

## A.4 Object Avoidance second\_OA.py

```

from scipy.spatial import distance as dist
import numpy as np
import cv2
import imutils
import tello_drone as tello
import time

host = ''
port = 9000
local_address = (host, port)

# Pass the is_dummy flag to run on a local camera
drone = tello.Tello(host, port, is_dummy=False)
#drone = tello.Tello(host, port, is_dummy=True) #local camera

frame_read = drone.get_frame_read()

frame1 = frame_read.frame

```



```

frame2 = frame_read.frame

cap = drone.get_video_capture()
#functions for dodging
#still need to test but for right now
#only move to right works
def dodge_right_center(x):
    drone.move_right(x)

#still need to test but for right now
#only move to left works
def dodge_left_center(x):
    drone.move_left(x)

#function for reaching goal
#still need to test but for right now
#only move forward works
def goal(x):
    drone.move_forward(x)

#works great with finding an object based on color
def hsv_Edge(frame):
    #yellow color detection range
    l_b = np.array([22,153,62])
    u_b = np.array([80,255,255])
    #testing other colors
    #l_b = np.array([51,51,00])
    #u_b = np.array([255,255,204])

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    mask =cv2.inRange(hsv,l_b,u_b)

    res = cv2.bitwise_and(frame, frame, mask=mask)

    gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
    edge = cv2.GaussianBlur(gray, (5,5), 0)
    _, edge = cv2.threshold(edge, 20, 255, cv2.THRESH_BINARY)

    return edge
#will keep since it does work for focusing on the object I want
def f_Marker( mark ):

    """
    finds largest contour/object among contours in frame
    ln#74-75
    returns minAreaRect and contourArea of largest object
    """

    contours, hierarchy = cv2.findContours(mark, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

```

```

cnts = [(cv2.contourArea(contour), contour) for contour in contours]

if len(contours)==0:
    return 0, 0, None
else:
    c = max(cnts, key=lambda x: x[0])[1]
    return cv2.minAreaRect( c ), cv2.contourArea(c), c

moved0=True
moved1=False
moved2=False
left= False
right =False
c=0

while True:
    edged = hsv_Edge(frame1)
    marker, area, cnt = f_Marker( edged )

    height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
    width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    # Calculate frame center
    center_x = int(width/2)
    center_y = int(height/2)

    obj_center_x = center_x
    obj_center_y = center_y
    z_area = 0

    """
    if area from f_Marker is below threshold this case 1000 and found
    continues to be false rotate drone which acts as a search for object.
    """

    if area >= 5000 or moved1==True or moved2==True:

        # draw a bounding box around the image and display it
        if moved0 ==True or area>= 5000:

            box = cv2.cv.BoxPoints(marker) if imutils.is_cv2() else
cv2.boxPoints(marker)
            box = np.int0(box)
            cv2.drawContours(frame1, [box], -1, (0, 255, 0), 2)

            M = cv2.moments(cnt)

```

```

(startX, startY, endX, endY) = box

obj_center_x = int((startX[0] + endX[0]) / 2.0)
obj_center_y = int((startY[1] + endY[1]) / 2.0)
z_area = (endX[1]+endY[1]) * (endX[0] + endY[0])

# using moments to calculate center ln. 255
o_x = int(M['m10']/M['m00'])
o_y = int(M['m01']/M['m00'])
mz_area = M['m00']

# Calculate recognized face offset from center
# offset_x = obj_center_x - center_x
offset_x = o_x - center_x
# Add 30 so that the drone covers as much of the subject as possible
# offset_y = obj_center_y - center_y - 30
offset_y = o_y - center_y - 30

cv2.putText(frame1, "Target Acquired:", (10, 20),
cv2.FONT_HERSHEY_SIMPLEX,
1, (0, 0, 255), 3)
cv2.imshow("feed", frame1)
if moved1==True and area < 5000:
    time.sleep(3)
    moved1=False
    moved2=True
    drone.move_forward(120)
    time.sleep(3)
elif moved2==True and area < 5000:
    if left==True:
        time.sleep(1)
        moved2=False
        drone.move_left(90)
        time.sleep(1)
    elif right==True:
        time.sleep(1)
        moved2=False
        drone.move_right(90)
        time.sleep(1)
elif moved0==True or area >= 5000 and c<2:
    if not -90 <= offset_x <= 90 and offset_x is not 0:
        if offset_x < 0:
            left=True
            c+=1
            moved0=False
            moved1=True
            dodge_right_center(90)
            time.sleep(2)

```

```

        elif offset_x > 0:
            right=True
            c+=1
            moved0=False
            moved1=True
            dodge_left_center(90)
            time.sleep(2)
        else:
            right=True
            c+=1
            moved0=False
            moved1=True
            dodge_left_center(90)
            time.sleep(2)
    else:
        right=True
        c+=1
        moved0=False
        moved1=True
        dodge_left_center(90)
        time.sleep(2)

    else:
        goal(70)
        cv2.imshow("feed", frame1)
        cv2.imshow("edged", edged )

    frame1 = frame2
    frame2 = frame_read.frame

    if cv2.waitKey(1) == ord('q'): # Land the drone once q key is hit
        break

drone.end()
cv2.destroyAllWindows()

```

## A.5 Facial Tracking Readme Document

# Face Detection and Tracking with Dji Tello drone

This is an implementation of face detection and tracking on the dji Tello drone based on a HAAR Cascade using OpenCV and Python 3.8.

