Automated Drone Laser Tracking System
ADLTS

Tommy To
Samantha Chou
Fernando Trevino
Aakash Gupta
Jon Rohrback

Computer Science Senior Design

Advisor: Dr. Richard Leinecker

Abstract

Because there are technologies that exist to harm the livelihood of people, there also exists technologies that are meant to counter or destroy potential threats. In the defense industry, military grade technologies include the ability to track and destroy threats, which employs the use of different tracking methods and countermeasures.

This project exists to study and deploy such methods and countermeasures. The ADLTS team will be studying different algorithms and create a system that mimics the real technologies used to track and target a potential threat such as a flying drone. In order to achieve full automation of this system, the team will study different ideas listed and explained in this paper.

Table of Contents

Motivations

Tommy To

CS is my second BS degree, I graduated with an Accounting BS 2 years ago. I chose CS because I realized that Accounting was too boring. During my accounting jobs I would "create" tools to make my job easier and enjoyed this, so I figured I could go back for a second degree in CS. The idea of ADLTS was created when Dr. Leinecker mentioned about drones from previous projects. I'm currently an intern at Lockheed Martin, and I created this idea based on some of the real stuff I know we use in the defense industry. So, I wanted to do a SD project that will help anyone that is interested in applying a job for in the defense industry.

During my time at my internship at Lockheed Martin, my managers would tell me that the defense industry highly values the hardware side more than the software development. This is due to their high sales in the hardware, because as one of the top defense companies in the world, Lockheed Martin sells a lot of hardware such as missiles and planes. As a result, the top engineers are often those that have worked with hardware in some capacity. So, it was my main goal for this project to learn as much as I can about simple embedded devices in order to order to impress my future employers.

I came up with this idea purely for this opportunity to further my career goals, and hopefully with this project under my belt I will be able to impress my interviewers.

Samantha Chou

I'm currently a student at UCF studying Computer Science. I used to be in the nursing major at a private university, and that's a long story there. However, I always wanted to be do something with related to engineering because I liked to create things and I didn't have as much of a passion for the health field. After a couple semesters I ended up switching to Computer Science and I have been enjoying every part of it.

What I like about CS is the project creation aspect and process of it, being able to create something, go through that process, and watch it work. I am interested in ADLTS because I like the software hardware correlation within this project and it definitely looks good on resumes, especially since I want to work in the defense industry. I would like to work with software or any front-ended needed part but learning and being familiar with all portions of the project would be great.

I have been a part of the same internship program for the past 2 years at Lockheed Martin, which is why I have the desire to continue working in the defense industry. All of my mentors that have led me throughout the years have pointed out that hardware experience in the defense industry is highly sought after. That is why I wanted to do this project in particular as it has a lot of opportunities to work with hardware and the firmware related to it.

Fernando Trevino

I am an undergraduate student at UCF finishing my degree in Computer Science B.S. with a minor in mathematics. I worked as a Math tutor for a few years and I'm currently working as a developer working with databases, websites, game simulations and AI for healthcare. I chose this project because of my interest in AI, Computer Vision and Simulations. For this project I developed the simulation to implement the Middleware responsible for transforming the coordinates sent by the detection system into movement of the motors by approximating velocity between other things.

My interest is in simulations and A.I. and must of the times that comes in hand with Data Science and Computer Vision. My interest in this project first sparked when I saw the possibility of using computer vision and a simulation for experiments. My goals for this project are to practice my programming skills and learn more from computer vision and close the cycle by working closer to the hardware as well.

Aakash Gupta

As a Physics and Computer Science major, I am very much devoted towards working with motors and microcontrollers. With my experience of working with Arduino, I found this project to be very fascinating where I can truly implement my microcontroller and working with motor skills to a good use. This project requires to have a good teamwork and understand the dynamics of the drone from detection to movement of the motors in order to track the drones. I always was leaned toward the machine learning/artificial intelligence projects that also caught my interest to work on this project.

As I am looking myself to be a future, physicist and a computer scientist, it is crucial for me to understand how I can integrate different concepts of these vast field into one application. Based on my previous experiences, I have worked individually on projects related to microcontrollers and computer vision individually, but this integration will help me understand the concepts they both may be lagging in. I plan to understand the concept to integrate the different area of this project and make on a complete task which I may be able to apply in future applications. For this project, I am working with the hardware related tasks so that I can use my knowledge to the full extent. I am looking forward to moving the arm, that contains both laser and camera, in all directions based on the specific angle provided by the Compute Vision with help of certain backend work.

This Project will help me understand the dynamics of different motors along with the microcontroller with the serialization which can help me in future while pursuing the robotics path as a career.

Jon Rohrback

After halfway through the Electrical Engineering program, I decided that I wanted to focus more on the software engineering aspect with a streamlined focus in machine learning and robotics which is why I switched my major to Computer Science with a minor in Mathematics. I had a strong motivation to work on this project due to the ability to gain experience in computer vision, software to hardware encoding, and a bit of robotics. Within our group, I was chosen to specialize in the computer vision aspect and developing the object tracking system that would be most beneficial for this project.

I wanted to work on ADLTS because it was a great way to get experience in several different fields such as robotics, computer vision, and machine learning while also developing something that has potential to be used in real world applications. This will be a challenging yet rewarding project to see through to completion and kick start my career after graduation. I am very interested in the field of robotics and machine learning with the goal of one day owning my own robotics company. With this project, I am hoping to be able to learn a lot in regards to computer vision and tying it together with robotics and potentially other machine learning algorithms through both research and actual hands-on experience over the course of the following two semesters, that will allow me to get a jump start into the industry.

Communication Tools

Due to the COVID quarantine that heavily affected the communications of our team project for Senior Design, we had to use internet tools in order to make up for the fact that we could not meet physically to develop. Our team meetings thus far have all been virtual, and have been a success due to two specific communication software tools, named Discord and Zoom.

Discord

Discord served as our overall communication hub, with our daily text communications on its servers. Through the servers, we had constant access to our chat history, as well as a means of keep track of our shared resources such as documents and files. Another way we used Discord was to keep track of all the Zoom meeting links. We had a channel that was organized just for Zoom links that were posted in order for the team to refer to previous meeting records.

Zoom

Although Discord has voice and video chat capability, we chose Zoom as our primary weekly meeting medium because of its capability of easy recording of the meeting. Zoom then stores the meeting in its cloud servers, and only required the link and password in order to access the records. This made it a lot easier to record and keep track of our meetings. These links and their respective passwords were then posted onto our Discord channel that was specifically designed for the Zoom records.

Arduino IDE

In order to interface and code with the Arduino Microcontrollers, we had to use their built in IDE, which is called Arduino. It is open sourced and has many different projects that users can download and learn from in order to easily get started with the Arduino hardware and firmware development. The tutorials were especially important for our team in order for everyone to get a handle on the new tool. The Arduino IDE is where the coding would take place, which would then be pushed onto the board through USB connection.

Languages

C++

The primary language that we ended up choosing as our code base language is C++. This is due to its easy compatibility with the Arduino boards as well as the OpenCV software. OpenCV is the opensource program for the computer vision, so being able to use C++ for all our tools and software was a huge plus. Many people recommend using C++ as the primary language when working with Arduinos.

Python

Python was another language that we had considered in our project, simply because of its simply use with OpenCV. It was completely possible to use Python only for OpenCV, and then make it work with our main program that uses C++, but then our developer for the camera vision chose to learn C++ for the sake of making this project more centralized.

Other Languages

Some other languages we were considering were QT, in order to develop a GUI so that we can display our data from the Laser Sensor and Laser turret onto the screen for easier testing and understanding of our systems. This is a further stretch goal that we might be considering in the future.

Overview and Goals

The Automated Drone Laser Tracking System (ADLTS) will be a fully automated system that integrates different technologies, hardware, and firmware in order to detect and track a threat in motion. We have decided to use a flying drone as the threat, which our system will target and aim a laser at as a countermeasure. While there are many different methods of detection, our system will be using a camera and computer vision algorithm in order to detect, track, and predict the movement of the threat. Using these coordinates and vector data, our 2-degrees of freedom turret motor system will smoothly guide our laser onto the target drone's sensor.

Constraints

- Drone will have a spherical sensor array attached that has a diameter of 4in (101.6mm)
- Drone will have a max speed of 10mph (16.09kph)
- Drone will initially hover at a radius of 10ft (3.048m) from the CLMS but will have 360° movement around the x axis and 180° around the y axis as seen from the CLMS

Requirements

- The Camera-Laser Mount System shall automatically detect and target the drone's sensor.
- The middleware algorithm shall transform the pixel-coordinate data from the camera in order to navigate the motors of the CLMS to align the laser onto drone's sensor.

- The drone's sensor system shall record the consecutive time of detection of a laser, and upon the arbitrary 2 consecutive seconds kill time, turn on blinking LEDs in order to communicate success of the overall project
- The middleware algorithm shall produce telemetry data in the form of a .csv file.

Stretch Goals

In every project, it is unrealistic to accomplish all future goals at once due to constraints such as personnel power, time constraints, and budget. Therefore, it is important to create a goal that is realistic and adjust the scope in order for the team to succeed. However, projects in technology always need to evolve and improve in order to keep up to date so it is important to create stretch goals and future plans. This section exists to explain the stretch goals that our team have planned should there be reason to believe that they can be achieved.

Multiple Integration

One of the stretch goals that our team is considering is to be able to use multiple laser turret systems in conjunction in order to target multiple threats. In order to do this, we would have to make our laser systems module and able to communicate with each other in order to process the threats. This is a stretch goal that we most likely will not be able to get around to, as the overall project would already be considered a success with just one turret being able to target a threat. However, this idea was still considered and document simply because in the real-world applications of defense turrets, there will often be multiple units of defenses working in conjunction in order to prioritize and target multiple threats.

GUI Development

Another stretch goal that our team was considering was adding a GUI that communicates all the data from our systems. The ability to read data from our systems will allow for easier testing, and further development. It would allow for communications between multiple systems, as the GUI would allow the user(s) to have more control over their systems.

Data Transfer via Laser to Sensor

One of the stretch goals that we currently have is for the laser sensor that is to be attached onto the drone. Currently the laser sensor is only to detect the laser signal and only uses that true or false detection in order to satisfy the requirements. In the future, with time permitting, we are going to implement data transfer over to the laser sensor. Transferring data via laser wavelengths will be covered in a later section, but it is possible, and we want to use this data to determine different outputs on the LED lights on the sensor. For example, we can have a multi-color LED light, and based on the laser signal transferred and received, the LED can turn on a different color in order to signal success. This will not only show success of a laser signal detected, but it will show that we were able to specifically transfer data via the laser wavelength.

LED Lights Automation

Another stretch goal is to automate the success signal. Currently the way we are determining success of the project is by looking at the LED lights on the sensor that is carried by the drone. If they light up, then the sensor is signaling that there was a detection of the laser from the laser system. The detection was able to meet the threshold of 2 seconds of consecutive contact of the laser signal. When the LED light up, that is currently the only way for the testers on the team to know that the test was a success.

WIFI/Bluetooth for Laser Sensor

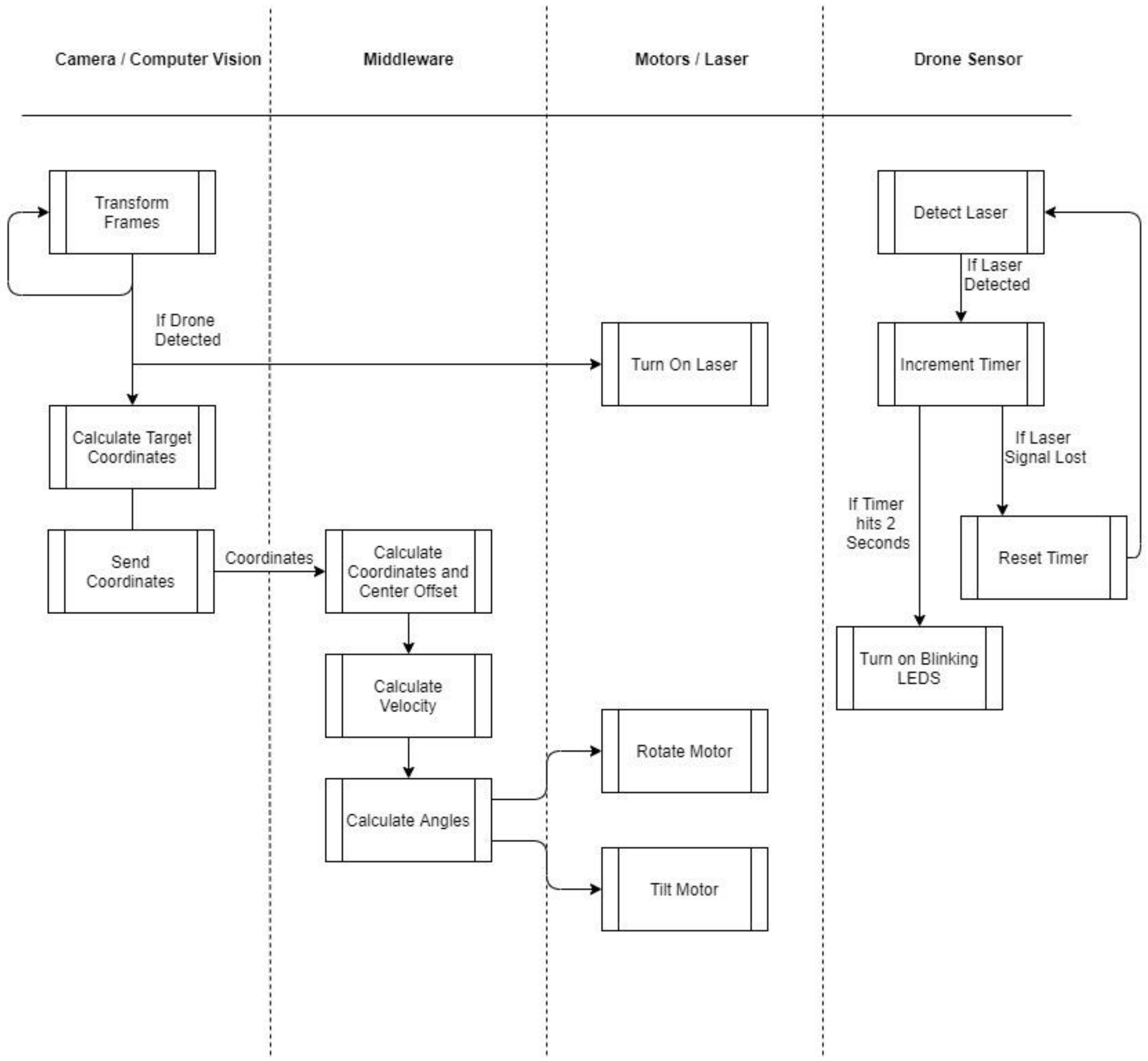
One of the ideas in order to make this process less manual is to use an Arduino microcontroller board with a WIFI port or Bluetooth capability. If we were to use this, then laser sensor will be able to transmit its data directly to our hub which would make testing easier in general. We would also be able to have sufficient data in order to figure out what is wrong. By using this idea, we would also be able to connect this data to our laser turret so that it will directly know when it is able to successfully hit the target or not. This will open doors to further stretch goals of using machine learning for our laser turret.

Legal Issues

While we will not need to have a license to fly the drone for recreational purposes, we will have to register our drone through the FAA due to it weighing more than .55 pounds. Our drones are chosen specifically for the carrying weight capacity in order to carry our sensor system.

All software code and other algorithms developed for this project will be openly available on our repository, as we will be using some other open sources such as OpenCV. Any materials or hardware used will be owned and kept at the end of the project by the original purchaser, unless otherwise agreed upon by the group.

Technical Ideas and Research



ADLTS can be broken down into 4 different sections which includes the camera/CV, middleware, motors and laser hardware, and drone sensor. Each of these parts can be implemented in different ways, so we researched the optimal setup and methods in order to integrate a successful system.

CAMERA / COMPUTER VISION

Objective

To achieve the overarching project goal, the ADLTS system must first be able to detect the laser sensor target which will be attached to a drone. Once the drone flies within the current field of view (FOV) of the camera-laser mount system (CLMS), the object detection algorithm will determine the pixel-coordinate location of the target and relay this information to the middleware algorithms that will then transcribe the required movements for the motors.

Initial Thoughts

The primary goal of this project is designed in a way that provides several constraints on the drone's movement and also the CLMS, however these constraints can be loosened as we progress through the development process to allow for stretch goals for this system. Currently, we have several constraints as listed below:

- Drone will have a spherical sensor array attached that has a diameter of 4in (101.6mm)
- Drone will have a max speed of 10mph (16.09kph)
- Drone will initially hover at a radius of 10ft (3.048m) from the CLMS but will have 360° movement around the x axis and 180° around the y axis as seen from the CLMS
- The CLMS will follow the target sensor array's movement for a consecutive two seconds after initial lock-on

- The testing environment will be primarily static, but solutions for dynamic environments will be researched for implementation

After reviewing the constraints, several things have become clear as to implementing an efficient and working solution. First, the camera must be able to process video at a high frame rate with a workable resolution depending on the computer vision algorithms used in order to accurately detect and track the sensor array. Second, since CLMS will be working in real-time, the algorithm must be efficient and pair well with the middleware algorithms and calculations being used to achieve the goal of tracking for two consecutive seconds. Last, since the current plan is to move everything to an embedded system towards the end of development, the software and hardware must pair well with the embedded system chosen while still achieving each of the required project goals.

Camera

In order to determine the best camera suited for this project, we must understand a few of the key characteristics of cameras, specifically for live video feed usage. These attributes include resolution, frames per second, and field of view.

Resolution

Resolution refers to the number of pixels in a given image or video frame. For videos, the resolution can range from 360p all the way to 4k. However, for the camera modules that were researched there are two resolutions that are commonly found, 720p and 1080p. 720p resolution refers to an image or frame with dimensions of 1280 pixels wide by 720 pixels tall, while 1080p refers to a resolution of 1920 x 1080 pixels. Resolution primarily affects the quality of the given image meaning an identical image portrayed by both 720p and 1080p will have more pixels in 1080p resolution allowing more pixels per object in the image thus being able to show more detail overall.

Frames Per Second

Frames per second (FPS) is a unique trait attributed to video in which it refers to the number of images taken concurrently within one second of time. While there are a multitude of cameras that can record at higher FPS, all of the camera modules researched had two different modes: 30fps and 60fps. These cameras are only able to increase their FPS rates by reducing the quality of the images, i.e. the resolution. However, there is a positive tradeoff with the higher speeds which is that higher frame rates allow smoother movement since it's capturing more snapshots of the object in motion during the same amount of time meaning that the object won't appear to be jumping around the frame which is very useful for object tracking in computer vision.

Field of View

Field of view (FOV) is everything in the world that can be seen inside the camera frame. FOV can be split into a horizontal angle and vertical angle which can determine how much is included within the frame at any given distance from the camera. While the idea of more being seen with higher viewing angles sounds like a positive, it could have some negative effects based on the resolution. If two images have the same resolution, but different FOV, objects in the image with the larger field of view will have less detail than the one with the smaller angles thus possibly resulting in poorer results for computer vision.

For this project, it may be beneficial to have a better understanding of the size comparison of the sensor array target compared to the entire frame in regards to pixel ratio to insure that the algorithm we use is capable of tracking this target with specific components. We can calculate this ratio given certain constraint information such as the distance from the CLMS to the drone, the size of the target, and the camera's FOV and resolution. The image below is a simple 2D example drawing of a camera frame with a horizontal and vertical FOV placed a certain distance away from the drone.

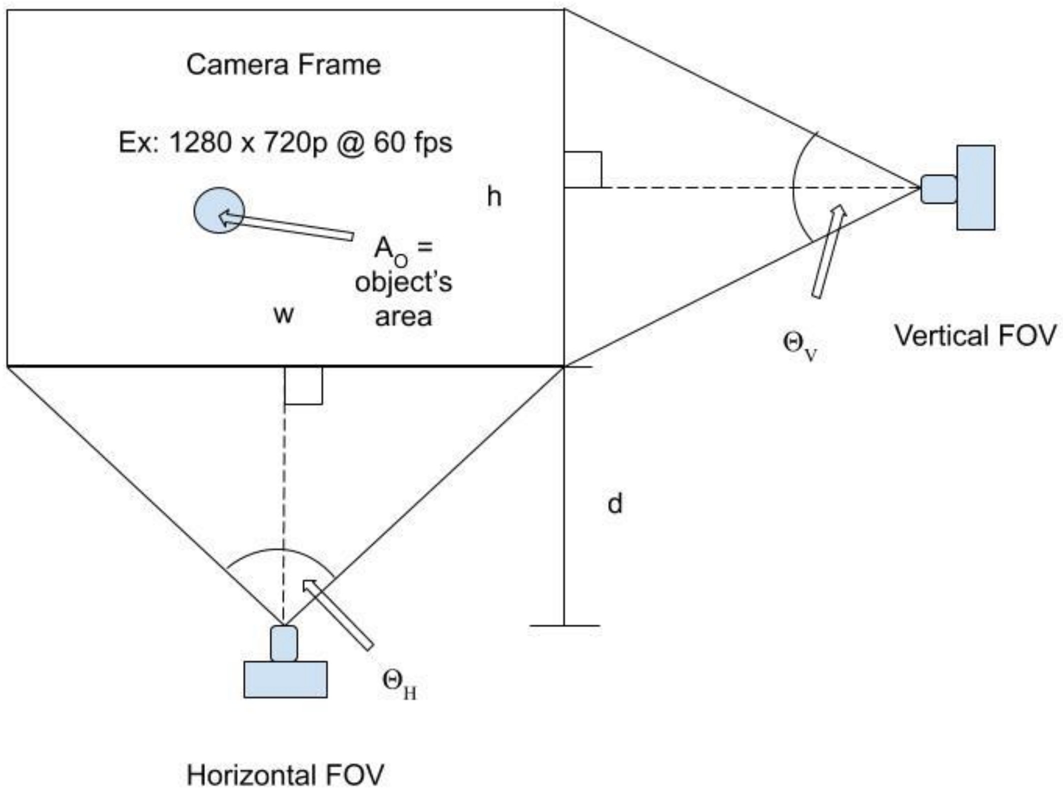


Figure 1 - Example of Camera FOV

The first step in calculating the ratio r is to find the width and height of the world that is viewable within the frame based on the distance from the camera to the desired point in the image and the horizontal and vertical FOV. This is done by calculating the following two equations:

$$w_d = 2\,(tan(A_H) * d)$$

Equation 1 - Width of Viewing Frame

$$h_d = 2\,(tan(A_V) * d)$$

Equation 2 - Height of Viewing Frame

Next, we must divide the area of the object by the area covered by the frame using the width and height calculated in the previous step to get the ratio and then we can determine the number of pixels attributed to the object in the image by multiplying that number by the number of pixels represented in the image based on the resolution's width and height pixel counts, w_p and h_p as shown in the following equations.

$$r = \frac{A_o}{w_d * h_d}$$

Equation 3 - Ratio of object size to frame size at distance d

$$n_p = r * (w_p * h_p)$$

Equation 4 - Number of pixels for the given object in the image

For example, if we were to use a camera that had a resolution of 1280 x 720p for 60fps, an object, with a diameter of 4 inches, a distance of 10ft away from the camera and the camera had a horizontal FOV of 54° and vertical FOV of 41°, there would be a total of approximately 1055 pixels featuring the object.

After understanding these characteristics, we researched some camera modules, specifically ones that would integrate seamlessly with Raspberry Pi boards. Below is a table showcasing each of the chosen cameras' specifications related to this project.

Model	Resolution	FOV	Dimensions / Weight	Price
Arducam 5MP OV5647	2592x1944 1080p30fps 720p60fps	54° horizontal 41° vertical	~25x24x9mm ~3g	$8.79
Raspberry Pi Camera Module V2	3280x2464 1080p30fps 720p60fps	62.2° horizontal 48.8° vertical	~25x23x9mm 3g	$27.50
Raspberry Pi Camera Module 12.3MP HQ w/ 16mm Telephoto Lens	4056x3040 1080p30fps 720p60fps	44.6° horizontal 33.6° vertical	Mount 38x38x18.4mm ~5g Lens 39x39x50mm 133.7g	$112.50

Table 1 - Raspberry Pi Camera Module Options

After considering many different camera module options, these three in particular stood out. For this project, the Arducam 5MP OV5647 seems like it will be the most cost efficient choice while still being just as capable as performing the required job.

Image Noise

One of the major concerns in computer vision and digital image processing is image noise, which “is a random variation of brightness or color information in the images captured” (Swain). Noise can cause images to skew randomly from pixel to pixel which in our case could lead to returning false positives or false negatives during the runtime of the object detection and tracking algorithms. There are different types of noise and with them different types of filters for reduction; however, we will primarily focus on the Gaussian Filter as it is one of the more popular methods used in computer vision applications. Another thing to be considered with noise is the variations between foreground and background objects within each frame. Since our goal is to solely focus on the drone’s movement within the frame, we can use a technique called background subtraction to hone in on the focal point that is the drone and its sensor array and further reduce any noise caused by the image’s background. One popular technique used for background subtraction is Mixture of Gaussians (MOG) which is a “model to subtract backgrounds between frames” (Grevelink Intel).

Raspberry Pi Communication

Since our team has decided to utilize a camera module that works seamlessly with the Raspberry Pi in order to make the CLMS more modular and concise, we have to determine the best way to transfer the camera frame data to our computer to run the object tracking and middleware software without losing speed or quality. By doing so with a wired connection means it can reduce the latency of the data transfer, however our team will have to set up a timetable for when the raspberry pi should send the image data and when it should receive the stepper motor instructions otherwise there could be a serialization conflict that would make the system run on irregular cycles. Wirelessly sending the video frame data every frame wouldn't be better due to having latency delays which would make the system unable to keep pace with the drone's movements. Our team will be able to test the different options available during the integration phase.

Computer Vision

Computer vision can be split into three unique sections: object detection, object recognition, and object tracking (Intel). The implementation of this project requires either object detection or recognition immediately followed by subsequent tracking of the object once it moves in the initial frame. There are many different object tracking algorithms and several computer vision libraries and frameworks that are capable of handling object tracking. However since this project requires using a live video feed to then perform complex calculations to provide real-time feedback in the form of motor movement, we must carefully choose the most optimal and efficient object tracking algorithm to implement in our design. Below, we explore some of the different technologies and algorithms that are possible options.

Common Problems in Computer Vision

Object detection and tracking aren’t as trivial and straightforward as it may seem. Humans are trained over years to be able to quickly recognize objects within their field of vision, but computers don’t work that quickly or accurately yet. In Ankit Sachan’s article on different object tracking modules, he describes eight different pitfalls that occur in computer vision. These challenges that limit the extensibility of object tracking include occlusion, identity switches, motion blur, viewpoint variation, scale change, background clutters, illumination variation, and low resolution. These problems must be taken into consideration when deciding which algorithms are best suited for the requirements of this project.

Libraries

The main constraints for deciding which library to use is that it must be compatible with Raspberry Pi, use C++ as, and it must be robust and efficient so as to meet the requirements set for tracking the drone for a consecutive period of time. It would also be beneficial to select a library with extensive documentation and support as a way to minimize errors or issues that might arise during implementation.

OpenCV

OpenCV is a library designed specifically for solving machine learning and computer vision problems. It has many tools that can be effective for object detection and even has several built-in tracking algorithms that can be easily utilized as well as built-in classes for background subtraction and noise filters. Many of the most popular algorithms are used designed with OpenCV.

TensorFlow

TensorFlow is a library that specializes in using deep learning alongside machine learning's neural networks to solve a vast array of programming challenges. While not used as much as OpenCV in regards to computer vision problems, it has greater flexibility and potential for all types of applications.

Algorithms

Since computer vision consists of various algorithms that all strive to accomplish different tasks, the following research has been split into three different sections; object detection, object recognition, and object tracking. We will explore these methods that pertain to each different section and determine their viability in respect to the project requirements in order to establish a design methodology that will be deemed the most effective.

Object Detection

Object Detection is a method in computer vision that can find an object in an image, not by recognizing what the object is (whether it's a car or a cat), but by the object's size, shape, or color. These algorithms are somewhat more trivial than the ones used for recognition and tracking, however they can prove to be quite useful if the object's shape or color is known before development. We explore the methods below as a way to determine if any of them may be a good starting point for our design methodology required for this project.

