Autonomous Racing Drone

ECE4012 Senior Design Project

Section L8A, Auto Quads
Project Faculty Advisor: Jennifer Hasler

Max Rudolph, EE, mrudolph8@gatech.edu
Dave Patel, CmpE, dpatel344@gatech.edu
Eddie Stevens, CmpE, estevens9@gatech.edu
Michael Bermudez, CmpE, mbermudez3@gatech.edu
Nyaire Najieb, CmpE, nyaire.najieb@gatech.edu
Rishov Sarkar, CmpE, rishov.sarkar@gatech.edu
Suhani Jain, CmpE, sjain340@gatech.edu

Submitted

2020 April 29

Executive Summary

In this modern age of autonomy, robotics has become more ubiquitous in society and our everyday lives. For instance, autonomous package delivery is beginning to become viable technologically and legally [6]. These drone systems need to be reliable, safe, and, most importantly, fast. For this reason, we were originally designing an autonomous racing drone that can navigate a course without the interaction with a human pilot. After COVID-19, the final goal of the team switched from delivering a fully autonomous race drone to a proof of concept that would showcase the possible autonomy of a drone.

Due to the switch to remote learning, we had to convert our project to have more simulation-based deliverables. Our project has laid the groundwork for a fully autonomous drone in the future. This report discusses the simulated deliverables that we built; these deliverables include various disjoint pieces of software such as computer vision scripts, velocity control algorithms, and a drone simulator. The software we have written acts as a proof of concept for the autonomy of the drone. Finally, we can also deliver the physical drone complete with hardware to use the NVIDIA Jetson Nano as the compute unit and Pi Cam as the vision unit. The bulk of our cost consisted of the hardware required. In total, we needed $575.92 for drone parts and compute units. We were able to use the several documented successes of autonomous drones navigating a course in order to give inspiration and ideas to our own work [1]. Although we encountered many hiccups due to COVID-19, we were able to make ample progress in designing our autonomous quadcopter.

Table of Contents

Autonomous Racing Drone

1. Introduction

Our team, AutoQuads, is requesting $1,000 of funding to design and develop an autonomous racing drone: a drone that is able to autonomously pilot itself through professional drone racing courses. The design will require a drone kit for simplified customization, a Jetson Nano Development Kit for computations, and a Pixhawk flight controller to interface with the motors. **Figure 1** helps visualize the structure.

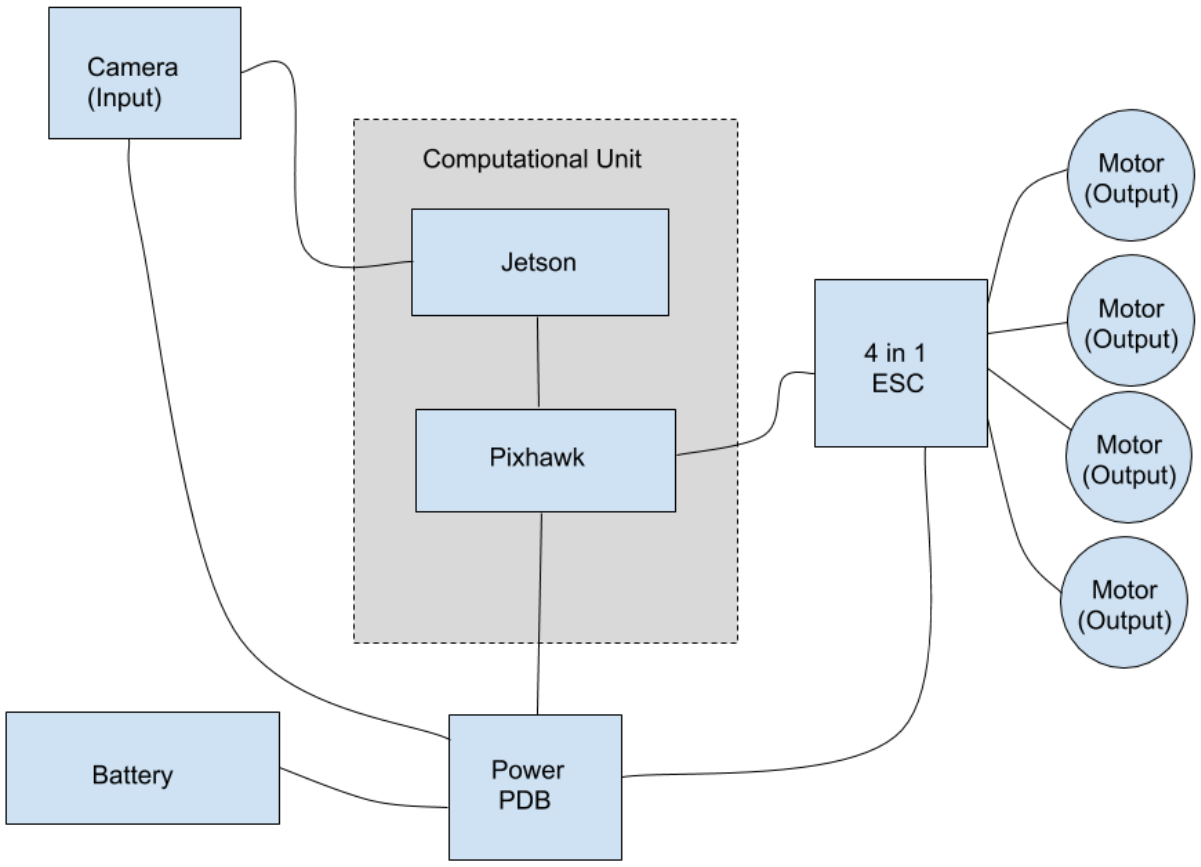


Figure 1. Block diagram of proposed autonomous flight system

1.1 Objective

The objective of the drone is to autonomously fly in an unmapped domain. The original intention was for the racing drone to successfully compute a path to the goal and adjust the trajectory in real time as it senses its local environment and detects and prevents collisions. With the adjusted circumstances and timeline, our objective shifted to simply autonomously flying the drone and detecting a single checkerboard for localization.