The current implementation allows the user to:

- Launch the drone by using the command line `python main.py`

- Receive video feed from the drone to the computer and visualize the face detection carried out by the drone

It allows the drone to:

- Detect multiple faces at any given frame
- Position the user at the center of any shot by deciding the best movement based on the users x, y and z coordinates

**Note:** Current implementation allows only tracking of 1 user.

### Quick Start

To initialize your drone and get it up and running, simply clone the repository and download its dependencies with:

```
pip install -r requirements.txt
```

Afterwards, connect to the drones wifi and run:

```
python main.py
```

This will make the drone take off and initialize a video feed directly from the drone to your computer.

Press "Q" to quit the program

### Used Sources

- <https://github.com/DrexelLagare/Senior-Design-1.git>
- <https://github.com/UTRGV-CS-Projects/202020-spring-2020-projects-afs-2.git>

## A.6 Requirements Text File

numpy==1.16.4 ,

opencv-contrib-python==4.1.0.25

Pillow==6.2.0

rope==0.14.0

## A.7 Facial Tracking Main.py

```
import numpy as np
import cv2 as cv
import tello_drone as tello

host = ''
port = 9000
local_address = (host, port)

# Pass the is_dummy flag to run the face detection on a local camera
drone = tello.Tello(host, port, is_dummy=False)

def adjust_tello_position(offset_x, offset_y, offset_z):
    """
    Adjusts the position of the tello drone based on the offset values given from
    the frame

    :param offset_x: Offset between center and face x coordinates
    :param offset_y: Offset between center and face y coordinates
    :param offset_z: Area of the face detection rectangle on the frame
```

```

"""

if not -90 <= offset_x <= 90 and offset_x is not 0:
    if offset_x < 0:
        drone.rotate_ccw(10)
    elif offset_x > 0:
        drone.rotate_cw(10)

if not -70 <= offset_y <= 70 and offset_y is not -30:
    if offset_y < 0:
        drone.move_up(20)
    elif offset_y > 0:
        drone.move_down(20)

if not 15000 <= offset_z <= 30000 and offset_z is not 0:
    if offset_z < 15000:
        drone.move_forward(20)
    elif offset_z > 30000:
        drone.move_backward(20)

face_cascade =
cv.CascadeClassifier('cascades/haarcascade_frontalface_default.xml')

frame_read = drone.get_frame_read()

while True:
    # frame = cv.cvtColor(frame_read.frame, cv.COLOR_BGR2RGB)
    frame = frame_read.frame

    cap = drone.get_video_capture()

```

```

height = cap.get(cv.CAP_PROP_FRAME_HEIGHT)
width = cap.get(cv.CAP_PROP_FRAME_WIDTH)

# Calculate frame center
center_x = int(width/2)
center_y = int(height/2)

# Draw the center of the frame
cv.circle(frame, (center_x, center_y), 10, (0, 255, 0))

# Convert frame to grayscale in order to apply the haar cascade
gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, minNeighbors=5)

# If a face is recognized, draw a rectangle over it and add it to the face
list
face_center_x = center_x
face_center_y = center_y
z_area = 0
for face in faces:
    (x, y, w, h) = face
    cv.rectangle(frame,(x, y),(x + w, y + h),(255, 255, 0), 2)

    face_center_x = x + int(h/2)
    face_center_y = y + int(w/2)

```



```

z_area = w * h

cv.circle(frame, (face_center_x, face_center_y), 10, (0, 0, 255))

# Calculate recognized face offset from center
offset_x = face_center_x - center_x
# Add 30 so that the drone covers as much of the subject as possible
offset_y = face_center_y - center_y - 30

cv.putText(frame, f'[{offset_x}, {offset_y}, {z_area}]', (10, 50),
cv.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv.LINE_AA)
adjust_tello_position(offset_x, offset_y, z_area)

# Display the resulting frame
cv.imshow('Tello detection...', frame)
if cv.waitKey(1) == ord('q'):
    break

# Stop the BackgroundFrameRead and land the drone
drone.end()
cv.destroyAllWindows()

```

## A.8 Facial Tracking tello\_main.py

```

import socket
import threading

```

```

import cv2 as cv

class Tello:
    """
    Handles connection to the DJI Tello drone
    """

    def __init__(self, local_ip, local_port, is_dummy=False,
tello_ip='192.168.10.1', tello_port=8889):
        """
        Initializes connection with Tello and sends both command and streamon
instructions
        in order to start it and begin receiving video feed.
        """
        self.background_frame_read = None
        self.response = None
        self.abort_flag = False
        self.is_dummy = is_dummy

        if not is_dummy:
            self.socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

            self.tello_address = (tello_ip, tello_port)
            self.local_address = (local_ip, local_port)

            self.send_command('command')
            # self.socket.sendto(b'command', self.tello_address)

```

```

print('[INFO] Sent Tello: command')
self.send_command('streamon')
# self.socket.sendto(b'streamon', self.tello_address)
print('[INFO] Sent Tello: streamon')
self.send_command('takeoff')
# self.socket.sendto(b'takeoff', self.tello_address)
print('[INFO] Sent Tello: takeoff')
self.move_up(160)

# thread for receiving cmd ack
self.receive_thread = threading.Thread(target=self._receive_thread)
self.receive_thread.daemon = True

self.receive_thread.start()

def __del__(self):
    """
    Stops communication with Tello
    """
    if not self.is_dummy:
        self.socket.close()

def _receive_thread(self):
    """
    Listen to responses from the Tello.
    Runs as a thread, sets self.response to whatever the Tello last returned.