Edge Detection

One of the most popular computer vision applications as well as one of the first projects many software engineers start with when learning about this field is edge detection. This algorithm focuses on a multi-stage process that detects edges by calculating the gradients of the image and filters out edges based on the given threshold parameters. For this project, edge detection seems to be too taxing on the hardware and must be given very specific constraints to operate as expected.

HSV Detection

Another popular method for detection is using the color of the object to filter it out from other parts of the image. This operates by the user first defining the hue, saturation, and value of the specific object to be detected and then the program will filter out anything that lies outside of those thresholds. This method is only useful if the object that needs to be detected is visible before testing as well as if there are no other objects in the frame with the same HSV values that could throw off the algorithm.

BLOB Algorithm

Since our project goal has a set of constraints to aid in the successful tracking of the sensor array, it may be possible to use a simpler method such as the BLOB object detection and tracking algorithm. BLOB stands for Binary Large OBject and it's used to group pixels with similar properties such as color and brightness. This algorithm most commonly uses convolution with the Laplacian of Gaussian (LoG) to detect the objects and then implements contours from OpenCV for mapping.

Hough Circle Transform

The Hough Transform was created as a way to detect and recognize shapes, and specifically the Hough Circle Transform (HCT) is used to detect circles within an image as seen in the example below.

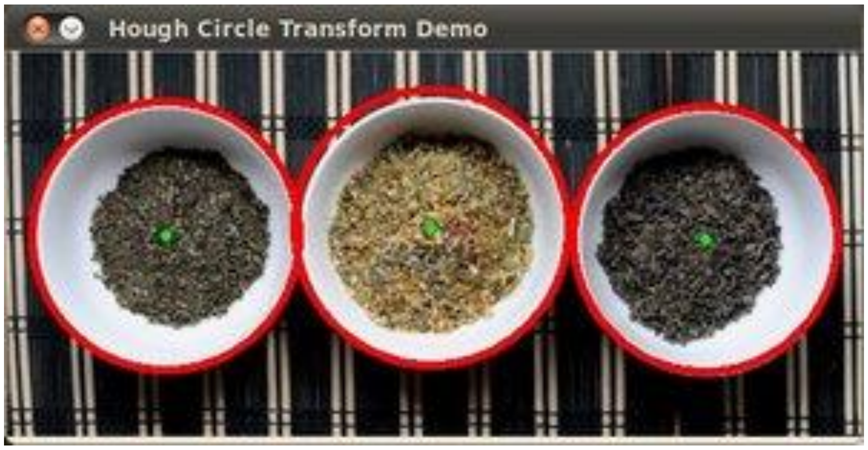


Figure 2 - HCT Example Result (OpenCV)

For this project, it's known that the first phase of sensor arrays attached to the drone will be surrounded by a spherical one-way mirror enclosure which means that no matter what angle the CMLS is looking, the array will consistently appear as a circle in the frame. This constraint makes HCT seem like an ideal semi-trivial method to detect the drone's sensors.

However, with this method there are still a few shortcomings. Firstly, the algorithm requires several different parameters that our team may not initially know what to set to and so we must test it repeatedly on real objects to narrow down the required values. The next possible issue will vary from application to application. The HCT algorithm is capable of detecting all circular objects in the frame as long as they meet the given requirements derived from the parameters, but our project only requires one specific circle to be detected every frame. This indicates a possibility of higher error during detection since this algorithm can't verify the identity of the sensor array without assistance from another part of the system in which case our system may start tracking a completely unrelated object. Another concern is that in the stretch goals for this application, the drone may not have a spherical sensor array which would make the CV implementation obsolete and we would have to resort to another method of detecting the drone.

Object Recognition

Detection and recognition go hand-in-hand many times although object recognition goes a bit farther by not only determining if an object is in frame, but by defining and returning that object's label such as being a dog or human for example. These "applications can be based on matching, learning, or pattern recognition algorithms" (Grevelink). Essentially, these algorithms only work due to being trained beforehand using machine learning practices to classify different types of objects required for the application. While there are many training datasets available, it’s yet to be determined whether a dataset exists for recognition of commercial drones in which case our team would have to construct our own set of training data samples.

R-CNN, Fast R-CNN, and Faster R-CNN

The concept of a Region-based Convolutional Neural Network (R-CNN) was introduced by Ross Girshick back in 2014 as a way to improve upon the other methods of object detection by using a localization and selective search technique. R-CNN constructs 2000 region proposals which are then passed through a neural network such as the image shown below (Saxena).

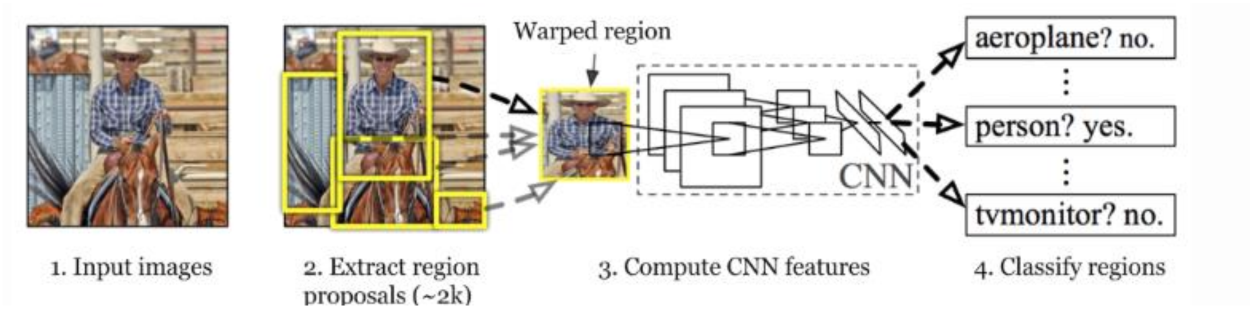


Figure 3 - Region-based Convolutional Neural Network (R-CNN) Architecture

The issue with this method is that it is very slow due to having to train the CNN and then process 2000 regions per image. However, Fast R-CNN was proposed as a solution to this problem by passing in the entire image and region proposals as arguments to the neural net and by expanding on the previous architecture of the system which can be seen in Figure 4. Despite the speedup during training, this model still struggles from a long wait time during the detection process due to a “bottleneck” caused by the selective search process (Saxena).

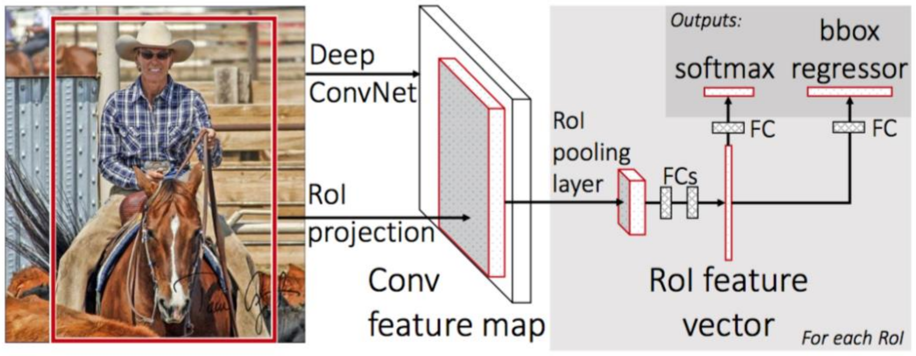


Figure 4 - Fast R-CNN Architecture

This leads us to the last model in this series of R-CNN’s, Faster Region-based Convolutional Neural Network which fixed the aforementioned bottleneck by replacing the selective search with a region proposal network. Without going into details, this algorithm, which is depicted below, reduced the object detection process by 99.96% compared to the original R-CNN algorithm (Saxena).

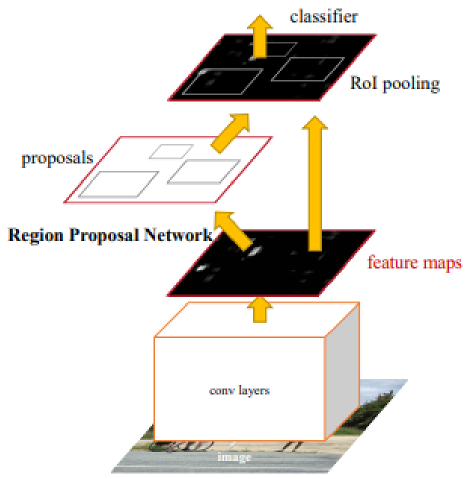


Figure 5 - Faster R-CNN Architecture

Single Shot Detector (SSD)

Another popular object recognition algorithm is SSD or Single Shot Detection which uses a “single deep neural network” and a multibox, which is the term to describe the multitude of various sized filters that work to process the image in a single pass over the network as seen in Figure 6 which in turn then creates a feature map of that input image similarly to the R-CNN models. Due to removing the region-based proposals that R-CNN models use and implementing a CNN with filters that gradually decrease in size, this algorithm manages to improve both the speed and accuracy of detection based on testing with SSD, Faster R-CNN, and YOLO (Khandelwal).

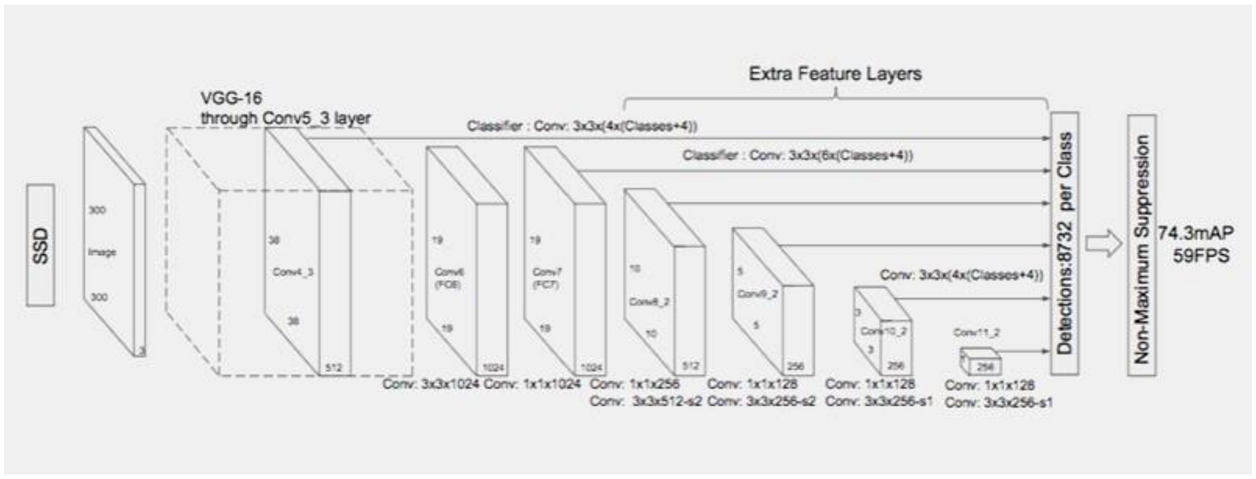


Figure 6 - Single Shot Detector (SSD) Architecture

YOLO (You Only Look Once)

You Only Look Once (YOLO) is an real-time algorithm that uses a multibox similar to SSD as well as deep learning to efficiently classify various objects in an image. Also, similarly to SSD, YOLO only takes one pass through the neural network, hence the name. This is where the similarities stop as YOLO splits the initial image into a grid-like pattern and uses a fully convolutional neural network (F-CNN) to create a class probability prediction map (Sharma). In the few years that this algorithm has existed, there have already been a few iterations made to its core model which started with version 1 and then YOLOv2 and YOLOv3. The original architecture for this algorithm can be seen below.

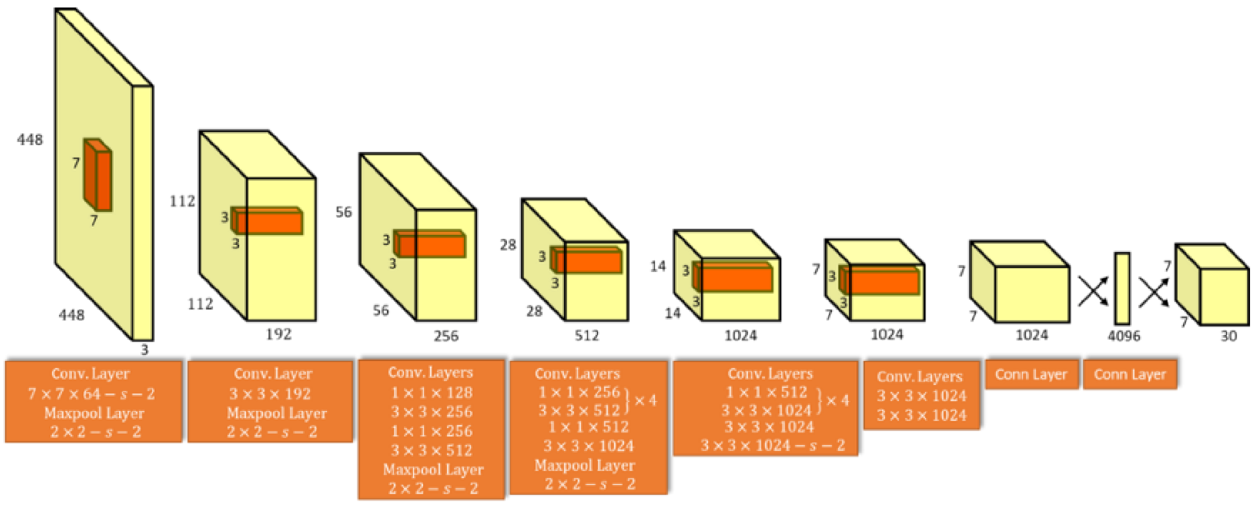
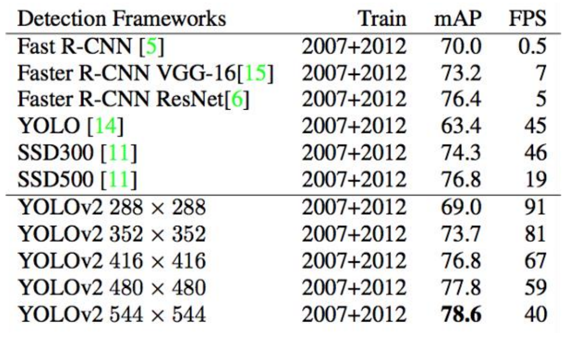


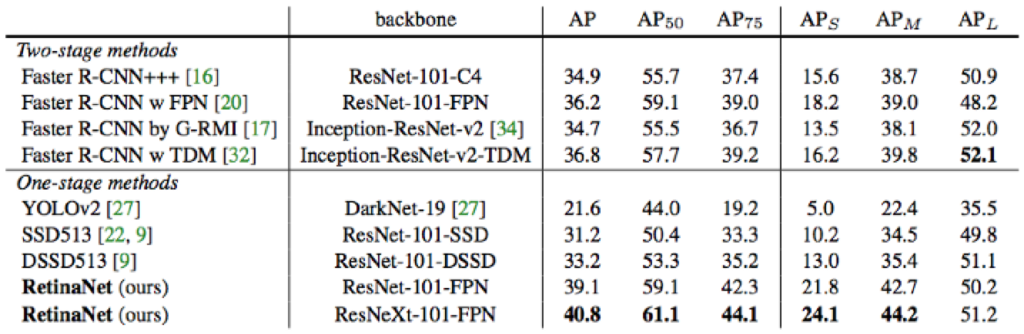
Figure 7 - YOLOv1 Architecture

However, even with this new take on object recognition algorithms, it’s yet to be seen whether it is actually a better choice compared to some other options. Various tests have been conducted to compare the speed and accuracy of these algorithms of which some results are shown in Figures 9 and 10 below (Hui). From the data given, there isn’t any discernible difference between SSD’s best model, SSD300, and YOLO’s best version, YOLOv3. YOLO is a bit faster while SSD is more accurate. Despite this, there are a good number of researchers that have further tested these different algorithms and determined SSD to be better overall.



Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Figure 8 - Object Detection Comparison with VOC 2007 Challenge Test



	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

Figure 9 - Object Detection Comparison with COCO database test

Tracking

Due to the fact that object detection and recognition algorithms are CPU intensive and cumbersome if run on multiple images, our project will require an efficient tracking algorithm that can take an input from the object detection and monitor it's movement within the frame, especially when trying to reach the stretch goal of having our entire application run on an embedded system. According to Ankit Sachan in his article, he claims that there are four distinct parts of object tracking; target initialization, appearance modeling, motion estimation, and target positioning (Sachan). Target initialization will be handled by the object detection/recognition algorithm and pass it's information, generally in the form of an initial bounding box around the object, to the tracking algorithm. Afterwards, the tracking method will then determine the appearance of the object through learning, predict it's motion and next possible state of occurrence within the frame, and finally scan that predicted area to determine if the object is truly in that position.

OpenCV Built-In Algorithms

OpenCV already has eight built-in tracking algorithms within its tracker class. These include BOOSTING, MIL, KCF, CSRT, MedianFlow, TLD, MOSSE AND GOTURN. Several of these are very outdated and way too slow for the purpose of our application. Based on Adrian Rosebrock's research using these algorithms, he only recommends three of the eight as being viable for current OpenCV project; CSRT, KCF, and MOSSE where each one has its own strengths and weaknesses compared to each other. CSRT is useful when an application needs higher accuracy at the loss of frame rate, KCF operates at a faster speed with slightly worse accuracy, and MOSSE is only worth using when the fastest speed is required (Rosebrock). For this project's purpose, we will explore these tracking algorithms during initial testing to determine if any are feasible in meeting our requirements, but we will continue to look for other, more modern alternatives.

Centroid Tracking

While there isn't necessarily a specific algorithm to track the central point of an object, many modern tracking algorithms incorporate the centroid in their output. For this project in particular, it's key to track the center of the sensor array since the middleware of CLMS requires a set of coordinate points for it to rotate and aim at within the frame. No matter the tracking algorithm chosen for this project, the output will be the coordinate point of the centroid for the object's bounding box.

SORT (Simple Online and Realtime Tracking)

Simple Online and Realtime Tracking (SORT) is a "tracking-by-detection framework for the problem of multiple object tracking (MOT)" (Bewley et al. 2017). The team from Queensland University of Technology devised this algorithm as a way to create a more efficient real-time tracking system by only using the detections from two frames, the previous and current ones. They utilized the Faster R-CNN framework as their object detection model as the object detection framework played a large role in the results of SORT. While this team were only concerned with tracking pedestrians, they ran a multitude of comparison tests against other state-of-the-art online trackers that were more focused on accuracy over speed and found that SORT actually performs the best compared to the competition while retaining both speed and accuracy. The team claims that this framework, due to its simplicity and effectiveness, could be used as a baseline in researching improved object detection algorithms that can better handle the problems of occlusion and object re-identification. It should be noted that there is an extension to this framework called DeepSORT that was released a year after SORT and uses a deep learning approach to better stabilize the tracking and improve its performance.

DeepSORT is one of the most widely used tracking frameworks currently.

Design Methodology

After researching the many different types of computer vision algorithms that range from object detection, to recognition, and tracking, it becomes possible to narrow down which ones may prove useful in our application and which ones we can remove from the list of options. For our particular system and due to the constraints set for this project, it seems ideal to utilize three phases of object tracking algorithms throughout the lifecycle of the project up until its finalized submission. These three phases can be labeled as the integration phase, specialized phase, finalized phase.

Integration Phase

Since this project is a mix of software and hardware, there are a lot of moving pieces that need to be taken into consideration with each component's implementation.

Our team has acknowledged that integrating each of the components together into the fully functioning system is going to be a challenge in itself and potentially shed light on unforeseen issues. In an effort to reduce the time necessary for debugging, it is critical to start the integration stage as soon as possible to hit our project deadline; which is where the integration phase for object tracking is required. In order to be able to test our system this early on, we will need to improvise a solution that allows us to test it working with the middleware component, but may not be up to the standards that we are aiming to deliver for the final version of ADLTS.

The middleware component requires the central coordinates of the sensor array for it to compute the necessary calculations that will be passed onto the CLMS. This means that the detection component doesn't have to perform its own detection, but rather just track the sphere through enough consecutive frames to provide adequate information on how the entire system is cooperating. To do this, we can implement a trivial solution for the "object detection" and then use one of the built in tracking algorithms from the OpenCV library. Since the tracking algorithm only requires a bounding box for the object that needs to be tracked, our team can ensure the drone is in the first frame of the video feed and then manually create a bounding box around the sensor array. A diagram of how the integration phase tracking system will work is shown below.

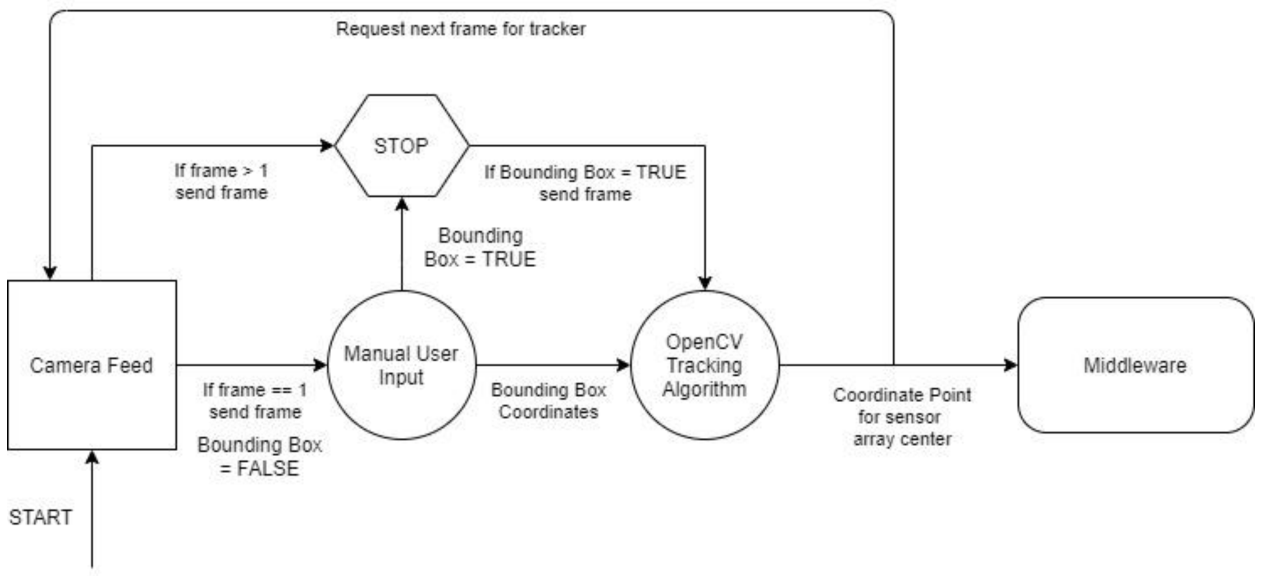


Figure 10 - Integration Phase Object Tracking Flowchart

While this particular setup for the tracking system isn't expected to be the most accurate or fastest option available, it's primary goal is to allow our team to get started as early as possible in integrating all of the components and run tests to determine where, if any, pitfalls might occur.

Specialization Phase

This phase of our tracking system design is aimed to introduce a proper object detection algorithm that will determine if the object is within the frame and then proceed to track it. For this, the specialization phase will focus more on the constraints of ADLTS, primarily the design of the sensor array. Since it's known that initial build of the sensor array will be a sphere measuring approximately 4 inches in diameter attached to the bottom of the drone, it's possible to use an object detection algorithm that implements the Hough Circle Transform and then pair it with either one of OpenCV's built in trackers or an improved framework such as DeepSORT.

HoughCircles is a built in method that can be imported by the core OpenCV library and requires nine parameters to be passed in. The first three are the input image, a vector that stores the coordinate pair for the central point as well as the radius for each circle detected, and the detection method which the only one that currently exists in OpenCV is the Hough Gradient. These parameters always remain the same relatively, while the rest of the parameters will vary depending on the particular use case. These parameters include the inverse ratio of resolution, minimum distance between detected centers, upper threshold for the Canny edge detector, threshold for center detection, and minimum and maximum radius to be detected.

For this project, the parameters that affect our results the most will be the threshold for center detection as well as the minimum and maximum radius that will be allowed for detected circles. The center detection threshold must be set appropriately as to not accrue too many false positives or false negatives in the circles generated. Also, since our goal is to only track the sphere that is the sensor array, we will need to select the minimum and maximum thresholds for the radius with a minimal difference between the two to increase the probability that the only circle detected is the sensor array, otherwise, our tracking algorithm will be attempting to track the wrong object.

This phase will work similarly to the last except that it will handle the object detection by itself without user input, other than the parameters that must be passed into the HoughCircles method. A flowchart of this phase is shown below.

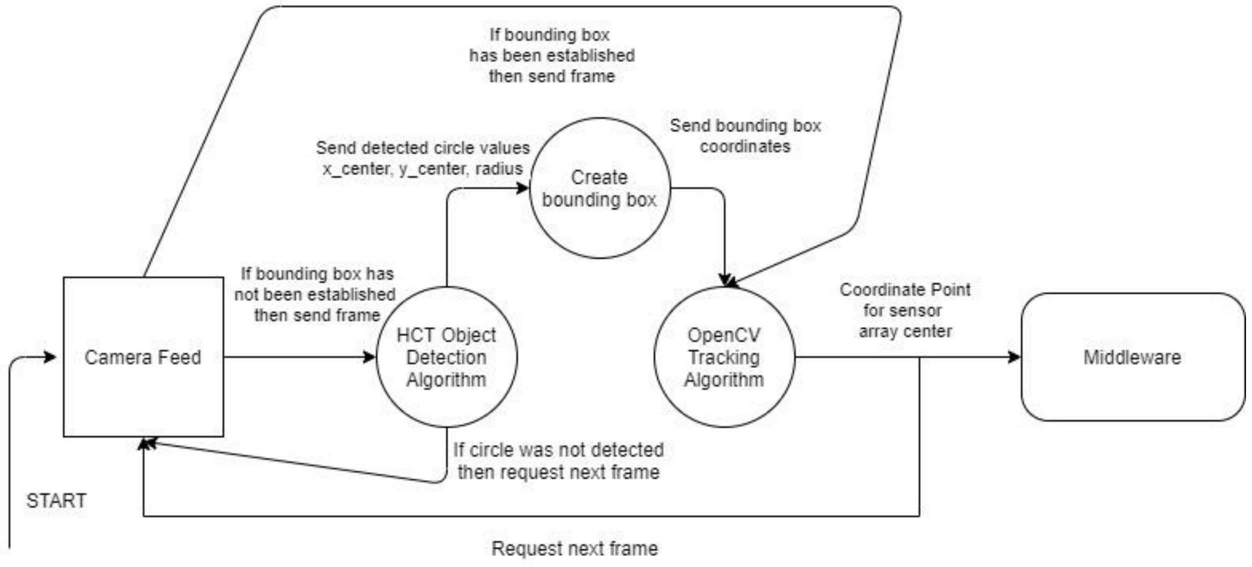

If bounding box has been established then send frame
Send detected circle values x_center, y_center, radius
Create bounding box
Send bounding box coordinates
If bounding box has not been established then send frame
Camera Feed
HCT Object Detection Algorithm
OpenCV Tracking Algorithm
Coordinate Point for sensor array center
Middleware
If circle was not detected then request next frame
START
Request next frame

Figure 11 - Specialization Phase Object Tracking Flowchart

It should be stated that this phase of our system, just like the integration phase, is not a perfect solution. It's uncertain how accurate and fast this object detection algorithm will perform compared to other architectures like the deep learning models discussed earlier. Another concern is that the object detection module won't accurately be able to determine if the circle detected is actually the sensor array which is why the parameters must be accurately determined prior to the tracker testing.

However, investigating the usefulness of this design is beneficial as it may turn out to be more practical for ADLTS without requiring the time necessary to set up and train a more complex, deep learning framework. It is also a good step into determining what type of object detection algorithm is best suited for a project with constraints such as the ones for this system. At this stage, we will have a better understanding of what should be done for the finalized phase.

Finalized Phase

At this point in the life cycle of this project, our team will have a better understanding of the limitations of the previous phases and determine the rationale of transitioning to one of the deep learning object recognition frameworks that can be trained to detect a commercial drone. If a faster and/or more accurate system must be used to achieve the goal of ADLTS then we will have to transition sooner for testing and integration, otherwise it will only be necessary to do so in order to reach one of our stretch goals of tracking a drone that doesn't have a predefined sensor array attached.