1.2 Motivation

Beyond the team's fascination with autonomy, we chose to do this project because we believe autonomous drones are the future of package delivery, warfare, and surveillance. **Delivery:** Companies such as Wing and Amazon Prime Air have already shown that there is a demand for autonomy in drone flights; they use large autonomous drones to deliver packages all over the west coast. **Warfare and Surveillance:** along with most three letter government agencies, there are several companies, like Equinox Innovations, that are developing tethered drone systems for surveillance. They have already implemented autonomy by creating flight routines that launch, fly, and land the drone without human interaction. Clearly there is a market for dynamic, autonomous drones so we have decided that creating race drones is a good way to break into the industry.

1.3 Background

The drone will perform image processing onboard using a GPU-enabled embedded computing platform, such as the NVIDIA Jetson or Raspberry Pi. There are several controllers that are already enabled with this capability. For example, Georgia Tech's own AutoRally research platform uses a model predictive controller to plan paths for off-road autonomous rally cars. The controller, called Model Predictive Path Integral, samples thousands of stochastic and achievable

trajectories and averages them to obtain one reliable and optimal path. MPC's are highly parallelizable and thus well suited to real-time control using discrete GPUs. Along with path planners we need fast and accurate perception sensors. Another example that touches on all of these factors involves the Artificial Intelligence Robotic Racing (AIRR) from the Drone Racing League (DRL) responsible for performing AI racing. The goal of this DRL Drone AI is to defeat a human in a physical sport through navigating challenging obstacles at a top speed of 70 miles per hour. The DRL Drone AI has a configuration to provide the CV with non-obstructive frontal view during racing. It is equipped with a NVIDIA Jetson AGX Xavier, and is connected to four stereoscopic cameras that help the AI to detect and identify objects with twice the field of view as compared to human pilots [2].

2. Project Description and Goals

Our team assembled an autonomous drone using commercial off-the-shelf drone parts and a single-board computer, the NVIDIA Jetson Nano [3]. We wrote and demonstrated the possibility for custom tracking software to take input from the drone's camera modules and perform computer vision in real time.

We simulated a proof of concept in MATLAB/Python that located the targets, and we separately demonstrated how flight commands would be issued to the drone. However, given the COVID-19 situation and separated team members, we were unable to package the software into one cohesive system. With this in mind, the list of goals below reflects what we believe our ideal, complete software suite should be able to do.

- The final product should be able to autonomously navigate a course through non-predetermined goal points and avoid obstacles.
- The drone should be able to autonomously fly through goal points, measured by the tolerance between a goal point and the point on the actual flight path closest to it.
- It should be able to avoid obstacles, measured by the number of times the drone hits an obstacle on a test course.
- It should be small and lightweight.
- The target price for the product is $500, considering that a drone kit, camera, battery, and NVIDIA Jetson board together cost around $450.
- The targeted end user is drone enthusiasts.
- Originally, the drone was expected to maintain high speeds during its performance, but hardware limitations, specifically from the camera interface, caused severe hiccups in frame processing. The rate of motion blur was excessive at higher speeds, and we lacked the ability to accurately identify obstacles. Given more time and a faster camera module, we could improve our object detection algorithm to compensate for the motion blur.

3. Technical Specifications & Verification

3.1 Hardware Specifications

Table 1 contains the specifications for the hardware components of the drone.

Item	Requirement	Satisfied?
Total Weight	< 7 lb	Yes
Computation Power	> 4 cores + discrete GPU	Yes
Horizontal Speed	> 40 mph	No
Communication Range	200 meters	N/A
Communication Frequency	5.8 GHz	Yes
Computation Power consumption	< 5 Watts	Yes
Total Power Consumption	~ 70 Watts	Yes

3.2 Software Specifications

Table 2 contains the specifications for the software requirements necessary to navigate the course at high speed

Item	Requirement	Satisfied?
Data transfer	50 Mbits/sec	N/A
Image lag	30 ms	Yes
Structure	Highly parallelizable and	No
Language	C++	No -- python
Real Time Image Analysis	5 Hz	Yes, 15 Hz

3.3 Final Product Specifications

Table 3 contains the specifications for the final product as a cohesive unit with hardware and software specifications

Item	Achieved?
Take Off Autonomously	Yes
Reach Given Waypoint	Yes
Track Goal Post (Checkerboard)	Yes
Approach Goal Post Given Path	Yes
Test on Drone	No

4. Design Approach and Details

4.1 Design Approach

In this section, we will discuss our design approach from before and after the start of remote learning. As will be mentioned in the scheduling and tasks section of the report, before we left school, we were able to fully build the drone, test that it flies stably, and begin on the computer vision scripts. After remote learning began, we decided to continue working on the computer vision and prove that it could work if/when it was integrated with the drone.

Our post-COVID design approach was two-fold. Firstly, we wanted to simulate the drone. Secondly, we wanted to simulate the computer vision.

