CyDrone

Design Document

Team sdmay20-47

Client: Ali Jannesari
Advisors: Ali Jannesari

Team Leader: Kenneth Lange,
Chief Hardware Developer: Alain Njipwo,
Chief Software Developer: Daniil Olshanskyi,
Chief Interface Developer: Luke Bell,
Chief Backend Developer: Max Medberry

E-Mail: sdmay20-47@iastate.edu
Website: https://sdmay20-47.sd.ece.iastate.edu

Revised: 12/08/2019 - v3

Executive Summary

Development Standards & Practices Used

Hardware Standards:

- PX4 controller interface
- CAN

Software Practices/Standards:

- GIT
- REST API
- Websockets
- AirSim API

Summary of Requirements

- Functional
  - Set up a simulation environment to test the software.
  - Set up a website to see the drone/simulation camera stream
  - Implement object detection and compute the volumetric analysis.
  - Assemble the drone itself.
- Non-functional
  - Software will have an architecture that is easy to understand and navigate.
  - Code will be well documented, both via comments and a wiki explaining to future users what is being accessed by what and how.
  - Code will be modular, so it is easy to fix, extend, and/or replace.
  - Hardware will be documented and taken care of
- Technical and/or other constraints
  - Learning how to build the drone and calibrate it
  - Learn new software platforms: Unreal Engine and AirSim

Applicable Courses from Iowa State University Curriculum

- ComSci 252 - Linux
- ComSci 352 - Linux/C programming/Multithreading
- CprE 308 - Linux
- ComSci 309 - Software design practices

- ComSci 319 - Software design practices
- ComSci 327 - C programming
- EE 201 - Circuit testing

New Skills/Knowledge Acquired Outside of Coursework

- Unreal Engine - world-creation and simulation
- AirSim - simulation
- ArduPilot - calibration of drone
- NVIDIA Jetson - mobile computing platform
- PixHawk - drone control and communication
- Python
- TensorFlow
- Volumetric Estimation
- CMake

Table of Contents

Tables and Figures

1 Introduction

1.1 Acknowledgement

Huge thanks to Dr. Ali Jannesari for mentoring us on this project.

1.2 Problem and Project Statement

Nowadays, all aspects of human lives are moving towards automation. This has been true for as long as humanity has existed, but with the development of computers and growth of machine learning, more and more operations can be efficiently done by computers. Since we, humans, are receiving a big portion of information via our eyes, it's logical for us to try and teach our machines in a similar way; to see and process what they see, making decisions based on that.

This is why we decided to work with this problem. Our goal is to make a platform with high mobility (a drone), load it with exceptional computational powers (nvidia SoM from the Jetson family), and make it process a camera input with machine learning algorithms. As a starting point in our machine learning we want to teach the drone to: identify objects, follow or avoid them, and analyze them (including volumetric analysis). Further development will be done on creating more complex algorithms based on computer vision and machine learning, like adding the possibility of a master-slave system, where a fleet of drones are controlled by a single leader.

We are hoping to build a drone-based computational system capable of solving complex tasks via machine-learning and computer-vision-based algorithms. Object detection and tracking should serve as examples of how our system can handle complex machine-vision and machine-learning-based operations while our architecture and implemented control functionalities will allow us to use and expand the solution with ease.

1.3 Operational Environment

The operational environment on the software side is simple: the front end will be launched from the browser, so any browser supporting JavaScript, html5 and webgl will be sufficient. The back end will be launched on the drone itself, so it will represent the server, the computer-vision analyzer and the drone controller at the same time. On the other hand, our hardware - the drone itself - should be able to withstand any environmental influences that a drone can encounter (wind, unexpected collision, etc). Obviously, the balancing algorithms should be present to stabilize the machine mid-air and negate the windflow. In general, we don't plan to use the drone during the rain, but we'll have water protection to protect the drone from occasional moisture and small splashes. The protection should also cover the vulnerable electronics from small debris such as leaves that a drone can encounter in the air at normal conditions.

1.4 Requirements

Functional requirements:

- Incorporate information about old iterations of the project received from the client.
- Set up a simulation environment to test the software.
- Set up a website to see the drone/simulation camera stream
- Implement object detection and compute the volumetric analysis.