The end goal of the computer vision aspect of this system is to implement the most efficient object detection and tracking pairing that will provide the best results for a real-time system such as ADLTS. From the research conducted, merging YOLO object detection with DeepSORT tracking has proven to be one of the best combinations available currently with it capable of running at high frame rates while maintaining superior recognition and tracking accuracy.

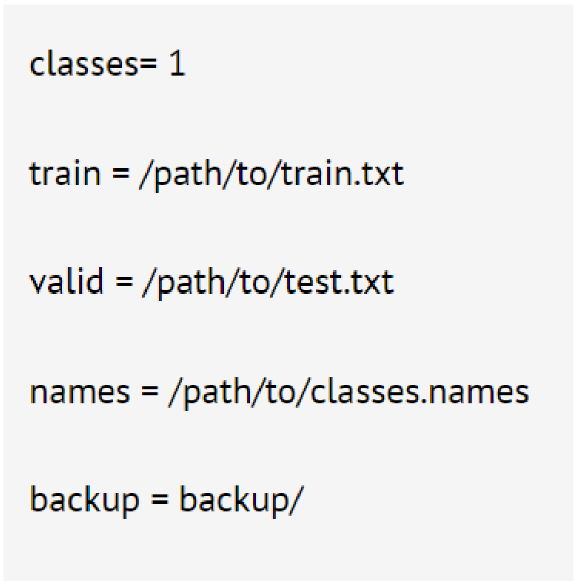
The reason these deep learning frameworks work this well is due to the fact that they are trained on a dataset of images and can efficiently utilize the "experts" created by that training. This, however, makes these algorithms trickier to use and debug if something goes wrong during the testing process. While there are several public datasets available for individuals and teams working in computer vision such as ImageNet and COCO (Common Objects in Context), none of them appear to have a set of images for training to detect commercial drones. Luckly, there is a public Github repository, created by Github user chuanenlin, called drone-net which showcases over 2,600 images of camera-mounted drones that should prove useful in training our system. This will save us valuable time that could be better utilized in integrating and testing ADLTS by not having to establish our own dataset of images.

In order to train our custom dataset onto the YOLO framework, we will need to use another framework known as Darknet. Darknet is a custom framework developed by Joseph Redmon as a more versatile method of training object detection models and commonly used among software engineers due to the improved results compared to other training frameworks.

The first step is to ensure that we have the dataset of drone images downloaded in a shared folder on our system. Then, if it hasn't already been done by a previous individual or team, we must annotate each image. This means that we have to draw bounding boxes around the objects that we're trying to train our object detector to recognize and also label it under the appropriate class, which in our case would be categorized as drone, for each image of our dataset. This part of the training process will take up a large amount of time so it would be beneficial to start the annotations during the integration phase.

The annotations can be created in any photo manipulation software such as Photoshop or LabelImg, however the latter can be seen as the better option due to the fact that it can save the annotated images in the format that YOLO requires. The directory that will be used for the saved annotated images must have a text file with the class name inside. After creating the annotation and label for the image and upon saving it, LabelImg will create a text file with the same name as the image that contains all of the annotation details, which for YOLO is "object-id center_x center_y width height" (Ponnusamy).

The next step using Darknet is to create the remaining files that are required for dataset training. This includes a classes.names file, which is the same as the classes.txt file just with a different extension, train.txt file, test.txt file, and obj.data file. Train.txt consists of a list of all of the images that will be used training which usually varies between 60% and 90% of the total dataset according to Arun Ponnusamy. Test.txt is similar in that it lists the remaining images that will be used for the training validation while obj.data includes the text as seen below.



```
classes= 1

train = /path/to/train.txt

valid = /path/to/test.txt

names = /path/to/classes.names

backup = backup/
```

Figure 12 - obj.data contents

After the training dataset has been constructed, it will be time to actually train the YOLO framework. First, the pre-trained weights must be downloaded so they can be used for the convolutional layers and stored inside build\darknet\x64. Then to commence training, we have to run the command "darknet.exe detector train data/obj.data yolo-obj.cfg yolov4.conv.137", at which this part of the training process will need to be repeated. While it varies with different datasets, the general understanding is that the training process should iterate no fewer than the number of images in the dataset and no fewer than 6000 total iterations. Once the training process has completed and the average error of detection is minimal, the YOLO framework will be ready for use in detecting drone objects.

Once YOLO detection has been set up and verified that it can accurately detect drones, the next step will be to merge it with the DeepSort tracking algorithm. The first part of DeepSort is a Kalman filter which is the primary spatio-temporal detection and prediction. This filter maintains a state of eight variables which include the x and y coordinates for the bounding box center, aspect ratio, image height, and then four values for the respective velocities. It essentially uses a linear velocity model along with the prior state to predict the bounding boxes. The filter also maintains parameters to track and delete objects' states if that particular object hasn't been detected in a while.

The next part of DeepSort handles assigning any new detections with new predictions. This is done by using a distance metric and efficient algorithm. The distance metric is handled by using a squared Mahalanobis distance, which is more accurate than Euclidean distance calculations, and then applying thresholds to create improved associations between the detections and predictions. In most cases, the Hungarian algorithm is used as an efficient data association algorithm as it is capable of running in polynomial time.

Since the Kalman filter has some issues with occlusion and differing viewpoints, deep learning is introduced at this point to improve results by introducing "another distance metric based on the 'appearance' of the object" (Maiya). This distance, which the equation for it is shown below, is created by stripping the final classification layer from the training classifier which leaves the network with a dense layer that produces a single feature vector which will have the bounding box crops passed through it. This process will calculate the new distance metric value which can be seen in the equation below, where D_K is the Mahalanobis distance, D_a is the cosine distance between the appearance feature vectors, and Lambda is the weighing factor (Maiya). Now, between the use of the Kalman filter and Hungarian algorithm, the DeepSORT framework will be ready to provide fast and accurate tracking of the drone.

$$D = Lambda * D_k + (1 - Lambda) * D_a$$

Equation 5 - Distance Metric for DeepSORT Tracking

As for the architecture of the finalized phase, there will be very little difference compared to the specialized phase. The biggest change is that the object detection algorithm won't be searching for circles, but rather it will detect the drone. This means that we will have to calculate the sensor array coordinates based on the drone's bounding box coordinates and the offset of the center of the sphere with the center of the drone. The diagram below shows how this will fundamentally work.

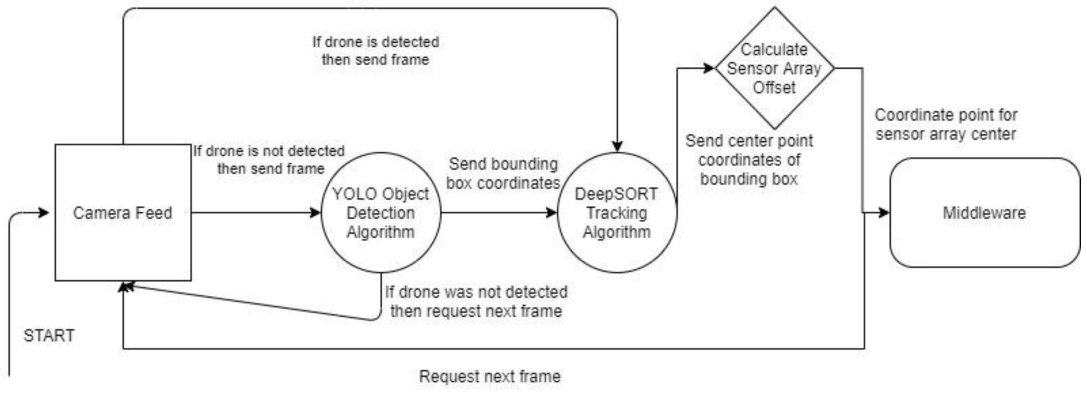

If drone is detected then send frame
Calculate Sensor Array Offset
Coordinate point for sensor array center
Send center point coordinates of bounding box
If drone is not detected then send frame
Send bounding box coordinates
Camera Feed
YOLO Object Detection Algorithm
DeepSORT Tracking Algorithm
Middleware
If drone was not detected then request next frame
START
Request next frame

Figure 13 - Finalized Phase Object Tracking Flowchart

The main concern attributed to using these deep learning techniques for object detection and tracking is the setup of each due to the process being daunting and long-winded. However, if implemented correctly, this approach in theory will show drastic improvements compared to the other phases developed for this project.

Testing

Since it will be necessary to ensure the algorithms work prior to integrating with the other components of the system, some testing will have to be conducted in a location that might differ from the team's designated testing site as well as without the availability of the drone that will be used for this project since it is being constructed and tested by Tommy and Sammie.

The integration phase testing can be done by just checking that the manually created bounding box is capable of consistently tracking the object within that box at a reasonable frame rate. We can test the average frame rate, how often the algorithm loses sight of the object, and ensure that it is transmitting the central coordinates as an output. Testing for the specialization phase requires a bit more effort as it not only has to track the object, which will appear circular in the frame, but accurately detect it in first place. Since we can use the equations discussed in the camera section earlier to calculate the relative size of the sensor array, we can use that information to set precise values for the Hough Circle Transform parameters and adjust based on the detection results. Once the proper values have been determined, we can create the bounding box from the detection results and pass it onto the tracking algorithm to find out how efficiently it runs.

The finalized phase is more difficult to test as it will require the drone that we are using to determine if the trained YOLO framework can detect our particular drone. One way to initially test whether the setup process works is to train it on a dataset of images of people and test it on ourselves to ascertain that it can truly detect people or we can test it using static images of drones, including our own drone. If the algorithm can accurately recognize the drone no matter where it is in the frame then it should prove viable once our team reconvenes real-time testing with the full system. If it can detect the drone image through the webcam stream, then the next step would involve testing whether it can accurately track that printed image while it’s moving around the camera frame.

Conclusion

Through the culmination of research and thorough deliberation of the constraints and requirements for this project, it’s clear that a phased approach for implementing a solution would probably work best, especially considering the time limitation and vast array of usable detection and tracking algorithms. Establishing a trivial solution quickly will allow the team to get started on the integration of the entire system, while allowing myself to then focus on a more systemic approach to the design of the object detection framework. Then, if the testing for that phase of the solution shows a need for improvement, then our team can focus on efforts on utilizing a more robust solution that can provide overall better results.-/

This project, and specifically the computer vision component, has made it more clear how diverse and constantly changing the field of computer vision actually is which can make it difficult to ascertain the best solution for the given problem. However, it also provides the ability to properly learn how these implementations will work on the fundamental level and potentially be inspired to create the next groundbreaking innovation.

MIDDLE WARE

Definition

The “Middleware”, in our context, is the program that acts as the connecting bridge between the hardware and software. This program will be responsible to translate pixel coordinates from an image (given by the detection system) to angles for the motors to rotate to point at the target.

The Problem

How to start the development of the Middleware if the motors and detection system aren’t ready to be integrated? Is our only option to finish all components before we can test the middleware?

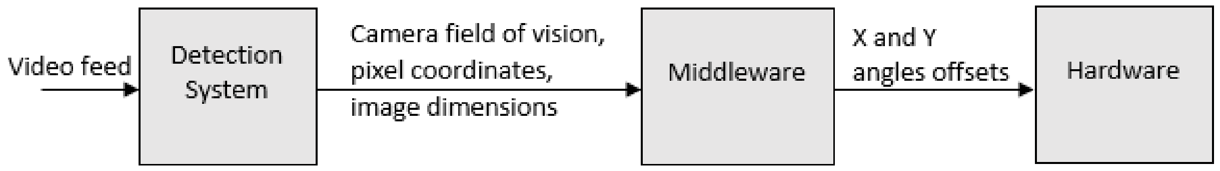


Figure 1: Flow chart of the project showing some inputs and outputs of each component.

Each of these main elements will need to align in tasking to achieve our goal. The problem is that, at first glance, they depend in each other. The motor can’t know where to move to follow the drone without the Middleware telling it what to do; The Middleware can’t translate the drone’s coordinates to physical movement if it doesn’t have a motor to test on and a detection system; and the Detection System can’t detect a drone if there is no drone.

These problems are solved by creating unit tests for each individual element, with the intention of working in parallel and not have to wait for each other to finish. To achieve this an idea was proposed to build a simulation in a Game Engine to simulate our project. To do this, we have to choose the right game engine (platform), simulate a drone flying, a room to contain it, the hardware or more specifically the motors, the laser hitting the drone, the detection system coming from a camera, and the algorithm itself to translate image coordinates to motor movement.

Platform

We looked into Unity, Unreal and Godot as possible engines for the simulation. All three having at least one free version, being written in C++ (Godot used C as well), and able to simulate 3D physics well. It narrowed down to community support, engine limitations, and the right complexity for the job. Highlights for each engine:

Figure 2: Godot Logo.

- **Godot**: Lacks documentation, and the community support is slowly gaining momentum. It uses a custom language called "GDScript". This language is similar to Python in syntax and is a high-level, dynamically typed language optimized for the Godot engine.

Figure 3: Unity Logo.

- **Unity**: Strong community support and best documentation out of all three. It uses C# as the main scripting language. Presumably known for being a user-friendly engine. Not a lot of built in features, meaning most things will be done from scratch or grab from the Asset Store.

- **Unreal**: It has good documentation and community support. It uses C++ as the main scripting language. Reputation for not being user friendly. The only AAA ready engine by default. It comes with many built in features.

Figure 4: Unreal Logo.

Summary: Godot had no meaningful advantage for our purposes. Unreal's main advantage is that it uses C++ which is one of our candidate languages for the computer vision algorithm. Using C++ as the scripting language would allow us to minimize integration time, but some of the built-in features in Unreal might get in the way for our only physics simulation. And Unity on the other side has the best documentation and support, as well as give in us enough control over the physics components to let us build the simulation very easily with minimum effort. Extra side note is that our developer in charge of the simulation (Fernando Trevino Ramirez) is already familiar with the Unity game engine which will eliminate the time spent learning the engine.

Conclusion: Unity advantages outweighed the other two engines for our purposes. Regarding time minimization we compared time of learning the engine (Unreal) versus translating the C# code (from Unity) to C++ to integrate it with the computer vision algorithm. Note: we are exploring options to import one of the two algorithms as a reference to not have to rewrite code.

Simulating the Room

To contain and limit how far the drone can go, we started by creating a room formed by something that looks like the inside of a cube. The 6 walls are square planes of 15 $units^2$ with a plane mesh collider of the same size. To be able to judge depth and follow the movement of the drone, we used a chess like tile material consisting of 4 black and white squares and extended it across the entire surface of the planes. This resulted in an environment looking like Figure 5 below:

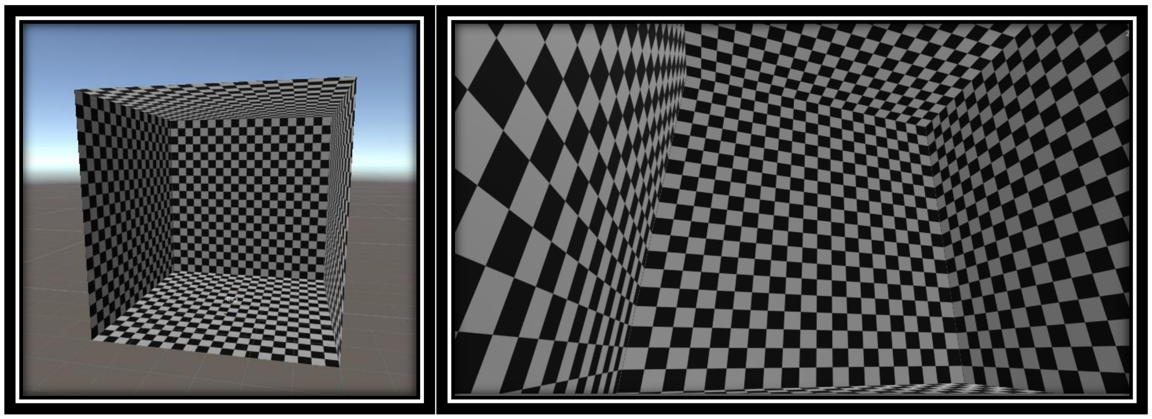


Figure 5: Screen shot from Unity Engine showing the virtual room where the simulation was done.

Simulating the Camera and Laser

To get realistic results we need to simulate the laser in a way it reflects the real setup. Two setups were proposed:

- One with a stationary camera with the laser next to it: This set up would imply that a 3D coordinate be calculated. Which proves to be difficult with one camera and the assumption that we know nothing about the target and our surroundings. We believe if we had at least 2 cameras it would be possible to triangulate the drone's coordinates to calculate the angles and move the laser. This setup would be beneficial if we had more than one laser, since the number of cameras is constant even if we increase the number of lasers.

- And another setup with the camera mounted on top of the laser: This is the setup we chose to use since we only need one laser and this setup minimizes the number of cameras with one laser. The idea for this setup is to mount the camera on top of the laser such that the tangent of the center of the camera is parallel to the laser pointing direction, as seen in Figure 6 below.

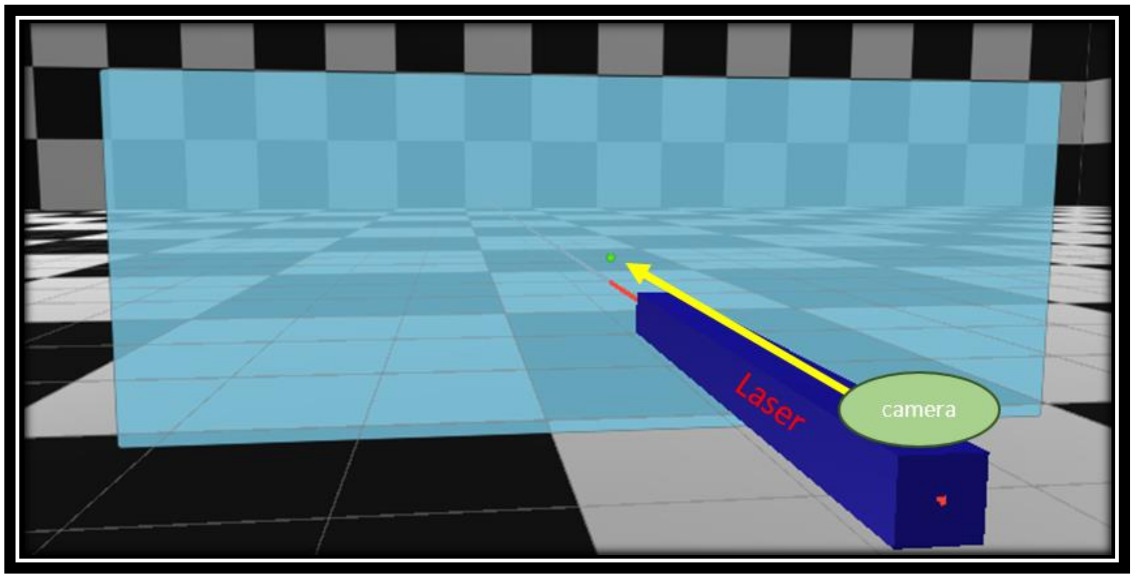


Figure 6: the blue rectangle is a slice of the cone (pyramid) of vision of the camera, the green dot in the middle of the blue rectangle is the center of the rectangle and the vector orthogonal to it is where the camera is facing.

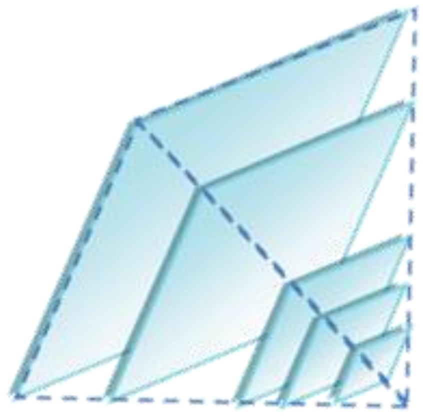


Figure 7: Field of vision of the camera

So, if we were to get a slice closer to the camera, the physical dimensions of the slice would be smaller, and bigger if the slice is farther form the camera, as seen in Figure 7 to the left. It is important to note that even if the physical dimensions are different, the number of pixels to represent each slice is constant (we will use this fact in the future). This is because of how pictures are created. Cameras don't increase the number of pixels in a picture if the objects are far away, the pixels simply represent a bigger portion of the physical view.

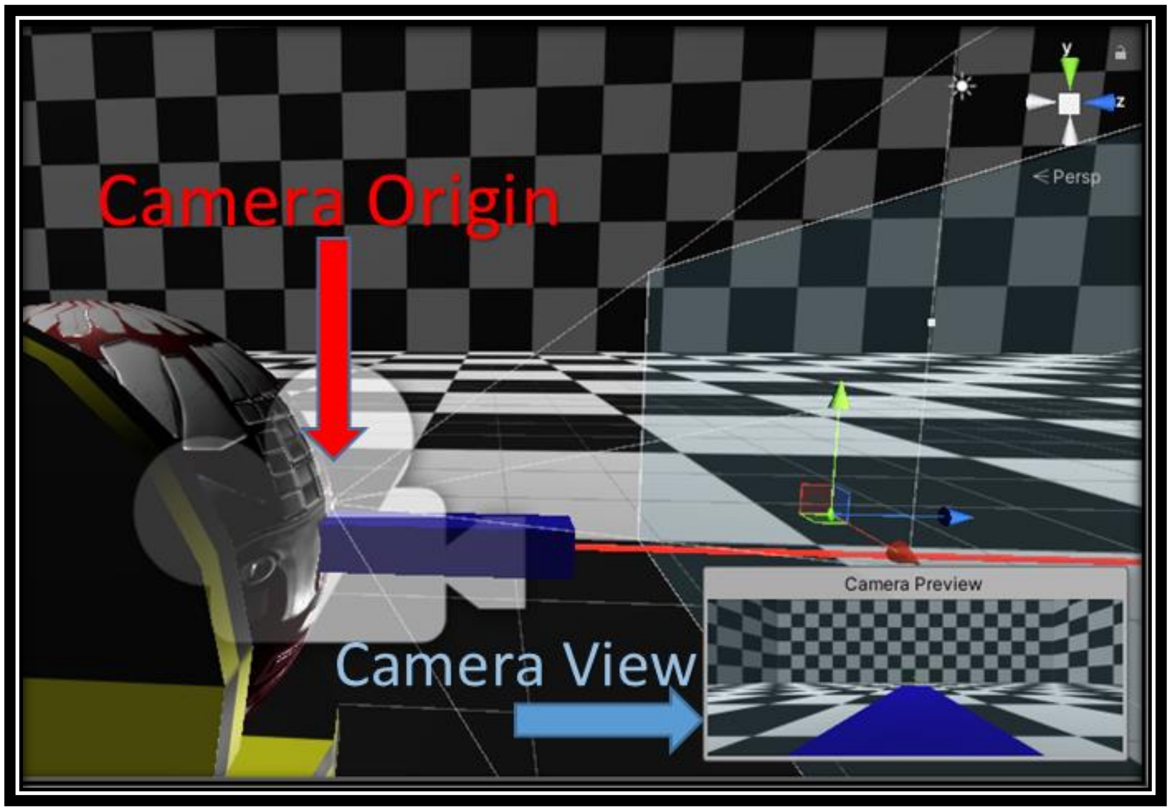


Figure 8: Side view of the camera and laser in the simulation. Showing field of vision of the camera and a preview of the camera perspective in the bottom right corner.

In Figure 8 the camera is the single point above the laser where all the white lines originate from. And since the camera and laser will have a physical volume, their line direction cannot cross and be parallel at the same time. So, for this reason, the laser will always be off by a constant amount. The other option is to angle the laser to intersect with the camera at the desired point but that would require different angles for different distances, making the laser inaccurate at long distances. But angling the laser to intercept the camera at half of the "maximum range" would yield the best result for a fixed position.

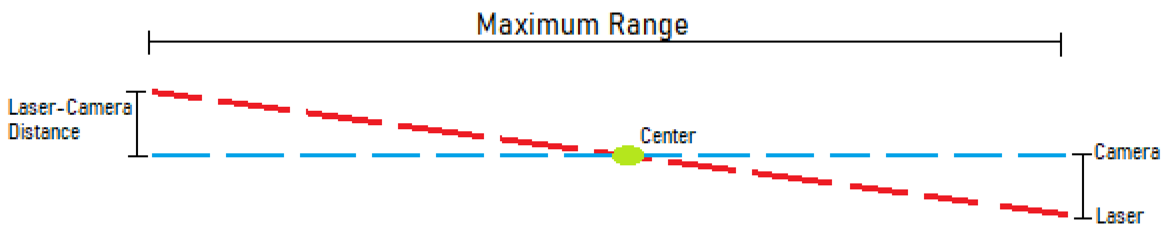


Figure 9: Diagram of angled laser line intersecting camera view line at halfway of maximum range.

As see in Figure 9, if we have the "Maximum Range" (r) and the distance between the camera and laser (d), then an angle can be calculated with the following formula: $\arctan(2d/r)$. Finding the distance between the laser and camera id trivial, but to find the "Maximum Range" we can look at the laser's range or the camera's limitations pared with the detection system to see how far it can detect the drone.

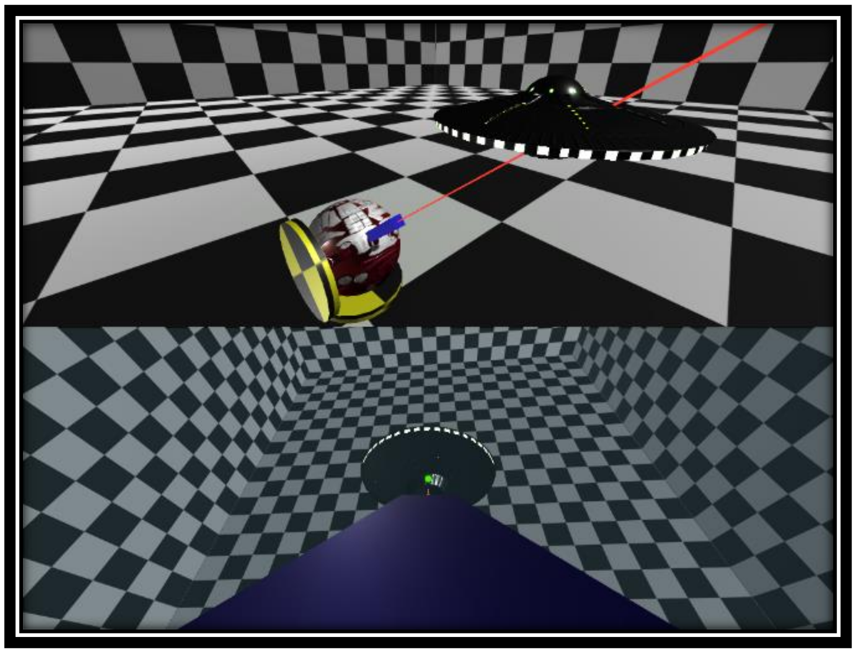


Figure 10: Screenshot of simulation in action. The bottom panel being the camera perspective and the top panel a camera following the drone. In this approach the camera and laser are parallel, and the result looks promising for small distances between the laser and the camera.

Simulating the Motors

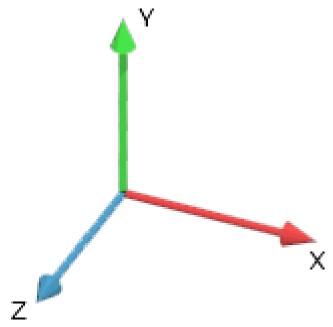


Figure 11: A three dimensional Cartesian coordinate system with the Y-axis as the vertical axis.

We can't just simulate the motors; we need to get closer to an emulation if we want to make the translation go smoother. For this we set up 2 axis of rotation "*Vertical*" and "*Horizontal*", where *Vertical* rotates around the X-axis and *Horizontal* around the Y-axis (See Figure 11).

We will use two identical motors positioned with their axis orthogonal to each other. Doing this will allow the laser to have a full 360-degree view from each axis, giving the laser the capabilities to rotate and point to any point of the spherical coordinate system. Below are pictures of the 2 representations of the motors in the simulation. The left one (Figure 12) capable of rotating around the Y-axis (green) and the one in the right (Figure 13) is capable to rotate around the X-axis (red).