Simulation

We used the dronekit and dronekit_sitl libraries in python to simulate the ArduPilot software and physical drone [7]. The two libraries are open-source python packages that are SITL (software-in-the-loop) simulators that allow for the testing and creation of ArduPilot python applications without the need for hardware. The simulators are not visual/physics simulators and are solely used for the testing of flight commands. **Figure 2** is a code snippet, and **Figure 3** is a visualization from the drone simulation.

```
print("Start simulator (SITL)")

sitl = dronekit_sitl.start_default()
connection_string = sitl.connection_string()

# Import DroneKit-Python
from dronekit import connect, VehicleMode, LocationGlobalRelative

# Connect to the Vehicle.
print("Connecting to vehicle on: %s" % (connection_string,))
vehicle = connect(connection_string, wait_ready=True)
```

Figure 2. Screen shot of dronekit simulator built in Python

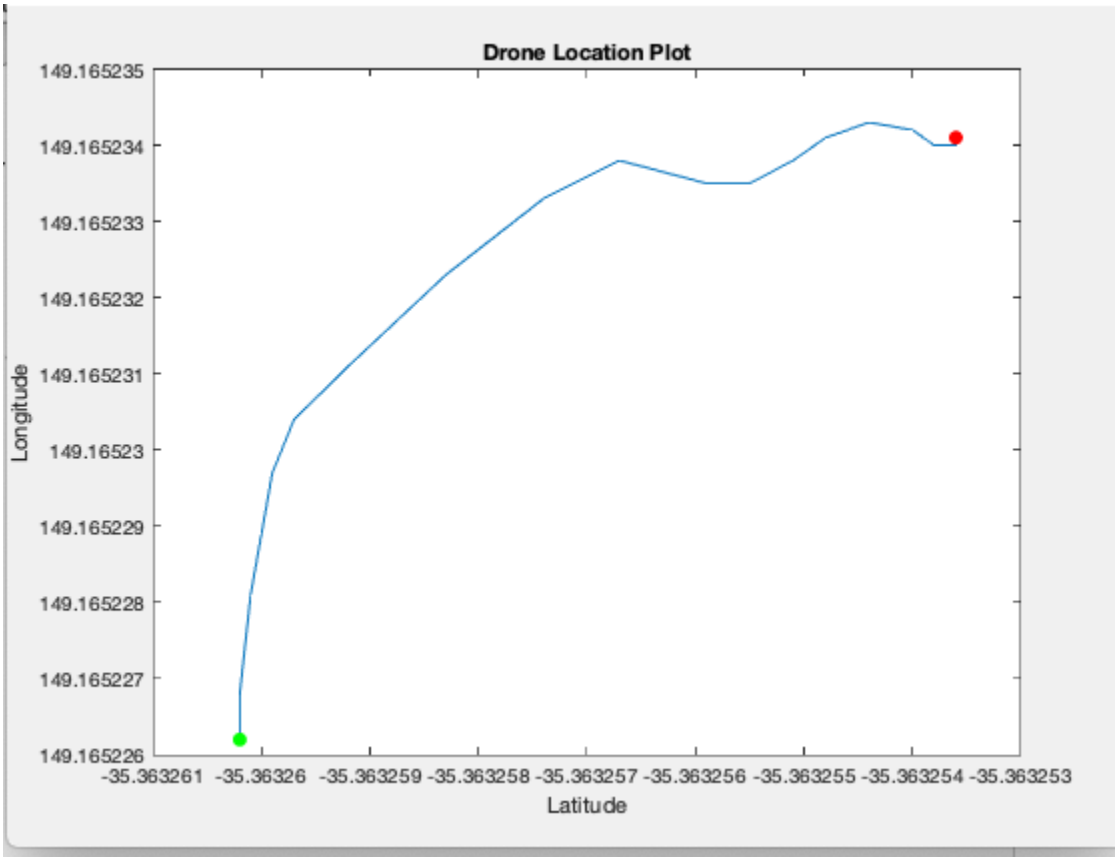


Figure 3. Plot of the simulated travel of drone commanded with hypothetical velocities

Computer Vision

For the computer vision aspect of our design approach, we were able to implement numerous CV libraries that make it simple to determine camera location and orientation. For this project, we used both OpenCV in python and the MATLAB Computer Vision Toolbox. To simulate drone movement, we captured video (without a drone) while approaching a target. These videos were then used in our computer vision software stack to detect location and movement. From our desired pose (directly in front of the target) and current pose (calculated using CV scripts), the autonomy scripts can determine the appropriate velocity to set in order to approach the target. These velocities are sent to the drone simulation scripts to “fly” the drone in real time.

Motion blur was a major obstacle when capturing simulated video for the CV scripts; the blurred images add uncertainty to the localization and the ‘drone’ loses its ability to determine its location. Because we were capturing simulated video at 60 FPS, it is likely that the real drone will also struggle with this issue because the Pi Cam also operates at 60 FPS. We suggest future teams use either a higher quality traditional camera or a depth camera.

Simulation Overview

1. Capture Simulated Video
2. Detect checkerboard of variable size in each video frame
    a. In simulation this step is done at 60Hz but only needs to operate at max. frequency of 10Hz when implemented on the drone because flight controller cannot respond faster than 10Hz
3. Use MATLAB/Python to extract location/orientation data
    a. Plot the calculated location for accuracy measures of drone simulator
4. From any given location, calculate and record velocity necessary to bring drone to target
5. Send real-time calculated velocities to drone simulator
6. Compare dronekit path to MATLAB path to ensure accuracy

Proposed Flight Order of Operations Overview

1. Pi camera feeds image data to Jetson Nano
2. Jetson Nano runs checkerboard detection algorithm to localize and orientate
    a. If drone is within goal distance, the drone yaws to the where is expects the next target to be
    b. If drone is without goal distance, the Jetson Nano sends flight velocities to the drone that approach target
3. Pixhawk flight controller receives the flight commands from the Jetson Nano through the JST interface
4. Pixhawk issues desired propeller speeds to the four ESCs in order to guide the drone's flight path

As was stated in the goals section above, we were unable to fully implement the target detection algorithm on the drone due to the COVID-19 situation and group member separation. However, we successfully demonstrated the individual software components that located targets and issued flight commands to the flight controller. Although we did not get the opportunity to combine all of the software on the drone, we are confident that the cohesive software suite would

perform as we expected. The one design aspect that we definitely cannot resolve is the speed of the drone. Through multiple tests with our computer vision algorithm, we realized that we were limited by the speed of our camera. At high drone velocities, the images would experience excessive motion blur that proved impossible for object detection. Nonetheless, our software would still detect objects at moderate speeds. Future design teams could benefit from a better camera module and perhaps, given adequate time to work, a more efficient object detection algorithm.