```

```

"""
while True:
    try:
        self.response, ip = self.socket.recvfrom(3000)
    except socket.error as exc:
        print (f"Caught exception socket.error: {exc}")

def send_command(self, command):
    """
    Send a command to the Tello and wait for a response.
    :param command: Command to send.
    :return (str): Response from Tello.
    """
    self.abort_flag = False
    timer = threading.Timer(0.5, self.set_abort_flag)

    self.socket.sendto(command.encode('utf-8'), self.tello_address)

    timer.start()
    while self.response is None:
        if self.abort_flag is True:
            break
    timer.cancel()

    if self.response is None:
        response = 'none_response'
    else:

```

```

        response = self.response.decode('utf-8')

    self.response = None

    return response

def send_command_without_response(self, command):
    """
    Sends a command without expecting a response. Useful when sending a lot
    of commands.
    """
    if not self.is_dummy:
        self.socket.sendto(command.encode('utf-8'), self.tello_address)

def set_abort_flag(self):
    """
    Sets self.abort_flag to True.

    Used by the timer in Tello.send_command() to indicate to that a response
    timeout has occurred.
    """
    self.abort_flag = True

def move_up(self, dist):
    """
    Sends up command to Tello and returns its response.

```

```

        :param dist: Distance in centimeters in the range 20 - 500.
        :return (str): Response from Tello
        """
        self.send_command_without_response(f'up {dist}')

def move_down(self, dist):
    """
    Sends down command to Tello and returns its response.
    :param dist: Distance in centimeters in the range 20 - 500.
    :return (str): Response from Tello
    """
    self.send_command_without_response(f'down {dist}')

def move_right(self, dist):
    """
    Sends right command to Tello and returns its response.
    :param dist: Distance in centimeters in the range 20 - 500.
    :return (str): Response from Tello
    """
    self.send_command_without_response(f'right {dist}')

def move_left(self, dist):
    """
    Sends left command to Tello and returns its response.
    :param dist: Distance in centimeters in the range 20 - 500.
    :return (str): Response from Tello
    """

```

```

self.send_command_without_response(f'left {dist}')

def move_forward(self, dist):
    """
    Sends forward command to Tello without expecting a return.
    :param dist: Distance in centimeters in the range 20 - 500.
    """
    self.send_command_without_response(f'forward {dist}')

def move_backward(self, dist):
    """
    Sends backward command to Tello without expecting a return.
    :param dist: Distance in centimeters in the range 20 - 500.
    """
    self.send_command_without_response(f'back {dist}')

def rotate_cw(self, deg):
    """
    Sends cw command to Tello in order to rotate clock-wise
    :param deg: Degrees bewteen 0 - 360.
    :return (str): Response from Tello
    """
    self.send_command_without_response(f'cw {deg}')

def rotate_ccw(self, deg):
    """

```

```

Sends ccw command to Tello in order to rotate clock-wise

:param deg: Degrees bewteen 0 - 360.

:return (str): Response from Tello
"""

self.send_command_without_response(f'ccw {deg}')

def get_udp_video_address(self):
    """

    Gets the constructed udp video address for the drone

    :return (str): The constructed udp video address
    """

    return f'udp://{self.tello_address[0]}:11111'

def get_frame_read(self):
    """

    Get the BackgroundFrameRead object from the camera drone. Then, you just
need to call

    backgroundFrameRead.frame to get the actual frame received by the drone.

    :return (BackgroundFrameRead): A BackgroundFrameRead with the video data.
    """

    if self.background_frame_read is None:
        if self.is_dummy:
            self.background_frame_read = BackgroundFrameRead(self, 0).start()
        else:
            self.background_frame_read = BackgroundFrameRead(self,
self.get_udp_video_address()).start()

    return self.background_frame_read

```



```

def get_video_capture(self):
    """
    Get the VideoCapture object from the camera drone
    :return (VideoCapture): The VideoCapture object from the video feed from
the drone.
    """

    if self.cap is None:
        if self.is_dummy:
            self.cap = cv.VideoCapture(0)
        else:
            self.cap = cv.VideoCapture(self.get_udp_video_address())

    if not self.cap.isOpened():
        if self.is_dummy:
            self.cap.open(0)
        else:
            self.cap.open(self.get_udp_video_address())

    return self.cap

def end(self):
    """
    Call this method when you want to end the tello object
    """

    # print(self.send_command('battery?'))

    if not self.is_dummy:
        self.send_command('land')

```

```

        if self.background_frame_read is not None:
            self.background_frame_read.stop()

# It appears that the VideoCapture destructor releases the capture, hence
when

# attempting to release it manually, a segmentation error occurs.

# if self.cap is not None:
#     self.cap.release()

```

```

class BackgroundFrameRead:

```

```

    """

    This class read frames from a VideoCapture in background. Then, just call
backgroundFrameRead.frame to get the

    actual one.

    """

    def __init__(self, tello, address):
        """

        Initializes the Background Frame Read class with a VideoCapture of the
specified

        address and the first frame read.

        :param tello: An instance of the Tello class

        :param address: The UDP address through which the video will be streaming

        """

        tello.cap = cv.VideoCapture(address)

        self.cap = tello.cap

        if not self.cap.isOpened():
            self.cap.open(address)

```

```

self.grabbed, self.frame = self.cap.read()
self.stopped = False

def start(self):
    """
    Starts the background frame read thread.
    :return (BackgroundFrameRead): The current BrackgroundFrameRead
    """
    threading.Thread(target=self.update_frame, args=()).start()
    return self

def update_frame(self):
    """
    Sets the current frame to the next frame read from the source.
    """
    while not self.stopped:
        if not self.grabbed or not self.cap.isOpened():
            self.stop()
        else:
            (self.grabbed, self.frame) = self.cap.read()

def stop(self):
    """
    Stops the frame reading.
    """

```

```
self.stopped = True
```

## A.9 OpenCV License

### License Agreement

For Open Source Computer Vision Library

(3-clause BSD License)

*Copyright (C) 2000-2019, Intel Corporation, all rights reserved.*

*Copyright (C) 2009-2011, Willow Garage Inc., all rights reserved.*

*Copyright (C) 2009-2016, NVIDIA Corporation, all rights reserved.*

*Copyright (C) 2010-2013, Advanced Micro Devices, Inc., all rights reserved.*

*Copyright (C) 2015-2016, OpenCV Foundation, all rights reserved.*

*Copyright (C) 2015-2016, Itseez Inc., all rights reserved.*

*Third party copyrights are property of their respective owners.*

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of the copyright holders nor the names of the contributors may be used to endorse or promote products derived from this software without specific prior written permission.