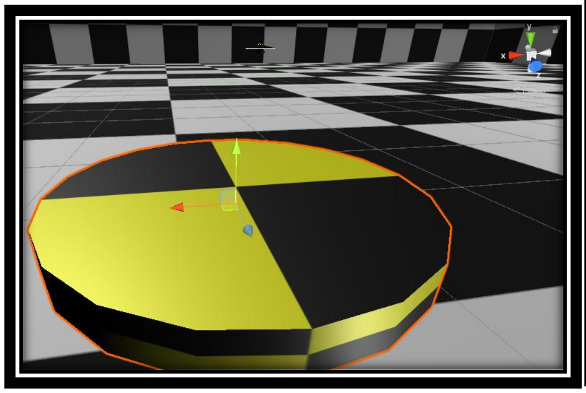


Figure 12: Motor-Y

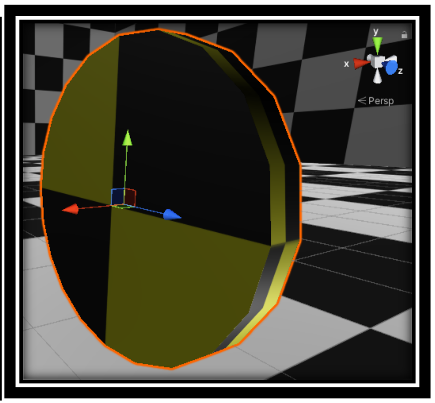


Figure 13: Motor-X

Motor-Y (*Horizontal* rotation) acts as the base of the laser and rotating it 360 degrees will give us a full horizon view. And Motor-X (*Vertical* rotation) will control the laser vertical inclination, where rotating it 90 degrees from horizontal (degree 0) will mean to look straight up in the direction of the Y-axis. Motor X and Y, and the laser are shown below in Figure 14:

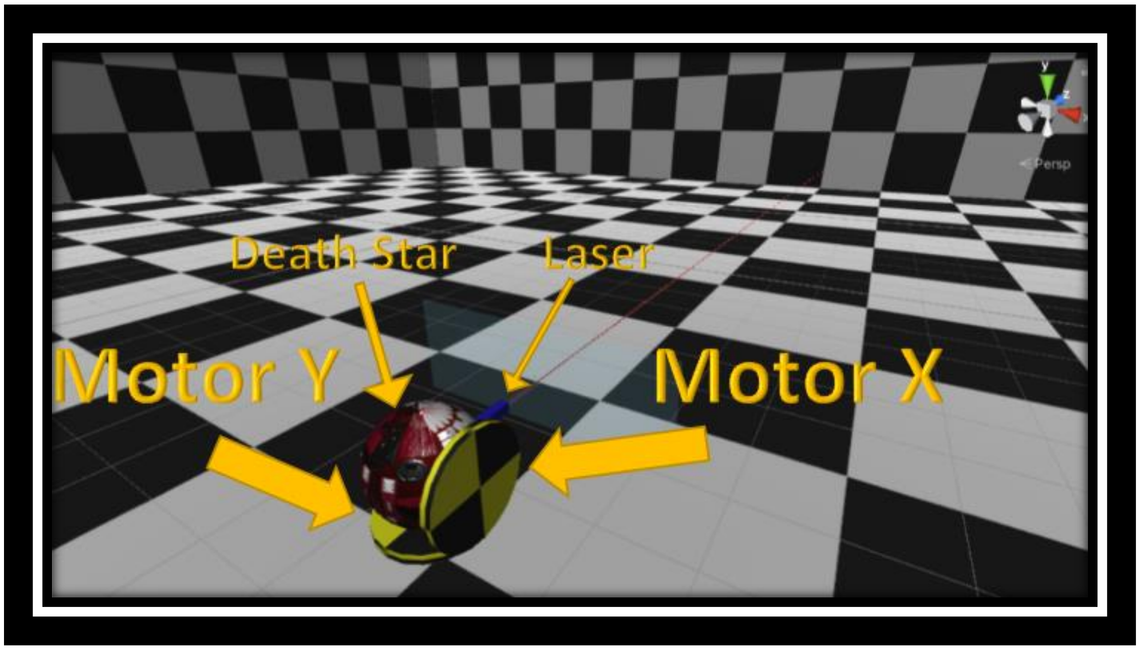


Figure 14: Screenshot of the simulated motors and laser with their corresponding labels.

Once the motors rotation is generalized, it is simple to translate it into a more interesting-looking model. Since we know the laser will be on the ground and it doesn't need to look straight down (where motor Y is located). We can translate the motors movement to any other models as long as Motor X is mounted on top of Motor Y. Or in other words, Motor Y is the parent of Motor X. Following this restrictions, we can change the esthetics of the simulation:

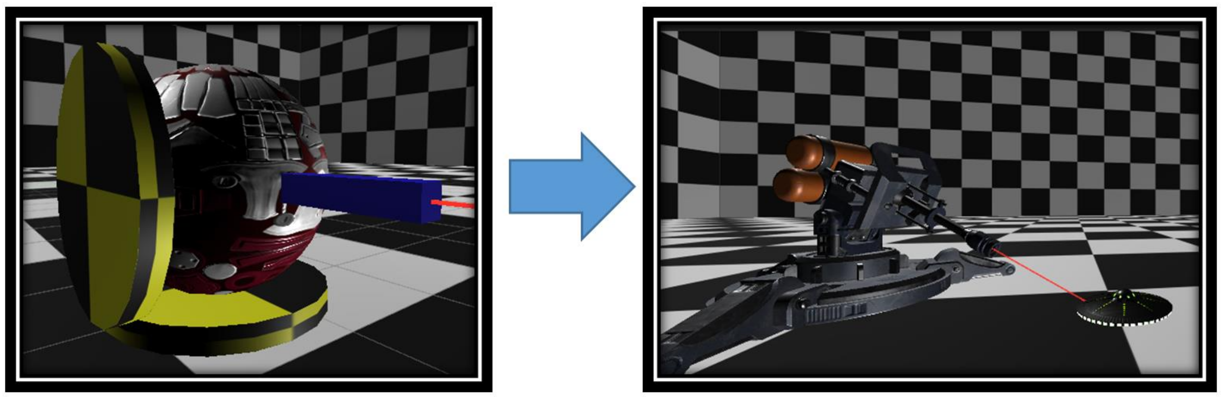


Figure 15: First model clearly showing Motor X and Y in the left. Second model with nicer graphics on the right. Both models operate with the same components and their performance is identical.

Simulating the Drone

To simulate the drone flying in the air we first grabbed a free asset from the Unity Asset Store since the drone looks and dimensions are not relevant to the simulated detection system because it won’t actually process the image of the drone to find it (more on this later). In the asset store we found a package with 4 drone models, from which we are using the UFO model below (Figure 16):

Figure 16: Top down and side view of one of the drone models used in the simulation.

After importing the model, a “Rigidbody” component was added to allow forces (like gravity) to be applied to the object. In addition, a mesh collider was added as well with the same mesh as the model to be able to accurately detect collisions (with the walls, floor and ceiling primarily).

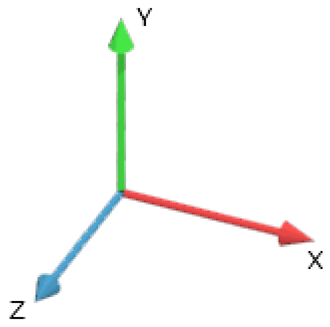


Figure 17: A three dimensional Cartesian coordinate system with the Y-axis as the vertical axis.

For the drone movement, a small script was created where it will gather 3 inputs from the user: *vertical*, *horizontal* and *forward* **input**.

- The *Vertical* input is controlled with the up and down arrows to determine if the drone will move up and down in the Y-axis, if not pressed, then vertical forces will be applied in the current frame. This will allow the drone to maintain the current height and float in the air if gravity is turned off.
- The *Horizontal* input comes from the left and right arrows, in the keyboard, and this input controls the rotation of the drone along the local up vector, which it will always be the Y-axis because the drone has restricted rotation around the X and Z axis. At this point the drone can float up and down, and it can rotate around the Y-axis while keeping a constant coordinate position in the X and Z axis.
- The *Forward* input, as its name suggests, adds a force to the drone in the direction of the current local forward direction of the drone. This means that if the drone rotates with the *horizontal* input, then the forward vector will rotate in the same direction and amount.

To confirm this controller is sufficient and that it can reach all point in R^3 (3D space) we just need to prove that the drone can move in 3 directional vectors that are linearly independent from each other. To show this we start with the forward vector, which point to where the drones' front is looking at. If we then rotate the drone with the *horizontal* input, it will produce a second vector which lies in the x-z plane. In fact, the rotation can produce infinitely many vectors in the x-z plane but we only need 4: $[F_1, -F_1, F_2, -F_2]$ where F_1 and F_2 any 2 linearly independent vectors. For demonstration we are showing to orthogonal vectors but note that they don't necessarily need to be orthogonal to each other to span R^2 (a plane), they only need to not be linearly dependent (or parallel in this context).

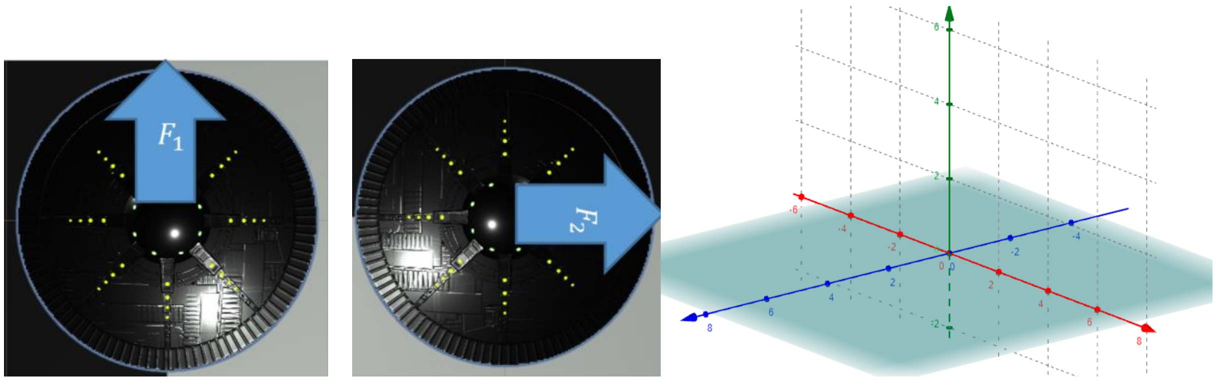


Figure 18: Top down view of a drone facing in 2 directions (F_1 and F_2). And the image farthest to the right is the resulting plane from the span of F_1 and F_2 in ground level (y = 0).

With the *Horizontal* and *Forward* input able to span R^2 (a plane), the system only needs a vector outside the plane (linearly independent from all vectors in that plane) to be able to move along the orthogonal axis to span R^3. And this is where the *Vertical* input is necessary. Although having the vector that is orthogonal to the plane is not strictly necessary to span R^3, it is convenient for avoiding unnecessary movement to reach most of the point and give a more natural feel to the controller. Using this new vertical vector, any point in R^3 is reachable and an intuitive way of thinking about this is to move the plane created by F_1 *and* F_2 and dragging it along the new vertical vector. If we then draw a copy of the plane on every place visited, the result would be a solid covering the entirety of R^3as seen in Figure 19.

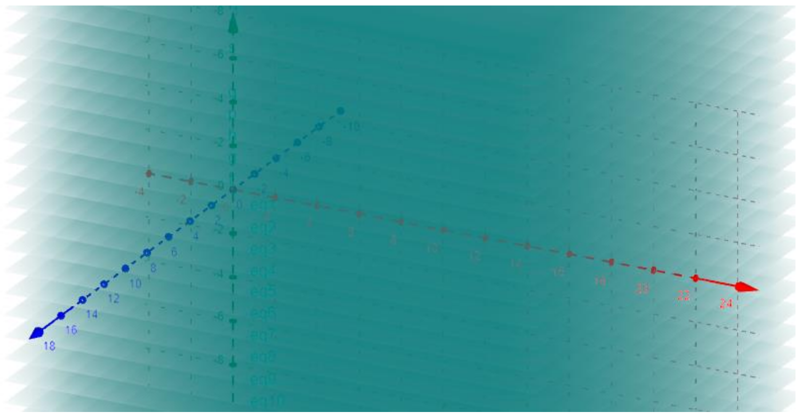
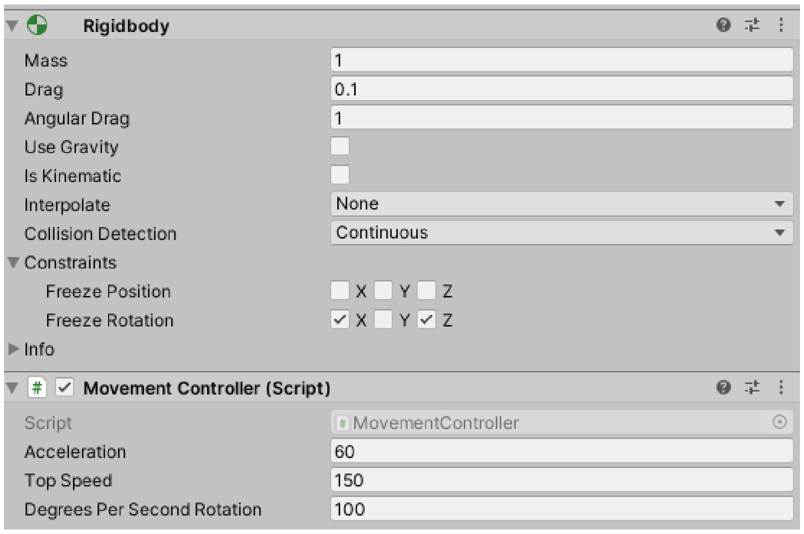


Figure 19: Visual representation of drawing multiple instances of the x-z plane with different y values.

In Unity, everything is measured in "Units" which means that it is up to the user to define what a unit of measurment is. For the Movement Controller script we set the Acceleration to 60 meters per second (134 mph) with a top speed of 150 meters per second (335 mph) which is more than twice the acceleration and speed of the fastes drone information we could find (In the next section we will give a better explanation on why we used such big values). And the Degrees per second rotation variable was set too 100 degrees just so that the drone can turn quickly. For the Rigidbody component, the key aspects to look at are the "Constraints", which we used freezed rotation on the x and z axis. This with the intention to not have the drone crash and turn itself up side down. Turning the drone up side down would complicate the navigation controls even if it is theoretically possible to still reach any point on R^3 (3d space) with the current implementation of the mvement controler.



Rigidbody	
Mass	1
Drag	0.1
Angular Drag	1
Use Gravity	☐
Is Kinematic	☐
Interpolate	None
Collision Detection	Continuous
Constraints	
Freeze Position	☐ X ☐ Y ☐ Z
Freeze Rotation	✓ X ☐ Y ✓ Z
Info	

✓ Movement Controller (Script)	
Script	MovementController
Acceleration	60
Top Speed	150
Degrees Per Second Rotation	100

Figure 20: Drone Inspector showing metadata values of each variable field

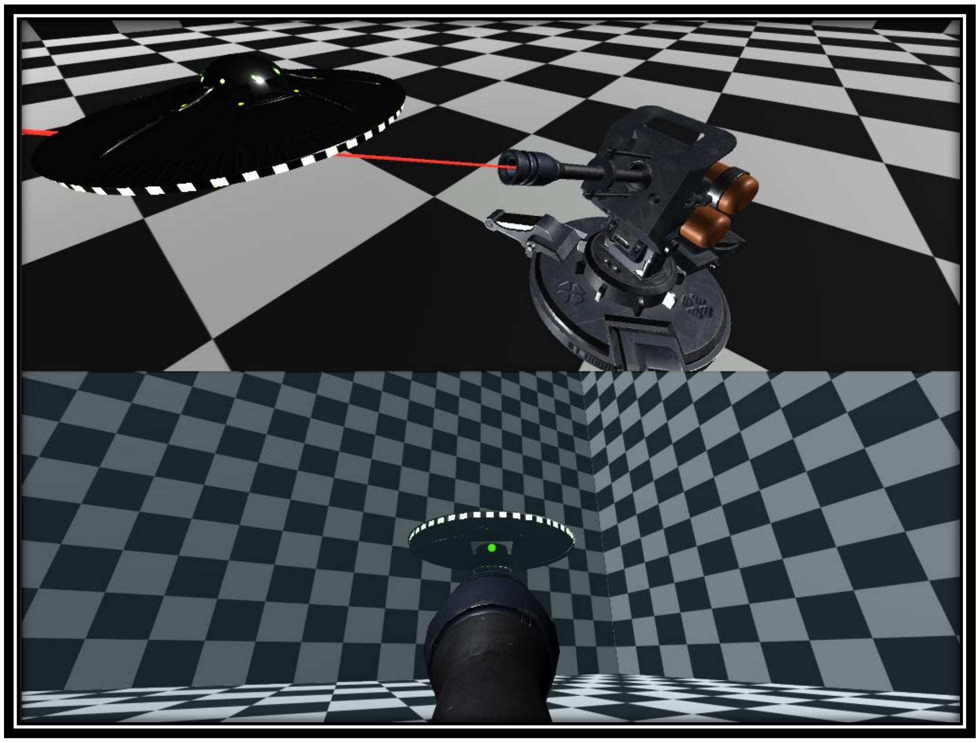


Figure 21: Screenshot of the simulation in action with the new laser model.

The Algorithm

The Base

With all of the components in place for the simulation, it is finally time to add the algorithm.

This algorithm is the one responsible for moving the motors to rotate the laser. But how to know what angle is necessary to rotate the laser to point to the drone, if the setup only contains one camera and any sense of depth is lost? As explained in section 5 (Simulating the Camera and Laser) of the Middleware, the trick is to mount the camera on top of the laser. Normally if we wanted to measure the angle needed to rotate the camera we would need to know 2 of 3 distances:

1. The distance from the camera to the object (drone/UFO).

2. The projection of the distance from the camera to the object onto the line formed in the direction of the camera view, assuming the camera is at $(0, 0)$.

3. The distance from the object to the closest point in the line described in "2."

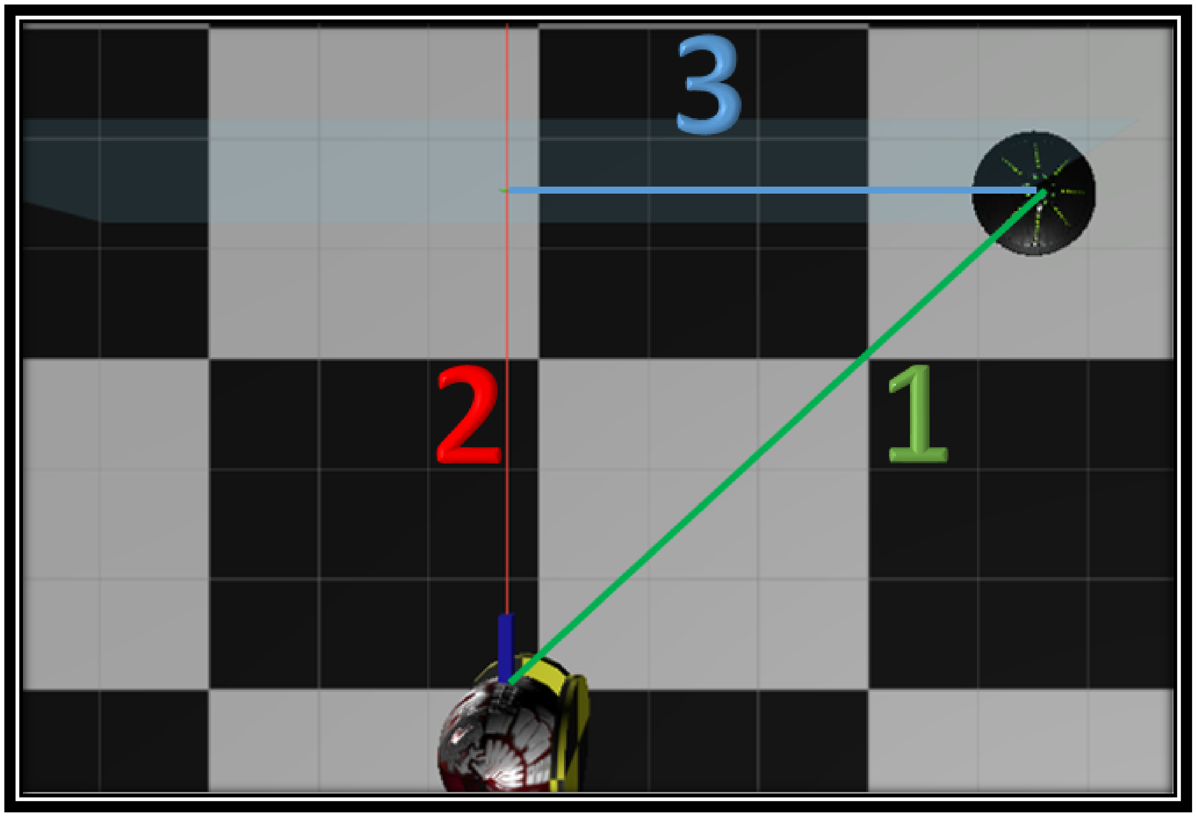


Figure 22: Diagram showing the 3 unknown distances that can help find the necessary angle for rotation.

From Figure 22 above, none of the three unknown distance can be found. Especially if we do not know the real size of the object, which we shouldn't know because that would mean that we know the drone beforehand and the algorithm would need to be updated with every new drone with different dimensions.

Even if we can't find the ***physical*** distances between the 3 points, it is possible to find our angle with another method. The unknown distance number 3 can be written in terms of pixels. If we assume the center of the screen is $(0, 0)$ and since we know the pixel coordinates of the drone in the screen thanks to the detection algorithm, we can conclude that the x component of the drone's pixel coordinate is distance 3.

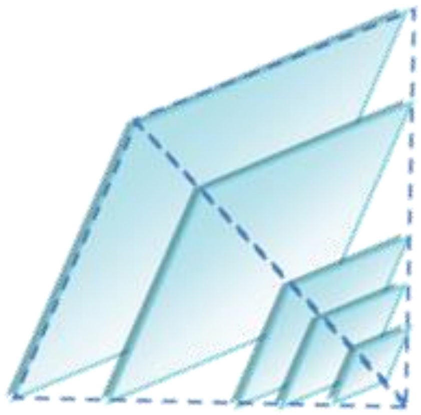


Figure 23: Field of vision of the camera.

As explained in section 5, we can see in figure 23 that the physical plane that represents the camera screen in the real world scales with depth while keeping a constant pixel size in the image produced. We used this information to calculate the degrees per pixel. If we have a camera with a horizontal field of view of 60 degrees that produces a rectangular image of 1024 by 576 pixels. Then we can get the angle difference per pixel with the following formulas (note that the screen size depends on the field of view, if it is a horizontal or vertical FOV):

$$Degrees\ Per\ Pixel\ = \frac{Field\ Of\ View}{ScreenSize}$$

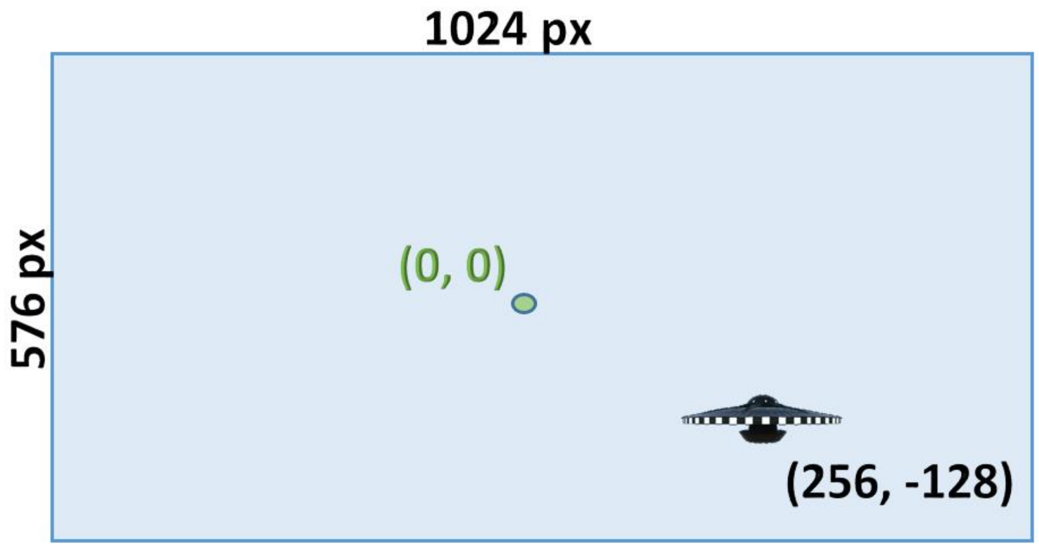


Figure 24: Diagram of the drone in a picture example with dimensions of 1024 x 576 pixels (aspect ratio of 16:9). *Where the center of the picture is the origin (0, 0) and the drone is at the pixel coordinates of (256px, -128px)*.

To calculate the horizontal and vertical angle difference of the drone in Figure 24, we can calculate the degrees per pixel using the $Degrees\ Per\ Pixel$ formula shown earlier:

$$Degrees\ Per\ Pixel = \frac{60\ degrees}{1024\ px} = \frac{15}{256}\frac{degrees}{piexel}$$

Then using this information, we can calculate horizontal angle difference:

$$\theta_x = 256\ pixels * \frac{15}{256}\frac{degrees}{piexel} = 15\ degrees$$

And the vertical angle difference:

$$\theta_y = -128\ pixels * \frac{15}{256}\frac{degrees}{piexel} = -\frac{15}{2}\ degrees = -7.5\ degrees$$

Now that we have the horizontal and vertical angle differences (15 and -7.5 respectively), we can rotate the motors with their corresponded desired angle.

As explained in section 6, Motor-Y rotates around the Y-axis and is responsible for horizontal movement and Motor-X rotates around the X-axis and is responsible for vertical movement. Therefore Motor-Y will need to be turned by 15 degrees (to the right) and Motor-X by -7.5 degrees (downwards). This is assuming that any positive angle means rotation upwards or to the right, and a negative angle means to rotate downwards or to the left.

The effectiveness of this method depends on how many frames can be processed per second. If the frame rate is too low then the result would show a laser turning like a clock in the direction of the drone but most of the time not pointing at the drone.

The laser will turn once a frame. After the turning of the motors ends, the drone will be in another position. With this approach, even if the framerate is increased there's always going to some lag of the laser drifting behind the drone when the drone is moving at high speeds. Shown below is a diagram of a cycle of the entire system:

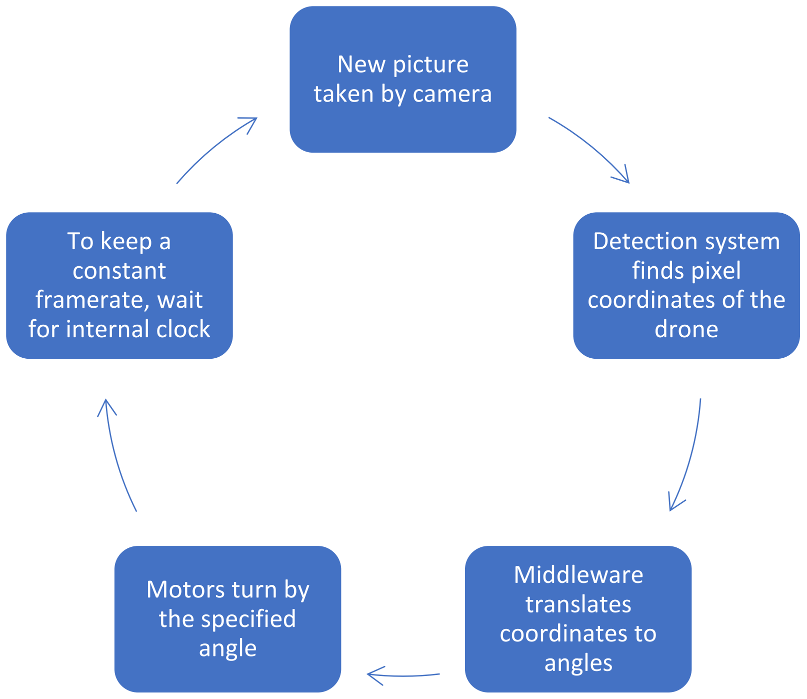

New picture taken by camera
Detection system finds pixel coordinates of the drone
Middleware translates coordinates to angles
Motors turn by the specified angle
To keep a constant framerate, wait for internal clock

Figure 25: One cycle of ADLTS

Velocity Prediction

After some testing in the simulation, we came to the conclusion that the “Base” algorithm is not good enough for tracking fast moving objects. Even with a fixed frame rate of 20 frames per second, the laser lags behind the drone. To solve this problem, we proposed a new complimentary method to calculate the current velocity of the object and compensate in the next frame. The formula for the velocity of a moving object is:

$$v = \frac{d}{t}$$

Where v stands for velocity, d for distance and t for time. From the equation, we can get the time t if we know the frame rate of the system. For this we will assume the frame rate is constant (20 fps was used for the simulation) thanks to an internal clock that will halt the program until a new cycle is ready to start.