4.2 Codes and Standards

FAA regulations will need to be followed during testing. Since the drone will weigh less than 55lbs we can operate as recreational flyers as long as we abide by the FAA guidelines [4]. These guidelines should be easy to follow with our design process.

- Fly below 400 feet and outside of controlled airspace
- Never fly over crowds or vehicles
- Be aware of temporary flight restrictions (TFR)

The MAVLink Protocol forms the underlying communication medium for the drone and its components [5]. Commands are sent as messages, and the library gives us the functionality we need to perform flight operations.

- Issues commands to the flight controller using the laptop ground station.
- Allows for us to remotely communicate with the flight controller through the Jetson Nano

4.3 Constraints, Alternatives, and Tradeoffs

One major constraint that has affected us in the spring 2020 semester was COVID-19. We won't go into all the trials and tribulations that we faced in this section as we address the challenges in detail in multiple sections of our proposal. However, in short, we were not able to work together on the hardware to get everything working together. Instead, we developed several pieces of software that have the potential to work together on the drone but because of our remote-ness, we are not able to perform the system integration.

Our general hardware constraints and considerations are outlined in the rest of the section as follows:

An early hardware decision we had to make was going with the Jetson Nano by NVIDIA over the Raspberry Pi, both of which are small computers that are responsible for running cameras and sensors. The Jetson Nano allows for faster computations to be performed, while the Raspberry Pi falls short. For example, the Raspberry Pi 3 only offers 6.2 gigaflops while the Jetson provides us with 472 gigaflops. Gigaflops are a measure of the performance of the computer's floating-point unit. If we consider price to performance, the Raspberry Pi is a better budget choice, but since we needed to avoid any bottlenecks with processing ability, we opted for the more expensive Jetson Nano and are now happy with our more performance oriented decision.

Another important tradeoff that we were eventually forced to make was the loss of speed in favor of accuracy. On one hand, traveling as fast as possible through a course would in theory result in crossing the finish line first; however, this gives less time to perform calculations and avoid incoming obstacles. We originally pursued high speed flight, but quickly realized that we were unable to accurately locate obstacles at higher speeds. Faster flight resulted in increased motion blur in the image captures, and we were simply unable to detect objects at those velocities. We had to face our hardware and time limitations, and consequently, we prioritized accuracy over speed. The main goal of our drone is to autonomously avoid objects, and speed was the constraining variable in achieving that.

We also considered purchasing a store-bought drone to use for our project; however, the main concern was that with the added weight of the Jetson, the store-bought drone would not have enough power to maintain optimal flying capability. Looking back, although it was initially hard to understand the software interface of our custom drone kit hardware, once we got comfortable with the software, we were able to access greater functionality. A store-bought drone would have been more straightforward out of the box, but with reduced capabilities and customization.

Additional tradeoffs

i. Including more cameras for improved input and spatial awareness will not only add weight, but also increase the computational power and bandwidth required to process extra data

ii. We could choose to have two input processing loops one fast and one slow
    i. fast loop would be used for object avoidance
    ii. slow loop would be used to develop trajectory to next hoop
iii. If one loop is preferred, all input information will flow fast, but only differences will be processed to reduce computational power required

Design Factors

- Global: Global factors played a role in the design of the autonomous racing drone as it affects the current market. Since we purchased a drone from the available market, global factors shaped and limited our purchasing options.

- Cultural: There were no cultural factors that affected our design. If anything, our drone is intended for any user regardless of cultural background.

- Economic: Economic factors definitely played a role in the design factor for the autonomous drone. We only had a limited budget to buy the drone and its parts, so we had to make wise choices in terms of both components and the sources where we purchase those components. The drone is also meant to be somewhat affordable, so we had to make realistic part choices.

- Environmental/Sustainability: Environmental factors did not really play a role in the design aspect of the autonomous racing drone. However, the drone's batteries had to be recharged for flights. Solar rechargeable batteries would be a better environmental solution.

- Manufacturability: Since our parts were sourced from a drone kit and two reputable companies, NVIDIA and Raspberry Pi foundation, the drone itself is rather easy to manufacture. The software components could also be distributed and deployed as necessary.

- Ethical: There were no ethical issues that were taken into consideration as we were designing our drone.

- Health and Safety: We have to abide by a set of code and regulations that have been placed on flying drones for the safety of the general public. For example, we can't fly the drone near an airport or military installations.

- Social: With an autonomous racing drone, there aren't social issues when designing the drone. As long as the drone is being raced in a closed off area away from the public, there are no issues.

- Political: Political factors do not affect our design factors when deciding on our drone and the other hardware components we are going to use.

5. Schedule, Tasks, and Milestones

The AutoQuads team has decided to break the project into five building blocks: Hardware, Computer Vision, Autonomy, Budget/Procurement, and Final Testing. In Appendix A, the provided GANTT chart illustrates the proposed tasks, owners, and deadlines. In our project proposal, we outlined several project phases in which we complete hardware, computer vision, data pipelining, path planning and final testing. We were able to build the drone and have the maiden test flight before we switched to remote learning. After we switched, the remaining tasks included computer vision, data pipelining and path planning. Because we were separated, there was no need to data pipeline and we just focused on computer vision and simulation.

6. Final Project Demonstration

Our project demonstration will consist of three parts, as there are three separate components to the drone's functionality. Unfortunately, due to the pandemic, we were unable to combine these components into one cohesive demonstrable product. However, it is possible (and easy) to integrate the components when the opportunity arises.