## A.10 Scipy License

### SciPy license

---

Copyright © 2001, 2002 Enthought, Inc.  
All rights reserved.

Copyright © 2003-2019 SciPy Developers.  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Enthought nor the names of the SciPy Developers may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## A. 11 Pathfinder\_grid.py

```
from pathfinding.core.diagonal_movement import DiagonalMovement
from pathfinding.core.grid import Grid
from pathfinding.finder.a_star import AStarFinder
from scipy.spatial import distance as dist
```

```

import numpy as np
import cv2
import imutils
import tello_drone as tello
import time

host = ''
port = 9000
local_address = (host, port)

#Pass the is_dummy flag to run on a local camera
drone = tello.Tello(host, port, is_dummy=False)
#drone = tello.Tello(host, port, is_dummy=True) #local camera

#drone.rotate_ccw(90)
#drone.rotate_cw(90)
frame_read = drone.get_frame_read()

frame1 = frame_read.frame
frame2 = frame_read.frame

cap = drone.get_video_capture()
#functions for dodging
def dodge_right_center(x):
    drone.move_right(x)

def dodge_left_center(x):
    drone.move_left(x)

#function for reaching goal
def goal(x):
    drone.move_forward(x)

#function for moving forward in the space(checks if the space infront of
drone is taken, then moves or rotates)
#works great with finding an object based on color
def move_in_space(blocked,cardinal_direction,grid,positionx,positiony):
    if cardinal_direction== 'N':
        if grid[positionx][positiony-1]==0 and blocked==0:
            positionx=positionx
            positiony=positiony-1
            drone.move_forward(20)
        else:
            if cardinal_direction== 'N':
                cardinal_direction= 'E'
            elif cardinal_direction== 'E':
                cardinal_direction= 'S'
            elif cardinal_direction== 'S':
                cardinal_direction= 'W'
            elif cardinal_direction== 'W':
                cardinal_direction= 'N'
            drone.rotate_cw(90)
            grid[positionx][positiony-1]=1

```

```

elif cardinal_direction== 'E':
    if grid[positionx+1][positiony]==0 and blocked==0:
        positionx=positionx+1
        positiony=positiony
        drone.move_forward(20)
    else:
        if cardinal_direction== 'N':
            cardinal_direction= 'E'
        elif cardinal_direction== 'E':
            cardinal_direction= 'S'
        elif cardinal_direction== 'S':
            cardinal_direction= 'W'
        elif cardinal_direction== 'W':
            cardinal_direction= 'N'
        drone.rotate_cw(90)
        grid[positionx+1][positiony]=1
elif cardinal_direction== 'S':
    if grid[positionx][positiony+1]==0 and blocked==0:
        positionx=positionx
        positiony=positiony+1
        drone.move_forward(20)
    else:
        if cardinal_direction== 'N':
            cardinal_direction= 'E'
        elif cardinal_direction== 'E':
            cardinal_direction= 'S'
        elif cardinal_direction== 'S':
            cardinal_direction= 'W'
        elif cardinal_direction== 'W':
            cardinal_direction= 'N'
        drone.rotate_cw(90)
        grid[positionx][positiony+1]=1
elif cardinal_direction== 'W':
    if grid[positionx-1][positiony]==0 and blocked==0:
        positionx=positionx-1
        positiony=positiony
        drone.move_forward(20)
    else:
        if cardinal_direction== 'N':
            cardinal_direction= 'E'
        elif cardinal_direction== 'E':
            cardinal_direction= 'S'
        elif cardinal_direction== 'S':
            cardinal_direction= 'W'
        elif cardinal_direction== 'W':
            cardinal_direction= 'N'
        drone.rotate_cw(90)
        grid[positionx-1][positiony]=1
print("moving")
return 0,cardinal_direction,grid,positionx,positiony
def hsv_Edge(frame):
    #yellow color detection range
    l_b = np.array([22,153,62])
    u_b = np.array([80,255,255])

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

```

```

mask =cv2.inRange(hsv,l_b,u_b)

res = cv2.bitwise_and(frame, frame, mask=mask)

gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
edge = cv2.GaussianBlur(gray, (5,5), 0)
_, edge = cv2.threshold(edge, 20, 255, cv2.THRESH_BINARY)

    return edge
#will keep since it does work for focusing on the object I want
def f_Marker( mark ):

    """
    finds largest contour/object among contours in frame
    ln#74-75
    returns minAreaRect and contourArea of largest object
    """

    contours, hierarchy = cv2.findContours(mark, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    cnts = [(cv2.contourArea(contour), contour) for contour in contours]

    if len(contours)==0:
        return 0, 0, None
    else:
        c = max(cnts, key=lambda x: x[0])[1]
        return cv2.minAreaRect( c ), cv2.contourArea(c), c

def drone_front_cell(positionx, positiony, cardinal_direction):
    if(cardinal_direction=='N'):
        positionx=positionx
        positiony=positiony-1
        return positionx, positiony
    elif(cardinal_direction=='E'):
        positionx=positionx+1
        positiony=positiony
        return positionx, positiony
    elif(cardinal_direction=='S'):
        positionx=positionx
        positiony=positiony+1
        return positionx, positiony
    elif(cardinal_direction=='W'):
        positionx=positionx-1
        positiony=positiony
        return positionx, positiony

def new_path(x, y, cd, my_grid, blocked, blocked_path, endx, endy):

    if(blocked==1):
        mx=blocked_path
        a, b= drone_front_cell(positionx=x,positiony=y,cardinal_direction=cd)
        print(b-1,a-1)