The distance can be calculated by subtracting 2 consecutive points given by the detection system. And the time can be calculated with the framerate. If the frame rate is 20 frames per second then $t = \frac{1\,frame}{20\frac{frames}{sec}} = 0.05\ sec$ per frame. But if the frame rate is constant, it is not necessary to calculate the real velocity in terms of seconds but instead, assuming the frame rate is constant, we can refer to the time passed in a frame to a “unit of time”. This will make the math cleaner since each frame will last exactly “one unit of time”. And given that on each cycle or frame the motors move to the desired target position, it is not necessary to subtract the current target position with the previous one because by the time the new target position is given, the camera will have moved its center to the previous position. And because the center of the camera is the origin (0, 0), the new coordinates are the subtraction of the previous point from the new one. In other words, if no adjustment is made, then each new point given by the detection system is equivalent to the velocity vector of the drone on the screen.

Let’s consider the following scenario where the camera is not moving and the detection system sends this 5 coordinates over 5 cycles/frames as seen in Figure 25:

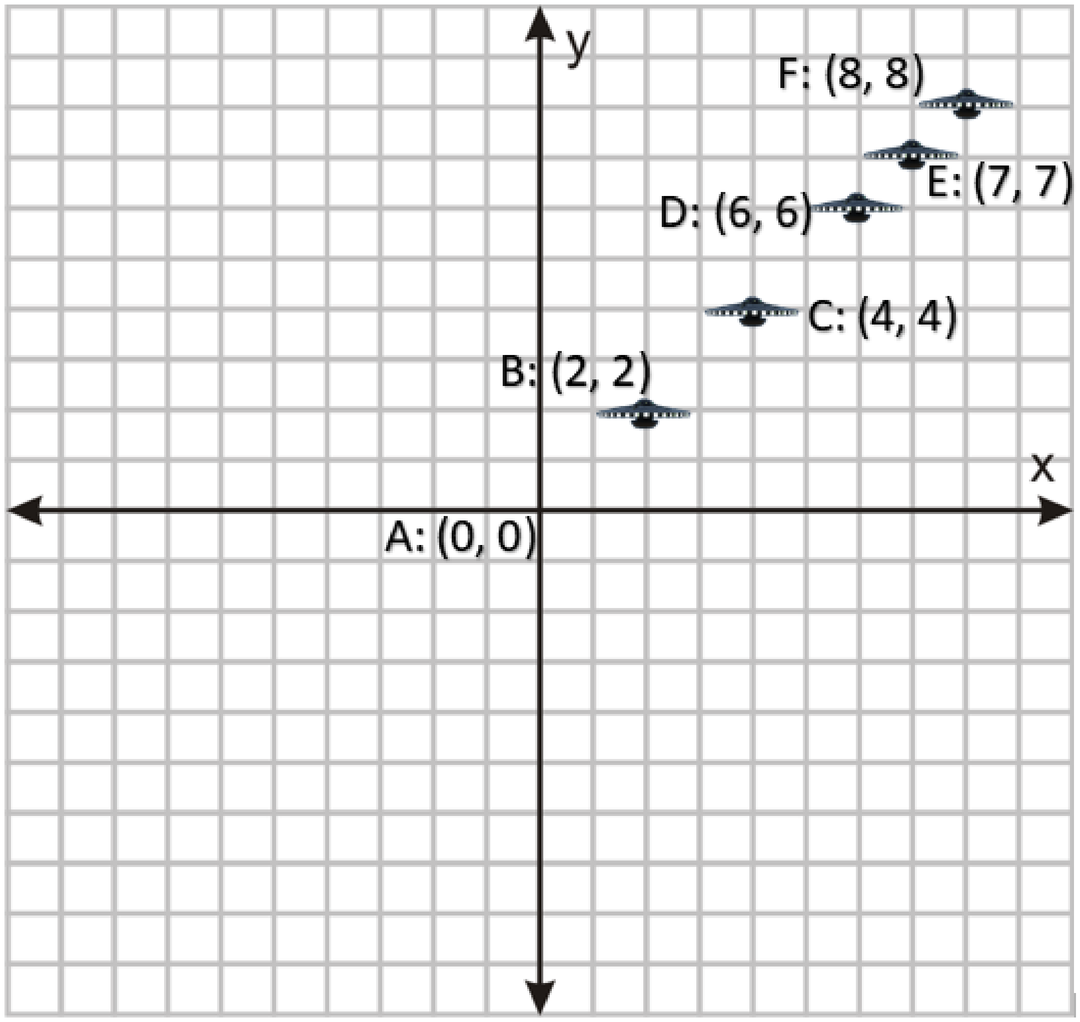


Figure 25: Example scenario where the camera is stationary, and the detection system sends 5 coordinates over a time span of 5 cycles. Labeled B through F in that order.

In the first frame the system receives the first coordinate "B" of (2, 2). Since it is the first point, we can't approximate any velocity because the drone could stationary or moving in any direction. If the system then moves the origin to the first coordinate B, then once the detection system sends the next coordinate, it will be with base of the new origin position. Therefore the coordinate point C of (4, 4) will be relative to coordinate point B as the origin, giving a new coordinate C of (2, 2), which is equivalent to subtracting the two points:

$$\vec{V}_C = C - B = (4,4) - (2,2) = (4x + 4y) - (2x + 2y) = 2x + 2y = (2,2)$$

With this in mind, the new coordinate is the change in distance over one cycle, or in other words the velocity in terms of pixels per cycle. If we then calculate the velocity vector for the following points:

$$\vec{V}_D = D - C = (6,6) - (4,4) = (6x + 6y) - (4x + 4y) = 2x + 2y$$

$$\vec{V}_E = E - D = (7x + 7y) - (6x + 6y) = x + y$$

$$\vec{V}_F = F - E = (8x + 8y) - (7x + 7y) = x + y$$

It is important to note that with the base algorithm, the center camera will always be off from the drone center position by the drone's velocity vector length.

An intuitive solution to solve the velocity gap is to calculate the velocity at each cycle (assuming no frame loss or drone out of the picture). For this, the velocity will be initialized to 0x + 0y and the velocity $\vec{V}_{new}$ of each new frame will be the sum of the target position $\vec{P}$ and the previous velocity $\vec{V}_{prev}$.

$$\vec{V}_{new} = \vec{P} + \vec{V}_{prev}$$

Then using the new formula for the velocity and adding the new velocity V_{new} to the target position P as the new target position to rotate the laser:

Current Cycle #	Drone Position	Base Algorithm		Base Algorithm with Velocity Compensation		
		Camera Center	Distance to next Point	Camera Center	Distance to next Point	Velocity calculated
1	B = (2, 2)	(0, 0)	$2\hat{x} + 2\hat{y}$	(0, 0)	$2\hat{x} + 2\hat{y}$	$0\hat{x} + 0\hat{y}$
2	C = (4, 4)	(2, 2)	$2\hat{x} + 2\hat{y}$	(2, 2)	$2\hat{x} + 2\hat{y}$	$2\hat{x} + 2\hat{y}$
3	D = (6, 6)	(4, 4)	$2\hat{x} + 2\hat{y}$	(6, 6)	$0\hat{x} + 0\hat{y}$	$2\hat{x} + 2\hat{y}$
4	E = (7, 7)	(6, 6)	$\hat{x} + \hat{y}$	(8, 8)	$-\hat{x} - \hat{y}$	$\hat{x} + \hat{y}$
5	F = (8, 8)	(7, 7)	$\hat{x} + \hat{y}$	(8, 8)	$0\hat{x} + 0\hat{y}$	$\hat{x} + \hat{y}$

Where the distance to the next point d is defined as the directional distance from the center of the camera $\vec{P}_C$ and the drone $\vec{P}_D$:

$$\vec{d} = \vec{P}_D - \vec{P}_C$$

And because mobbing the camera's center shifts the origin giving coordinates of the drone local to the new center, we can redefine our velocity formula for this example where the coordinate system stays constant:

$$\vec{V}_{new} = \vec{d} + \vec{V}_{prev}$$

And the camera position for the next cycle $\vec{P}_{C\_new}$ when accounting for velocity can be calculated by adding the current drone position $\vec{P}_D$ and the current calculated velocity $\vec{V}_{new}$

$$\vec{P}_{C\_new} = \vec{P}_D + \vec{V}_{new}$$

Walkthrough of the first 3 cycles:

1. At cycle 1 the center of the camera is at position A from Figure 25. At this time the detection system sends the coordinates of the drone currently at position B. Since the velocity was initialized as $0\hat{x} + 0\hat{y}$, both: the base algorithm with and without velocity will turn the motors to coordinates at B.
2. At cycle 2, both algorithms have the center of the camera at position B ant the detection system sends the coordinates of the drone as position C. This will make the base algorithm to turn to positon C and the base algorithm with velocity to turn to C + V, where V is the current calculation of the velocity, which is $2\hat{x} + 2\hat{y}$.
3. At cycle 3, the drone is at position D and the base algorithm is at C giving an error of D – C ($2\hat{x} + 2\hat{y}$) and the base algorithm with velocity is at position D, giving no error of D – D (
$0\hat{x} + 0\hat{y}$).

The distance from the center of the camera to the drone at any given cycle is the instant error or lag in the system. It is important to note that the base algorithm always have an error equal to the instantaneous velocity of the drone. While the base algorithm with velocity compensation is off by a factor of the instantaneous acceleration.

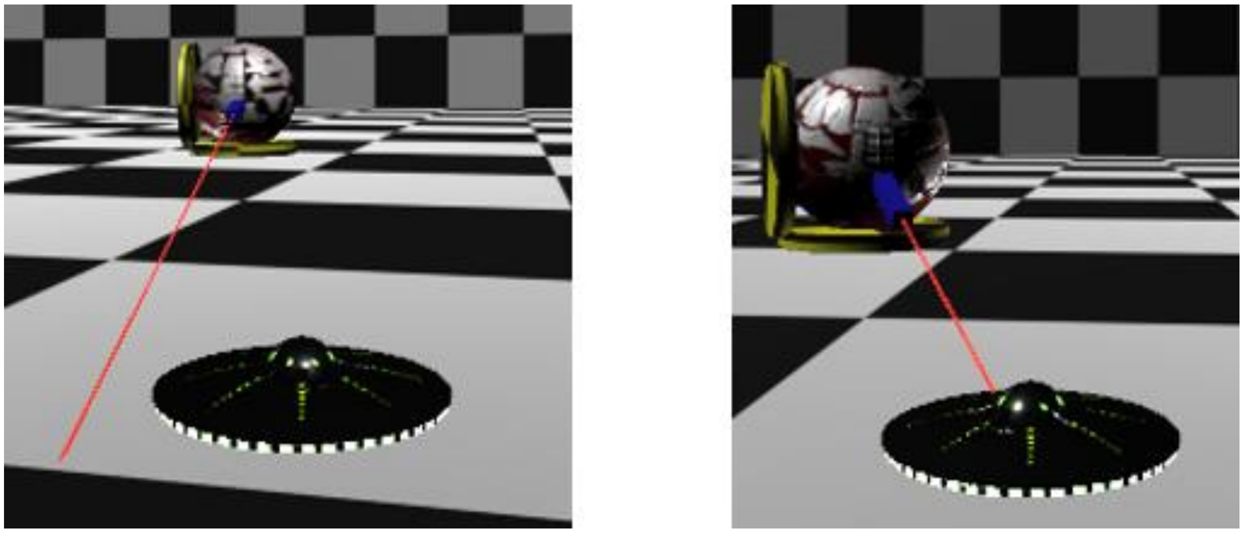


Figure 26: Left Panel with the Base Algorithm lagging behind drone. Right Panel with the Base Algorithm plus velocity prediction accurately following the drone regardless of speed.

Frame loss

The simulation is not completely representative of the real setup because the current state of the simulation is running in a perfect scenario. But as we all know, nothing is perfect and errors can happen. One common error that the middleware can encounter is frame loss. Or more specifically a cycle on which the drone is not detected in the picture, either by motion blur, obstruction, or any other external reason.

We implemented frame loss in the simulation by simply adding a random chance to not find the coordinates of the drone. After some observation and testing, the result was that the laser stops in place where and snaps back to the drone. The error only lasts for as many frames that are lost consecutively and the following 2 frames.

The error is very small but it can still be solved by dividing the new target position P by the number of cycles C since the last detection of the drone when calculating the velocity V_{new}:

$$\vec{V}_{new} = \frac{\vec{P}}{C} + \vec{V}_{prev}$$

Noise

Another scenario that the simulation hasn't explored is the introduction of noise in the system.

And by noise we refer to small deviations in the measurements. For example the drone coordinates are supposed to be the center of the drone in the image, but getting the center every time could be difficult for the detection system, in reality the middleware will be getting the coordinates close to the center of the drone but not perfectly every time like the simulation.

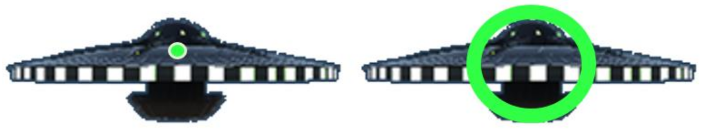


Figure 27: Dot accuracy of simulation for finding the center of the drone on the left. And the drone on the right has a circle representing the where the detection system could possibly mark as the center of the drone (radius of the circle depends on the detection system level of accuracy).

To imitate this behavior in the simulation, a noise vector V_{Noise} with a random direction and smaller than a predetermined length L can be added to the target position vector P to create a new target position with noise P_{Noise}:

$$\vec{P}_{Noise} = \vec{P} + \vec{V}_{Noise}\ , where\ |\vec{V}_{Noise}| < L$$

After adding noise to the system and looking at the results in the simulation by comparing the base algorithm with and without velocity predictions, it was observed that the base algorithm with velocity predictions amplified the noise in the system, resulting in a bigger chaotic shake of the laser. The reason of this outcome is due the way the velocity is calculated and how sudden changes of velocity can cause the system to shake out of control.

To minimize the sudden changes of direction in the system, the Noise Dampening method was proposed. The Noise Dampening method is in charge of restricting the rate of change of the drone's direction from the camera perspective while keeping a component of the desired movement.

One of the base scenarios is if the drone is stationary with no velocity, then it can potentially have any direction in the next time interval. Therefore, if the drone has a velocity close to 0, then no restrictions will be made for the next frame. But if the drone is already moving, the current velocity vector will be used to determine the next velocity vector.

Because of inertia and the maximum amount of thrust the drone can deliver, the drone can't change direction as fast. Since we do not know the weight nor the maximum acceleration of the drone, then we will have to add an adjustable variable to dictate the angle difference range θ_{Range} of the possible new drone directions.

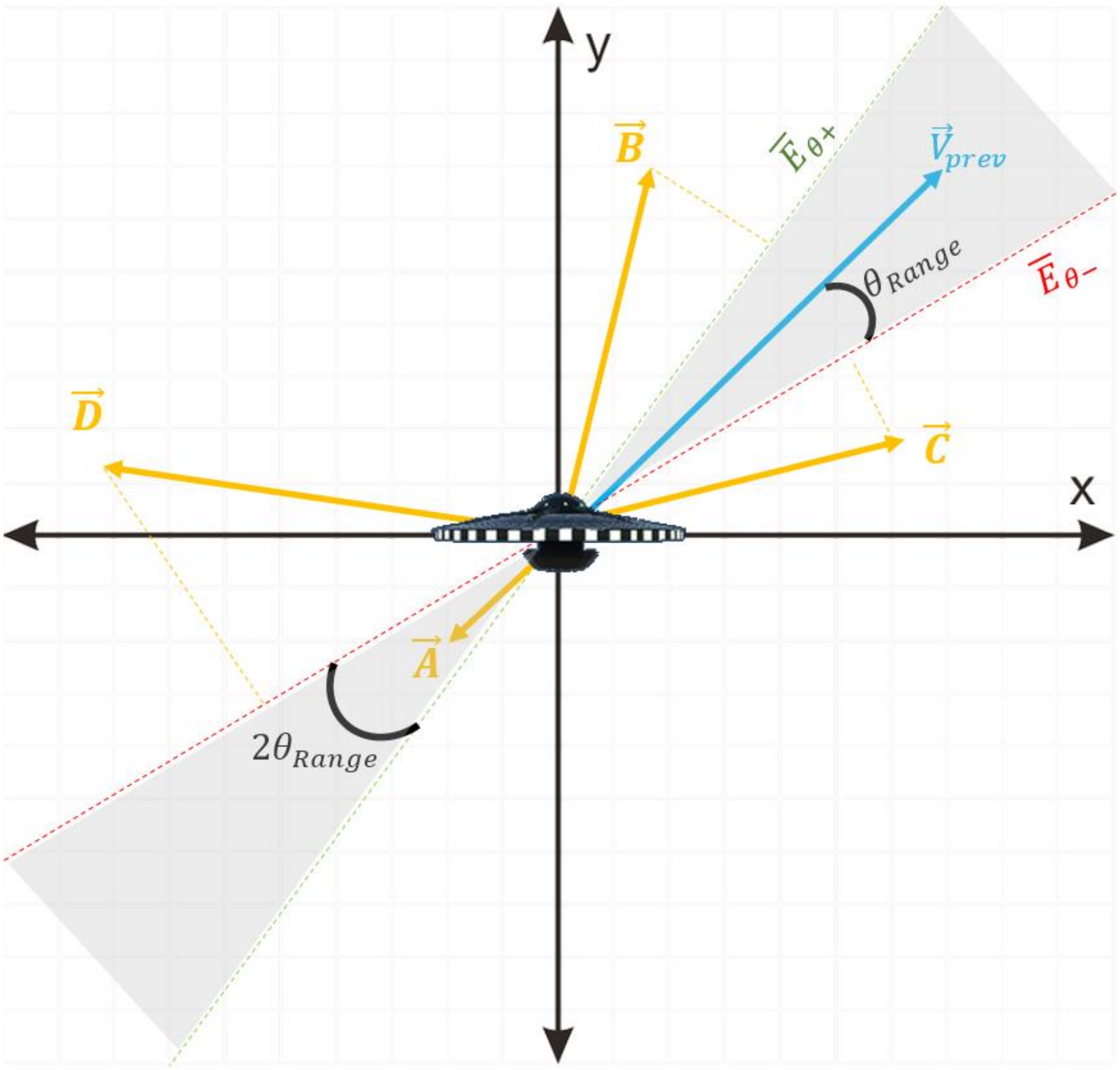


Figure 28: Example diagram showing components of the noise dampening method: counterclockwise edge $\overline{E}_{\theta+}$ of the field of possible directions; the clockwise edge $\overline{E}_{\theta-}$ of the field of possible directions; the velocity of the drone in the previous frame $\vec{V}_{prev}$; Angle θ_{Range} between the previous vector velocity $\vec{V}_{prev}$ and either edge of the field of possible directions ($\overline{E}_{\theta+}$, $\overline{E}_{\theta-}$) or shaded area; And finally 4 example vector velocities $\{\vec{A}, \vec{B}, \vec{C}, \vec{D}\}$ that the detection system can send .

The aim of the noise dampening method is to identify and modify incoming coordinates from the detection system that would result in a velocity with a direction outside the area formed by $\overline{E}_{\theta+}$ and $\overline{E}_{\theta-}$ that encloses $\vec{V}_{prev}$ and $-\vec{V}_{prev}$.

We can calculate a vector in the direction of the counterclockwise line $\overline{E}_{\theta+}$ by rotating counterclockwise the previous velocity vector $\vec{V}_{prev}$ by θ_{Range} degrees using the matrix rotation formula:

$$\vec{E}_{\theta+} = \left(\vec{V}_{prev}\hat{x} * Cos(\theta_{Range}) - \vec{V}_{prev}\hat{y} * Sin(\theta_{Range})\right)\hat{x} + \left(\vec{V}_{prev}\hat{x} * Sin(\theta_{Range}) - \vec{V}_{prev}\hat{y} * Cos(\theta_{Range})\right)\hat{y}$$

And we can calculate $\vec{E}_{\theta-}$ in a similar way:

$$\vec{E}_{\theta-} = \left(\vec{V}_{prev}\hat{x} * Cos(\theta_{Range}) + \vec{V}_{prev}\hat{y} * Sin(\theta_{Range})\right)\hat{x} + \left(-\vec{V}_{prev}\hat{x} * Sin(\theta_{Range}) + \vec{V}_{prev}\hat{y} * Cos(\theta_{Range})\right)\hat{y}$$

To check if we need to modify the incoming vector, we first need to verify that the new vector falls in the shaded area. Falling in the shaded area is equivalent to measuring the angle Φ between the previous velocity $\vec{V}_{prev}$ and the new one $\vec{V}_{\text{new}},$ for example $\{\vec{A}, \vec{B}, \vec{C}, \vec{D}\}$ and seeing f it's smaller than θ_{Range}. Angle Φ can be calculated by taking the dot product of both normalized velocities:

$$Cos(\Phi) = \frac{\vec{V}_{prev} \cdot \vec{V}_{\text{new}}}{\left|\vec{V}_{prev}\right| * \left|\vec{V}_{\text{new}}\right|}$$

If $Cos(\Phi)$ is greater than 0, then Φ is less than 90 degrees and $\vec{V}_{\text{new}}$ is closer to $\vec{V}_{prev}$ compared to $-\vec{V}_{prev}$. And in the contrary, if $Cos(\Phi)$ is less than 0, then Φ is greater than 90 degrees and $\vec{V}_{\text{new}}$ is closer to $-\vec{V}_{prev}$ compared to $\vec{V}_{prev}$ and if this is the case, then $\vec{E}_{\theta+}$ and $\vec{E}_{\theta-}$ need to be changed by a factor of -1 ($-\vec{E}_{\theta+}$ and$-\vec{E}_{\theta-}$) to reflect that $\vec{V}_{\text{new}}$ is in the other side of the coordinate system.

After analyzing where $\vec{V}_{\text{new}}$ is located with respect to $\vec{V}_{prev}$, we can modify $\vec{V}_{\text{new}}$ by replacing it with its own projection onto the closest edge of the shaded area. To figure out which edge is closer to $\vec{V}_{\text{new}}$, we can compare Cosine of the angle between both edges ($\vec{E}_{\theta+}$ and $\vec{E}_{\theta-}$) and the one with the biggest value will have the smallest angle. The cosine of the angle of each edge can be calculated as followed:

$$Cos(\theta_{+}) = \frac{\vec{E}_{\theta+} \cdot \vec{V}_{\text{new}}}{|\vec{E}_{\theta+}| * |\vec{V}_{\text{new}}|}$$

$$Cos(\theta_{\_}) = \frac{\vec{E}_{\theta-} \cdot \vec{V}_{\text{new}}}{|\vec{E}_{\theta-}| * |\vec{V}_{\text{new}}|}$$

Note that if $Cos(\Phi)$ is less than 0, then $\vec{E}_{\theta+}$ and $\vec{E}_{\theta-}$ need to be changed by a factor of -1 ($-\vec{E}_{\theta+}$ and$-\vec{E}_{\theta-}$) in the above formula.

After comparing which edge is closest to $\vec{V}_{\text{new}}$, we can proceed to project $\vec{V}_{\text{new}}$ onto said edge:

$$If\left(Cos(\theta_{+}) > Cos(\theta_{\_})\right)\ then:$$

$$\vec{V}_{\text{new}} = proj_{\vec{V}_{\text{new}}}\vec{E}_{\theta+} = \frac{\vec{V}_{\text{new}} \cdot \vec{E}_{\theta+}}{|\vec{V}_{\text{new}}|}\left(\frac{\vec{V}_{\text{new}}}{|\vec{V}_{\text{new}}|}\right)$$

$Else$:

$$\vec{V}_{\text{new}} = proj_{\vec{V}_{\text{new}}} \vec{E}_{\theta-} = \frac{\vec{V}_{\text{new}} \cdot \vec{E}_{\theta-}}{|\vec{V}_{\text{new}}|} \left(\frac{\vec{V}_{\text{new}}}{|\vec{V}_{\text{new}}|} \right)$$

To clarify with some examples, let's explore some scenarios having the initial conditions of Figure 28:

- Scenario 1: $\vec{V}_{\text{new}} = \vec{A}$, then after calculating Φ and θ_{Range} we find out that $\Phi < \theta_{Range}$ and thus $\vec{A}$ falls in the shaded region. Therefore no modification on $\vec{V}_{\text{new}}$ is necessary and no farther calculations need to be done.
- Scenario 2: $\vec{V}_{\text{new}} = \vec{B}$, then after calculating Φ and θ_{Range} we find out that $\Phi > \theta_{Range}$ and thus $\vec{B}$ falls outside the shaded region. Then, because $Cos(\Phi) > 0$ we know that $\vec{B}$ is closer to $\vec{V}_{prev}$. However, we still don't know if $\vec{B}$ is closer to $\vec{E}_{\theta+}$ or $\vec{E}_{\theta-}$, hence we need to compare their angles. As a result we find that $Cos(\theta_{+}) > Cos(\theta_{-})$ which means that $\vec{B}$ is closer to $\vec{E}_{\theta+}$ and our new $\vec{V}_{\text{new}} = proj_{\vec{V}_{\text{new}}} \vec{E}_{\theta+}$.
- Scenario 3: $\vec{V}_{\text{new}} = \vec{C}$, then after calculating Φ and θ_{Range} we find out that $\Phi > \theta_{Range}$ and thus $\vec{C}$ falls outside the shaded region. Then, because $Cos(\Phi) > 0$ we know that $\vec{C}$ is closer to $\vec{V}_{prev}$. However, we still don't know if $\vec{C}$ is closer to $\vec{E}_{\theta+}$ or $\vec{E}_{\theta-}$, hence we need to compare their angles. As a result we find that $Cos(\theta_{+}) < Cos(\theta_{-})$ which means that $\vec{C}$ is closer to $\vec{E}_{\theta-}$ and our new $\vec{V}_{\text{new}} = proj_{\vec{V}_{\text{new}}} \vec{E}_{\theta-}$.

- Scenario 4: $\vec{V}_{\text{new}} = \vec{D}$, then after calculating Φ and θ_{Range} we find out that $\Phi > \theta_{Range}$ and thus $\vec{D}$ falls outside the shaded region. Then, because $Cos(\Phi) < 0$ we know that $\vec{D}$ is closer to $-\vec{V}_{prev}$, and we need to multiply $\vec{E}_{\theta+}$ and $\vec{E}_{\theta-}$ by a factor of -1, giving us $-\vec{E}_{\theta+}$ and $-\vec{E}_{\theta-}$. However, we still don't know if $\vec{D}$ is closer to $-\vec{E}_{\theta+}$ or $-\vec{E}_{\theta-}$, hence we need to compare their angles. As a result we find that $Cos(\theta_{+}) < Cos(\theta_{-})$ which means that $\vec{C}$ is closer to $-\vec{E}_{\theta-}$ and our new $\vec{V}_{\text{new}} = proj_{\vec{V}_{\text{new}}} -\vec{E}_{\theta-}$.

The results of testing the noise dampening method on the simulation using a small θ_{Range} showed a slight decrease of movement perpendicular to the drone's current direction, as expected, while not restricting the movement when changing direction. However, when tested with a large θ_{Range}, no difference in the laser behavior can be detected, must likely due to all incoming $\vec{V}_{\text{new}}$ falling inside the shaded area.

Conclusion, the noise dampening method didn't improve the algorithm by much in the simulation. We will keep it as an optional feature in case it comes useful on the physical setup.

We plan to explore more options to minimize noise in the system. One of them include modifying $\vec{V}_{\text{new}}$ to only reflect reasonable changes of velocity depending on the current velocity and the maximum acceleration of the drone.

Measuring Error

One of the purposes of creating the simulation was to test the algorithm in a controlled environment where we have perfect motors and a perfect detection system. Then after validating that the algorithm was working on perfect conditions, we tried to make it more realistic by introducing frame loss and noise in the system to identify errors or weaknesses in the system early on. But how can there be improvement if there is no method to measure error? In other words, we can't improve our project if we don't have a reliable way of testing its performance.

