Search and Shine

Authors: Nina Guo, Ronit Shah, David Wu
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— Manual intervention in rescue operation in disastrous, remote areas is often inefficient, expensive, unpractical, and error-prone. Hence, we are designing a search and detection rover with a grounded spotlight that is capable of finding lost humans in semi-flat, dangerous terrains. We also aim to accelerate object detection using a distributed system and provide a secure website that will remotely signal the locations of detected humans for rescue workers to use. We have designed a scaled-down testing scenario where the rover is expected to detect a human and accurately point a laser at them.**

***Index Terms*—Autonomous, Computer Vision, Object Detection, Real-Time System, Rover, Search and Rescue**

1 INTRODUCTION

For the longest time, traditional methods of search and rescue have involved the labor of rescue workers and even dogs. Often, SAR is deployed in areas where human lives are decimated by war, nuclear spills, radiation, and natural disasters. Due to the necessary exploration of these sites, rescue teams are vulnerable to the same risks of physical harm as the survivors they are trying to locate [1]. As a result, rescue workers face deadly issues such as contamination and spread of toxic material, further exacerbated by the rising costs of personal protective equipment.

In fact, PPE costs have grown exponentially due to the recent COVID-19 pandemic and supplies now average 1000% of its original cost [2]. This means US taxpayers are not just paying 1000% of what they used to for PPE supplies but multiple times that due to the depletion of raw material and also the repetitive cycling through of disposable equipment. Consequently, the use of human labor in SAR missions incurs extraneous costs given the priceless of one's life.

Clearly, such risks and expenses to people's lives need to be mitigated in a cost-effective manner which can be done through the incorporation of designated search and rescue rovers. Rescue workers need to be able to extract themselves from life-threatening situations while also being able to monitor the search in real-time. Our SAR rover can replace the need for human labor and methodically search through hazardous zones while sending live video and location updates to a remote rescue team through our tracking website. Not only would it eliminate the endangerment of rescue workers, it would also diminish the additional costs of PPE from SAR in general.

Our goals for this SAR system are aimed to aid search and rescue workers with a safer and more cost-effective method of detecting survivors in perilous situations while also showcasing the capabilities of object detection transported via rover and onto a website.

2 USE-CASE REQUIREMENTS

Our system will have the following use-case requirements:

1. Speed: Less than 5s latency. Rescue workers will need fast response times due to urgency of situation, and 5s typical for real-time systems.
2. Identification: Top-1 Accuracy: $> 80\%$ and Top-5 Accuracy: $> 90\%$, object detection module needs to be able to identify humans accurately in order to not waste time or resources.
3. Autonomous Control: Can drive in pre-specified pattern, and correctly navigate to laser-pointing position if person found ($\pm$ 1 feet)
4. Longevity: Maintain $<$ 5 minute drive time while searching and carrying a load of 0.5 kg.
5. Point Control: Offset in person's actual location and person's calculated location: $\pm$ 0.5 feet; Offset in person's actual location and laser-pointed location: $\pm$ 1 feet

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

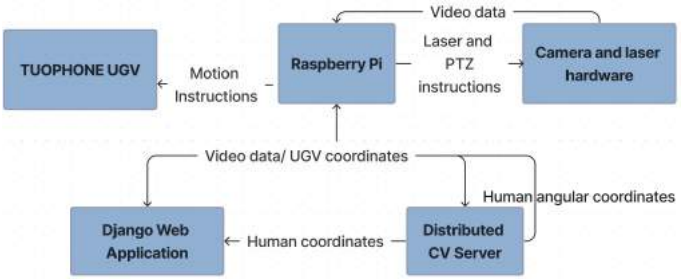
3.1 Subsystems

Our system architecture involves five subsystems: the TUOPHONE UGV, the PTZ IMX219 camera hardware, the Raspberry Pi, the Web Application, and the Distributed CV server. The TUOPHONE UGV involves the rover base and wood-based mount that's fitted on top of the rover based. The rover base is controlled by the Raspberry Pi and moves autonomously through UART communication from our controller program to a receiver that controls the rover's motors. The PTX IMX219 camera hardware involves an ArduCam camera that sends video data to the Raspberry Pi and a laser that's fitted on top of the camera. The Raspberry Pi serves as a hub for communication between the rover and the Distributed CV server, relaying information like video data and position over WiFi to other subsystems. The Web Application is hosted on the AWS