Flight Autonomy

Our flight autonomy was two-fold. The first approach (before we were disbanded) was to use a software called QGroundControl to add robustness to the drone. This software allows us, the drone architects, to put safeguards on the drone when using our custom autonomous algorithms. For instance, we can launch and land autonomously at the flip of a switch in case things go awry. We can also set the drone to hover in a loiter mode while the computer vision and path planning compute their tasks. Ensuring safety was the first step in creating drone autonomy. It is difficult to demonstrate the safety of a flight system; so, **Figure 4** below is an image of the drone's instability when the drone safety/autonomy is not engaged.

Figure 4. Unstable drone about to tip over

Figure 5. Drone in stable flight

Drone Simulation

Once we switched to remote learning, we needed a way to test our algorithms without actually touching the hardware. This led us to implementing software packages found in the dronekit and dronekit-sitl python libraries. These libraries allowed us to initialize, launch, and command a virtual drone as if it were real. Dronekit provides the architecture for sending flight velocities and commands to the flight controller software (ArduPilot). Dronekit-sitl offers a 'software-in-the-loop' solution that simulates the drone and works in tandem with dronekit to provide an alternate commanding architecture. Using dronekit, we can arm and command the simulated drone provided by dronekit-sitl. **Figure 6** below shows a screenshot of the terminal present when running dronekit. Note that dronekit does not offer a visual simulation of the drone, but rather we can collect all the relevant telemetry and visualize in post.

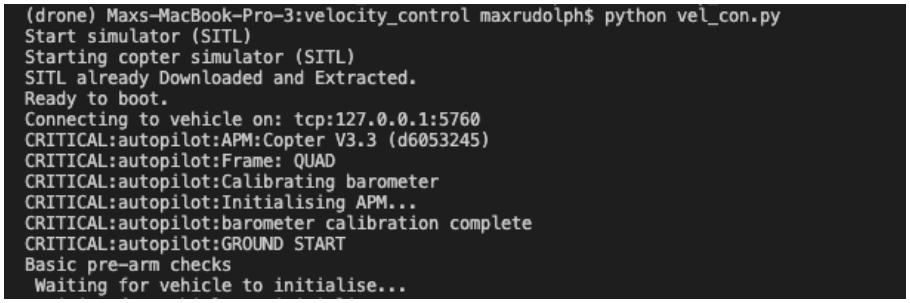


Figure 6. Screenshot of terminal output when drone is simulating

Computer Vision

To demonstrate the abilities of our computer vision software, we decided it would best to simulate a drone's flight path and present the computer vision that the drone would hypothetically complete. Then, we could use the computed theoretical velocities in the simulated drone environment provided by dronekit. To demonstrate our capabilities, we first show an image of our target and how the drone could localize itself. In **Figure 7** below, the image on the left was captured then we applied our CV scripts to calculate the orientation and location that the drone would have to be in to capture that same image (the red camera on the right is the 'drone').

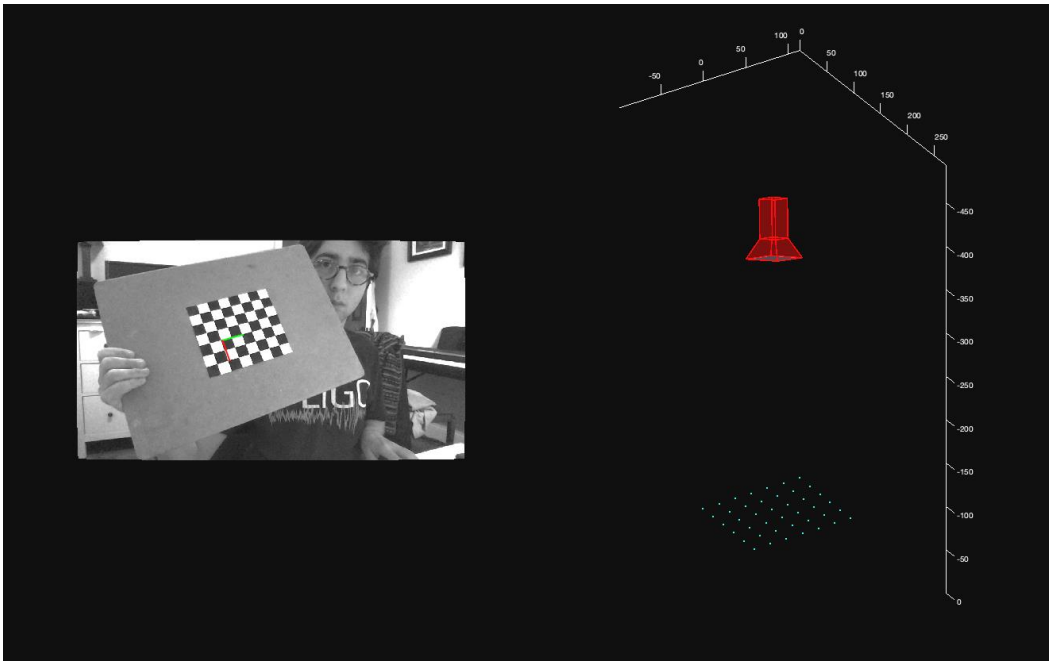


Figure 7. Computer vision localization of checkerboard with orientation/location visualization

Next, we can simulate drone movement by taking a moving video of our target and continuously apply the CV algorithms to compute our approximate location at any given point in the video. **Figure 8** on the next page shows the drone's hypothetical path. Each magenta point represents a timestamp at which the drone calculated a velocity to reach its target (the red dot on the bottom). These velocities are passed into the drone simulator for validation of path planning.