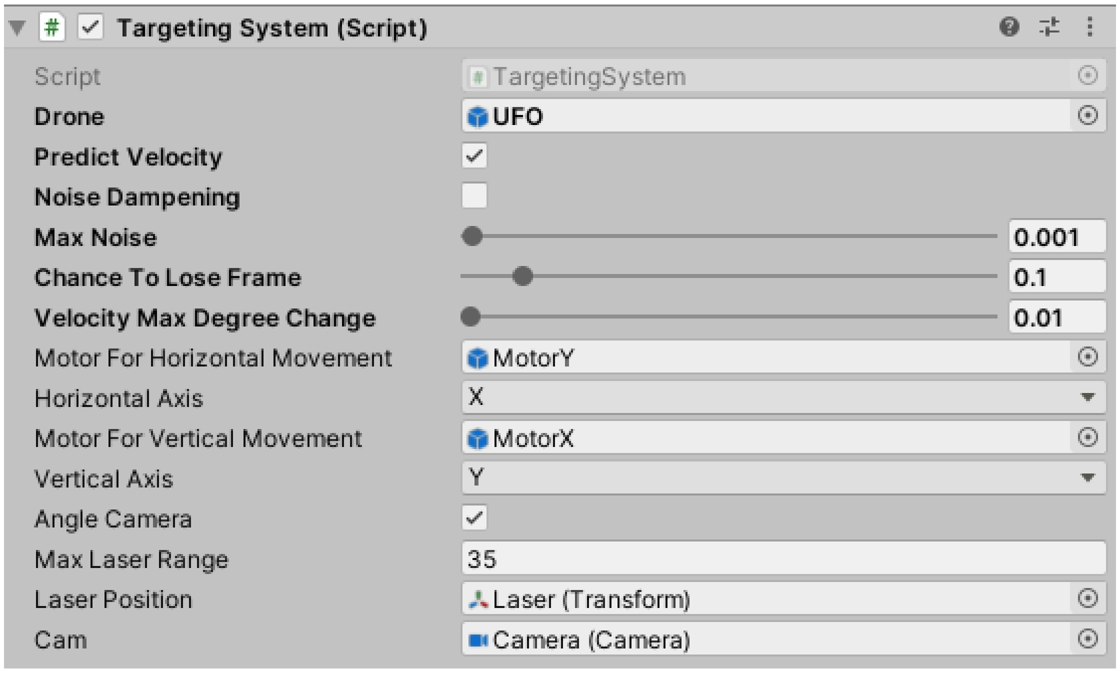
We implemented 2 measurements of error:

- One is the hit registration, which is the percentage of frames where the drone is in contact with the laser. We decided to check the drone for collisions because the drone won't have as many collisions as the laser and thus improve performance. And because we have a fixed framerate, we can simply count the number of frames when the drone and laser are touching and calculate the percentage based the time passed since the initial count. We display on the screen the hit registration for 1 and 10 seconds. Note that the measurements are not updated by accumulating them, but they get replaced after each time interval.
- The second measurement for error is the distance offset, which calculates how far is the drone to the closest point on the line formed by the laser. To calculate the distance from a point P to a line $L = \vec{r}(t) = Q + t\vec{u}$ in 3D space, we can calculate the distance from P to L with the following formula:

$$d(P, L) = \frac{\left|\left(\overrightarrow{PQ}\right) \times \vec{u}\right|}{|\vec{u}|}$$

With the distance formula, we calculate the distance between the drone and the laser at each frame similarly to the hit registration error measurement and take the average after the desired time interval, for our current implementation, we display the average distance over 1 and 10 seconds.

With these new measurements we tested the base algorithm with and without velocity predictions at different velocities and turning maneuvers; different probabilities of frame loss and different levels of noise in the system with and without the noise dampening method. As seen in Figure 29 for an example scenario:



Targeting System (Script)

Script	TargetingSystem
Drone	UFO
Predict Velocity	✓
Noise Dampening	
Max Noise	0.001
Chance To Lose Frame	0.1
Velocity Max Degree Change	0.01
Motor For Horizontal Movement	MotorY
Horizontal Axis	X
Motor For Vertical Movement	MotorX
Vertical Axis	Y
Angle Camera	✓
Max Laser Range	35
Laser Position	Laser (Transform)
Cam	Camera (Camera)

Figure 29: Screenshot of the Unity editor of the Targeting System script showing how the variables where exposed in the editor for quick tuning and testing. Velocity Prediction: On; Noise Dampening: Off; Max Noise: 0.001; Probability to lose frame: 10%; $\theta_{Range}: 0.01$.

The results helped us confirm and reevaluate our current findings. And by introducing an error function we opened the possibility of adding a neural network in the picture if things don’t work out as nicely in the physical setup.

MOTORS

Introduction

This project is to track a drone with the help of camera and point a laser gun towards the drone. In this project, camera is rotating in all directions which results in the tracking of the drone along with the pointing laser system to the drone. Due to which the tasks were basically divided as:

- Assembling the Hardware parts and implementing the movement of laser and camera to a specific angle.
- Communicating with the camera to receive the image from the camera and detection of the drone from that image in terms of X, Y pixels maps with the help of Computer Vision concepts.
- Calculations of the angle (spherical coordinates) required to move the camera and the laser gun such that the X, Y pixel maps (cartesian coordinates) can be read as center of the image along with the prediction of next step of drone. These polar coordinates are provided to the hardware part in order to move the camera and the laser.

For this project, it is assumed that the drone is moving maximum at the speed of 10 MPH; due to the constrain to move the camera, it is assumed that the drone will not arrive in 5 feet radius around the drone. It is crucial for the camera to have the ability to rotate in 360° in both axes, so that it can cover all views and could follow the movement of the drone.

For the ease of visualization, let us assume the earth surface is flat, and is XY plane, whereas normal to earth surface, i.e. pointing to sky is the direction of positive Z axis. If we consider spherical coordinates of point P be (r, θ, ϕ), where r is the radius or distance from the Origin (0, 0, 0). If we assume a line joining center and point P as X-axis called *l*, the angle required to rotate the line *l* (i.e. of radius r) along X-axis towards Y-axis direction is θ. This will form a plane in XY coordinate with the image of line OP in this plane, let us call it p. When we rotate this plane p, by an angle ϕ along Z-axis, we will be at point P itself. Hence if we rotate the camera around 360° in θ direction, we have a complete ability to move the camera in circular motion in 2D dimensions, which expresses that our camera and laser is able to cover across a plane of motion. In order to achieve 3D motion, it is required to rotate the camera and laser module in another axis, i.e. in ϕ direction. In this axis if we rotate by 180°, we are able to cover all points. Hence rotating for 360° is not giving us an extra advantage.

In order to have these rotation, 2 rotating device such as motors are required. To keep the camera positioning always up, it is important to not rotate the motors with the complete cycle across the Z-axis, which will help to prevent the camera to be upside down. So, θ is supposed to rotate a complete cycle, i.e. 360°, whereas ϕ is rotating only half the cycle, i.e. 180°.

To move the parts, it is important to correlate the computer vision parts and shaft that contains camera and laser to communicate with help of some controlling device like microcontrollers. For this project, Raspberry pi 4, is selected in order to control the shaft and communicate between the device implementing computer vision parts after comparing with other microcontrollers too as discussed in following section. As the other parts will provide the coordinates, motors can be moved based on the current operation. As the coordinates will be in spherical coordinates, it is important we are able to control the rotation angles of the motors.

Selection of Microcontroller

In order to develop the communication between motors and computer vision algorithms, it is crucial to have a microcontroller so transitions between every angle is smooth in order to track a drone. One of the ways to achieve is to select a micro controller powerful enough to compute computer vision algorithm along with motion of motors. For this project, we decide to use separate devices for computer vision and the movement of motors to rotate the camera and laser mounted on top of them.

For this project we considered 3 microcontrollers, Arduino Mega 2560 Rev3, Raspberry Pi 4, Jetson TX2 4GB. Looking up on the specifications of microcontrollers, Jetson TX2 4GB seems to be over promising to use as it will provide very fast computations which we may never use in this project. However, when we looked up for Arduino Mega 2560 Rev3, computation time to calculate and rotate the motor seems to be very lagging the computation of the computer vision algorithms. Our task is to rotate the motors by the time the computer vision can provide other coordinates for it to rotate. Arduino Mega 2560 Rev3 seems to compromise that time frame.

However, looking up at Raspberry Pi 4, with 8 GB RAM it seems it can have a high computation time to rotate the motors.

Selection between Jetson and Raspberry Pi seems very challenging, as high computation speed does not harm the applications. Even considering the costs of the microcontroller boards, it was decided to use Raspberry Pi 4 with 8 GB RAM for this project, however, if it also lags into computation time, Jetson can replace Raspberry Pi at that movement of time. Computation time for microcontroller is the time taken by the microcontroller to read the data (coordinates/angles) from serialization along with the calculations of the steps motor requires and sending the command to rotate motors along with rotation itself. This computation time should be less than the computer vision algorithm/predictions in order to have a smooth movement.

Selection of Motors

In order to move any mechanical part, it is essential to decide type of motors suitable for an application. For this project, we need to rotate the camera by certain angles, however speed of motor is not a crucial thing that plays role. For such applications, motors usually used are either servo motor or stepper motor. For our project we will be working with the stepper motor, however it is crucial to note what was the reasons we choose stepper motor.

Servo Motor

Servo motor works on the PWM (Pulse Width Modulation) principle, which means its angle of rotation is controlled by the duration of pulse applied to its control PIN. Basically, servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears.

A typical servo motor is a closed-loop servomechanism that uses position feedback to control its motion and final position. Moreover, the input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft. The motor is incorporating some type of encoder to provide position and speed feedback. In the simplest case, we measure only the position. Then the measured position of the output is compared with the command position, the external input to controller. Now If the output position differs from that of the expected output, an error signal generates. Which then causes the motor to rotate in either direction, as per need to bring the output shaft to the appropriate position. As the position approaches, the error signal reduces to zero. Finally, the motor stops.

The very simple servomotors can position only sensing via a potentiometer and bang-bang control of their motor. Further the motor always rotates at full speed. Though this type of servomotor doesn’t have many uses in industrial motion control, however it forms the basis of simple and cheap servo used for radio control models.

Servomotors also find uses in optical rotary encoders to measure the speed of output shaft and a variable-speed drive to control the motor speed. Now this, when combined with a PID control algorithm further allows the servomotor to be in its command position more quickly and more precisely with less overshooting.

There are 2 kinds of servo motor in general: Positional rotation servo motor and Continuous rotation servo motor. Positional rotation servo motor is the most important servo motor. Hence it is also the most common type of servo motor. The shaft output rotates in about 180°. Additionally, it includes physical stops located in gear mechanism to stop turning outside these limits to guard the rotation sensor. On the other hand, continuous rotation servo motor relates to the common positional rotation servo motor, but it can go in any direction indefinitely. The control signal, rather than setting the static position of the servo, is understood as speed and direction of rotation. For our functionality it is crucial to control the angle rather than the speed, hence leaving continuous rotation servo motor out of the picture as it will not stop at the desired angle very effectively and will produce more jumps that will make a rough movement of the camera. Moreover, to have complete freedom of ration in 3D plane it is necessary that at least one motor rotates by a complete angle of 360° which is not possible by positional rotation servo motor.

Stepper Motor

To implement such functionality where we need to change to a desired angle, stepper motors plays a crucial role. Stepper motors works on the principle of electromagnetic induction, where there are several pairs of coils as shown in the figure. However only one pair of coils will act as magnet and will attract the rotors. These coils are designed such that only one pair of coils will be perfectly aligned with the rotor at a time. Hence movement between every consecutive pair of coils is known as a step. Unlike DC motor, this motor moves in steps and the based on the number of coil pairs, number of steps required in one complete revolution is calculated. Typically, stepper motors have 200 steps per resolution, which results into a stepper size of 1.8°.

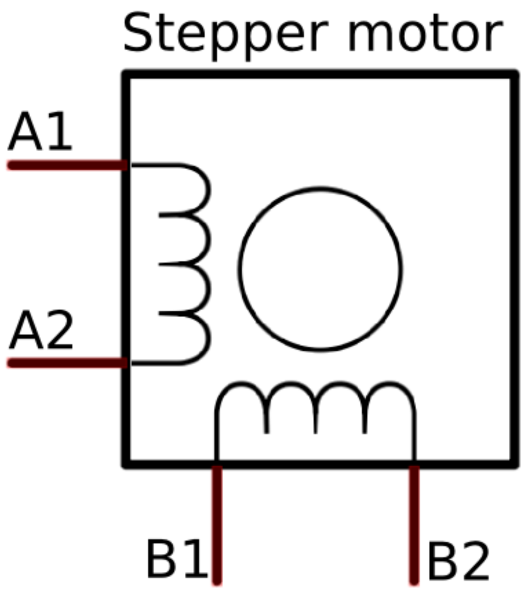


Figure 1: Internal Circuiting of a stepper motor

This principle can be manipulated with the help of bipolar stepper motor to achieve smaller steps also known as microstepping. The unipolar stepper motor operates with one winding with a center tap per phase. Each section of the winding is switched on for each direction of the magnetic field. Each winding is made relatively simple with the commutation circuit, this is done since the arrangement has a magnetic pole which can be reversed without switching the direction of the current. In most cases, given a phase, the common center tap for each winding is the following;

three leads per phase and six leads for a regular two-phase stepper motor. With bipolar stepper motors, there is only a single winding per phase. The driving circuit needs to be more complicated to reverse the magnetic pole; this is done to reverse the current in the winding. This is done with an H-bridge arrangement, however, there are several driver chips that can be purchased to make this a simpler task. Unlike the unipolar stepper motor, the bipolar stepper motor has two leads per phase, neither of which are common. Static friction effects do happen with an H-bridge with certain drive topologies; however, this can be reduced by dithering the stepper motor signal at a higher frequency.

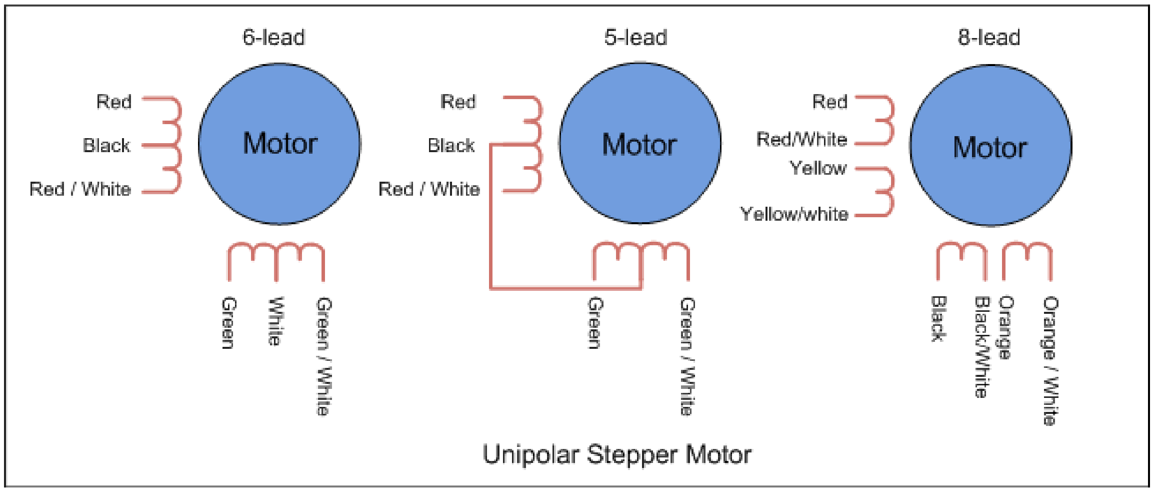


Figure 2: Internal wiring of a Unipolar Stepper Motor

Under no load conditions, a stepper motor has an angle accuracy within ±3 arc minutes (±0.05˚). The small error arises from the difference in mechanical precision of the stator and rotor and a small variance in the DC resistance of the stator winding. Generally, the angle accuracy of the stepper motor is expressed in terms of the stop position accuracy.

The stop position accuracy is the difference between the rotor's theoretical stopping position and its actual stopping position. A given rotor stopping point is taken as the starting point, then the stop position accuracy is the difference between the maximum (+) value and maximum (−) value in the set of measurements taken for each step of a full rotation. The stop position accuracy is within ±3 arc minutes (±0.05˚), but only under no load conditions. In actual applications there is always the same amount of friction load. The angle accuracy in such cases is produced by the angular displacement caused by the angle – torque characteristics based upon the friction load. If the friction load is constant, the displacement angle will be constant for uni-directional operation. However, in bi-directional operation, double the displacement angle is produced over a round trip. When high stopping accuracy is required, always position in the same direction.

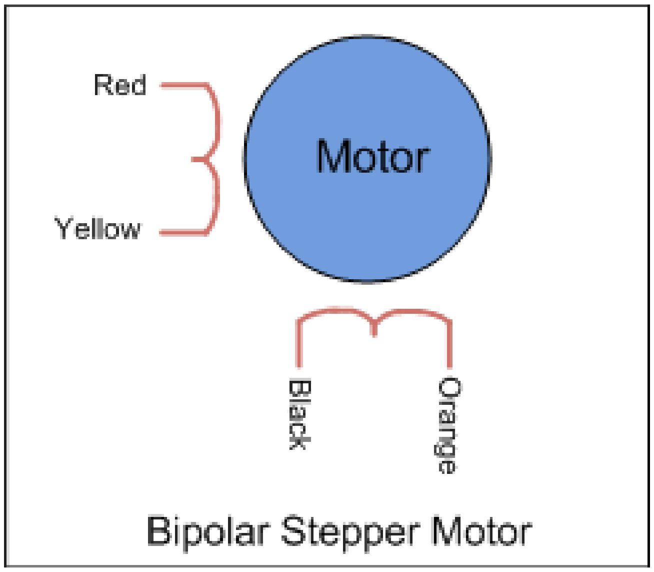


Figure 3: Internal wiring of a Bipolar Stepper Motor

The stepper motor rotates through the sequential switching of current flowing through the windings. When the speed increases, the switching rate also becomes faster and the current rise falls behind, resulting in lost torque. The chopping of a DC voltage that is far higher than the motor's rated voltage will ensure the rated current reaches the motor, even at higher speeds. Due to which stepper motors works really well at low speeds (<1000 rpm).

Microstepping

For this application, it is assumed that the motors may need to move 0.08° as the smallest angle, which is not possible in such scenario. Hence to increase number of steps it is essential to approach other techniques like attaching gears to the motors or providing a different set of current to the coils so that 2 pairs of coils can have the electric field of different ratio. This will result the rotor to be in between the both coils and will take smaller steps. This technique of small steps by manipulating the electric field across the coils is also known as the concept of micro stepping in stepper motors. These micro steps are achieved by just manipulating the electric field without adding any additional gears. So, such changes can be achieved by manipulating the driver of the stepper motor, so that it can provide the different fractions of the step. Such drivers can provide 1/2, 1,4, 1/8, 1/16, 1/32 steps in between one complete steps of the motor as shown in table 1. With each manipulation, the maximum torque a motor can rotate with decreases with the significance factor. Effect of such small angle rotations can be seen only if the torque provided to the motor by the weight of substance is within the range of decrease torque factor, else it will be shown at the complete steps. If the manipulation by electric field is done by the factor of 1/32 for a motor that has a step size of 1.8°, then after the manipulation, smallest step size will be approx. 1.8° x 1/32 = 0.05625°. Hence for this project manipulation by the factor of 1/32 is more than sufficient. With more micro stepping the average error to rate the motor by specific angle also increases as the rotor has to satisfy the both electric fields from the coil and could result in more uncertainty.

Step accuracy can be given by the manufacturer in the form of absolute (±1.0 degree, as an example) or relative (±5% of one full-step). Normally step accuracy is only specified for 2-phase-on stop positions. (Here a 2-phase-on stop position means a position with the same current level in both windings. A position with different current levels, or none, in the windings is a micro step position). Optimizing a motor for high full-step positioning accuracy and holding torque normally reduces micro- stepping accuracy. One important effect of the 2-phase-on step accuracy is shown by the following example. Consider a micro stepping design, using 1/32 -full-step mode with a 7.5-degree PM-stepper motor. One micro step theoretically corresponds to 7.5 π 32 = 0.23°. For this type of motor, a step accuracy of ±1 degree is common. This means that if the motor home position is calibrated at a randomly-selected 2-phase-on position (which can be positioned anywhere within ±1 degree from the theoretically-correct home position) the maximum deviation of the rotor at another 2-phase-on position can be [1-(-1)] / 0.23 = 8.5 micro steps from its theoretical position. This fact has to be considered when micro stepping is used in applications were absolute positioning is essential.

Step Mode	M	MODE3	MODE2	MODE1
Full step	1	0	0	0
Half step	2	0	0	1
Quarter step	4	0	1	0
1/8th step	8	0	1	1
1/16th step	16	1	0	0
1/32th step	32	1	0	1
1/128th step	128	1	1	0
1/256th step	256	1	1	1

Table 1: Step mode selection in STSPIN820

For this project, we are using a bipolar stepper motor (NEMA 14) where each phase draws 500 mA at 10 V, allowing for a holding torque of 1 kg-cm (14 oz-in). in order to achieve the micro stepping, STSPIN820 Stepper Motor Driver Carrier will be used. This breakout board for STMicro's STSPIN820 microstepping bipolar stepper motor driver offers microstepping down to 1/256-step and a wide operating range of 7 V to 45 V. It can deliver up to approximately 0.9 A per phase continuously without a heat sink or forced air flow (up to 1.5 A peak). The STSPIN820 is a stepper motor driver which integrates, in a small QFN 4 x 4 mm package, both control logic and a low R_{DSon} power stage. The integrated controller implements a PWM current control with fixed OFF time and a microstepping resolution up to 1/256th of the step. The device can be forced into a low consumption state. The device offers a complete set of protection features including overcurrent, overtemperature and short-circuit protection.

A microstepping driver such as the STSPIN820 allows higher resolutions by allowing intermediate step locations, which are achieved by energizing the coils with intermediate current levels. For instance, driving a motor in quarter-step mode will give the 200-step-per-revolution motor 800 microsteps per revolution by using four different current levels. The resolution (step size) selector inputs (MODE1, MODE2, and MODE3) enable selection from the eight step resolutions according to the table below. These three pins are floating, so the MODE pins must be connected to logic high or low before operating the driver. For the microstep modes to function correctly, the current limit must be set low enough (see below) so that current limiting gets engaged. Otherwise, the intermediate current levels will not be correctly maintained, and the motor will skip microsteps. The driver requires a logic supply voltage (3 – 5 V) to be connected across the VCC and GND pins and a motor supply voltage of 7 V to 45 V to be connected across VIN and GND. These supplies should have appropriate decoupling capacitors close to the board, and they should be capable of delivering the expected currents (peaks up to 3 A for the motor supply).

The driver ICs have maximum current ratings higher than the continuous currents we specify for these carrier boards, but the actual current you can deliver depends on how well you can keep the IC cool. The carrier’s printed circuit board is designed to draw heat out of the IC, but to supply more than the specified continuous current per coil, a heat sink or other cooling method is required.

Since the input voltage to the driver can be significantly higher than the coil voltage, the measured current on the power supply can be quite a bit lower than the coil current (the driver and coil basically act like a switching step-down power supply). Also, if the supply voltage is very high compared to what the motor needs to achieve the set current, the duty cycle will be very low, which also leads to significant differences between average and RMS currents. Additionally, please note that the coil current is a function of the set current limit, but it does not necessarily equal the current limit setting as the actual current through each coil changes with each microstep and can be further reduced if Active Gain Control is active.

The rising edge of each pulse to the STEP (STCK) input corresponds to one microstep of the stepper motor in the direction selected by the DIR pin. Unlike most of our other stepper motor driver carriers, the STEP and DIR inputs are floating, so they must be connected to logic high or low to ensure proper operation.

Chassis Design

Laser is supposed to be mounted on a home-made/3d printed chassis, as shown in the figure, that allows a platform to contain just camera and laser and rotate it by an angle of ϕ, which itself will be stored on another platform, that has a freedom to rotate along θ axis. These motors operate at 12-24 V DC power supply which requires either a power source or 110 V AC to 24 V DC conversion adapter. Input for the coordinates will be taken as a serial input to the microcontroller and will be processes to the nearest step size. This size will be transmitted to the driver of the motors that will allow the motors to rotate.

Based on the scenario, if a drone is at 10 miles/hr = 52,800 ft./hr = 14.66667 ft./s and it is 5 feet away, radius is 5 ft. Hence, if drone rotates in circle around camera at 5 ft then its angular speed is 2.933333 rad/s = 168.067619905°/s. = 0.4668544997 rotations per sec = 28.0112699842 rpm. Stepper motors usually provide rotation speeds of 1000 rpm. Which is very much faster than what we expect drone to rotate with. Hence it is feasible to assume that rotational speed of the motor, is not going to be a big factor in order to keep up with the drone's speed. Changing the polarity with of the coils changes the direction of the motor. As it is an Electromagnetic induction motor, changing the polarity will not create the lagging for motor to change directions.

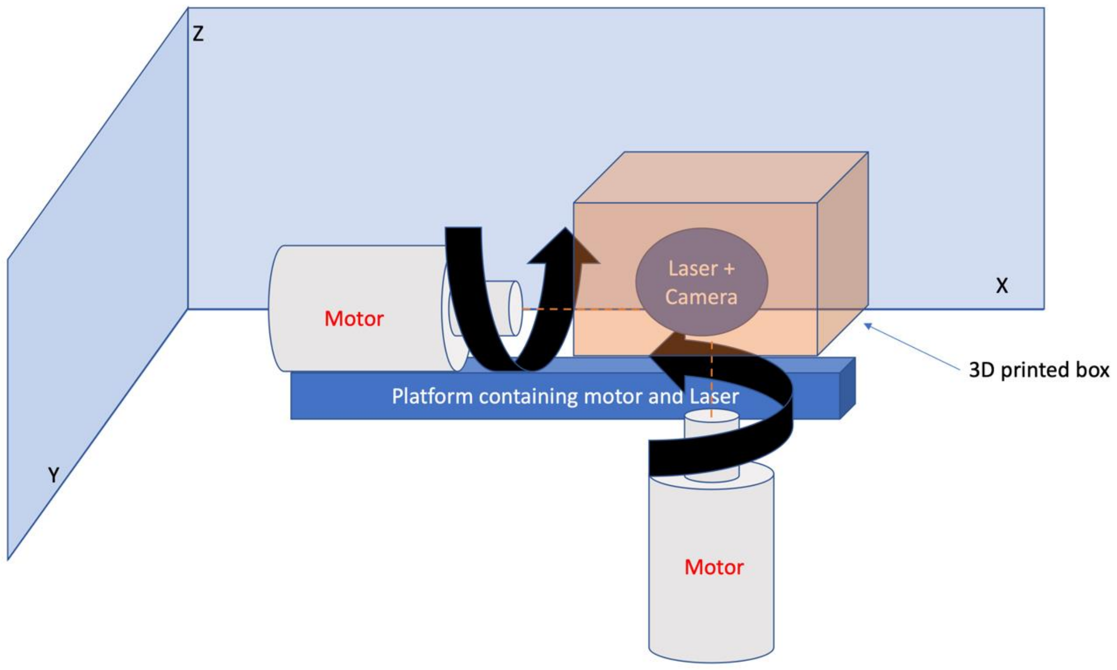


Figure 4: Schematic of chassis containing laser and camera. This chassis is connected directly to a motor controlling the angle in YZ plane. Both motor and chassis are placed on another platform connected to second motor controlling the angle in XY plane where center of Laser and camera is the center of axis for rotaion.

Serial Communication

UART (Universal Asynchronous Receiver/Transmitter) is a serial communication protocol in which data is transferred serially i.e. bit by bit. Asynchronous serial communication is widely used for byte-oriented transmission. In Asynchronous serial communication, a byte of data is transferred at a time. UART serial communication protocol uses a defined frame structure for their data bytes. Frame structure in Asynchronous communication consists:

- START bit: It is a bit with which indicates that serial communication has started and it is always low.
- Data bits packet: Data bits can be packets of 5 to 9 bits. Normally we use 8-bit data packet, which is always sent after the START bit.
- STOP bit: This usually is one or two bits in length. It is sent after data bits packet to indicate the end of frame. Stop bit is always logic high.

Usually, an asynchronous serial communication frame consists of a START bit (1 bit) followed by a data byte (8 bits) and then a STOP bit (1 bit), which forms a 10-bit frame as shown in the figure above. The frame can also consist of 2 STOP bits instead of a single bit, and there can also be a PARITY bit after the STOP bit.

