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: 10/06/2019 - v1

Executive Summary

Development Standards & Practices Used

List all standard circuits, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

Summary of Requirements

List all requirements as bullet points in brief.

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

Table of Contents

List of figures/tables/symbols/definitions (This should be similar to the project plan)

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.

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
- Become able to control the drone in the simulation via code/console commands
- Make a custom world in the simulation for basic drone training
- Come up 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.

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.

Include any concerns or details that may slow or hinder your plan as it is now. These may include anything to do with costs, materials, equipment, knowledge of area, accuracy issues, etc.

3.5 Project Proposed Milestones and Evaluation Criteria

What are some key milestones in your proposed project? Consider developing task-wise milestones.

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

Link below goes the Project's timeline:

https://docs.google.com/spreadsheets/d/1G79frzZnDQSUJYY2QotfJKLrOazHoSnM1jotNjP152k/edit?usp=sharing

Timeline rev 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. Once we get going steadily 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 our estimates change.

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

Each member is putting forward as much of an effort as they can. Each task we take on, we are far ahead of what we initially plan. That being said, we all also have full semesters of work from other more pressing classes. Our effort will change per task based on how busy each member is and thus is hard to give an exact at this time. Each team member usually pays about 5-6 hours a week to solve current project tasks.

For better details in exact expectations of the project, see project timeline section.

4.4 Other Resource Requirements

Hardware Components:

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.

Software Components:

One of our core software requirements is 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

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case 5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested 8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

5.1 Interface Specifications

– Discuss any hardware/software interfacing that you are working on for testing your project

5.2 Hardware and software

– Indicate any hardware and/or software used in the testing phase

– Provide brief, simple introductions for each to explain the usefulness of each

5.3 Functional Testing

Examples include unit, integration, system, acceptance testing

5.4 Non-Functional Testing

Testing for performance, security, usability, compatibility

5.5 Process

– Explain how each method indicated in Section 2 was tested

– Flow diagram of the process if applicable (should be for most projects)

5.6 Results

– List and explain any and all results obtained so far during the testing phase

- – Include failures and successes

- – Explain what you learned and how you are planning to change it as you progress with your project
- – If you are including figures, please include captions and cite it in the text

• This part will likely need to be refined in your 492 semester where the majority of the implementation and testing work will take place

-**Modeling and Simulation**: This could be logic analyzation, waveform outputs, block testing. 3D model renders, modeling graphs.

-List the **implementation Issues and Challenges**.

6. Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly reiterate your goals. Then, reiterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

6.2 References

This will likely be different than in project plan, since these will be technical references versus related work / market survey references. Do professional citation style(ex. IEEE).

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc. PCB testing issues etc. Software bugs etc.