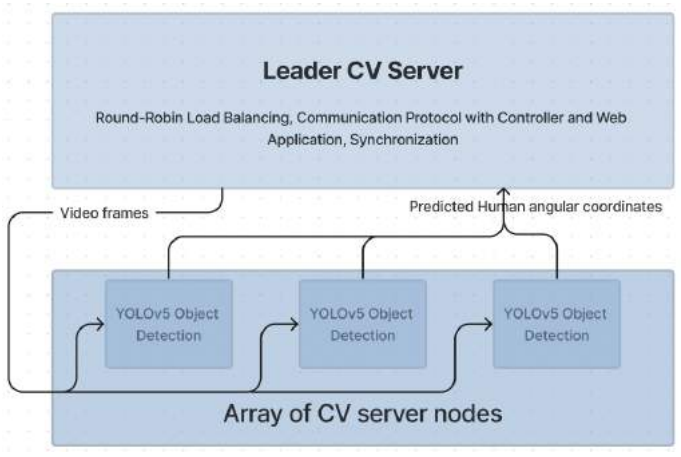
Cloud as an AWS EC2 instance, and serves as a tool to remotely monitor the UGV video feed and display information like the rover's GPS coordinates and live video feed in a secure, user-friendly manner. Finally, the Distributed CV server runs on the Gates-Hillman Cluster machines at Carnegie Mellon University, and runs YOLOv5 object detection on video data received from the UGV and sends angular turn instructions of humans detected to Raspberry Pi.

3.2 Data Flow

Fig. 1(a) gives an overview of data flow between the five subsystems. The UGV controller program running on the Raspberry Pi first computes a series of JSON commands to send to the TUOPHONE UGV such that the rover moves in a creeping line search pattern. The rover, powered by 18650 Rechargeable Batteries, receives these commands through an ESP32 WiFi module, and moves accordingly. As the rover moves, it collects video data through the front-facing PTZ Camera.



(a)



(b)

Figure 1: System description. (a) *Communication between the five subsystems - the TUOPHONE UGV, the PTZ IMX219 camera hardware, the Raspberry Pi, the Web Application, and the Distributed CV server.* (b) *Communication within the Distributed CV Server.*

In order to relay video frames over to the UGV Controller, the Raspberry Pi board is connected with the rover and the camera. A HiLetgo Laser Head laser is also mounted on the rover and connected to the Raspberry Pi board. The board collects data from the camera and sends it over to the Distributed CV server through the Raspberry Pi's ESP32 WiFi module in the form of a byte stream. Additionally, the Raspberry PI relays real-time GPS information to the Django web server.

Fig. 1(b) describes communication within the Distributed CV server. The Leader CV server collects the received byte stream and serializes it into video frames and GPS coordinate information. It then distributes each video frame across an array of "follower" CV server nodes. If a human is detected using the YOLOv5 object detection algorithm, resulting relative angular coordinate information about the human is calculated using trigonometry and is sent back to the leader server. The leader server then encodes the data in the form of a byte stream and relays it over to the Raspberry Pi and the Django Web Server using WiFi. Upon receiving this information, the Raspberry Pi calculates how many and what kind of motor commands the PTZ mount needs to execute to point precisely at the human. The Raspberry Pi also turns on the laser once the PTZ mount has centered the camera on the person. The entire process of communication within the CV server is summarized in Fig. 2.

The Django Web Server is used to a display real-time video feed, as well as GPS positioning of the rover. It decodes the byte streams received from the controller and the Leader CV server. The server then sends a HTML response to web browsers that rescue workers use to monitor the rover's feed and access GPS information using the Google Maps API in an efficient, user-friendly manner.

3.3 Principles of Operation

Our final product heavily utilized principles of engineering to meet its operational goals effectively. One fundamental engineering principle applied was the modular design approach, where the complex problem of creating an autonomous search and rescue system was decomposed into manageable subsystems. Each of these subsystems was designed to perform specific tasks independently yet interoperate seamlessly through communication protocols to achieve the collective objective.

Additionally, the principle of redundancy and reliability was critical, especially in ensuring the rover could maintain functionality in hazardous environments. This was implemented by having multiple backup systems, such as the distributed CV server setup, where multiple nodes ensured continuous operation even if one failed. Furthermore, the engineering principle of optimization was applied in the design of the rover's movement patterns. The creeping line search pattern allowed the rover to cover the search area methodically and efficiently, optimizing the search process and reducing the time to locate humans.

Scientific principles were integral in the development of our rover's operational capabilities. The rover's ability to navigate and detect humans autonomously was underpinned by the application of physics and computer science,

particularly in the fields of optics and image processing. Moreover, the rover's autonomous navigation was based on principles from robotics and kinematics, ensuring it could maneuver through rough terrains without human intervention. The scientific principle of feedback loops was also employed in the rover's control system, where real-time data from the environment (such as obstacles and terrain changes) was used to adjust its path and search strategy dynamically.

Finally, principles of mathematics are crucial to our project. The application of trigonometry allowed the system to calculate the precise angles and distances required for the rover to point its laser towards detected humans accurately. These calculations were essential for ensuring that the location of detected individuals was marked accurately, facilitating quick and efficient rescue operations. Furthermore, the principles of probability and statistics supported the optimization of the object detection algorithms. By analyzing the probability of detection errors across multiple frames, we could refine the algorithm to minimize false negatives and enhance the reliability of human detection.

