Requirements Specification

May 1, 2022

Team Sponsor: **Alexander (Allie) Shenkin**

Team Mentors: **Tom Prys-Jones, Anirban Chetia**

Document produced by

Austin Malmin, **Conrad Murphy**, **and ShanHong “Kyle” Mo**

Table of Contents

1. Introduction

In the world of automation, normal mundane tasks are being replaced by artificial intelligence that can perform these jobs at higher efficiency and with better accuracy. The world is starting to see a breakthrough in the field of robots and drones that can move independently of an operator, which are referred to as autonomous. Today, humans are currently performing tasks that can easily be automated with self-guided drones. Search and rescue missions can prove to be dangerous in places not easily navigated by humans, such as collapsed caves, house fires, war zones, etc., but a drone could make these missions safer and more efficient. Amazon has already taken use of autonomous drones in their delivery of goods throughout the United States and the United Kingdom. The dusting of crops with drones and planes piloted by humans can be inaccurate and expensive, whereas autonomous drones offer precision and require no manual labor which cuts down on costs. Our focus is the study of rainforest ecosystems through the use of autonomous drones.

Mapping the upper and lower forest canopy currently requires a researcher to manually move a ground-based sensor through the forest which is time-consuming and difficult. The sensors used are **Light Detection and Ranging (LiDAR).** LiDAR sensors measure the time for a laser beam to return once fired to determine the distance of the object from the sensor and create a 3D model of the environment, whether on land or on the seas. Ecologists perform research focused on how forests around the world respond to climate change and simulate the effects certain changes in the climate would have on the environment. Team Mockingbird is working with Dr. Allie Shenkin to build an autonomous drone navigation system.

2. Problem Statement

Dr. Alexander Shenkin studies forest 3D structure, and his current workflow involves traversing the forest on foot using a **LiDAR sensor** on a tripod to scan the canopy from the ground level. Groups of researchers trudge through the forest until they find an adequate spot, at which point they will set up the 360-degree LiDAR device to map the

nearby surroundings. Each LiDAR scan comes from a single vantage point. Therefore, researchers must repeatedly traverse the forest and find numerous different vantage points in order to create a satisfactory point cloud map.

There are many problems with this current workflow that render it unsatisfactory. Firstly, the process takes an unreasonable amount of time. According to Dr. Shenkin, a hectare (100m x 100m) of the forest requires a full week for a team of 2-3 people to cover with a ground-based LiDAR. Secondly, scanning from fixed points at ground level is a bad way to scan the details of tall trees. Surrounding overgrowth hinders the line of sight to the upper canopy, which results in some details being missed. As a result, forest ecologists receive an incomplete picture of 3D forest structure. Since forest structure dictates how the forest functions, such as a tree's resilience to climate change and the strategies employed to survive, incomplete data results in an incorrect assessment of what the forest is doing. A more detailed picture of forest function is important because it would allow researchers to more accurately simulate these functions and determine how the forest ecosystem responds to changing variables in the environment.

Dr. Shenkin wants an alternative solution that takes less time and hassle while providing more accurate results, all of which can be achieved using autonomous drones.

3. Our Solution: Self-Autonomous Navigation System

In order to resolve the issues inside our client's workflow, we have been tasked with creating a navigation system for a self-autonomous drone. This system will detect objects in its path and find safe directions to travel in order to maneuver around them. Over time, the system would build a map of the space to be saved as data for his research.

An autonomous drone provides a faster, easier, and more effective alternative to Dr. Shenkin's current workflow. Instead of a team of researchers, a drone would be able to

scout the plot of forested land with no outside intervention from human crews. The drone would be capable of flying up into the canopy, collecting data that is inaccessible to researchers confined to the ground level. Additionally, while a handheld LiDAR needs to be mounted in fixed spots throughout the forest, a drone with a sensor can take rapid, continuous snapshots as it moves. This drastically increases the speed at which the mapping process is completed while ensuring a more complete picture of the forest environment.

Our system's hardware consists of a Raspberry Pi and a **LiDAR sensor.** The Raspberry Pi is a minicomputer that will serve as the main processing power for our navigation system. It will be fitted with **Robot Operating System (ROS),** a standard platform for use in robots and autonomous drones. ROS supports a number of different software packages and libraries that enable a computer to convert raw scan data from the LiDAR sensor into a usable format. The scan data is utilized to perform **Simultaneous Localization and Mapping (SLAM),** a procedure that generates a map of the surrounding environment in real-time. The map can then be used to detect and avoid obstacles. The system will also store the map as a **point cloud** file on an onboard SD card.