Raspberry Pi has two in-built UART: PL011 UART, mini UART. PL011 UART is an ARM based UART. This UART has better throughput than mini UART. In Raspberry Pi 4, mini UART is used for Linux console output whereas PL011 is connected to the On-board Bluetooth module. And in the other versions of Raspberry Pi, PL011 is used for Linux console output. Mini UART uses the frequency which is linked to the core frequency of GPU. So as the GPU core frequency changes, the frequency of UART will also change which in turn will change the baud rate for UART. This makes the mini UART unstable which may lead to data loss or corruption. To make mini UART stable, fix the core frequency. mini UART doesn’t have parity support. The PL011 is a stable and high performance UART. For better and effective communication use PL011

UART instead of mini UART. It is recommended to enable the UART of Raspberry Pi for serial communication. Otherwise, we are not able to communicate serially as UART ports are used for Linux console output and Bluetooth module. All UARTs on the Raspberry Pi are 3.3V only - damage will occur if they are connected to 5V systems. An adaptor can be used to connect to 5V systems. Alternatively, low-cost USB to 3.3V serial adaptors are available from various third parties.

By default, mini UART is mapped to UART pins (TX and RX) while PL011 is connected to on-board Bluetooth module on Raspberry Pi 4. In previous version of Raspberry Pi models, PL011 is used for Linux Console output (mapped to UART pins) and there is no on-board Bluetooth module. After making above configuration, UART can be used at UART pins. To access mini UART in Raspberry Pi 4, ttyS0 port is assigned. And to access PL011 in Raspberry Pi 3 ttyAMA0 port is assigned. But in other models of Raspberry Pi, there is only ttyAMA0 port is accessible.

Hardware UART port i.e. GPIO14(TXD) and GPIO15 (RXD) is also known as serial0 while other UART port which is connected to Bluetooth module is known as serial1.These names are created as serial aliases for Raspberry Pi version portability. We can check which UART i.e. mini UART (ttyS0) or PL011 UART (ttyAMA0) is mapped to UART pins. We can perform UART based serial communication using Python and C. In Python, serial.Serial(port,baudrate) is a class for Serial which is used for opening port from the Serial Library and data is read and written using read(Size) and write(Data) functions respectively. However in terms of C, WiringPi library is used to establish UART communication on Raspberry Pi.

Using this UART communication, data, i.e. angles θ, ϕ will be transmitted to the raspberry pi, which will calculate the approximate number of steps Raspberry Pi need to move each motor such that those angles are achieved. Once number of steps is calculated, micro stepping mode will be selected from the Raspberry Pi to the motor Driver and the Driver will be instructed to move the motors.

Standards

In the following section, several hardware and software standards are brought to light for better examination. While these standards are analyzed, there will also be discussion on how it relates to our project. In this section, discussion on general safety procedures and concerns will be touched on to further ensure the safety of the group in their specific design implementation.

IPC PCB Standards

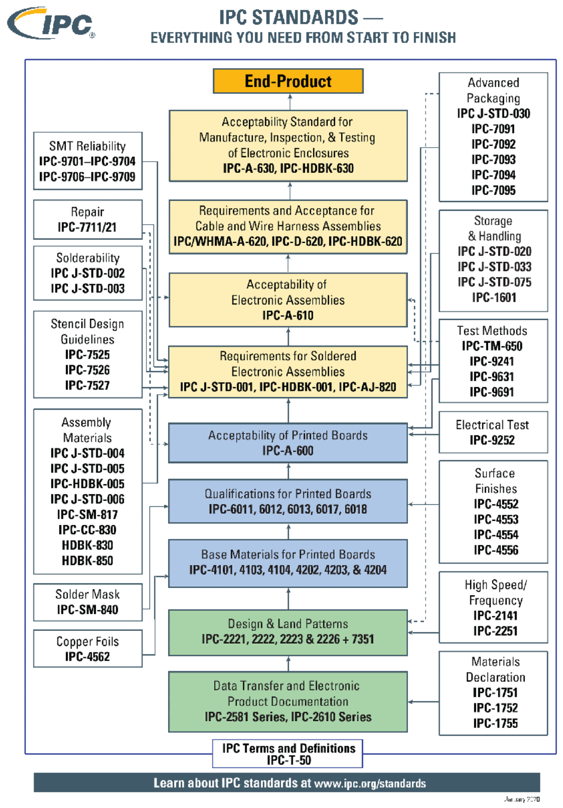

IPC STANDARDS —
EVERYTHING YOU NEED FROM START TO FINISH
End-Product
Acceptability Standard for Manufacture, Inspection, & Testing of Electronic Enclosures IPC-A-630, IPC-HDBK-630
Requirements and Acceptance for Cable and Wire Harness Assemblies IPC/WHMA-A-620, IPC-D-620, IPC-HDBK-620
Acceptability of Electronic Assemblies IPC-A-610
Requirements for Soldered Electronic Assemblies IPC J-STD-001, IPC-HDBK-001, IPC-AJ-820
Acceptability of Printed Boards IPC-A-600
Qualifications for Printed Boards IPC-6011, 6012, 6013, 6017, 6018
Base Materials for Printed Boards IPC-4101, 4103, 4104, 4202, 4203, & 4204
Design & Land Patterns IPC-2221, 2222, 2223 & 2226 + 7351
Data Transfer and Electronic Product Documentation IPC-2581 Series, IPC-2610 Series
IPC Terms and Definitions IPC-T-50
SMT Reliability IPC-9701–IPC-9704 IPC-9706–IPC-9709
Repair IPC-7711/21
Solderability IPC J-STD-002 IPC J-STD-003
Stencil Design Guidelines IPC-7525 IPC-7526 IPC-7527
Assembly Materials IPC J-STD-004 IPC J-STD-005 IPC-HDBK-005 IPC J-STD-006 IPC-SM-817 IPC-CC-830 HDBK-830 HDBK-850
Solder Mask IPC-SM-840
Copper Foils IPC-4562
Advanced Packaging IPC J-STD-030 IPC-7091 IPC-7092 IPC-7093 IPC-7094 IPC-7095
Storage & Handling IPC J-STD-020 IPC J-STD-033 IPC J-STD-075 IPC-1601
Test Methods IPC-TM-650 IPC-9241 IPC-9631 IPC-9691
Electrical Test IPC-9252
Surface Finishes IPC-4552 IPC-4553 IPC-4554 IPC-4556
High Speed/ Frequency IPC-2141 IPC-2251
Materials Declaration IPC-1751 IPC-1752 IPC-1755
Learn about IPC standards at www.ipc.org/standards
January 2020

Figure 5: IPC Standards

The Association Connecting Electronics Industries, or IPC, is a trade association determined to standardize the manufacturing process of electronic equipment. This association has published many standards that are used by commercial PCB manufacturers to ensure reliability in their products and uniformity within the market. The IPC-2221 is the generic standard on printed board design that establishes the requirements for component mounting and interconnecting structures. Other standards include IPC-2615, which covers printed board dimensions and tolerances. IPC-ET-652 for guidelines and requirements for electrical testing of PCB with no components. IPC-A-600 describes the acceptability of printed boards in inspection settings. IPC-

A-610 for acceptability of electronic assemblies. Some material specification standards include IPC-4562 for metal foil for printed wiring applications and IPC-4202 for flexible base dielectrics for use in flexible printed circuits. IPC standards range from general topics to design specifications, materials, performance, and inspections.

The Institute for Printed Circuits was originally founded by six individual printed circuit board manufacturers but in 1957. After 20 years, in 1977 the six manufactures officially changed the name of the institution to the Institute for Interconnecting and Packaging Electronic Circuits (IPC). Since then the institution has been able to publish several standards for when it comes to building a circuit from scratch. The IPC has gone into depth on design specifications, material specifications, performance and inspection documents, flex assembly and materials standards, and general documentation. It is recommended to uses the IPC standards when it comes to building your own custom circuit.

During this project, the team will be manufacturing their own PCB board. In the process of doing gather the elements and collecting research, it is important for the team to follow several guidelines that are stated within the IPC standards. As illustrated in the image above the IPC standards are step by step documents that thoroughly go into detail on what are some of the expectation for getting a PCB to be checked out properly, how to add components through soldering, what type of flux to use to create the best joint. Throughout the building process. The team will be going back through the IPC standards to insure an accurate and efficient board for the ADLTS.

IEC 60950-1

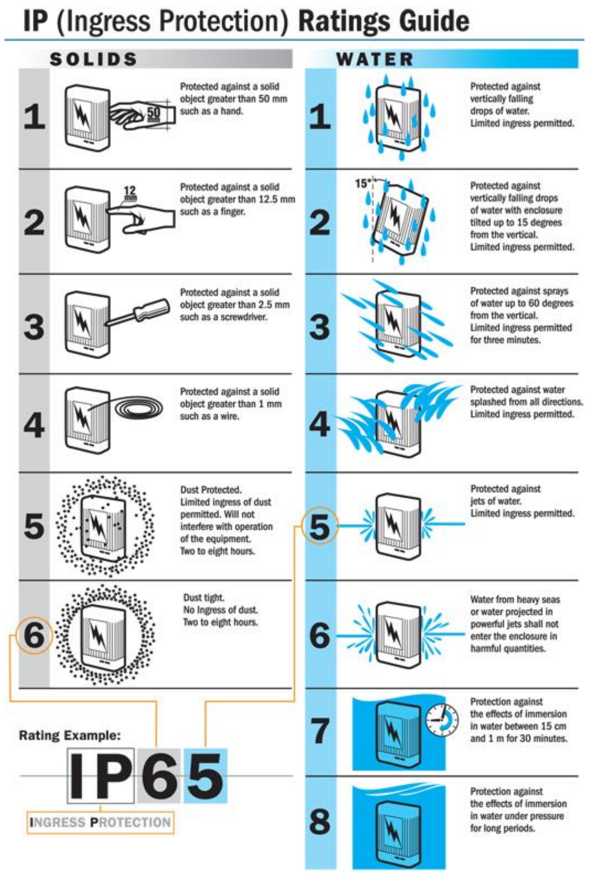
The safety of information technology equipment standard is applicable to mains, battery-powered equipment and office electronic devices with rated voltages under 600V. Its main purpose is to prevent hazards such as fire, electric shock, and mechanical instability. It divides equipment into three main classes. Class 1 equipment protects from shock by basic insulation and protective earth grounding. In other words, if the insulation of a conductive segment fails for any reason, the conductor is also connected to a protective earth conductor. Class 2 equipment requires no ground for protection since it uses double or reinforced insulation. Finally, Class 3 equipment operates from a safety extra low voltage (SELV) supply circuit, which inherently protects from electric shock since the equipment is unable to generate a hazardous voltage. Some definitions are required for the complete understanding of the standard:

- Hazardous Voltage: Any voltage over 42.2 VAC or 60 VDC without a limited current circuit
- Extra-Low Voltage (ELV): A voltage in a secondary circuit under 42.2 VAC or 60 VDC that is separated from a hazardous voltage by at least insulation.

- Safety Extra-Low Voltage Circuit (SELV): A secondary circuit unable to reach a hazardous voltage. SELV circuits must be separated from hazardous voltages by two levels of protection such as double insulation.
- Limited Current Circuits: Are designed to output safe currents even in the case of shorts or any fault conditions. For frequencies lower than 1 kHz, the steady state current cannot surpass 0.7 mA AC or 2 mA DC. For higher frequencies, the 0.7 mA limit is multiplied by the frequency in kHz without exceeding 70 mA.
- Limited Power Source (LPS): Designed with a set output voltage, current, power, and short circuit current limit.

IEC 60529

The International Electrotechnical Commission (IEC) 60529 is a standard that goes over what would classify an object to be protected from environmental elements. This standard is also known as the Ingress Protection Marking. These standard rates a device based on the degree of protection from dust, intrusion, accidental contact, and water. The Ingress Protection (IP) has its own grading scale in order to classify devices that are claiming to be protected for a environmental element.



	SOLIDS	WATER
1	Protected against a solid object greater than 50 mm such as a hand.	Protected against vertically falling drops of water. Limited ingress permitted.
2	Protected against a solid object greater than 12.5 mm such as a finger.	Protected against vertically falling drops of water with enclosure tilted up to 15 degrees from the vertical. Limited ingress permitted.
3	Protected against a solid object greater than 2.5 mm such as a screwdriver.	Protected against sprays of water up to 60 degrees from the vertical. Limited ingress permitted for three minutes.
4	Protected against a solid object greater than 1 mm such as a wire.	Protected against water splashed from all directions. Limited ingress permitted.
5	Dust Protected. Limited ingress of dust permitted. Will not interfere with operation of the equipment. Two to eight hours.	Protected against jets of water. Limited ingress permitted.
6	Dust tight. No Ingress of dust. Two to eight hours.	Water from heavy seas or water projected in powerful jets shall not enter the enclosure in harmful quantities.
7		Protection against the effects of immersion in water between 15 cm and 1 m for 30 minutes.
8		Protection against the effects of immersion in water under pressure for long periods.

Figure 6: IP Rating Chart

In the grading scale format is as follows "IP##". The "IP" would stand for it being officially graded by the Ingress Protection. The first number or digit would tell the user the intensity of the object's protection from a solid surface. The second number is how resistant the device is to water. The higher the rating the more protected the device is. The Ingress Protection goes through aggressive and extensive testing to scale the protection of devices. No other company can mimic what they do, therefore no other company can create their own grading scale to classify the protection of their own device or that of others.

The highest level of waterproof that the IP has to offer is IPX9K which means that the tested device is able to resist high-pressures and, high temperatures sprays at close range. This high rating is very rarely used in day to day market devices. A more commonly seen rating would be the IPX8, which is more commonly used in modern day technology such as mobile phones. This rating states that the device is capable of being submerged under water for more than one meter. The exact depth of testing can be changed based on the manufacture needs.

For this project, the team will be attempting to accomplish a water-resistant rating of IPX4 for motors, camera and laser with the help of 3D-chasis. This classification shows that the product is water resistant from any direction. Due to the fact the team does not plan on making this project a marketable device we are setting this rating as a goal. From lack of time and budget the team will not be reaching out to contact with the International Electrotechnical Commission to ask for and IP marking. Although the IP marking of certified protection will not be received from our device, we will still be attempting to achieve water resistant from any direction.

ACM Code of Ethics

Computing professionals' actions change the world. To act responsibly, they should reflect upon the wider impacts of their work, consistently supporting the public good. The ACM Code of Ethics and Professional Conduct ("the Code") expresses the conscience of the profession.

The Code is designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way. Additionally, the Code serves as a basis for remediation when violations occur. The Code includes principles formulated as statements of responsibility, based on the understanding that the public good is always the primary consideration. Each principle is supplemented by guidelines, which provide explanations to assist computing professionals in understanding and applying the principle.

Section 1 outlines fundamental ethical principles that form the basis for the remainder of the Code. Section 2 addresses additional, more specific considerations of professional responsibility. Section 3 guides individuals who have a leadership role, whether in the workplace or in a volunteer professional capacity. Commitment to ethical conduct is required of every ACM member, and principles involving compliance with the Code are given in Section 4.

The Code as a whole is concerned with how fundamental ethical principles apply to a computing professional's conduct. The Code is not an algorithm for solving ethical problems;

rather it serves as a basis for ethical decision-making. When thinking through a particular issue, a computing professional may find that multiple principles should be taken into account, and that different principles will have different relevance to the issue. Questions related to these kinds of issues can best be answered by thoughtful consideration of the fundamental ethical principles, understanding that the public good is the paramount consideration. The entire computing profession benefits when the ethical decision-making process is accountable to and transparent to all stakeholders. Open discussions about ethical issues promote this accountability and transparency.

IEEE Code of Ethics

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems. Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.

The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships. The Clauses of each Principle are illustrations of some of the obligations included in these relationships. These obligations are founded in the software engineer's humanity, in special care owed to people affected by the work of software engineers, and the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

It is not intended that the individual parts of the Code be used in isolation to justify errors of omission or commission. The list of Principles and Clauses is not exhaustive. The Clauses should not be read as separating the acceptable from the unacceptable in professional conduct in all practical situations. The Code is not a simple ethical algorithm that generates ethical decisions. In some situations standards may be in tension with each other or with standards from other sources. These situations require the software engineer to use ethical judgment to act in a manner which is most consistent with the spirit of the Code of Ethics and Professional Practice, given the circumstances.

Ethical tensions can best be addressed by thoughtful consideration of fundamental principles, rather than blind reliance on detailed regulations. These Principles should influence software engineers to consider broadly who is affected by their work; to examine if they and their colleagues are treating other human beings with due respect; to consider how the public, if reasonably well informed, would view their decisions; to analyze how the least empowered will be affected by their decisions; and to consider whether their acts would be judged worthy of the ideal professional working as a software engineer. In all these judgments concern for the health, safety and welfare of the public is primary; that is, the "Public Interest" is central to this Code.

The dynamic and demanding context of software engineering requires a code that is adaptable and relevant to new situations as they occur. However, even in this generality, the Code provides support for software engineers and managers of software engineers who need to take positive action in a specific case by documenting the ethical stance of the profession. The Code provides an ethical foundation to which individuals within teams and the team as a whole can appeal. The Code helps to define those actions that are ethically improper to request of a software engineer or teams of software engineers.

The Code is not simply for adjudicating the nature of questionable acts; it also has an important educational function. As this Code expresses the consensus of the profession on ethical issues, it is a means to educate both the public and aspiring professionals about the ethical obligations of all software engineers.

Environment Health and Safety

Soldering Safety

The ADLTS Team is expecting to use a high purity alloy such as Sn63Pb37, which is composed of 63 percent tin and 37 percent lead. The soldier must be high priority so that the solder joint may be properly joined with the board. The use of lead and lead byproducts poses a danger to the human body. This is well known from the continuous studies of modern science, and therefore require cautions actions when working with lead. Lead poses numerous chronic health effects due to its neurotoxic composition. Some of these side effect includes (but are not limited to), muscle and joint pain, concentration problems, digestive problems, reproduction problems, and many more. Therefore, our team will take the precautionary steps when it comes to handling lead when soldering.

When it comes to soldering in general, one of the first dangers comes from the soldering iron. In order to solder, the soldering iron must be approximately 750 degrees Fahrenheit to melt and manipulate the solder paste. With such extreme temperatures, the first danger comes from possibility of getting burnt. Therefore, when handling the soldering iron, it is crucial that the team handle the iron with respect and caution. This may be practiced by soldering little bits at a time and having the iron only out if needed. If not needed, to have it be placed in the cradle and far from human touch. Throughout the building of the PCB board there will be minimal human interaction with the iron, how there are many components that will be placed on the board using a pick and place machine and then heating the items into place through an oven. Another instance that is common to occur is the product coming off the line, such as a misaligned part. In order to fix this issue, hand soldering will be the resolution.

There are various factors that must be placed into perspective when it comes to handling anything that may contain lead. The first and utmost important is the user's health. As stated in the beginning of this section the dangers and side of effects of lead have been well documented in the scientific community. However, based on these documents, there are numerous applications, which includes ours, that lead will not necessarily pose a threat human health. It would take a large amount of lead to be ingested so that it poses a threat to the human body. How the handling of lead will still occur the team must obey the RoHS compliance. The RoHS Directive states that is there are certain elements and chemicals that must be handled in accordance to specific guidelines if that said product is being shipped abroad. Currently, the 37 percent of lead is not decent for what we are attempting to create. Being that is at a lower percentage it is easier to manipulate and work with when making the joints. Soldering is very precise when it comes to the temperature gradient each paste may need a Fahrenheit in order to blend well. With this solder that will not be an issue. The solder the team plans on using is very friendly and easy to overcome. Although it contains a dominant portion of tin, it is found it work well with small projects as this one.

Another concern that comes from soldering is the solder fumes when working with lead. At Carnegie Mellon University, it has been documented that safety precautions for these fumes must not be taken lightly. The documents at Carnegie Mellon University state that when soldering it would be advised to solder in a well-ventilated area or at least work under an exhaust fan of some sort. They also advise in wearing protective attire when soldering in the lab, this includes and is not limited to closed toe shoes, eyewear, and a mask. When certain parts of the product are being cooked in the heating oven, the fumes will not pose a threat, as the oven is equipped with an exhaust fan built in. The use of safety equipment is still advised while the lab is in use.

As a general summary when taking on project that involves soldering in some way there are specific safety precautionary steps that must be followed. In this paragraph, there will be discussion of these said safety procedures. To begin when handling lead there will always be the production of fumes from the flux that contains rosin. To combat this issue, one must be in a well-ventilated area whether it is an open area, or the area contains equipment such as an exhaust fan. A good example of such an area would be the UCF TI lab, which is a very open and even is equipped with a soldering station that gives good ventilation. It is also very important to never touch the soldering iron when heated as it may cause severe burns. A soldering iron may reach heat indexes more than 750 degrees Fahrenheit. Therefore, it is critical to ensure that the iron is unplugged and on a stable surface, given that the cradle may easily fall and leaving the iron plugged my lead to unforeseen dangers. While a member is using the lab, it is critical that they are following lab safety wear protocols. This includes and is not limited to wearing eyewear and closed toe shoes. It is also important to wash your hands thoroughly after the use of the soldering iron to eliminate any chances of lead exposure through the hands. Although it has not been discussed in thorough detail in this passage it is of utmost important to never solder when the circuit is live. One must ensure that the circuit is not plugged to any sort of power source when soldering. This is done for several reason. One being the safety of the parts on the board and the safety of the user, by avoiding electrocution.

With any type of soldering there will be waste. In the case of soldering with lead there will be waste created that must be disposed of properly. This waste will be packaged in a metal container and properly labeled as hazardous waste and later properly disposed of. This is done in efforts to avoid having any remains of lead go into landfills which is very harmful to the environment. In the UCF IT lab there are dross which are used to dispose of any waste that is produced by soldering with lead. The UCF EHS also provides specific labels that are to the student disposal so that they may properly label the waste they are disposing.

If the statements above are followed properly it will ensure safety in the lab environment when handling lead and soldering. Any further question in regard to handling and disposing of waste may be answered by contacting the Environmental Health and Safety (EHS) at UCF, phone number (407) 823-6300.

RoHS Compliance

RoHS, stands for the Restriction of Hazardous Substances Directive. Short for the restriction of use of certain hazardous substances in various electrical and electronic components. This Directive was adopted in February 2003 by the European Union. The RoHS lists approximately ten substances that are to be limited. These ten items being: mercury, lead, cadmium, polybrominated biphenyls, polybrominated diphenyl ether, bis(2-ethylhexyl) phthalate, butyl benzyl phthalate, dibutyl phthalate, hexavalent chromium, di-iso-butyl phthalate. RoHS is associated with Waste Electrical and Electronic Equipment Directive (WEEE), which is the Waste Electrical and Electronic Equipment Directive. WEEE main purpose is to set the standards on the collection, recycling, and recovery standards for various electronic goods. There intentions are to remove harmful substances, such as lead, from electronic devices. The RoHS is not required to contain special stickers to ensure that the order is complaint under the RoHS regulations, however they due require special restriction when it evolves shipping to various locations abroad.

The only concern that the Baby Buoy team has from the substances that the RoHS covers would be Lead. Lead is often found in solder paste. Having that the team will be constructing our individual PCB soldering will be occurring. Therefore, the team will be RoHS compliant. Based on the specifications of RoHS 1 < 1000 parts/millions of concentrations must be maintained over the board at all times to ensure compliance. A very common alternative to lead would be a 98 percent tin solder, but tint does not produce a high-quality solder joint and is more likely to be rejected in examination. Therefore, the team will be compiling with the RoHS regulation when it comes to soldering the PCB together.

Battery/DC Power Supply Safety

One of the most common sources of DC is from a stored electrical energy device such as a capacitor bank, battery, or fuel cell. You might be able to wait for a capacitor to bleed down, and you might be able to turn off a fuel cell but waiting for the energy to run down on a battery system could take years. Therefore, working on a battery system is always considered energized electrical work — or what used to be called “hot work” (referring to electrical energy, not heat). Batteries are somewhat unique in that they have a chemical hazard as well as an electrical one. Doing electrical work can in some cases expose a person to the chemical hazard, and vice versa. To be on the safer side the following steps will be followed while handling with Battery/DC Power Supply:

- Haste and inattention can cause many accidents. To avoid it, we will work deliberately and carefully. We will plan your activities prior to the execution, by familiarizing ourselves with equipment prior to actual operation, and verifing your work as we progress. It is important to keep a track of the location of power switches and ground fault interrupters.

- We will minimize exposure to live circuits. It is important to connect to the source of power as the last step when wiring a circuit and disconnect from the power source as the first step when disassembling or modifying a circuit.
- We will try to use only one hand as far as practical, keeping the other hand disengaged from circuitry as we do not want any part of your body to complete a circuit.
- Keeping watches, rings, and other metallic objects out of contact with live parts will be the safest step to operate with the Electrical Components. Wet, sweaty, or bleeding hands increase shock hazard. It is crucial to note exposed wires and other potential shock hazards.
- We will close switches quickly and positively. It is very dangerous to grope for switch handles with your head turned away or using damaged or misapplied parts! Wires that have poor insulation, setscrews that are loose, and insecure connections that may come apart are hazards.

Electric current damages the body in three different ways: it harms or interferes with proper functioning of the nervous system and heart; it subjects the body to intense heat; and it causes the muscles to contract. Electrical shock can be lethal. It's the current that kills. Voltage is not a reliable indication of danger because the body's resistance varies so widely it is impossible to predict how much current will result from a given voltage. The current range of 100 to 200 mA. is particularly dangerous because it is almost certain to result in lethal ventricular fibrillation. Victims of high-voltage shock usually respond better to artificial respiration than do victims of low-voltage shock, probably because the higher current clamps the heart and hence prevents fibrillation. If a person does suffer a severe shock, it is important to free the victim from the current as quickly as can be done safely. One must not touch the person until the electric power is turned off as one cannot help by becoming a second victim. The victim should be attended to immediately by a person trained in CPR (cardiopulmonary resuscitation). Also, an ambulance should be called immediately.

LASER / DRONE

Hardware Specifications

There are currently 4 different sections of the project, but 2 different systems. 1 of the systems is laser turret, which includes the motors, the Raspberry Pi, and the code base such as computer vision and algorithms in order to move the motors. The second system is the laser sensor that is carried by the drone. These two systems are independent of one another, although in the future may be able to communicate to each other. This section is to talk about the different hardware specifications that the ADLTS team has decided to use and the alternatives.

Laser Sensor

The laser sensor is independent from the main system, which is the laser turret and its processing unit. For that reason, the hardware and firmware need to be discussed independently from the main system. The laser system will require its own Arduino microcontroller board and battery source. Ideally the laser system would have shared the battery source from the drone in order to reduce the overall weight, but connecting and rerouting the voltage is unnecessarily overly complicated for the purposes of this project, so we decided to just have its own battery for the laser sensor. Since the laser sensor requires its own microcontroller board, we had to research on the most optimal board that will be able to complete the current task as well as future stretch goals.

To start, one of the first considerations that the ADLTS team talked about was the use between Raspberry Pi's versus Arduino brands and other major brand of microcontrollers. The Raspberry Pi's are generally more expensive than the Arduino counter parts, due the higher processing capabilities. The Raspberry Pi microcontrollers also have the additional capability of connecting greater SD memory for an additional cost. For the purposes of our goals, including stretch goals, the ADLTS team has decided that the Arduino brands are more than enough for the embedded system of the Laser Sensor. The laser sensor does not require tremendous processing power as it is only receiving a signal, and processing that signal at most for our data transfer stretch goals. These have traditionally been done in the past with just the Arduino microcontrollers.

So after choosing the Arduino brands, we had to decide which microcontroller in particular was the best one suited for this project. There are a lot of different options to consider and they all have different capabilities. Our team had some Arduino Uno's laying around so our initial prototype is using that, but it does not have the WIFI/Bluetooth capabilities that we might require in the future, so we might end up upgrading as our team continues to work on the project.