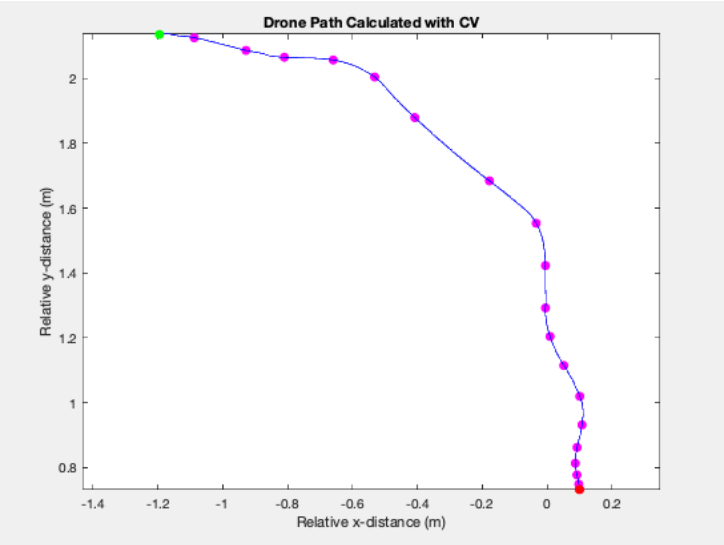


Figure 8. Plot showing path traveled as computed by CV scripts

At each time step, we simulate drone movement in dronekit. The validated path that the drone flew is plotted in **Figure 9**. Because the simulation assumes ideal conditions, when a velocity is commanded, it is reached immediately and there are no environmental factors affecting the flight path of the drone. As such, the path looks relatively clean and straight. Furthermore, within the simulation, the drone tracks its own location using global coordinates but that are *not* used in our algorithms. Clearly, this software stack can be used on the actual drone when we return to campus. By simply comparing the shape of the two curves presented, we can see that our algorithms are successful. The remaining tasks include testing and integration.

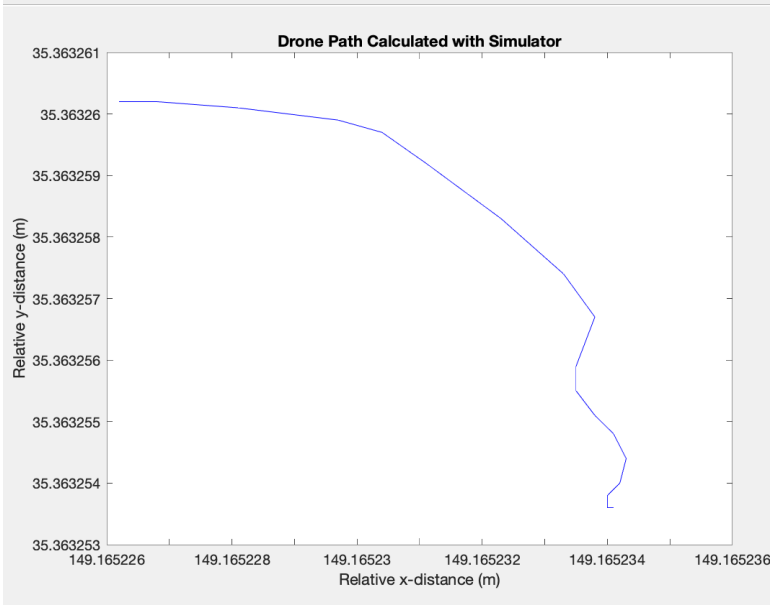


Figure 9. Drone path as computed by dronekit simulation after using calculated velocities

Product Specifications

As has already been said, our product specifications changed a great deal since the inception of the project. Our goals pivoted for two main reasons; 1) we had to change because of COVID-19 and 2) our initial project scope was possibly too large. So, our updated product specs are as follows: create a drone that could autonomously navigate to a target in *real time*. The most important spec we outlined was that the algorithms must run in real time. This is because, to provide any hope at completing the original project, we need real time localization and tracking. Additionally, it was important to us that our algorithms were easily applied to a real drone; this led to our specifications requiring accurate drone simulations that are true to real drone commanding architecture.

Demonstrated Capabilities - Brief

- 3D path reconstruction using computer vision
- Real-time Drone localization using checkerboard
- Velocity calculation/path planning using computer vision
- Arbitrarily accurate drone simulation used for further testing and validation
- Drone autonomy using ArduPilot and dronekit

7. Marketing and Cost Analysis

7.1 Marketing Analysis

The target market would mostly be made up of people who are interested in the use of racing drones, which in the future could lead to a wide array of uses but for now would mostly be utilized for entertainment purposes. There are other autonomous drones currently on the market, such as the Skydio 2, which is currently being sold for $999, but is not a racing drone like the drone for this senior design project. [1]

7.2 Cost Analysis

The total development cost in order to make a prototype autonomous racing drone would be about $1157.00. The majority of this cost comes from the price of the actual drone. Since a lot of the invention is dependent on the software and algorithms, there are not a lot of materials that are necessary in order to create this product. The price breakdown is represented by the following table.

Table 4. Equipment Costs

Product Description	Quantity	Unit Price ($)	Total Price ($)
Drone	2	280.00	560.00
NVIDIA Jetson	3	119.00	357.00
Camera	2	80.00	160.00
Batteries	4	20.00	80.00
Total Cost			1157.00

Assuming that we are paying laborers $45.00 per hour, development costs were able to be determined and are shown in the following table. The highest number of labor hours are designated to coding for the drone. The determined development costs to design the drone summed out to be $20,957.00.

Table 5. Development Costs

Project Component	Labor Hours	Labor Cost ($)	Part Cost ($)	Total Component Costs ($)
Autonomous Drone				
Building	80	3600	1157	4199
Simulation/Testing	40	1800	-	1800
Racing & Collision Detection				
Coding Algorithms	100	4500	-	4500
Debugging	50	2250	-	2250
Testing	50	2250	-	2250
Demo Preparation	20	900	-	900
Meetings	150	6750	-	6750
Total Labor Costs	-	19800	-	-
Total Part Costs	-	-	1157	-
Total	-	-	-	20957

The total development costs for the autonomous racing drone is $50,590 which is shown in the table below. We utilized a 35% fringe benefit of the total labor and 130% overhead of materials and labor.