- The hardware will be Master-Slave oriented so that new hardware can be added and removed as desired in the future.
- Assemble the drone itself.

Non-functional requirements:

- All software should store logs of the past important information and crash reports.
- Software will have an architecture that is easy to understand and navigate.
- Code will be well documented, both via comments and a wiki explaining to future users what is being accessed by what and how.
- Code will be modular, so it is easy to fix, extend, and/or replace.
- Open source software will be used to make the process cheaper and more maintainable using the help of an open source community.

1.5 Intended Users and Uses

We intend the product to be used by big companies as well as small enthusiasts to rely tasks on the drones. The system should allow anyone to use the object detection and volumetric analysis functionality with the help of our platform. We also see other developers as our user, so it is sufficient that we make our program modular and expandable. Third-party developers will be able to use our system to perform their needed tasks as well as an opportunity to extend our hardware and software solutions to fit their needs. We shall provide basic instruments as well as examples of useful algorithms so the system has a good performance as well as a high level of flexibility.

1.6 Assumptions and Limitations

Assumptions:

- Client/end user is familiar with operating a drone
- The drone will be utilized for object volumetric calculations.
- The drone is used where it can be wirelessly connected to the network

Limitations:

- The hardware might struggle to accurately record parameters for thinner objects.
- Input Noise might impact some of recorded measurements.
- Limited to use ROS or AirSim on the drone.

We are assuming that the client/end user will be familiar with the basic operations of a drone. We are also assuming that the drone will be operated as intended. To calculate the volume of any solid integral objects. We assume that the drone will be used in weather conditions that will not compromise the hardware's ability to accurately predict the size of objects.

The volume calculations will be limited to the hardware's capability to translate analog values to digital values. In an extreme case like a leaf where the length and width can be easily calculated the hardware will struggle to detect the dept. Cases like these should be considered and a possibility of a greater margin of error. The decrease in accuracy should be accounted for when the product is utilized.

1.7 Expected End Product and Deliverables

At the end of this project, our team should deliver a fully assembled drone that has unmanned automated vehicle (UAV) capabilities. Attached to the drone will be other hardware components that are needed to enable to drone to perform the required project function. There will be a GPS module to enable the drone to be tracked and relay it's position in real time. A camera to capture the data from the objects of interests. A GPU to directly process some of the data onboard.

There will be communication will the backend of the project. Hosted on a server, will be Artificial Intelligence portion of the software that will be able to communicate with the drone in real time during its flight. Also hosted remotely will be the algorithms necessary to calculate the volume of each object encountered.

For visuals, we will boundary test cases of objects that have been scanned by the drone as well as the obtained volume.We will manually do these same volumetric calculation. From both obtained results, we will do a percent error calculation to give an estimate of the expected accuracy of the drone.

We will include a video demonstrating the drone starting, taking flight and calculating the volume of a test object.

2. Specifications and Analysis

2.1 Proposed Design

This project was given to us by the client with an already-established approach to solving the problem. A previous senior design team was working on the same project, and they ran into a lot of issues out of their control that rendered them unable to fully complete the project. Having learned from those issues, our client did some research before giving us the project for how we should best go about implementing the solution to their problem. They identified multiple technologies for us to use: ROS or AirSim as an industry-standard operating system for controlling robotic systems like drones, Gazebo or Unreal Engine as a suitable simulation environment for flying the drone, and Hector Quadrotor as a useful interface for sending the drone commands through the CyDrone web interface. Using these technologies will allow us to remotely operate the drone, both in simulations and the real world, and run machine learning algorithms on the web server that will control the simulated drone.

2.2 Design Analysis

So far, our work has been focused on: construction and calibration of the drone, setup of an AirSim simulation environment, and operation of the CyDrone website. While doing all of this, we've been attempting to work with the results of a previous group's senior design project. That team had a lot of issues getting the different technologies to work together reliably, so after looking through their work and attempting to use it, we decided it would be more efficient if we reworked some aspects of their project, using their results to identify how we could implement the different components most effectively. So far, we've been able to do this successfully, making progress in areas where the other team was stuck. We have decided to use AirSim as our simulated environment because it has given us a lot more documentation than Gazebo, the environment that the other team used. Our next steps are to establish communications between the CyDrone website and the simulation environment and get the drone flying using commands sent through our backend.