4. Assumptions Given Budget Uncertainty

Throughout this paper, our team was required to make assumptions given the uncertainty of our budget and the availability of hardware components due to supply chain delays. These assumptions may have to change as the project progresses; however, these assumptions must be made in order to complete this document. We have made the assumption that a 2D LiDAR will suffice for the needs of this project's first steps. The sensor we will be using is an RPLIDAR A1M8-R6 - 360 Degree Laser Scanner (Figure 4). However, recent experimentation with the hardware reveals that a 2D LiDAR does not meet our needs. Moving forward, we will experiment with other options for sensing objects in 3D space. Another assumption made by this team is that the power required from the system will be provided by the drone. However, in order to

get a working product, we will not have access to a drone, so our system will be powered by a battery. All of these things will be detailed heavily inside this paper but will be subject to change.

Figure 4 Hardware Visualization

Figure 4: An RPLidar A1M8 sensor which we assume is a given for our project.

5. Project Requirements

For this project, our team has broken down the requirements into 3 distinct sections: functional, performance, and environmental. Functional requirements are what our system is going to do. Each of these different requirements is broken down in detail in section 5.1. Performance requirements describe the minimum acceptable performance of our product and will be detailed in section 5.2. Lastly, the environmental requirements are the constraints imposed on the team by the physical environment and the users. These will be detailed in section 5.3.

5.1 Functional Requirements

Overall, our navigation system must be capable of guiding an arbitrarily-sized drone through the understory without crashing. This means it must detect objects of very small sizes and maneuver around them if they will pose a threat to the drone. The drone should also create a point cloud map of the space and store it in memory for future use. In addition to these considerations, the system must be compatible with existing drone hardware and be able to interface with its movement system built on ROS. Since our

system will not be used on a physical drone within the scope of this project, it will simply display a basic output of the direction that the drone should move.

5.1.1 Minimum Requirements

The minimum requirements are those needed to create the most basic viable product for our project specifications. In this section, we list three basic functions that the drone must be able to perform, then break these functions down into specific instructions and algorithms within the software.

5.1.1a Automatic detection of objects

This is the most important minimum requirement as this is the starting point of the project since detection of objects as a requirement will be used in other requirements.

Per our starting assumptions, the navigation system will use a **LiDAR sensor.** A LiDAR sends out many laser pulses in the form of photons at many different angles depending on the make and model of the sensor. In our case, the RPLidar A1 has an angle of about 1 degree between each beam. Even though the degree difference between beams seems small, as the distance that the LiDAR needs to scan increases, the distance between each beam also increases. As a result, the LiDAR might miss small objects positioned in between pulses. That is why it has a guaranteed data accuracy for a range of 6 meters, even though it can scan up to 30 meters but with less certain accuracy as detection range increases.

When a laser pulse strikes an object, it will reflect back to a receiver within the sensor. A point in space is generated based on the time that it took for the photons to arrive back to the receiver. For example, if it took the photon 0.01 seconds to come back to the receiver while another photo took 0.02 seconds to come back, then the 0.01 point will be closer to the LiDAR and the 0.02 point will be twice as far away. Each point is stored within a **point cloud** in the form of its XYZ coordinates. Currently, we are using a 2D LiDAR, which means that even though it only scans the XY coordinates, it will output the XYZ coordinates with the Z value set to the default value of 0.

Because our sensor and its relevant vision libraries depend on the Linux operating system, our navigation system will be connected to a Linux-based computer, such as a Raspberry Pi. While waiting for the Raspberry Pi to arrive, we will use a Linux virtual machine (an emulated copy of the Linux OS which can be run within a different machine) to operate the sensor. This computer will have a vision library which performs **Simultaneous Localization and Mapping (SLAM),** called HectorSLAM. This vision library takes in data from the sensor's receiver and calculates the position of each point, accounting for how much the LiDAR sensor has moved from its starting position. For example, if the LiDAR moves north, HectorSLAM updates the LiDAR's position based on how far north it has moved since it was powered on. If it begins moving east, the library offsets the LiDAR's position east as well. Point clouds are displayed on a map with a fixed size. For example, if the map display is set to be 3000 pixels (which is 10 inches on a computer screen), the distance that was calculated will have to fit in the 10-inch map. So if an object is 32 meters away, it will be drawn close to the edge of the map, while an object that is 1 meter away will be closer to the center. Figure 5.1 summarizes how HectorSLAM draws the map. In summary, all systems (sensor, vision library, and ROS) will function together to create one instance of a map. Every time the LiDAR sweeps across all 360 degrees of space, a new map is created. These map instances are all overlaid onto one main map.