Table 6. Total Development Costs

Parts	$1157
Labor	$19,800
Fringe Benefits	$6,930
Subtotal	$26,830
Overhead	$23,760
Total	$51,248

In order to determine the selling price and profit per unit, it will be based on a production of 5,000 units over a period of 5 years at $999.00 per unit forming about a 52%

margin between cost per unit and selling price and matching the price of the other autonomous drone that is on the market. In bulk, the Jetson will be 16.67% off. Advertising is budgeting at 5 percent of the cost per unit. Therefore, the expected revenue from this product is estimated to be $995,000, which is a profit of $199 per unit. This information is summarized in the following table.

Table 7. Selling Price and Profit per Unit (5000 units over 5 years)

Parts Cost	$479
Assembly Labor	$20
Testing Labor	$20
Total Labor	$40
Fringe Benefits	$14
Overhead	$129
Advertising	$48
Average Development Costs	$50
Selling Price	$999
Profit	$199

8. Conclusion

8.1 Current Status

Due to COVID-19 and the switch to remote learning, we have been partially stifled in creating one cohesive drone. However, we were able to develop technologies separately that, when put together, would create a functioning autonomous drone. Firstly, the drone itself has been built, calibrated, tested, and flown. While it may seem trivial on the surface, the hardware of this project posed several challenges because we needed to pipeline the Nvidia Jetson Nano, Pi Cam, and the Pixhawk Flight Controller. We were able to send commands to the drone through the Jetson Nano and capture video from the Pi Cam. The drone is ready to be used by the next design team that takes on this project.

As for software, we successfully developed CV scripts that can localize and control the drone in simulation. Our software takes in images that include checkerboards (which are fixed to the goal posts) of known size and determines the location and orientation of the camera (which is fixed on the drone) with respect to the checkerboard. From these localizations, we can compute splines and velocities that would bring the drone to the checkerboard. These scripts function at 15 Hz in simulation, so they are ready to be used on the actual drone. The software we developed, while only tested in simulation, can be applied to the physical drone with little effort.

Lastly, we developed our software for use in the dronekit quadcopter simulator. In this simulator, we initialize a drone equivalent to our physical hardware and interact with it as if it were the actual drone. We can send velocity commands and receive all telemetry data that would be useful in testing the real drone. The next team can use this simulation as a good starting point to begin design on their drone's software.

8.2 Further Improvements

We have laid the groundwork for a team to begin working on an autonomous drone at the beginning of the next senior design cycle. This section will discuss next steps and potential future works that a senior design team should accomplish to further the progress of this project. The first necessary task to accomplish would be to synthesize all the disparate technologies that we have developed in order to make one autonomous drone (pipeline Pixhawk communication with python CV scripts, Pi Cam communications, etc…) In other words, the data pipelining needs to be done because we have been prevented from doing so by the move to remote learning. The next crucial improvement would be to use OpenCV to detect goal posts instead of relying on fiduciary checkboards for localization. Currently, we use a printed-out checkerboard to denote the location of a goal post; this is not generalizable. The next necessary improvement would be to integrate a camera with a higher frame rate (or use a depth camera). When the drone is moving at high speeds, the current camera (Pi Cam) captures blurry images that cannot be used for localization. Finally, it would be prudent of the next team to be split into several senior design teams. We found it difficult to tackle the path planning, hardware integration, computer vision, and data pipelining all at the same time. Perhaps if one team worked with a motion capture system to develop path planning algorithms, another team worked with the Jetson Nano and Pi Cam to develop robust CV scripts, and so on, the progress of the entire project would be greatly accelerated.

9. Leadership Roles

Max, Software Lead. In charge of all software and algorithms that are implemented on the drone and compute unit.

Rishov, Webmaster. In charge of creating a website to showcase the project, as well as handling the videography and mapping of the drone's path.

Eddie, Hardware Lead. In charge of the physical design of the drone, as well as deciding all additional components to be added for functionality.

Nye, Meeting Timekeeper. In charge of ensuring that the team stays on topic during team meetings.

Suhani, Team Coordinator. In charge of managing meetings, deadlines, and the scope of the project.

Dave, Design Lead. In charge of all schematics of the drone and planning the integration of all components.

Michael, Research Expert. In charge of researching similar projects and determining what components are needed.

10. References

[1] Skydio, “Pricing Page-Skydio” *Skydio, 2019.* [Online]. Available: https://www.skydio.com. [Accessed: Nov. 17, 2019].

[2] “AIRR,” *The Drone Racing League*. [Online]. Available: https://thedroneracingleague.com/airr/. [Accessed: 29-Apr-2020].

[3] Jetson Nano Specifications, “Nvidia Developer site” *Nvidia, 2019.* [Online]. Available: **https://developer.nvidia.com/embedded/jetson-nano-developer-kit**[Accessed: Nov. 18, 2019].