One of the strengths of our design is that we are taking a simple and modular approach. We are keeping in mind that there will likely be another senior design team continuing to work on this project after us, so we're trying to make our work easily digestible and modular to accommodate for that. Another strength is that we're using open-source software, so we'll have technical support coming directly from other members of the open-source community, plus additional support from Microsoft if we have any questions about AirSim.

A potential weakness of our design is that the training of the neural network controlling the drone is done in simulation. Due to this fact, it will be a while before we'll be able to see if it functions as expected on the real drone. We will be doing test flights with the drone itself, but we do not yet know if the calibration in the simulated environment will function the same as the real-world environment.

2.3 Development Process

Our team is following a one-week Agile development process. We have a weekly meeting with our client where we present the work we've completed over the previous week and identify what tasks we will work on in the coming week. This process works well for our team and our client, as we meet frequently enough to continuously identify tasks to work on and are able to make solid progress on our project, but not so frequently that we can't find time in our schedules to meet or work on the project.

2.4 Design Plan

We want to create a modular, reusable system for current and future development of our volumetric analysis drone project. To do this, we will create a model system using ROS or AirSim to simulate an environment for the autonomous drone with an image processing ability. This will be used to prevent crashes and allow for algorithms, specifically collision-avoidance machine learning algorithms, to learn from the flight patterns. On our real-world drone, we would like the user to be able to see from the drone's view. We plan on setting up a web-based interface that the user can see these images from. One of our core technical goals is to allow the drone to have an object detection functionality. This is what the volumetric analysis will be built upon. We plan on streaming the camera footage to our algorithm so that it can calculate the volume of an object while mid-flight.

3. Statement of Work

3.1 Previous Work And Literature

From what we know there is currently no product on the open market that is similar to what we are doing. Yes, indeed machine learning and computer vision are applied to drone technologies in the field, but none of the projects uses the outcomes to control drones. As it is widely known, Google, Tesla, and other smaller competitors are using computer vision for their auto piloted cars, but no widespread product uses machine learning in addition to computer vision to control and analyze information from drones.

We have had previous work on this project. Last year, another senior design group was working on this. We got our hands on their work and tried to extend it. Unfortunately for us, it became clear only after a month of processing that the work and results achieved by the last year team are poorly done, badly documented, and even hard to launch at all (no one was able to run their work to at least prove it is working). We have had a meeting with a person from that team in hope he could clarify some of our questions, when in fact he wasn't able to explain anything to us. This led us to a decision that we will get much more profit if we completely discard their work and start over from scratch, what we did. In addition to that, we overthought the technology stack that included ROS and Gazebo to understand if it is feasible to replace it with Unreal Engine and AirSim from Microsoft.

3.2 Technology Considerations

Our most valuable technological advantage on the hardware side is the Nvidia Jetson portable computer, that has a powerful GPU able to handle machine learning and small enough to be mounted on a drone. The main trade off is that there is nearly no information of how to control a drone from such platform (while there is an excessive info about how to do this with RaspberryPi). We are going to write our own software controller to be able to use this board to control the drone. There was an alternative to just use RaspberryPie, but that SoC is too weak for our needs - it has no GPU at all, and small amount of RAM with a relatively weak CPU makes it impossible for RaspberryPie to process the incoming information with a required speed.

On the software side, we had to come up with a simulation environment that we will use to train the neural network and test our solutions while avoiding risks of using the actual drone before we are certain. The most widespread solution is using ROS as a control system and Gazebo as a simulation environment. That was the approach the last year team tried. We also gave it a try. The problem is that ROS is poorly documented, not made for our task, and has excessive capabilities that we won't ever use. On the other hand, ROS provides good modularity and can be run on top of Linux or even in a container to provide more portability which is also good - since we will be using it on lab computers first and only after the neural network is ready will transfer it to the Nvidia board.

When we encountered trade-off of ROS we considered another option - the use of AirSim - an open-source set of drone simulation utilities developed by Microsoft. AirSim simulates the drone using the UnrealEngine 4 environment. The solution is modular - AirSim provides a server that represents the drone in the simulation in the UnrealEngine, and a set of API to be used by the client that don't even have to be on the same machine and can be used in any Python or C++ program by simply using a provided library. The main trade-off of the AirSim is that it requires UnrealEngine to be running and simulates the drone in it which is noticeably resource-costly. Also, the AirSim build for Linux(Ubuntu in our case) is relatively new and causes some troubles on launch, so we might have to use Windows for our needs in case any problems arise with the Linux build.