Figure 5.1 How LiDAR Sensors Generate Images

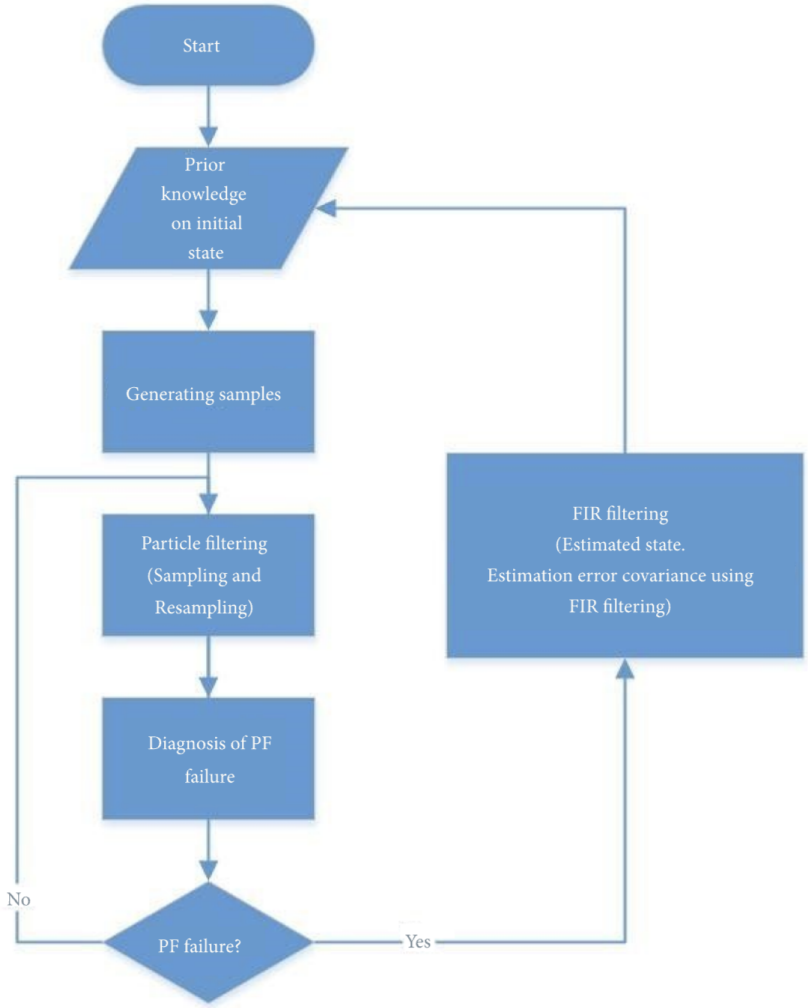

Figure 5.1: How the vision library and sensors generate images. PF stands for particle filtering

Link: https://www.hindawi.com/journals/jr/2018/7806854/

The vision library and the sensor (RPLidar A1) mentioned above will also need to work with **Robot Operating System (ROS)** as well as the computer that the team is using. ROS interacts with the Linux computer via the command line. There are commands to launch ROS, launch certain packages and libraries within ROS, etc. Some of the vision libraries also open GUIs which can be used to control certain display options. For

example, a certain button on HectorSLAM's GUI can be toggled to begin generating maps. A binary value of 0 is used to denote False, and a 1 represents True. When a map is not being generated, the setting is set at a "0", but when the user toggles the option to generate a, the "0" will be replaced as a "1" meaning that it wants the system to generate maps.

Other than the details already mentioned with the interaction between ROS and LiDAR with the vision library, it will also remember the maps. As mentioned above, the maps are generated many times with new ones layered on top of each other. The system will remember details about places visited previously, so it will register certain clusters of points as objects. This data is very crucial, as it will support automatic detection of objects and eventually play a role in creating safe routes throughout the environment (a stretch goal covered in section 5.2)

5.1.1b Move towards a destination

While detecting and mapping the surrounding environment, the system must have goal-oriented movement towards a specific direction. This means the drone must be capable of making overall progress in a certain direction. The drone may take one step back to get out of dead ends as long as it can move two steps forward later on.

Using the point cloud created, there will be a method in the ROS (with the support of the sensor and its vision library) that will create a path that is mentioned in the requirements for automatic detection of objects (as mentioned above, the sensor shots thousands of beams per second, meaning that even if it is moving, it will still produce an accurate map of its surroundings). Since the map has a point cloud that is normally black (which can be changed to another color based on the user's preference), it also means there are white spaces that aren't part of the point clouds that represent empty space. The goal is to stay within the white space and avoid black points. There are many ways to tackle this problem but they all generally come down to the same idea of principle which is to calculate the distance between the same object. As stated above, the map is only X inches long and wide on the computer which represents a certain amount of pixels

(10 inches means 3000 pixels), and those pixels will represent the distance between objects (for example, one pixel will represent 1cm in some cases while sometimes will represent 2cm depending on how far we measure). One method of doing this is to calculate the distance between the black point clouds, using an example, in the codes, we can set the drone as 100 pixels long and wide and create a function that calculates if the drone can fit or not by returning a binary number such as “0” and “1” where “0” is false and “1” is true. This function will take the drone’s pixel length and the pixel length of distance between two point clouds and if the pixel between two point clouds is greater than the pixels of the drone, it will return a “1” as it will represent it will fit otherwise it will return a “0”.