```



```

        if(b-1!=5 and a-1!=5):
            mx[b-1][a-1]=0
            grid = Grid(matrix=mx)
            start = grid.node(x-1, y-1)
        else:
            mx=blocked_path
            grid = Grid(matrix=mx)
            start = grid.node(x-1, y-1)

        end = grid.node(endx, endy)
        finder = AStarFinder(diagonal_movement=DiagonalMovement.never)
        #print(str(finder.find_neighbors))
        path, runs = finder.find_path(start, end, grid)
        test_path=grid.grid_str(path=path, start=start, end=end)
        print(test_path)
        pathing=0
        for i in range(7):
            for j in range(7):
                if test_path[pathing]!='s' and test_path[pathing]!='e' and
test_path[pathing]!='x':
                    my_grid[j][i]=1
                else:
                    my_grid[j][i]=0
                    print(i,j)
                pathing+=1
            pathing+=1
        grid.cleanup()
        print(my_grid,mx)
        return my_grid, mx

```

```

distance=0
blocked= 0
positionx=5
positiony=5
cardinal_direction= 'N'
startx,starty =4,4
endx,endy = 4,0
matrix = [
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1],
]
my_grid= [
    [1, 1, 1, 1, 1, 1, 1],
    [1, 0, 0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0, 0, 1],
    [1, 1, 1, 1, 1, 1, 1],

```

```

]
#time.sleep(10)
#my_grid,matrix=new_path(x=positionx,y=positiony,cd=cardinal_direction,my_gri
d=my_grid,blocked=blocked,blocked_path=matrix,endx=endx,andy=endy)
#time.sleep(5)
while True:
    edged = hsv_Edge(frame1)
    marker, area, cnt = f_Marker( edged )

    height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
    width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    # Calculate frame center
    center_x = int(width/2)
    center_y = int(height/2)
    obj_center_x = center_x
    obj_center_y = center_y
    z_area = 0

    """
    if area from f_Marker is below threshold this case 1000 and found
    continues to be false rotate drone which acts as a search for object.
    """
    if area >= 1:

        box = cv2.cv.BoxPoints(marker) if imutils.is_cv2() else
cv2.boxPoints(marker)
        box = np.int0(box)
        cv2.drawContours(frame1, [box], -1, (0, 255, 0), 2)

        M = cv2.moments(cnt)

        #(startX, startY, endX, endY) = box

        #obj_center_x = int((startX[0] + endX[0]) / 2.0)
        #obj_center_y = int((startY[1] + endY[1]) / 2.0)
        #z_area = (endX[1]+endY[1]) * (endX[0] + endY[0])
        #focal_length= (marker[1][0]*24.0)/8.0
        #cv2.putText(frame1, "Dodge!!!", (10, 20), cv2.FONT_HERSHEY_SIMPLEX,
        #              1, (0, 0, 255), 3)
        #cv2.imshow("feed", frame1)
        distance= (3.5*775.4265270996094)/marker[1][0]
        #print(distan)
        if 3.5 <=distance<=5:
            blocked=1
            cv2.putText(frame1, "Blocked!!", (10, 20),
cv2.FONT_HERSHEY_SIMPLEX,
                        1, (0, 0, 255), 3)
            cv2.imshow("feed", frame1)
            print("this blocked")
            print(blocked)
            """
            time.sleep(4)
            print("before move:")

```

```

        print(blocked, cardinal_direction, positionx, positiony)
        (blocked, cardinal_direction, grid, positionx, positiony) =
move_in_space(blocked, cardinal_direction, grid, positionx, positiony)
        print("after move:")
        print(blocked, cardinal_direction, positionx, positiony)
        time.sleep(4)
        """
time.sleep(5)

        #print(distance)
        #time.sleep(5)
        #if blocked ==0
        #Will land if the drone reaches the end of th path
        #a,b=drone_front_cell(positionx,positiony,cardinal_direction)
        if positionx-1== endx and positiony-1==endy:
            print("reached end of path")
            break

my_grid,matrix=new_path(x=positionx,y=positiony,cd=cardinal_direction,my_grid
=my_grid,blocked=blocked,blocked_path=matrix,endx=endx,endy=endy)
        #print(my_grid)
        #time.sleep(5)
        #print("before move:")
        #print(distance,blocked,cardinal_direction,positionx,positiony)
        (blocked,cardinal_direction,my_grid,positionx,positiony)=
move_in_space(blocked,cardinal_direction,my_grid,positionx,positiony)
        #print(my_grid)

        #print("after move:")
        print(distance,blocked,cardinal_direction,positionx,positiony)
        time.sleep(5)
        #elif blocked==1

        cv2.imshow("feed", frame1)
        cv2.imshow("edged", edged )

        frame1 = frame2
        frame2 = frame_read.frame

        if cv2.waitKey(1) & 0xFF == ord('q'): # Land the drone once q key is hit
            break

drone.end()
cv2.destroyAllWindows()

"""
#grid = Grid(matrix=matrix)
#grid.neighbors
#start = grid.node(4, 4)
#end = grid.node(4, 0)