4 DESIGN REQUIREMENTS

We have developed a series of design requirements based on our use-case requirements. These design requirements cover all five of our subsystems and are further described in the following subsections.

4.1 Accurate Human Identification

A robust object detection algorithm that can accurately identify humans amongst other objects is crucial to meet our use case requirement of human identification and point control. The distributed CV server must be able to do this on the video feed received from the TUOPHONE UGV to locate humans needing assistance. We are using the popular open-source object detection algorithm YOLOv5 for this purpose.

The metric we are using to measure the accuracy of human identification involves Top-1 accuracy and Top-5 accuracy. These metrics measure the accuracy with which the object detection algorithm could classify various objects. Top-1 accuracy specifically deals with the accuracy with which the predicted label matches the target label. Top-5 accuracy, on the other hand, deals with whether any of the top 5 predicted labels from the model match the target label [3]. We aim to achieve a Top-1 accuracy of $> 80\%$ with our prediction model, as well as a Top-5 accuracy of $> 90\%$. This is so that we are able to locate humans needing help without relying on manual intervention, making this approach practical in disaster-relief scenarios and cost-effective. A high degree of accuracy is needed to make this approach viable; if humans remain undetected by the object detection algorithm, they would not be located by rescue workers.

Another important thing to note regarding the accuracy of the object detection model is that the objects are passed through object recognition multiple times. This is because the rover moves to different positions, capturing frames from different distances and angles to objects. Hence, the error rate of missing humans decreases exponentially with each frame analysed. This error rate can be quantified in the form of an equation:

$$E = \frac{(100 - A)^N}{(100)^{N-1}} \tag{1}$$

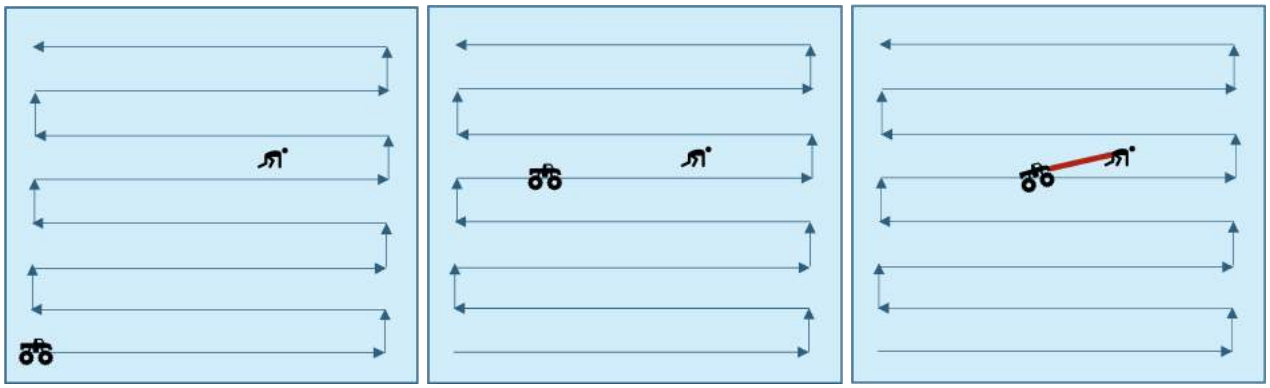
where E is the percentage likelihood of missing a human during search, A is the Top-1 accuracy in successfully detecting a human in a frame, and N is the number of frames with the human in it analyzed. This equation justifies why a Top-1 accuracy of $> 80\%$ is significant to accurately identify humans. For example, plugging in $A = 80$ and $N = 5$ into "(1)", we observe that the likelihood of missing a person during search is $< 0.032\%$. In our implementation, N would be larger, making E in "(1)" trivial. This requirement to autonomously execute a search pattern to capture numerous frames in different angles is described in more detail in Section 4.2.

Another important sub-requirement that goes with accurate human identification is a low error rate in mistakenly classifying non-human objects as humans. We quantify this requirement with a false positive rate. A false positive is defined as an incorrect detection of a non-existing object or a misplaced detection of an existing object [4]. Our system must have a false positive rate of $< 1\%$. The reasoning behind this strict criteria is that falsely detecting an object as human would clog up a lot of resources. Rescue workers would have to send aid to an area where humans are not present. Furthermore, incorrect detection delays the rover from continuing the search for humans, leading to a greater time taken in finding humans needing assistance.

4.2 Autonomous Search Pattern

The TUOPHONE UGV must be controlled through pre-programmed control logic so that it can autonomously comb an area, covering varying angles of objects in the area. This design requirement is motivated by the use-case requirements of autonomous control and longevity of drive.