Originally we were going to use Gazebo as our simulation, this is the alternative direction we could have taken the simulator. The reason we decided not to because Gazebo, although built to work seamlessly with ROS, had little to no documentation or tutorials on how to do what we were looking to do. Additionally, there was a lot of nitty-gritty stuff that had to be just perfect to work in each environment. Our choice to go with AirSim was made because the original solution Gazebo took too many man hours to do simple stuff, whereas AirSim was well documented and had a much more active support group if ever we needed the help.

3.3 Task Decomposition

Since we are trying to make our system modular - we set our tasks mainly based on particular modules we need to get done:

- Set up the simulation environment
- Control the drone in the simulation via code/console commands
- Make a custom world in the simulation for basic drone training
- Design with the Reinforced Learning algorithm
- Implement the reinforcement learning algorithm into the drone simulation environment
- Apply the algorithm and train a neural network to complete our tasks

Next portion of tasks is somewhat separate from the previous one since they are more hardware-based:

- Assemble the drone
- Be able to control the drone via RC for assembly testing
- Mount the Nvidia Jetson onto the drone and get it running
- Make the Nvidia Jetson be able to control the drone

Third major step is connecting previous two together:

- To port a neural network and reinforcement learning algorithm to Jetson on the drone
- Make a controller module so the reinforcement learning algorithm can control the actual drone movements
- Launch and test how everything works together

3.4 Possible Risks And Risk Management

One of our main risks is limitations due to simulation environment. Since we have to use one made by third-party developers we don't have a lot of control over it and will mostly be limited with the functionality it gives us. Unfortunately, there is nothing we can really do to mitigate this risk, we will just try to avoid it by smartly using the functionality available.

The second major risk is our hardware. Drone is built custom and with this much computational power on it. This can potentially result in a lot of problems that will require additional expenses, time for parts to arrive, knowledge of electrical engineering etc.

A third risk is the reinforcement learning algorithm. There is a chance the whole neural network can get corrupted due to some error in the algorithm and some bad decisions made by it. This way we may lose the whole training done and will have to start over. We can mitigate this risk by doing backups of our neural network and restoring them in case of a corrupted version. Moreover, the best strategy would be to avoid this risk by thinking-through and additionally testing the reinforcement learning algorithm.

3.5 Project Proposed Milestones and Evaluation Criteria

We will have many milestones throughout this project. Due to the nature of the project our first milestone is to get the drone simulation up and running. This will give us the ability to simulate test environments, weather effects, drone stabilization and much more before trying the software on the actual device. This will prevent the majority of our issues as we can test the drone worrying about damaging the hardware. It also gives us the opportunity to use a machine learning approach to better adapt to unknown changes and prevent damage to the hardware that may happen in a real flight without accounting for it.

Our second milestone will be getting our object detection and volumetric analysis functioning using the built in depth sensing camera in our environment. This is unique as it also directly correlates to the camera that we will be using thus allowing us to achieve both milestones at the same time via one milestone. The depth sensing will be the primary decision base for the algorithm to decide what the drone should do mid-flight. As the drone's environment changes, it will be able to fly from one point to another without fail due to this milestone.

The final milestone will be for the drone to use the prior milestone to navigate to a detected object, determine it is the correct object and start following it accordingly as it moves around. This milestone will be tested mainly in simulation, but once all tests conclude its success, it will be tested on the real-life drone. This milestone should be close to our endresult as everything we need to fly the drone and prevent it from crashing into walls or choosing to follow the wrong object are solved by this point.

We will be mainly testing our milestones on the simulation as mentioned above. This is to prevent any easily avoidable hardware damage that we would have otherwise run into if all our tests were on a live drone. This also allows for us to test the milestones at a large scale. This could be done via many flights of a similar environment, to changing the randomness of the object determined to be followed, all the way to a complicated course for the drone to navigate with a tricky target to follow.