```

```

finder = AStarFinder(diagonal_movement=DiagonalMovement.never)
#print(str(finder.find_neighbors))
path, runs = finder.find_path(start, end, grid)
test_path=grid.grid_str(path=path, start=start, end=end)
#print('operations:', runs, 'path length:', len(path))
#print(test_path)
#my_grid = [[0]*7 for i in range(7)]
pathing=0
#print(my_grid)
for i in range(7):
    for j in range(7):
        #if i == 0:
        #    my_grid[i][j]=1
        #elif i==6:
        #    my_grid[i][j]=1
        #elif j==0:
        #    my_grid[i][j]=1
        #elif j==6:
        #    my_grid[i][j]=1
        if test_path[pathing]!='s' and test_path[pathing]!='e' and
test_path[pathing]!='x':
            my_grid[i][j]=1
            pathing+=1
    pathing+=1
#print(my_grid)
#####
grid.cleanup()
print("new object in path")
my_grid = [[0]*7 for i in range(7)]
pathing=0
matrix = [
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 0],
    [1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1]
]
grid = Grid(matrix=matrix)
start = grid.node(4, 4)
end = grid.node(4, 0)
finder = AStarFinder(diagonal_movement=DiagonalMovement.never)
path, runs = finder.find_path(start, end, grid)
test_path=grid.grid_str(path=path, start=start, end=end)
print(test_path)
print(my_grid)
for i in range(7):
    for j in range(7):
        if i == 0:
            my_grid[i][j]=1
        elif i==6:
            my_grid[i][j]=1
        elif j==0:
            my_grid[i][j]=1
        elif j==6:
            my_grid[i][j]=1
        if test_path[pathing]!='s' and test_path[pathing]!='e' and
test_path[pathing]!='x':

```

```

        my_grid[i][j]=1
        pathing+=1
    pathing+=1
print(my_grid)
"""

```

## A. 11 free\_roam\_grid.py

```

"""
HSV edging and coordinate systems were pulled from the previous AFS group
Additions include: Dodge, Re-center, goal reached
Code Reference: https://github.com/DrexelLagare/Senior-Design-1/blob/master/Object%20Tracking/target\_A.py
"""

from scipy.spatial import distance as dist
import numpy as np
import cv2
import imutils
import tello_drone as tello
import time

host = ''
port = 9000
local_address = (host, port)

# Pass the is_dummy flag to run on a local camera
drone = tello.Tello(host, port, is_dummy=False)
#drone = tello.Tello(host, port, is_dummy=True) #local camera

#drone.rotate_ccw(90)
#drone.rotate_cw(90)
frame_read = drone.get_frame_read()

frame1 = frame_read.frame
frame2 = frame_read.frame

cap = drone.get_video_capture()
#functions for dodging
def dodge_right_center(x):
    drone.move_right(x)

def dodge_left_center(x):
    drone.move_left(x)

#function for reaching goal
def goal(x):
    drone.move_forward(x)

#function for moving forward in the space(checks if the space infront of
drone is taken, then moves or rotates)

```

```

#works great with finding an object based on color
def move_in_space(blocked,cardinal_direction,grid,positionx,positiony):
    if cardinal_direction== 'N':
        if grid[positionx][positiony-1]==0 and blocked==0:
            positionx=positionx
            positiony=positiony-1
            drone.move_forward(20)
        else:
            if cardinal_direction== 'N':
                cardinal_direction= 'E'
            elif cardinal_direction== 'E':
                cardinal_direction= 'S'
            elif cardinal_direction== 'S':
                cardinal_direction= 'W'
            elif cardinal_direction== 'W':
                cardinal_direction= 'N'
            drone.rotate_cw(90)
            grid[positionx][positiony-1]=1
    elif cardinal_direction== 'E':
        if grid[positionx+1][positiony]==0 and blocked==0:
            positionx=positionx+1
            positiony=positiony
            drone.move_forward(20)
        else:
            if cardinal_direction== 'N':
                cardinal_direction= 'E'
            elif cardinal_direction== 'E':
                cardinal_direction= 'S'
            elif cardinal_direction== 'S':
                cardinal_direction= 'W'
            elif cardinal_direction== 'W':
                cardinal_direction= 'N'
            drone.rotate_cw(90)
            grid[positionx+1][positiony]=1
    elif cardinal_direction== 'S':
        if grid[positionx][positiony+1]==0 and blocked==0:
            positionx=positionx
            positiony=positiony+1
            drone.move_forward(20)
        else:
            if cardinal_direction== 'N':
                cardinal_direction= 'E'
            elif cardinal_direction== 'E':
                cardinal_direction= 'S'
            elif cardinal_direction== 'S':
                cardinal_direction= 'W'
            elif cardinal_direction== 'W':
                cardinal_direction= 'N'
            drone.rotate_cw(90)
            grid[positionx][positiony+1]=1
    elif cardinal_direction== 'W':
        if grid[positionx-1][positiony]==0 and blocked==0:
            positionx=positionx-1
            positiony=positiony
            drone.move_forward(20)
        else:
            if cardinal_direction== 'N':

```

```

        cardinal_direction= 'E'
    elif cardinal_direction== 'E':
        cardinal_direction= 'S'
    elif cardinal_direction== 'S':
        cardinal_direction= 'W'
    elif cardinal_direction== 'W':
        cardinal_direction= 'N'
    drone.rotate_cw(90)
    grid[positionx-1][positiony]=1
print("moving")
return 0,cardinal_direction,grid,positionx,positiony
def hsv_Edge(frame):
    #yellow color detection range
    l_b = np.array([22,153,62])
    u_b = np.array([80,255,255])

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    mask =cv2.inRange(hsv,l_b,u_b)

    res = cv2.bitwise_and(frame, frame, mask=mask)

    gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
    edge = cv2.GaussianBlur(gray, (5,5), 0)
    _, edge = cv2.threshold(edge, 20, 255, cv2.THRESH_BINARY)

    return edge
#will keep since it does work for focusing on the object I want
def f_Marker( mark ):
    """
    finds largest contour/object among contours in frame
    ln#74-75
    returns minAreaRect and contourArea of largest object
    """

    contours, hierarchy = cv2.findContours(mark, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    cnts = [(cv2.contourArea(contour), contour) for contour in contours]

    if len(contours)==0:
        return 0, 0, None
    else:
        c = max(cnts, key=lambda x: x[0])[1]
        return cv2.minAreaRect( c ), cv2.contourArea(c), c
distance=0
blocked= 0
positionx=5
positiony=5
cardinal_direction= 'N'
#will setup grid
#1111111