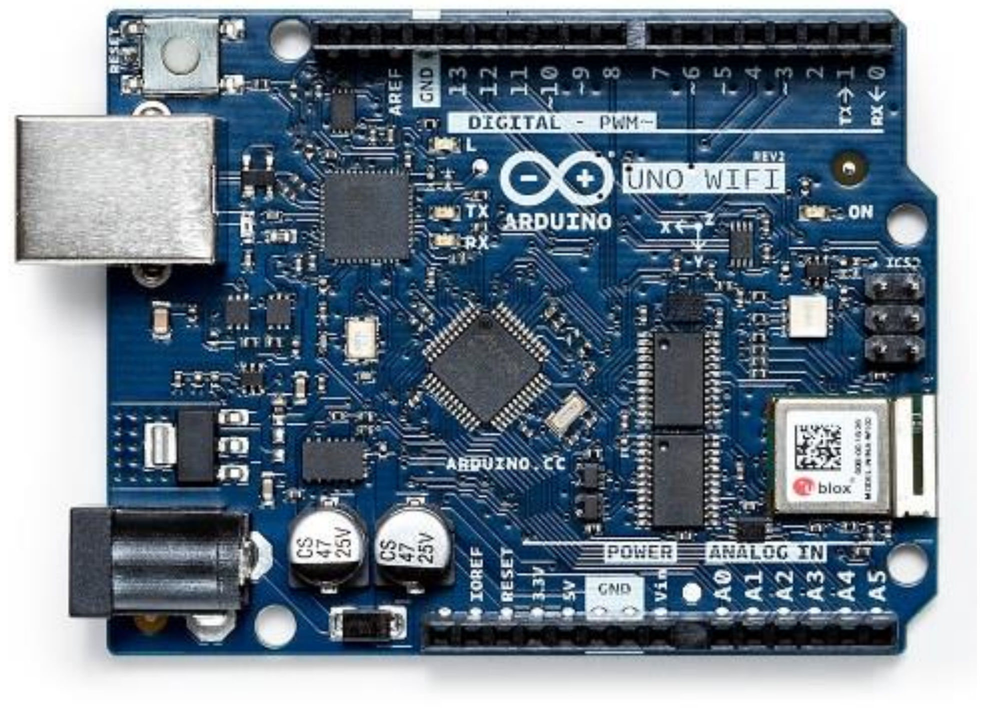
Adruino Microcontrollers

Arduino Microcontrollers are relatively inexpensive and are an open source platform that have allowed many engineers to get started on their early projects with hardware. These microcontrollers are the brains of a simple robot or any system. For any system that requires a simple “brain” these Arduino microcontrollers are a great start to look at.

Arduino Uno

The most common microcontroller of the Arduino brands is the Uno. It is considered to be the most “documented board” of the “Arduino family” according to the main Arduino website. This is the microcontroller that our ADLTS team has and so we started with this one. It weighs in at 25 grams by itself, which is heavier than some of the smaller ones that we were considering. However, unlike some of the smaller ones, the inputs and outputs for the Uno do not require soldering, which allows for easy testing as developers and users can easily rewire by plugging and unplugging the wires into the slots.

Arduino Uno WIFI

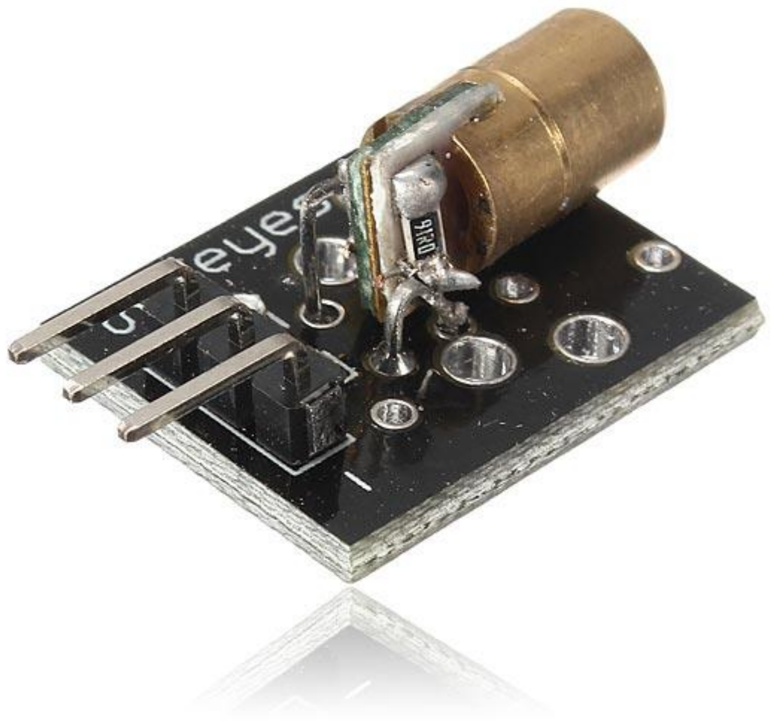


The Arduino Uno WIFI is one of the boards that our team is considering, as it has a WIFI AND Bluetooth capability build in already, so that makes it the perfect microcontroller to upgrade to when we want to work on our stretch goals. This board will allow ADLTS two systems to communicate with each other, as discussed in the stretch goals. The big issue that has stopped our team from upgrading to this board right from the start is that it’s twice as expensive as the regular Uno.

Laser Transmitter

There are many different lasers at different strengths. Some of the lasers are military grade, while others are smaller and meant for home projects requiring a distance of 10 feet or less. Because there are many different lasers out there the ADLTS team researched the ones that would be most appropriate for our project.

KY-008 Laser Transmitter Module



This 5V laser was chosen for its inexpensive yet powerful enough laser. Because it's a cheaper laser, at a distance of more than 10 feet the laser is very spread out. However, from our initial testing, we found that even though the laser was spread out, the laser receiver was still able to detect the laser signal. Even at really far distances where it was almost undetectable by the naked eye, the laser was still able to be picked up by the transmitter. The reason that the laser is so spread out is due to beam divergence, which is the process of a light being less bright and less concentrated at a further distance because of light wavelengths spreading out as the distance increases.

Laser Receiver Non-modulator Tube Sensor Module

We chose this laser receiver also for its cheaper price that was able to get the job done. The laser is detected by the square glass piece that sticks out from the module. From this, the laser receiver module will send the signal to the Arduino which would tell the microcontroller that there was a detection of a laser signal. This same module is able to receive data and send it to the Arduino, where the data would be decoded and determined on how to proceed.

Miscellaneous Wires and Capacitors

Some other hardware pieces that our team had to invest in are some misc. pieces such as the wiring and capacitors. These parts allow for the different modules to connect to the brain of the whole system with the right voltages as to not "fry" a vital piece.

Laser Divergence

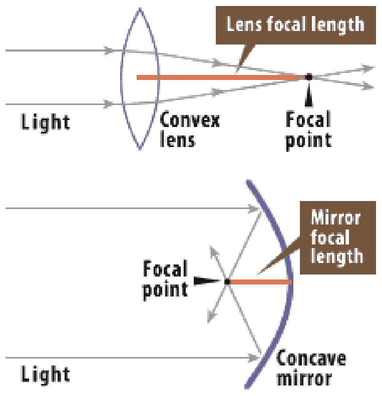
Some of the issues that require attention to when dealing with lasers is the divergence of the beam over distance. Every laser has an optimal distance where the laser is most concentrated and any further from the distance the laser will start to diverge. This divergence can be bad for certain applications, such as a distance finder or any other applications where precision is required.

In the case of beam divergence, the wavelength is often still the same, but it is more spread out. In other words, data in the form of the wavelength will not change, but it will be harder to detect and read. So it is important to consider this issue because at too great of a distance our laser receiver will have difficulty detecting the laser from our laser turret system. In the case of our tests, the distance will not be too great for our laser transmitter and receiver. Although there will be significant laser beam divergence, the transmitter is still able to detect the signal from our laser turret system.

Although this issue is not too relevant in our ADLTS system, we initially thought of ways to counter act the laser divergence. Our team initially thought of two different possible solutions in order to deal with the laser divergence.

Convex Lens

The use of convex lens is similar to the idea of prescription glasses that people with bad vision wear. The reason for their blurry vision is related to the light waves being spread out at a far distance, and when it reaches the eyeballs, the eyes are unable to concentrate the light into the center. Convex lenses refocus the light beams that were initially spread out and redirects them to the center of the eye, which allows for a clearer image. In the same way, our team wanted to use this idea in order to refocus any divergence of the laser beam. The convex lens would focus the laser beams that have diverged over distance so that the receiver will have a stronger beam to pick up.



This is similar to using a magnifying glass in order to make an object larger, except that its in the reverse. This situation is making an object that's too large and spread out, eg the laser beam, and making it smaller so that the laser beam is more concentrated and stronger for the transmitter to be able to pick up.

Also, by using multiple receivers that are all connected to the Arduino Microcontroller, it makes it more likely that the laser sensor will be able to pick up the laser from the turret system. While this would make the process easier by making a larger target, one of the things we had to take in consideration is the overall weight of the laser sensor system. Drones have a max carry weight capacity that they are able to lift, and this is why for our system we had to make sure we are within the safety bounds of our drone by and limit the overall hardware used and carried by the drone.

One-Way Mirror Ball

The other solution that our team had brainstormed had to do with one way mirrors and its ability to bounce a laser beam around. The one-way mirror would allow for the laser to pass through the glass, and then get trapped and redirected until it hits a laser receiver. Although this idea sounded good in theory, the main issue with it is that there is no true one way mirror. Although there are uses of “one-way” mirrors in different applications such as interrogation rooms, this concept only works due to the other room being slightly darker than the main room. In other words, the one-way mirror ball would end up refracting too much light away from the ball and not let much light in, because the light would be coming from the outside. So instead of strengthening our goal and making it more likely that the laser would be trapped within the ball to hit a receiver, it is more likely that the opposite would happen. In other words the light would be refracted and bounced away from the ball/sensor so much that it would end up making the laser signal weaker rather than stronger.

The other reason that we wanted to use a spherical sensor was to help the camera vision detect our target easier. This is because at any angle or elevation the target would still be a 2D “circle” in the “eyes” of the camera vision. This would allow our algorithms of the camera vision to easily detect our target and transmit that data to our laser turret. So although we might not be using the one way mirror, we will continue to try and implement the spherical object in one form or another. Instead of the sensors being on the inside of the one way mirror, this time it could be on the outside of the spherical target which would allow multiple angles to be targeted at the same time.

Drones

There were many different drones that our team was considering. The drones in the market all ranged for different purposes with wide ranging price points. So our team had to consider many different drones that would be the most appropriate for the studies of our ADLTS team.

The drones would range for different purposes such as hobby drones or professional drones that are used to carry high definition cameras used for movie production. These cameras are very expensive and actually require a state license in order to operate. These drones are not safe for indoor operations either due to their large size and updraft in order to life their weights. Most of these drones are illegal to be operated on private property and so these professional drones are inappropriate for our purposes.

Hobby drones can range from many different smaller drones that are relatively inexpensive. These small drones often do not have a camera attached and no extra weight carrying capacity. They are often used for a beginner to drones in order to practice flights or just for the sake of a hobby. These drones are mostly too small and due to their lack of extra weight carrying capacity are also no appropriate for our team. This is because the overall weight of the laser sensor system is approximately 100 grams, which includes the microcontroller and the battery as well as the other components.

So, our team researched different price ranges of drones that range in between a “hobby” drone and a “professional” drone. Currently we have a target drone called the Syma X8C which has a carrying weight capacity of 200 grams, which is more than enough for our sensor. The drone is also about $100 dollars as well, which allows for a relatively cheap testing drone for our purposes.

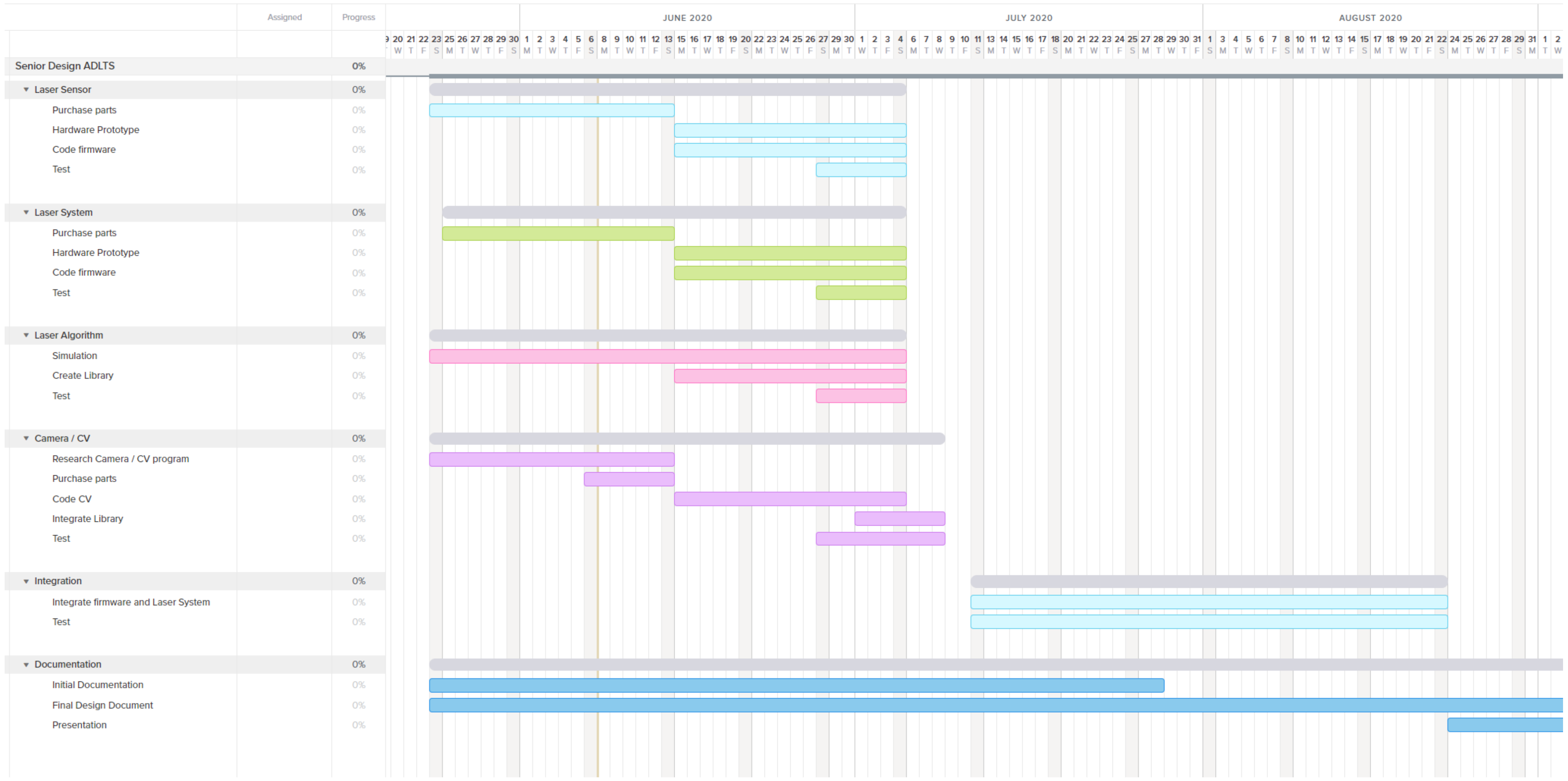
The Syma X8C, created by SYMA, is the drone that our team has chosen. It carries a 2MP camera which we do not plan on using. The main reason we chose this drone was for its 200g extra weight carrying capacity. Unfortunately, though, due to its overall weight 1.31

pounds, our team will have to officially register it with the state of Florida in order to operate. The legal laws of the state require all drones over the weight of .55 pounds to be registered.

Other drones were also considered for their price ranges and carry capacity, but after finding that this drone is more than capable for our purposes and considering the price it was officially chosen for our testing purposes.

Project Milestones

- 5/3/2020 Meet with Team and come up with common goals for project
- 6/06/2020 Research technologies and hardware needed
- 6/18/2020 Intermediate Check-In for SD1
- 7/04/2020 Purchase equipment and finish individual components
- 7/09/2020 Final Check-In for SD1 with Dr. Leinecker
- 7/12/2020 Finish 15 pages of final paper
- 7/25/2020 Finish 30 pages of final paper
- 7/30/2020 Edit and formalize Final Design Paper
- 8/22/2020 Finish integration of all systems
- 9/15/2020 Clean up and optimize systems
- 11/01/2020 Finish presentation preparation
- 11/05/2020 Practice with Team for presentations



	Assigned	Progress
Senior Design ADLTS		0%
▾ Laser Sensor		0%
Purchase parts		0%
Hardware Prototype		0%
Code firmware		0%
Test		0%
▾ Laser System		0%
Purchase parts		0%
Hardware Prototype		0%
Code firmware		0%
Test		0%
▾ Laser Algorithm		0%
Simulation		0%
Create Library		0%
Test		0%
▾ Camera / CV		0%
Research Camera / CV program		0%
Purchase parts		0%
Code CV		0%
Integrate Library		0%
Test		0%
▾ Integration		0%
Integrate firmware and Laser System		0%
Test		0%
▾ Documentation		0%
Initial Documentation		0%
Final Design Document		0%
Presentation		0%

Personal Takeaways

<u>Tommy To</u>

I surprisingly found this project to be a lot more complicated than I initially thought. Although I do suspect that most of the complication from my end was due to project management being a weaker skill of mine. This was the first time that I led a team for a large project of this size, and there were a lot of different things that I learned.

One of the lessons that I learned is that project management requires the ability to properly plan the timing of deliverables. It is easy to just set in stone the deadlines but it's a lot more difficult to be able to predict how long the tasks will actually take. This was one of the shortcomings of my project management, because I was not able to properly think about how long a certain task would take. For the most part I often overestimated how long something took, which would result in the team spending too much time on a single task.

Another lesson I learned was having to properly facilitate our team meetings. Our team meetings were particularly difficult when there was too much on the agenda or there was not a strict coherence to the meeting agenda. This would often result in some conversations that could have been discussed on a different meeting when some people had more time or understanding of expectations.

The final lesson I learned was that I personally was not flexible enough about team members and their different methods or styles of work. I realized that I had a lot of my own expectations of how things could be done, but not everyone is going to have the same mindset or solutions, and that I needed to be able to communicate better in this avenue.

Samantha Chou

My main takeaways from this project is that there were a lot more nuances relating to hardware than I initially thought. I always knew that it was a high learning curve that is different from pure programming, but the coding aspect for this project, in my opinion, was close to trivial compared to the hardware set up. When it came to the hardware there were so many different options and different capabilities needed to be considered. The different microcontroller boards and the determination of which one to use was surprising a lot more difficult than I thought. It wasn't as simple as simply picking any board and going through with it, because some boards were overly capable of carrying out the task but were too expensive.

In the case of the laser sensor that I helped develop, we didn't need a high capacity microcontroller, so we had to choose to cheapest yet most efficient board that is able to carry out the task. In other words, we simply did not need a supercomputer only to do simple arithmetic calculations. We didn't need a huge capacity microcontroller such as the raspberry pi or the Nvidia ones. So in the end I think that we did choose the best one, which was the Arduino Uno. Currently in our stage of the project, we haven't been able to get around to upgrading the board to the WIFI/Bluetooth Arduino board, but the good thing about upgrading in the future is that the code base isn't different between the boards. Although we would have to add additional code in order to transmit data via Bluetooth to main system, the current code base of receiving and decoding the laser signal can be unchanged between the two boards.

<u>Fernando Trevino</u>

For this project I reinforced my abilities related to problem solving, teamwork, time management, vector math, quality assurance, project design and integration at the simulation level. I had to analyze the different components of the project to create a meaningful simulation of the project to develop and test the middleware. Making the simulation early on allowed me to concentrate my time on improving the middleware to optimize results in different scenarios, for example in the case of needing to overcome limitations in the motors or detection system. My biggest take away is that there's always room for improvement, and we can't improve something without a measurement of quality.

<u>Aakash Gupta</u>

This project helped me to understand that a team need not to be around, in order to work together. However, it may be difficult or time consuming as you don't personally know anything about your team members, but this project especially in time of COVID-19, taught me how a good team can work in order to have best communication and time management skills. As there was no time for any of us to do some good research as I was lacking my access to physical books at libraries along with the good communications with professors; in this project, I learned that if resources are not enough, one can find its way by understanding the dynamics of problem and communicating with the team mates as it will be the greatest assert to one person. Apart from this, this project will teach me the synchronization of motors with different electrical components and how to control the minor step with less window of errors. It is crucial that most people works with motors, but it is important to see that how different motors can be integrated into one unit, such that a single motor does not come into picture as one motor, rather than it comes as a complete tracking system that undergoes into a constant communication among each other. This will give me vast view of understanding how does a thing in field of physics, computer vision, Machine learning, and many more works.

Jon Rohrback

This project turned out to be both challenging and exciting to work on. It allowed me to continue to improve skills such as teamwork, communication, and time management while also giving me the ability to expand my knowledge of computer vision. Due to the time constraints and amount of research that needed to be conducted, I didn't have too much time to really dive into the structure of all the different frameworks and algorithms so it became more about finding the pros and cons of each and then determining what suited our project specifications the most. Even with this amount of research, it is still difficult to find the most optimal solution until we start testing the different algorithms, for both speed and accuracy, against each other and making our decisions based on those results. The second semester will largely consist of running these tests for object detection and tracking until we find the one that fits our project's requirements. My biggest takeaway is that even if an algorithm might seem like a good fit for a given problem, testing and trying alternative solutions won't hurt and can actually help you gain a deeper understanding of why one might work better than the others. Nevertheless, everything I’ve learned while working on this project will help me get started in my career and be a foundation for my continued learning in the fields of computer vision, machine learning, and robotics.

COVID-19 Impacts

Due to the unprecedented circumstance of the worldwide pandemic, our team had to adapt from previous semesters versions of Senior Design. The impact was particularly hard on our team due to the wide dependence of hardware integration and the different components that make up the ADLTS project. Although we were able to efficiently break up and task the components, the team is still struggling with integration and doing that safely. COVID-19 affected our team members differently and we each had to adapt so that we can continue to contribute to the team and make ADLTS successful.

Tommy To

I've only ever had experience leading a team in person that I would meet on a weekly basis. In my experience in the past I was able to gauge the team environment easier and by spending personal one on ones with my team members I was able to learn of their strengths and weaknesses. However due to COVID-19 I had to change my leadership style and I had to figure out what was the most effective way of doing so. Communication through the internet, even though we used the Zoom meeting tool in order to voice chat, is still very nuanced and can be easy to misunderstand each person. This is due to the fact that the only signals you can read other than their words is their tone, and this could often be misunderstood.

Samantha Chou

The impacts of COVID-19 for me was the last of resources and safety concerns. Due to importance of keeping people safe, it was important that people practiced social distancing. With this in mind the UCF campus was not available to students, and all the students had to vacate off the premises and moved out of their dorms. Due to this, many peer resources were not available as they would have been otherwise. I personally learn best from being with people and learning in person, so it was tough for me to learn these new hardware tools all by myself during quarantine. Nevertheless, I believe I was able to succeed mostly because there are many online resources when it comes to Arduino and other projects exists that I could refer to.

<u>Fernando Trevino</u>

The pandemic definitely changed my lifestyle, but I was fortunate enough to keep my job during these unprecedented times. The main thing that the pandemic changed in my life that affected this project is that I had to move away from my apartment in Orlando and complicated meetings with my teammates face to face. This wasn't the reason for the need to create the simulation, but it played out well that I recommended making one.

<u>Aakash Gupta</u>

As an international student, I had to stay in this pandemic away from my family and friends, as many of my friends around has already left the town during the early stage of this virus. This has resulted in slowing down some of my work as I used to get emotionally drained more frequently. Apart from these I had really hard time looking up on the working of the motors and its drivers as I was unable to reach out to UCF library and some of my previous professors. Getting the availability about the motors and getting the schematic diagram from the company was also proved to be a hard task as delivery of the goods were directly impacted by this pandemic. However communication was one of the other problem caused by COVID-19, as I was unable to hold any face-to-face interactions with any of my teammates along with the innovation lab that could result in solving my doubts regarding the 3D-printing at UCF.

Jon Rohrback

Just like most of the other students, COVID-19 really threw a curveball at us while working on this project for our Senior Design. The biggest issues were not being able to fully utilize all of the resources that UCF offers and not being able to meet and collaborate with the other members of this group in person. With a project that has multiple moving pieces, it becomes even more important to be able to coordinate our efforts in ensuring everything will work together, but our team has done a good job of communicating with each other during this semester.

References

"Hough Circle Transform — OpenCV 2.4.13.7 Documentation." *OpenCV*, docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html

"A Closer Look at Object Detection, Recognition and Tracking." *Intel*, 18 Dec. 2017, software.intel.com/content/www/us/en/develop/articles/a-closer-look-at-object-detection-recognition-and-tracking.html

Swain, Anisha. "Noise in Digital Image Processing - Image Vision." *Medium*, 24 Nov. 2019, medium.com/image-

vision/noise-in-digital-image-processing-55357c9fab71

Saxena, Pawan. "R-CNN vs Fast R-CNN vs Faster R-CNN | ML." *GeeksforGeeks*, www.geeksforgeeks.org/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-ml

Khandelwal, Renu. "SSD : Single Shot Detector for Object Detection Using MultiBox." *Towards Data Science*, 30

Nov. 2019, towardsdatascience.com/ssd-single-shot-detector-for-object-detection-using-multibox-1818603644ca

Hui, Jonathan. "SSD Object Detection: Single Shot MultiBox Detector for Real-Time Processing." *Medium*, 13 Mar.

2018, medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06

Choudhury, Ambika. “Top 8 Algorithms For Object Detection.” *Analytics India Magazine*, 16 June 2020,

analyticsindiamag.com/top-8-algorithms-for-object-detection

Sharma, Nikita. “Introduction to Basic Object Detection Algorithms.” *Medium*, 18 Dec. 2019,

heartbeat.fritz.ai/introduction-to-basic-object-detection-algorithms-b77295a95a63

Hui, Jonathan. “Object Detection: Speed and Accuracy Comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet

and YOLOv3).” *Medium*, 27 Mar. 2018, medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

Mallick, Satya. “Object Tracking Using OpenCV (C++/Python).” *Learn OpenCV*, 13 Feb. 2017, www.learnopencv.com/object-tracking-using-opencv-cpp-python

Rosebrock, Adrian. “OpenCV Object Tracking.” *PyImageSearch*, 18 Apr. 2020,

www.pyimagesearch.com/2018/07/30/opencv-object-tracking.

Bewley, A, Ge, Z, Ott, L, Ramos, F & Upcroft, B 2017, ‘Simple Online and Realtime Tracking’, *Queensland*

University of Technology, University of Sydney, pp. 1-5

Maiya, S. (2020, April 24). DeepSORT: Deep Learning to track custom objects in a video.

https://nanonets.com/blog/object-tracking-deepsort/

Bochkovskiy, A. (2019, February 18). AlexeyAB/darknet. https://github.com/AlexeyAB/darknet

Ponnusamy, A. (2019, October 6). Preparing Custom Dataset for Training YOLO Object Detector.

https://www.arunponnusamy.com/preparing-custom-dataset-for-training-yolo-object-detector.html

Enlin, C. (2019, May 25). Chuanenlin/drone-net. https://github.com/chuanenlin/drone-net/tree/master/images

Advanced 256 microsteps integrated motor driver with step-clock and direction interface. STMicroelectronics. December 2017. https://www.pololu.com/file/0J1609/STSPIN820.pdf

Blue Sea Systems. IP Rating Chart. Nov. 8, 2018. https://www.bluesea.com/resources/117

GPIO. Raspberry Pi. https://www.raspberrypi.org/documentation/usage/gpio/

IPC . IPC Standards. Nov. 8, 2018. http://www.ipc.org/ContentPage.aspx?pageid=Why-Should-OEMs-Use-IPC-Standards

Raj, Aswinth.What is Stepper Motor and How it Works. Oct. 01. 2018. Circuit Digest. https://circuitdigest.com/tutorial/what-is-stepper-motor-and-how-it-works

SPI. Raspberry Pi. https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md

STSPIN820: Microstepping Management. STMicroelectronics. October 2019. https://www.st.com/resource/en/application_note/dm00647202-stspin820-microstepping-management-stmicroelectronics.pdf

"Rotation Matrix." *Wikipedia*, en.wikipedia.org/wiki/Rotation_matrix

Ortiz, Elijah. "FREE PACK" *Unity Asset Store*, 29 May 2018, assetstore.unity.com/packages/tools/physics/free-pack-117641#description

"Manual." *Unity Game Engine*, docs.unity3d.com/Manual/index.html

Epic Games. "Documentation." *Unreal Engine 4*, 2020, docs.unrealengine.com/en-US/index.html

Linietsky, Juan. "Godot Docs." *Godot*, 2020, docs.godotengine.org/en/stable/index.html