Additional tests will be on the hardware as we want to make sure that whatever simulated environment exists, we have the proper calibration for hardware. Once this is in place, the simulation should be close to if not exactly as functional as the actual drone in flight. Using all the machine learned data, the drone should then be able to fly around any given environment and follow any object that is determined by the user until otherwise suggested.

3.6 Project Tracking Procedures

Our group will track progress by identifying milestones to achieve and by maintaining a Trello board for our tasks. All of our work is motivated by needing to reach milestones throughout the project, such as: getting our drone flying, running machine learning algorithms in a simulated environment, and implementing the volumetric analysis algorithm on the drone. Once one of these milestones has been reached, we know we've progressed further towards our end goal. To track the progress towards each of these milestones more incrementally, we're using a Trello board with tasks that we create/update/complete weekly. When progress is made towards a milestone, it's indicated on the Trello board by an update to or completion of a task.

3.7 Expected Results and Validation

The desired outcome of this project is to have an autonomous drone that uses a camera to detect, remember, and follow objects using volumetric analysis. The drone will be able to navigate throughout a space, avoiding collisions with objects and surfaces. A camera feed from the drone as well as the location of detected objects and their volume will be shown on the CyDrone website.

To determine whether the drone is able to navigate autonomously, we'll test it in an environment where it needs to avoid a large number of obstacles, some of which will be moving while the drone is flying. If the drone crashes or doesn't navigate the space successfully, it won't have met the requirements. After the drone has collected volume information about the objects it detects in the space, we'll compare its results with the actual volume of the objects it detected. If the drone's results are within a forgivable margin of error of the actual volume, it will have met the requirement for detecting the volume of objects. The requirement of showing a live feed of the drone's camera and object information is simple to evaluate. We can view the website while the drone is flying and see whether the camera and object information is correctly shown. If so, the drone will have met that requirement

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Our project timeline can be seen below in Fig. 1. As it's still early in the project, the initial timeline we've created is rough. We need it simply to gain an idea of how much time we have compared to how much we need to get done. Later in the project we'll be able to better estimate the amount of time it will take to complete each task, at which point we'll update our timeline if necessary.