Our system will take a greedy approach to move in the specified direction. The drone will make as much progress as it can in the preferred direction, as long as it doesn’t detect obstacles that cannot be avoided without backtracking. This means that as long as it doesn’t detect an obstacle, it will have permission (the function will return a “1”) to continue to move in that direction. If the drone does reach an obstacle (set the function to return a “0”), our system will have a function that will find whether to move left or right based on point clouds generated and calculated by finding a path that has enough space for the drones to travel (so the function will return a “1” again when found a suitable path to take). And it will continue to do so until it reaches a dead-end, meaning it can’t go forward or to the sides and can only go backward. Because the system remembers the path that it took previously, the drone will go back to the previous area where it made a decision to go a certain path, for example, if previously encountered an obstacle where it can go right and left and the system chose “right”, this time it will go back to that place and will choose “left” (we will set that if there are two options of right and left, the system will always pick right first). If the previous section only had one choice of left and it already went left, it will go back to the previous section before this section that it made a choice, it will continue to make this decision until it reaches the destination that it was meant to go.

5.1.1c Output a direction in which to move

Our system must periodically output a direction in which to move. This direction will be defined as an angle relative to the drone's current facing direction. The direction can be further generalized into a simple Forward, Left, Right, and Backward based on which quadrant the angle occupies. The type of output (angle or cardinal direction) can be chosen by the user by including a flag in the command line when run. Since our project assumes a 2D LiDAR, we do not need to worry about 3D movement up and down for now.

Drones running ROS have a pre-existing interface to fly the physical drone in a specified direction. This interface contains functions that, when called, fly the drone in a certain direction for a certain amount of time. The fine-tuning of how fast each motor should spin in order to move in any given direction will already exist on the drone. This means we do not need to invest time into developing the movement algorithms ourselves. Also, we will not be working with a physical drone within the scope of this project. Instead, our hardware will be mounted onto a piece of wood or another hard platform as a replacement for the drone. Therefore, our directional output will be left as a simple print to the command line. Human operators will move the platform in the direction the system wants to move in order to simulate drone flight. If the platform can be moved throughout the space according to our system's orders without crashing into obstacles, then that is an indication that our project has succeeded. In the future, when the scope of the project broadens to incorporate a real drone, our simple output can be translated into a function call to the drone's interface.

5.1.2 Stretch Goals

Following the completion of the core requirements, the following stretch goals are listed in feasibility from most to least. The first stretch goal is storing the 2D map onto an external SD card. The second stretch goal is that the drone will display a planned path to the user before actually doing it so the user can interrupt it in the case that is not what the user desired. The third stretched goal is that if we are to give a designated area to

the drone, the drone has to be able to avoid crashing into obstacles and visit as much 3D space as possible to get a detailed map of the area.

5.1.2a Store and Display/Print the area visited

As the drone moves throughout the environment, it will be stitching together a map of the environment by plotting the point clouds given from the LiDAR. This process can be viewed in real time via HectorSLAM's graphical interface. Once the flight is complete, the map from HectorSLAM will be complete. This map needs to then be saved and written to the SD card. The map is saved as a list of all points in the point cloud.

5.1.2b Show a Display Screen of the Planned Path

Please read the minimum requirements for automatic detection of objects as this requirement is built upon the previous requirements, meaning a lot of information is relevant and important.

Before the drone takes off or starts navigating, the sensor will send photons out and receive information based on the photons that got reflected back(explained in 5.1.1 and 5.1.2). Since the system can already detect an object and determine if it can move left or right depending on if the open space is big enough for the drone to move into, all we have to do is have a function that can draw all possible paths that the drones can take by measuring the open space. All this function will do is return the possible path that the drone can take to the user's screen. When this is completed, we will add a function that can override the system (remember that the drone will always take right when it meets an obstacle in front of it) to allow the user to determine the direction for the drone to move to the user's preference.

5.1.2c Visit as much of the designated 3D space as possible

After giving the drone commands, with the ability to detect objects, the drone will move in the XYZ (3D) plane by avoiding objects and stopping at the point that it was designated. Due to limitations on our budget, this will only happen in a 2D plane as the Z coordinate will always be 0. If the drone detects an object, it will stop and the user has

to manually give commands to make it move around it. The way that the drone will visit as much 3D space as possible is a working progress but the idea is that based on the previous requirement of showing a display screen of the planned path, we will use the map where the black/red dots are point cloud that the drone can't access and the knowledge of the space that is big enough for the drone to go through, the system will have a flag saying true or false for if a path has already been crossed through if it has, we will probably color the path so that the user can look and tell that drone already look through this area and will then go and continue to go to the path that hasn't been colored red yet (the colors so far is hypothetical to make a point).