```

```

#1000001
#1000001
#1000001
#1000001
#1000001
#1111111
grid = [[0]*7 for i in range(7)]
for i in range(7):
    for j in range(7):
        if i == 0:
            grid[i][j]=1
        elif i==6:
            grid[i][j]=1
        elif j==0:
            grid [i][j]=1
        elif j==6:
            grid [i][j]=1
#for changing direction when rotating clockwise
#
#if cardinal_direction== 'N':
#    cardinal_direction= 'E'
#elif cardinal_direction== 'E':
#    cardinal_direction= 'S'
#elif cardinal_direction== 'S':
#    cardinal_direction= 'W'
#elif cardinal_direction== 'W':
#    cardinal_direction= 'N'
#forward direction position
"""
if cardinal_direction== 'N':
    if grid[positionx][positiony-1]==0:
        positionx=positionx
        positiony=positiony-1
        drone.move_forward(40)
    else:
        if cardinal_direction== 'N':
            cardinal_direction= 'E'
        elif cardinal_direction== 'E':
            cardinal_direction= 'S'
        elif cardinal_direction== 'S':
            cardinal_direction= 'W'
        elif cardinal_direction== 'W':
            cardinal_direction= 'N'
elif cardinal_direction== 'E:'
    if grid[positionx+1][positiony]==0:
        positionx=positionx+1
        positiony=positiony
        drone.move_forward(40)
    else:
        if cardinal_direction== 'N':
            cardinal_direction= 'E'
        elif cardinal_direction== 'E':
            cardinal_direction= 'S'
        elif cardinal_direction== 'S':
            cardinal_direction= 'W'
        elif cardinal_direction== 'W':
            cardinal_direction= 'N'

```



```

elif cardinal_direction== 'S':
    if grid[positionx][positiony+1]==0:
        positionx=positionx
        positiony=positiony+1
        drone.move_forward(40)
    else:
        if cardinal_direction== 'N':
            cardinal_direction= 'E'
        elif cardinal_direction== 'E':
            cardinal_direction= 'S'
        elif cardinal_direction== 'S':
            cardinal_direction= 'W'
        elif cardinal_direction== 'W':
            cardinal_direction= 'N'
elif cardinal_direction== 'W':
    if grid[positionx-1][positiony]==0:
        positionx=positionx-1
        positiony=positiony
        drone.move_forward(40)
    else:
        if cardinal_direction== 'N':
            cardinal_direction= 'E'
        elif cardinal_direction== 'E':
            cardinal_direction= 'S'
        elif cardinal_direction== 'S':
            cardinal_direction= 'W'
        elif cardinal_direction== 'W':
            cardinal_direction= 'N'
"""
#will be needing time.sleep() to make movements and process images
while True:
    edged = hsv_Edge(frame1)
    marker, area, cnt = f_Marker( edged )

    height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
    width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    # Calculate frame center
    center_x = int(width/2)
    center_y = int(height/2)
    obj_center_x = center_x
    obj_center_y = center_y
    z_area = 0

    """
    if area from f_Marker is below threshold this case 1000 and found
    continues to be false rotate drone which acts as a search for object.
    """
    if area >= 1:

        box = cv2.cv.BoxPoints(marker) if imutils.is_cv2() else
cv2.boxPoints(marker)
        box = np.int0(box)
        cv2.drawContours(frame1, [box], -1, (0, 255, 0), 2)

        M = cv2.moments(cnt)

```

```

#(startX, startY, endX, endY) = box

#obj_center_x = int((startX[0] + endX[0]) / 2.0)
#obj_center_y = int((startY[1] + endY[1]) / 2.0)
#z_area = (endX[1]+endY[1]) * (endX[0] + endY[0])
#focal_length= (marker[1][0]*24.0)/8.0
#cv2.putText(frame1, "Dodge!!!", (10, 20), cv2.FONT_HERSHEY_SIMPLEX,
#           1, (0, 0, 255), 3)
#cv2.imshow("feed", frame1)
distance= (3.5*775.4265270996094)/marker[1][0]
if 3.5 <=distance<=5:
    blocked=1
    cv2.putText(frame1, "Blocked!!", (10, 20),
cv2.FONT_HERSHEY_SIMPLEX,
           1, (0, 0, 255), 3)
    cv2.imshow("feed", frame1)
    """
    time.sleep(4)
    print("before move:")
    print(blocked,cardinal_direction,positionx,positiony)
    (blocked,cardinal_direction,grid,positionx,positiony)=
move_in_space(blocked,cardinal_direction,grid,positionx,positiony)
    print("after move:")
    print(blocked,cardinal_direction,positionx,positiony)
    time.sleep(4)
    """

    #print(distance)
    #time.sleep(5)
    #if blocked ==0
time.sleep(3)
    print("before move:")
    print(distance,blocked,cardinal_direction,positionx,positiony)
    (blocked,cardinal_direction,grid,positionx,positiony)=
move_in_space(blocked,cardinal_direction,grid,positionx,positiony)
    print("after move:")
    print(distance,blocked,cardinal_direction,positionx,positiony)
    time.sleep(3)
    #elif blocked==1

cv2.imshow("feed", frame1)
cv2.imshow("edged",edged )

frame1 = frame2
frame2 = frame_read.frame

if cv2.waitKey(1) & 0xFF == ord('q'): # Land the drone once q key is hit
    break