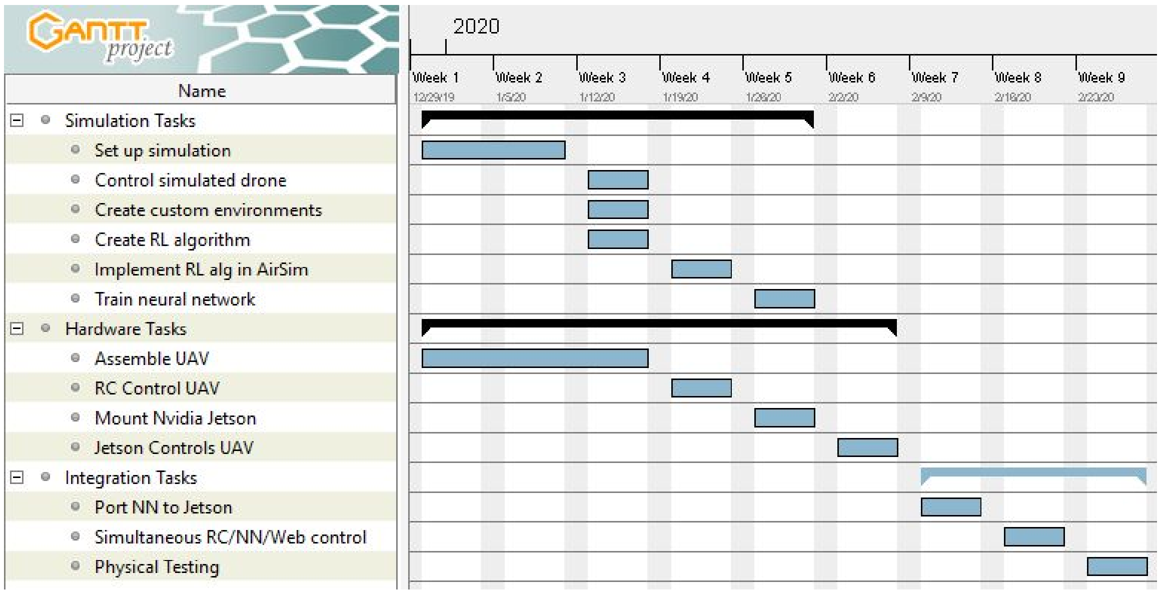

Fig. 1. Project Timeline

4.2 Feasibility Assessment

We should be able to complete the project in its entirety as mentioned above. We know that if we stick to our schedule and work at a steady pace, we'll be able to implement all of the features needed. Our project relies on a number of different technologies working well together, so the biggest challenge for our project will be getting everything to work together nicely. However, we are continuously evaluating what technologies we use to ensure that what we are using can be integrated most effectively. Another challenge we might face could come with the transition from flying the drone in simulation to flying it in real life. In theory it should fly as well in real life as in the simulation, but there's no guarantee. If it doesn't fly well in real life, we'll determine what's causing the discrepancy between the flight environments and modify our implementation accordingly.

4.3 Personnel Effort Requirements

Table 1 below shows a rough estimate of how many hours each team member will spend on each task. This information is by no means final, as we're attempting to have all team members gain experience in different areas, so it's likely that we'll switch up assigned tasks to give members exposure to different types of work.

Task	Luke B.	Daniil O.	Kennth L.	Max M.	Alain N.
Set up simulation environment	-	10	10	-	-
Add controlled drone to simulation	-	6	6	-	-
Create custom environments	-	10	10	-	-
Run RL algorithm in AirSim	6	-	-	6	-
Train neural network in simulation	6	-	-	6	-
Assemble drone	-	-	-	-	12
Control drone with RC remote	-	-	-	-	10
Mount NVIDIA Jetson on drone	2	-	-	2	4
Enable Jetson to control drone	4	-	-	4	-
Transfer neural network to Jetson	4	-	-	4	-
Enable simultaneous control from RC remote, neural network, and web	4	-	-	4	-
Functional testing	6	6	6	6	6
Total	**32**	**32**	**32**	**32**	**32**

Table 1. Individual work breakdown

4.4 Other Resource Requirements

From the hardware perspective, other non technical materials that are required include components such as Soldering Iron and wires used to create connections between components on the drone. Spare units of critical components such as the Pixhawk 5 Flight Controller, and the receiver for the RC controls are needed to be stored. This enables us to continuously work on the project without the interruption of waiting for new components to arrive.

From a software perspective, one of our core requirements is using Microsoft's open source AirSim to simulate the environment for drone. It also provides internal cameras with depth sensing that we can later use for our calculations for both detecting and following an object. Additionally, we have the requirements of sending the visuals to the webpage so a user can see through the "eyes" of the camera. This will allow the user to have a more immersive experience and see anything that they desire from a drone's view.

4.5 Financial Requirements

Our project is based around using as many open source applications as possible and reducing the amount of paid resources to a minimum. Due to this fact, we have no immediate costs foreseen in the software department. The only area that may have a cost is in the hardware area of our project. The drone itself is built from the ground up with parts that we assembled, purchased by our client. Therefore when flying our drone, under the rare circumstance that a piece goes bad or the drone crashes, there may be a later cost that we need to factor in for the client. Our client is well aware of this risk of crashing the drone and although we plan on doing most of our tests in a simulation, this is a possibility. This and hardware failure costs will be covered by the client and is not something that we will need to focus on in any way.

5. Testing and Implementation

5.1 Interface Specifications

At this point in the project we haven't yet started to fully implement our software/hardware connection yet since work needs to be done on both sides before putting them together as a complete drone-mounted machine-learning computer.

We are expecting a quite long control cycle for the drone with data coming in from the camera, going into neuralnet, being analyzed and turned into commands, commands passed to the px4 controller, controller turning them into actual control sequences for drone propellers. This includes one transfer point from hardware to software (input) and another transfer point back from software to hardware (output).

At this point all elements are tested autonomously. The camera input is tested via a helper program under different conditions and settings. The neural network and the reinforced learning algorithm are trained and tested in a virtual simulation environment based on the Unreal Engine 4.18. Drone controller is tested via giving it commands using the remote controller to test whether the hardware connections are made the right way and the drone can be controlled.

We are trying to do as much as possible before putting the elements together because a fault at any module can heavily impact our project by damaging the hardware component. More than that, after we finish testing modules autonomously we will put together and in safe conditions (with propellers of) test how well the modules communicate with each other to make sure there are no bugs and hardware problems along the whole stack of modules and their interfaces.