5.2 Performance Requirements

Our sponsor has listed several exact specifications for our navigation system's performance in order to minimize crash risk and increase mapping efficiency. The system must detect obstacles of a certain size, scan a certain field of view within the space, and scan up to a certain distance. Additionally, the memory writing component must write fast enough to keep the system operational and not stall the drone.

5.2.1 Obstacle Detection Size and Range

The jungle contains many small branches, lianas, and vines, all of which could damage or destroy the drone if crashed into. To navigate the thick jungle environment, our system must detect and avoid obstacles of width 1cm or larger from a distance of at least 1 meter in advance. In response to the presence of these obstacles, the system must be able to find spaces that are large enough to accommodate the drone's size.

Based on our feasibility research, some LiDAR sensors are sufficient for detecting these small objects, and some are not, depending on the cost and quality of the sensor. We will test the accuracy of our 2D LiDAR in upcoming stages of the project. Our testing will involve placing a small object, such as a marker or pencil, and attempting to scan its location with our LiDAR sensor. If the sensor can consistently detect the object from at least one meter away, then the sensor will be satisfactory for our needs. If this is not the

case, then we will explore other types of sensors and find one which will perform to the project's standards.

5.2.2 Field of View

The autonomous drone must be capable of identifying objects situated 20 degrees above and below it to ensure that it is safe to maneuver. This requires that the drone has a field of view that spans both a horizontal and vertical distance. By extension, the system must also be capable of plotting points in 3D space to account for this field of view.

Unfortunately, our low budget limits us to only a few options for LiDAR sensors, meaning it is very unlikely that we will have a sensor that rotates both horizontally and vertically. Our current working budget is expected to be around $500, allowing us to purchase a LiDAR sensor that only scans across a 2D plane. We had plans to periodically tilt the mounted LiDAR sensor up and down during flight to sweep over a 20-degree field of view. However, recent testing with a 2D LiDAR reveals that the sensor is not capable of detecting 3D motion or plotting in 3D space. Therefore, it is entirely impossible to scan a 3D field of view with our current hardware. There exist LiDAR sensors that can fire pulses at slight vertical angles; however, these products often range in the tens of thousands of dollars–far beyond our budget. We will continue to search for hardware which suits our needs during upcoming steps in the project's development.

5.2.3 Memory Write Speed

While flying and generating its point cloud map, the drone must be able to write the map into permanent storage. The map will be stored as a point cloud file which can then be exported to other systems for external use. For instance, a completed map of the rainforest could be imported into a virtual simulator or game engine, where a researcher could perform a wide range of experiments modifying the environment such as testing how different rainfall levels in the environment impacts the environment.

The memory writes speed should occur at a minimum of 100 MB/s (megabytes per second). Our sponsor expresses relatively little concern for memory write speed; however, it is possible that a drone with low write speed could stall in the air, wasting time and processing power writing to storage instead of collecting new data points. Therefore, the 100 MB/s limit is a suitable baseline for anticipating and preventing this problem.

5.3 Environmental Requirements

The environmental requirements are constraints imposed on the project by the drone's physical environment, the available technology, or the client. Each environment has its own requirements which will be covered in this section. Write speed could be considered an environmental requirement as it's something that the client requested but it has been covered under the performance requirements so it will be left off this section.

5.3.1 Physical Environment

For the physical requirements, the weather must support drone flight. This includes but is not limited to: no extreme winds, no rain or snowfall, and adequate light present. The LiDAR sensors must avoid direct sunbeams which are capable of causing damage to the inner components. This is only a concern near dawn or dusk when sunbeams come in laterally. Another requirement given to this team by the sponsor is that we use ROS. This is due to the fact that ROS is compatible with any drone movement system that this system may be used to navigate. Another thing to consider is the total weight of our hardware. Mounting extra components onto the drone will cause it to expend more energy to lift the additional weight. The weights of the LiDAR and Raspberry Pi are known: The LiDAR weighs 170 grams and the Raspberry Pi weighs 46 grams. The weight of the mounting gear will depend on the medium chosen. In order to power our prototype, which is separate from a drone, we will be using an external battery; this does not need to be included in our weight calculations as the system assumes it draws power from the drone battery.

5.3.2 Technological Environment

As for the technological environment requirements, it has been determined by this team inside our technical feasibility study that the Raspberry Pi is a requirement as there are no viable alternatives to the Raspberry Pi. The power that our system will require is also a requirement as this system cannot draw extensive power from the drone and take away from drone flight time. Our system will shorten the flight time but it must not hinder the drone completely.