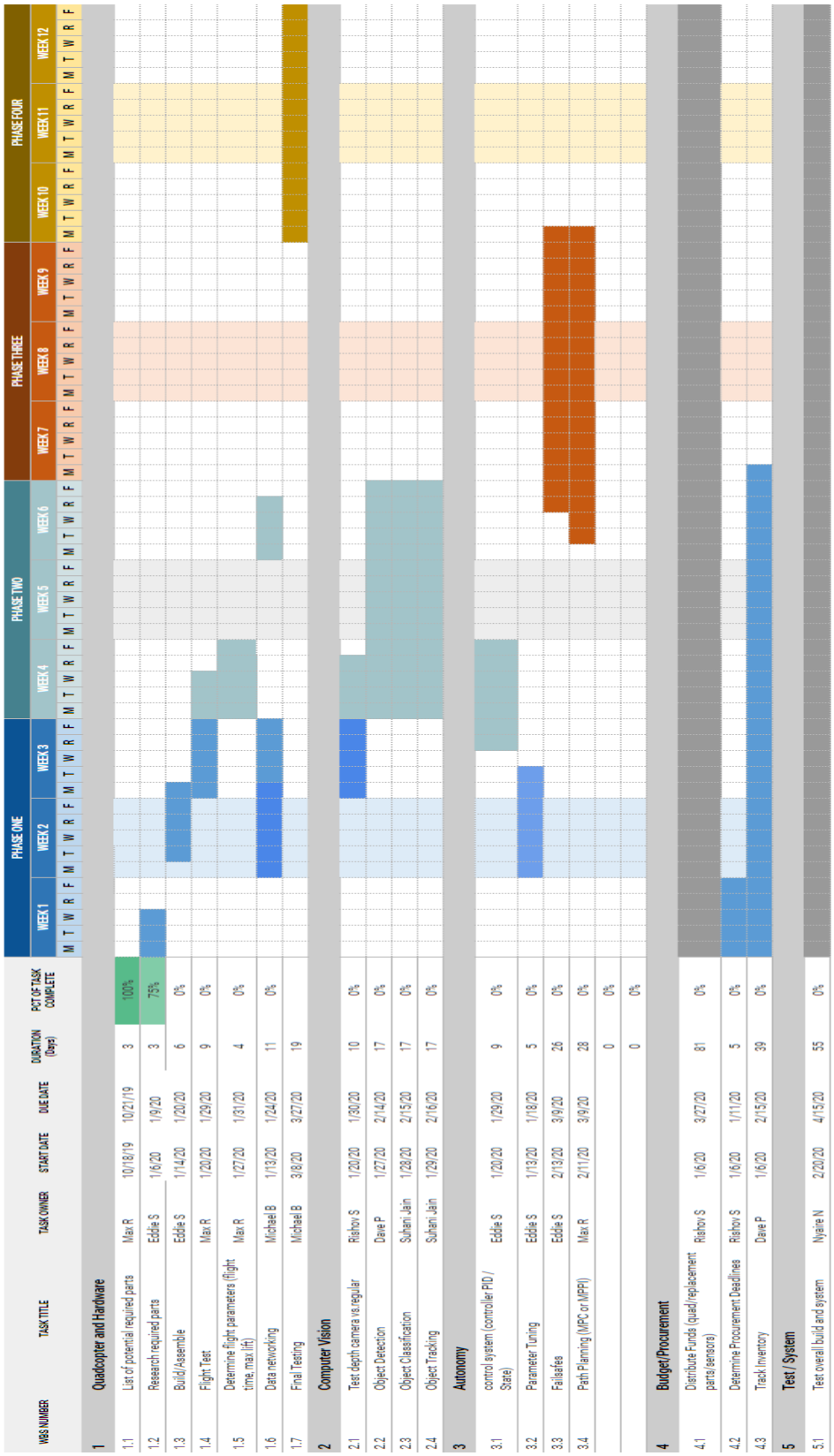
[4] L. Dorr, “Small Unmanned Aircraft Regulations,” *Small Unmanned Aircraft Regulations*. FAA, 2018. [Online]. Available: **https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=22615**

[5] Mavlink.io. 2020. *Introduction · Mavlink Developer Guide*. [Online] Available at: <https://mavlink.io/en/> [Accessed: 28 April 2020].

[6] “Drones Reporting for Work at 45 MPH,” *Draper*, 03-Aug-2017. [Online]. Available: https://www.draper.com/news-releases/drones-reporting-work-45-mph. [Accessed: 17-Sep-2019].

[7] Dronekit. (2019), DroneKit. [Online]. Available: https://github.com/dronekit/dronekit-python

Appendix A GANTT



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (Days)	PCT OF TASK COMPLETE
1	**Quadcopter and Hardware**					
1.1	List of potential required parts	Max R	10/18/19	10/21/19	3	100%
1.2	Research required parts	Eddie S	1/6/20	1/9/20	3	75%
1.3	Build/Assemble	Eddie S	1/14/20	1/20/20	6	0%
1.4	Flight Test	Max R	1/20/20	1/29/20	9	0%
1.5	Determine flight parameters (flight time, max lift)	Max R	1/27/20	1/31/20	4	0%
1.6	Data networking	Michael B	1/13/20	1/24/20	11	0%
1.7	Final Testing	Michael B	3/8/20	3/27/20	19	
2	**Computer Vision**					
2.1	Test depth camera vs regular	Rishov S	1/20/20	1/30/20	10	0%
2.2	Object Detection	Dave P	1/27/20	2/14/20	17	0%
2.3	Object Classification	Suhani Jain	1/28/20	2/15/20	17	0%
2.4	Object Tracking	Suhani Jain	1/29/20	2/16/20	17	0%
3	**Autonomy**					
3.1	control system (controller PID / State)	Eddie S	1/20/20	1/29/20	9	0%
3.2	Parameter Tuning	Eddie S	1/13/20	1/18/20	5	0%
3.3	Failsafes	Eddie S	2/13/20	3/9/20	26	0%
3.4	Path Planning (MPC or MPPI)	Max R	2/11/20	3/9/20	28	0%
					0	0%
					0	0%
4	**Budget/Procurement**					
4.1	Distribute Funds (quad/replacement parts/sensors)	Rishov S	1/6/20	3/27/20	81	0%
4.2	Determine Procurement Deadlines	Rishov S	1/6/20	1/11/20	5	0%
4.3	Track Inventory	Dave P	1/6/20	2/15/20	39	0%
5	**Test / System**					
5.1	Test overall build and system	Nyaire N	2/20/20	4/15/20	55	0%