5.2 Hardware and software

The following hardware components are used when performing test-flights:

- 4x Tarot 4114 brushless motor (330 kV).
- 4x Tarot 1355 Carbon Propellers.
- 1 Radio Control Receiver
- 1 Radio Control Remote
- 4x 40A Electronic Speed Controller (ESC) board.
- Pixhawk control unit

We are using the following software for simulation:

- Unreal Engine
- AirSim

5.3 Functional Testing

The hardware components are tested for functionality every time the drone is powered. Most of these components are connected in series, meaning that performing a successful test flight tests these components as a system. If one component fails to perform properly the drone won't behave properly, which we'll be able to observe. These components include: motors, propellers, ESC boards, RC receiver, and the Pixhawk module.

Testing our software isn't as straightforward as it's been in most of our classes. Rather than having mock data that we can write unit tests with, we are building the brain of a drone. This being said the reinforcement learning algorithm controlling the drone learns using a weight based system that attempts to keep the drone at an ideal distance from the object it is following without crashing. Later in our project, when implementing the volumetric analysis to help determine which object to follow, we may have some of these tests. We are also using a well-tested system called TensorFlow which will be in charge of detecting and identifying the object prior to volumetric analysis, so there is no need to test that until a later stage.

5.4 Non-Functional Testing

We're not currently using any non-functional testing for our project because the nature of our project makes such testing difficult or unnecessary. We're effectively creating a prototype as a proof-of-concept, so we're much more concerned with functionality than aspects like security or compatibility testing. However, as we get further into the project and begin running our code on the real drone, we may run into performance issues that cause our code to not run in real-time. If that happens, we'll implement performance tests to make sure that our code will be able to run in real-time on the drone.

5.5 Process

To perform test flights, the carbon fiber propeller is mounted onto a motor. This motor is connected to an Electronic Speed Controller (ESC). As a unit, the ESC communicates with the RC receiver to ensure that the motors are executing the commands it receives from the . The Pixhawk acts as an arbiter for the system, ensuring that everything is working as it's supposed to. Fig. 2 below shows a picture of the drone, including the hardware used for testing it.

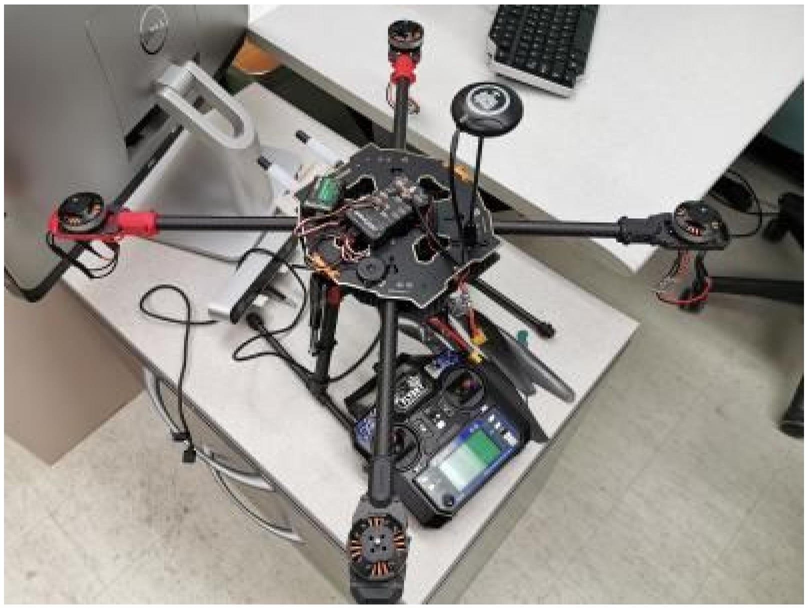

Fig. 2. Picture of drone hardware

The Unreal Engine is our simulation environment. It is very useful as it allows us to train the reinforcement learning algorithm in an environment that doesn't require active supervision or participation and doesn't risk damaging the drone. The drone model that we are using in the

Unreal Engine is Microsoft's AirSim. The Airsim model allows us to control the simulated drone in the same manner as we'd control the real drone, which makes our control code modular. Both pieces of software are invaluable because they are what the entire computer vision, object detection, and volumetric analysis is simulated on. Without the simulation we wouldn't be able to train or test the reinforcement learning algorithm before running it on the real drone. Fig. 3 below shows a snapshot of the drone navigating through the simulation environment.