6. Potential Risks

There are many risks associated with our project because we anticipate that this navigation system will be used in real fieldwork. We are programming a drone to navigate and scan surroundings so there are many risks involved with the team, the drone (hardware and software) and the environment. This section will be broken down into risks when our navigation system is attached to a drone, as well as risks unrelated to a drone.

6.1 Risks Related to Drones

The following section will detail the risks that relate to drone collisions. The first risk is collision damage to the drone itself; the second risk is injury to human operators or bystanders.

6.1.1 Damage from Collisions

Damage from collisions poses a danger to both the drone and its surroundings. Crashes can happen because of many factors that can happen at any moment–for example, a sensor malfunction where the drone can’t detect an object and crashes into it, or a connection interference which will immediately cause the drone to stop receiving commands. The drone has many sensitive components, such as sensors and rotors, that can be damaged and will be extremely costly to repair or replace. A drone collision involving a human could also result in severe injury to the human. While these risks

have a low likelihood (3), they have high severity. This is because the drone parts are expensive to replace and repair, as well as because it can damage the environment and injure people. These risks are given a low likelihood as the parts for these drones have undergone testing and have all passed before being deployed. Nonetheless, if the user flies the system under improper conditions, then the likelihood of collisions will increase.

6.1.2 Mitigation of risks for/from drones

- Let a professional pilot the drone or learn it from professionals
- Check drones before it flies such as battery life, connections to the computer, sensors attachments, etc
- Pilot the drone when the environment is clear of unwanted objects
- Fly the drone only when the environment is suitable, for example when there are no rain, lightning, wind and etc

6.2 Sensors Producing False Data

Every sensor has a weakness. For instance, our 2D LiDAR sensor has trouble detecting objects at very close range (<30cm), and it also tends to miss objects that are far away if the laser pulses are fired at angles which miss the object. Sensors can also be damaged; if so, they will need to be replaced or repaired. Any damage to sensors will produce errors in data. Out of the box, some sensors can be faulty. In order to avoid this, our team tested our sensor before implementing it into the system. Now in the event of a sensor producing a false reading the following can all happen:

- Wrong data could lead to wrong information produced for the sponsor
- Require new sensors
- Cause a collision

Due to these outcomes this has been given a severity score of 8. Not only can this lead to inaccurate research for our sponsor but it also can lead to damage to the drone and sensor.

6.2.2 Mitigating Sensors Producing False Data

In order to mitigate the risks detailed in 6.2, the user should perform a visual inspection of the sensor before each flight to ensure no physical damage has occurred. If this is the case do not fly the system and replace the sensor. The user should also fly the system in proper conditions as direct sunlight or direct light into the sensor can provide false readings and damage the sensor.

6.3 Damage from LiDAR Laser Pulses

LiDAR utilizes laser beams and the time of return to create a mapping of the environment but what happens to the objects they impact and reflect off of. In most cases, the reflection causes no damage as the object is wood or metal but there are some impacts that can cause damage such as human eyes and cameras.

6.3.1 Damage to Humans from LiDAR

Straight contact of the LiDAR laser beam with a human eye can cause vision loss, headaches, and blindness. While this does have a very low likelihood (3) of occurring due to the drone mapping a large 3d space it has such a large severity score (10) with such easy prevention it is mentioned in this paper.

6.3.2 Damage to Cameras from LiDAR

Similar to human eyes, cameras also can be damaged by the laser beams from LiDAR sensors. This does have a higher likelihood(5 to 6) of occurring as a camera around our system will be more likely actively taking a picture of the system exposing the camera to the laser beams. In the event of direct contact with the camera lens and a LiDAR laser beam, the camera will be rendered useless. This has a low severity score (2) as a camera can be replaced but can also be easily avoided.

6.3.3 Mitigation of Damage from LiDAR Lasers

Mitigating damage to both human eyes and cameras is easy and not costly. Simply wear proper eye protection and cover camera lenses when in parallel with the drone. This will not allow the laser beam to penetrate the vulnerable surfaces.

6.4 Loss of Team Member(s)

Because each team has a role, any member leaving or not doing their job means more work for other members. Because we are using electrical equipment and a lot of hardware that could harm each member, members that use the equipment must be able to know the basics before doing it.

- Members must be trained in the equipment that they are using
- Members must wear protective equipment if it is needed
- Members that might leave the team must inform the team so the team can have measures for it

While this is unlikely as everyone needs to complete this course to graduate it did happen to this group already and we have given this a likelihood score of 3. In the event we lose another member, the group would suffer heavy losses. Each member has their role, losing members means someone has to pick their role up and responsibilities.

6.4.1 Mitigation of Losing Team Members

As detailed in the course the members have a process to express concerns and to give any team member a chance to respond. Before this action needs to be taken, internal team communication is essential to not lose a team member.