```

```
drone.end()
cv2.destroyAllWindows()
```

## A. 12 Waypoint.py

```
from scipy.spatial import distance as dist,
from djitellopy import Tello
import numpy as np
import cv2
import imutils
import tello_drone_for_calculator as tello
import time
import pandas as pd
import matplotlib.pyplot as plt

host = ''
port = 9000
local_address = (host, port)
startCounter= 0

drone = tello.Tello(host, port, is_dummy=False)
fly = Tello()
fly.connect()
fly.for_back_velocity = 0
fly.left_right_velocity = 0
fly.up_down_velocity = 0
fly.yaw_velocity = 0
fly.speed = 0
frame_read = drone.get_frame_read()

frame1 = frame_read.frame
frame2 = frame_read.frame

cap = drone.get_video_capture()
#works great with finding an object based on color
def hsv_Edge(frame):
    #yellow color detection range
    l_b = np.array([22,153,62])
    u_b = np.array([80,255,255])

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    mask =cv2.inRange(hsv,l_b,u_b)

    res = cv2.bitwise_and(frame, frame, mask=mask)

    gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
    edge = cv2.GaussianBlur(gray, (5,5), 0)
    _, edge = cv2.threshold(edge, 20, 255, cv2.THRESH_BINARY)
```

```

return edge

def f_Marker( mark ):

    """
    finds largest contour/object among contours in frame
    ln#74-75
    returns minAreaRect and contourArea of largest object
    """

    contours, hierarchy = cv2.findContours(mark, cv2.RETR_TREE,
    cv2.CHAIN_APPROX_SIMPLE)

    cnts = [(cv2.contourArea(contour), contour) for contour in contours]

    if len(contours)==0:
        return 0, 0, None
    else:
        c = max(cnts, key=lambda x: x[0])[1]
        return cv2.minAreaRect( c ), cv2.contourArea(c), c

    while True:
        edged = hsv_Edge(frame1)
        marker, area, cnt = f_Marker( edged )

    height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
    width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    # Calculate frame center
    center_x = int(width/2)
    center_y = int(height/2)

    obj_center_x = center_x
    obj_center_y = center_y
    z_area = 0

    """
    if area from f_Marker is below threshold this case 1000 and found
    continues to be false rotate drone which acts as a search for object.
    """

    if area >= 1:

        box = cv2.cv.BoxPoints(marker) if imutils.is_cv2() else cv2.boxPoints(marker)
        box = np.int0(box)
        cv2.drawContours(frame1, [box], -1, (0, 255, 0), 2)

        M = cv2.moments(cnt)

        (startX, startY, endX, endY) = box

```

```

obj_center_x = int((startX[0] + endX[0]) / 2.0)
obj_center_y = int((startY[1] + endY[1]) / 2.0)
z_area = (endX[1]+endY[1]) * (endX[0] + endY[0])

cv2.imshow("feed", frame1)
cv2.imshow("edged",edged)

# GET THE IMAGE FROM TELLO
frame_read = drone.get_frame_read()
frame = frame_read.frame

l_b = np.array([22,153,62])
u_b = np.array([80,255,255])
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
mask =cv2.inRange(hsv,l_b,u_b)
result = cv2.bitwise_and(hsv,hsv, mask = mask)
mask = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)

imgBlur = cv2.GaussianBlur(result, (7, 7), 1)
imgGray = cv2.cvtColor(imgBlur, cv2.COLOR_BGR2GRAY)
kernel = np.ones((5, 5))

##### FLIGHT
if startCounter == 0:
    fly.takeoff()
    startCounter = 1

if dir == 1:
    fly.for_back_velocity= 60
elif dir == 2:
    fly.for_back_velocity= -60
elif dir == 3:
    fly.yaw_velocity= 60
elif dir == 4:
    fly.yaw_velocity = -60
elif dir == 5:
    fly.up_down_velocity= 60
elif dir == 6:
    fly.up_down_velocity= -60
else:
    fly.left_right_velocity = 0; fly.for_back_velocity = 0; fly.up_down_velocity
    = 0; fly.yaw_velocity = 0;
# SEND VELOCITY VALUES TO TELLO
if fly.send_rc_control:
    fly.send_rc_control(fly.left_right_velocity, fly.up_down_velocity,
    fly.yaw_velocity, fly.for_back_velocity)
    print(dir)

cv2.imshow("feed", frame1)
cv2.imshow("edged",edged )

if cv2.waitKey(1) & 0xFF == ord('q'): # Land the drone once q key is hit
    img_name='focalLengthAndDistance.jpg'
    cv2.imwrite(img_name,frame1)
    break

```

```

frame1 = frame2
frame2 = frame_read.frame

```

```

drone.end()
cv2.destroyAllWindows()
#####
#Need to change this for the file path of where you have the folder for this
project
#DO NOT change the text focalLengthAndDistance.jpg, since that is the name of
the file to be processed
image_path = r'C:\Users\Owner\Desktop\focalCode\202020-spring-2020-projects-
afs-2\spatialvisualawareness\focalLengthAndDistance.jpg'
#####
focalImage=cv2.imread(image_path)
edged = hsv_Edge(focalImage)
marker, area, cnt = f_Marker(edged)
#####
#These are the values you will be changing based on your circumstances
obj_width= 8.0
predefined_distance_between_obj_and_drone=24.0
#####
focal_length=
(marker[1][0]*predefined_distance_between_obj_and_drone)/obj_width
distance= (7.75*focal_length)/marker[1][0]
print("focal length: "+str(focal_length))
print("distance: "+str(distance))

```