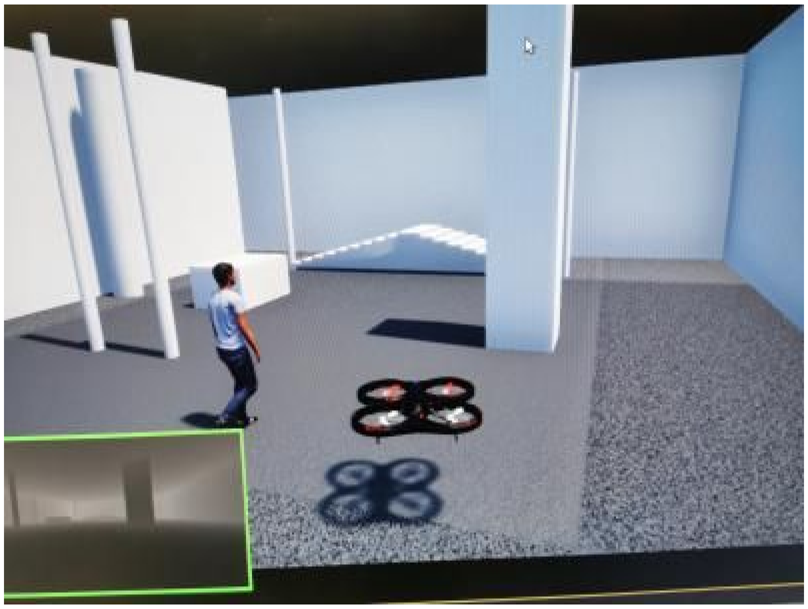

Fig. 3. Drone training in simulation environment

5.6 Results

To test the hardware, we've conducted a few test-flights. The tests have been successful for the most part. During one of the tests the drone crashed, but we believe that occurred because of our team's inexperience with flying drones and because the room being used for the test-flights is small. However, our tests have indicated that the drone is flying correctly, so the hardware is functioning properly.

For our software testing, we've begun running the reinforcement learning algorithm inside of our simulation. The algorithm allows the drone to get better at avoiding collisions, but it fails to improve the drone's ability to detect and follow an object (in our case, a human walking around). In the latest test, the drone ended up circling endlessly above the simulation environment, not following the human at all. From these results, we've learned that the reinforcement learning algorithm needs to be improved so that it provides harsher punishment for failing to follow the human, and/or greater reward for keeping the human in view of the camera.

6. Closing Material

6.1 Conclusion

At this point we have our simulation environment set up and tested. Currently the neural network is trained using this environment to make the drone able to make movement decisions based on what it sees from the camera. The current goal for the neural network is to fly the drone, operate it without crashing and bumping into obstacles, detection and segmentation of visible humans, and making the drone follow a person. On the other hand, the camera is tested separately while we identify and try different approaches to getting stereo images suitable for our neural network. As for the hardware, the drone assembled in test configuration and is currently tested using the trivial remote controller.

Our next step on the software side will be to make more complicated simulations for the neural net to solve and train. The camera will then be integrated to test how well our virtual drone can make decisions based on the image from the actual camera. Meanwhile, one of the Nvidia SoCs will be mounted on top of the drone (possible with a change in chassis assembly configuration) and hooked up to the rotor controllers. Next we will test how well the console commands from the SoC to the controller can manipulate the rotors and, after tested, the drone in midair. The last step in integration would be to put everything together and test how well can the drone make decisions and behave autonomously without human interactions. After that is successful, we will start developing and implementing a more complex computer vision algorithm - the volumetric analysis.

This way we can develop and test different components apart from each other thus eliminating inside-module bugs and malfunctions. We will reach a major milestone when he drone is put together and autonomous in avoiding collisions and making simple movement decisions (like following a person). After this is done, it will be time to develop and apply a more complicated algorithm. Such an approach also allows us to divide tasks between different team members according to specific skills and interests of each one of us.

6.2 References

None used.

6.3 Appendices

No additional material.