Figure 6 Summary of Risks

Risks	Likelihood 1-10, 10 is going to happen	Severity 1-10, 10 is most severe
1. Damage to the drone from collision	3	10
2. Injuries to the user or bystander due to collision	3	10
3. Sensors might produce data errors	4	8
4. Sensor that uses laser could damage people's eye	3	10
5. Sensor that uses laser can damage cameras	5-6	2
6. Losing team member	3	10

Figure 6: A summary of the risks that the project presents.

7. Project Plan

Our project plan involves budgeting, ordering, and testing our hardware in order to devise a prototype demo of our system.

So far, we have discussed budget restraints among our sponsor, Dr. Shenkin, along with the Capstone coordinators. Based on the project requirements and findings in our feasibility study, our budget is a severe limiting factor for obtaining and testing higher-quality LiDAR scanners with our system. For instance, the LiDARs that we researched in our feasibility study only swivel horizontally; this is not as ideal as a 3D LiDAR which can sweep vertically as well. Because of these limitations, we will need to reconsider our choices of hardware, such as using a camera instead of a LiDAR. In the best case, we could reach out to a hardware distributor and get them to donate or lend

one of their products. In the worst case, we would need to design additional features into the navigation system in order to work around our limited equipment.

While waiting to receive our final budget, we plan to purchase some pieces of equipment that we know we will need. For example, we have ordered a cheap RPLiDAR A1 which we will use for our demonstration. Although not ideal for our final model, the LiDAR will allow us to test our vision libraries with live data. This model may be returned and exchanged for a higher-end model if the budget allows. We will also order a Raspberry Pi, which is currently low in stock. Shipping times prevent us from receiving one until July. Another example is an SD card. This will be relatively easy to purchase, but memory storage is a secondary concern and therefore not something we need to address until later in our project schedule.

By the end of this semester, we will need a demo navigation system that we can present to our mentors. Since we have a LiDAR sensor ready for use, we can showcase point-cloud SLAM occurring in real time and demonstrate how the system reacts to nearby obstacles. The SLAM demo will take place on a computer running ROS.

Since we are new to ROS, we will meet with **Dr. Truong Nghiem** early in the fall semester for an introduction on how to work with ROS. This introduction to ROS will likely involve utilizing ROS's interface as well as the drone's existing motion system. The drone can already move in whichever direction the user commands it to move, so we will only need to know how to give the drone directions using ROS's interface.

Figure 7a summarizes our project schedule for the end of the Spring semester, and Figure 7b summarizes the schedule moving forward into the summer and fall months.

Figure 7a Project Schedule Gantt Chart

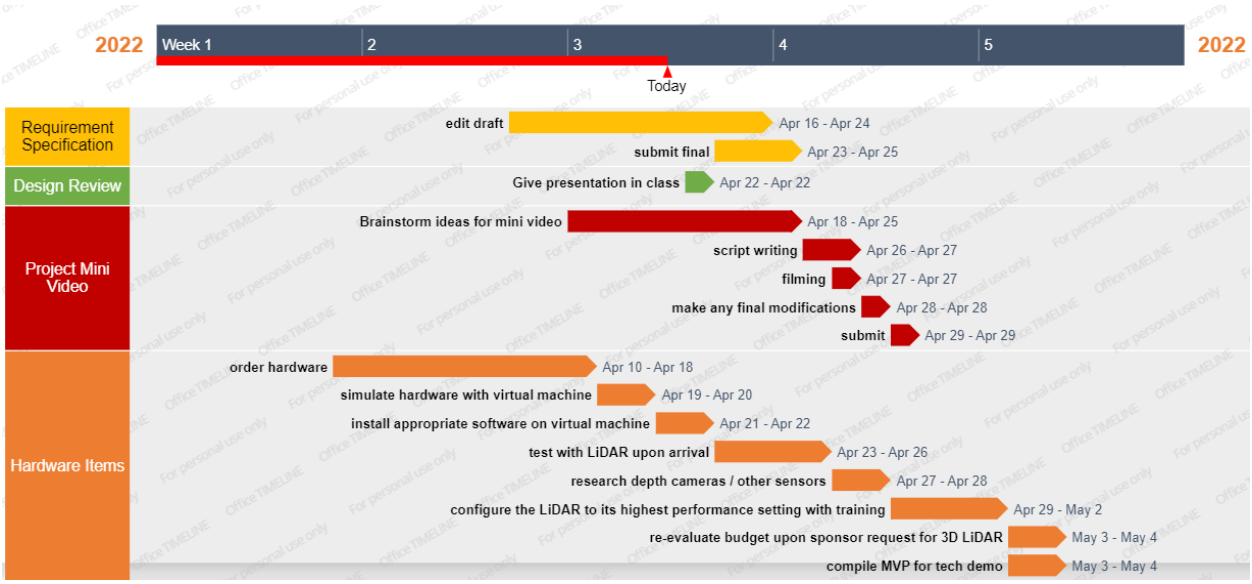

Figure 7a: A graphical summary of our semester project schedule.

Figure 7b Summer Schedule Gantt Chart

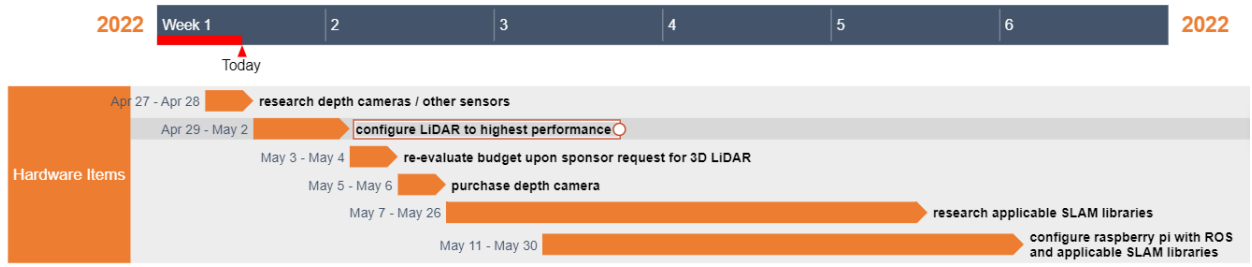

Figure 7b: A graphical summary of our summer project schedule. Starting at the end of April.

As stated in section 4, the information above is also subject to change as the project progresses. This is largely due to budget constraints. The timeline of this project is also thrown off as COVID-19 is affecting the availability of hardware such as a Raspberry Pi and certain LiDAR sensors. This disruption of the supply chain may push back the desired timeline for this project.

8. Conclusion

Autonomous drones are a major advancement in many different fields of study and service, including the study of intricate details in the rainforests. Dr. Alexander Shenkin is a researcher and climatologist who is attempting to create detailed maps of plots of forested land. His current workflow is slow and tedious, requiring large investments in time and manpower in order to create a map that is not even complete. This approach is unwieldy and unsatisfactory, and a better solution is needed.

Our mapping system will greatly improve Dr. Shenkin's workflow in his study of the rainforest, allowing for faster and more efficient data collection from our ecosystems. This data will help us understand our environment much more intricately, revealing important conclusions such as the effects of climate change and the mitigation of carbon in the atmosphere.

This document lays the foundation for everything we will need to create in the coming months. Following the completion of our requirements, we will continue to test available hardware and search for other hardware options that better suit our goals. We will also continue to familiarize ourselves with ROS and its vision libraries.

If we fulfill our goals, we will have a powerful module for automated navigation and mapping which could serve as the foundation for other data analysis tools. Dr. Shenkin could mount the drone with other types of sensors that measure light, humidity, heat, and other environmental factors, allowing for a more complete understanding of the environment's structure and function. Thanks to the drone's maneuverability, this comprehensive data would be quickly and easily obtainable. It would eliminate the trouble of sending a workforce of researchers to map by hand, saving hundreds of hours of labor.

9. Glossary

Light Detection and Ranging (LiDAR) Sensor: A device that is used to locate objects in space. It consists of a rapidly firing laser beam as well as a returning light reader. The laser beam fires pulses in various directions, which reflect off of objects and return to the sensor for reading. Based on the angle at which the pulse was fired and the amount of time it took to rebound, the LiDAR can determine the location of an object as a point in space.

Nghiem, Dr. Truong: Our technical advisor for this project. Offers his expertise and knowledge on different types of LiDAR and how to use ROS.

Point Cloud: A dataset consisting of a collection of points in 3D space, where each point represents a solid object detected by a sensor. Point cloud maps often contain millions of individual points.

Robot Operating System (ROS): An operating system which is considered standard for use in robots and autonomous drones. It includes a number of different interfaces and libraries that will allow our navigation system to command the drone to move in a specified direction.

Shenkin, Dr. Alexander “Allie”: Our sponsor for this project. His field of study involves collecting data about rainforests to better understand their structure, function, and response to changes in the environment.

Simultaneous Localization and Mapping (SLAM): A standard or protocol for mapping spaces in real-time. A SLAM system will use sensors to detect obstacles in its environment, then simultaneously upload the data into a procedurally-generating map which can be used during runtime and/or stored for future use. This data is often recorded as a collection of millions of points, collectively known as a point cloud.

Accepted as baseline requirements for the project:

For the client:

_____

Alexander Shenkin

For the team:

_____

Austin Malmin

_____

Conrad Murphy

_____

ShanHong “Kyle” Mo