To comb an area of 20ft by 20ft, the UGV must be able to travel at a speed of 1 m/s. The UGV must also be able to perform a creeping line search pattern. We quantify creeping line search as moving 20ft across, then 1ft above, then 20ft across again, and so on. This means that the rover must cover a total distance of $(20 + 1) * 20 =$ 420ft. Additionally, due to battery life constraints of being powered by 6 5000mAh rechargeable batteries, the rover must complete the search in < 5 min. The rover, during this duration, must be able to hold 0.5kg of weight and remain stable. The weight requirement comes from holding the camera mount, the HiLetgo Laser Head laser, the ArduCam PTZ Camera, the wooden mount to elevate the camera, the Raspberry Pi, and other relevant circuitry.



(a) (b) (c)

Figure 2: UGV process in human identification. (a) *UGV moves in a creeping line search pattern* (b) *CV server detects human in video frame* (c) *Rover receives instructions to stop, rotate, and turn on laser towards detected human*

It is imperative that these constraints are met by our system. If the rover travels at a speed greater than 1 m/s, insufficient video frames will be received by the Distributed CV server, which could lead to missed humans by the object detection algorithm. If the rover travels at a speed less than 1 m/s, there is a chance that the rover would be unable to perform the entire search, leading to missed humans. This would be unacceptable, as it is crucial that the rover can gather enough frames to detect humans. If this happens, rescue workers might not be able to confirm whether humans needing assistance exists in a particular area, necessitating manual intervention in potentially dangerous, remote areas. Furthermore, if the rover is unable to complete the search in < 5 min, there is a possibility that the rover might not be able to return to a location where it can be retrieved by humans. This would put additional burden on rescue operation budgets, which is not ideal. The rover also must be stable enough to carry the additional weight of different components mounted on it, as it cannot topple over in uneven terrains. This might lead to an incomplete search and the need for manual intervention.

There is also a sub-requirement that the controller must be able to correctly navigate to a position where it can accurately point the laser at a detected human. We quantify this requirement by considering an offset from the desired position, which should be $\pm$ 1 feet. It is important to have this accuracy so that rescue workers can accurately find the human in rescue operations. The rover must be able to stop the search pattern, rotate itself, and turn on the laser through the Raspberry Pi after attaining the right orientation.

4.3 Efficient Web Application

Having an efficient and secure web application is crucial. This deals with the use-case requirement of having fast responses and crucial data available promptly to rescue workers. We quantify this design requirement in the form of latency. For this project, latency is defined as the delay between "seeing" a human on the PTZ camera and reporting the detected human on the web application. Achieving a latency of 5s is important, as promptness in emergency situations is necessary. This latency is typical for real-time systems, and achieving this latency will allow crucial data to be received by rescue workers promptly so that plans for aid or extraction could be made.

A sub-requirement that goes with this is acceleration of the object detection algorithm. The object detection algorithm is the most computationally expensive part of our system, and using a distributed system to accelerate this algorithm would allow us to achieve this latency. Hence, another requirement is to achieve at least a 5x speedup of the algorithm compared to the sequential single-threaded YOLOv5 algorithm.

The website must also be secure and resistant to hacking attacks. Such attacks might involve altering the data displayed on the website, so having security is paramount, as misleading data would result in rescue workers being sent to the wrong location in case a human is detected. We aim to achieve this by using novel user authentication techniques, Cross-Site Request Forgery tokens, and encryption of data through the Django API. Having a user-friendly website is also important for the ease of use of rescue workers, so we aim to make the website easy to navigate through and use the Google Maps API to display where the rover and detected humans are. This would allow users to visualize where exactly humans are, making rescue efforts easier.

5 DESIGN TRADE STUDIES

5.1 Vehicle Design

Below are some design trade-offs regarding our system.

5.1.1 Vehicle Choice

From a very high-level analysis, the decision to use a rover for the main body of our project was an involved process. We originally examined using a drone instead of a UGV to gather the video feed data. While a drone

does have its perks, notably having a better aerial view for searching, it proved to have many disadvantages as well. Drones were incredibly expensive, inconvenient, and fragile, relative to rovers. To achieve the same requirements of having a software API and being able to carry a laser were much more challenging to find with drones than rovers, and those that were found were an order of magnitude larger in terms of price. For example, drones with ArduPilot (a software API for controlling drones), such as ELANUS DUO, cost upwards of $5000, while the rover we chose cost under $300. Drones were also less capable of carrying items (e.g. external paraphernalia like the laser), drastically reducing their efficiency for an already limited flight duration.

On a related note, because the rover was now grounded, it offered a more limited view of the environment. Being able to see from a higher vantage point would mean more of the camera contains the surroundings rather than the ground.

To counteract this problem, we decided on implementing a mount to elevate the positioning of the camera. However, there was a careful decision to be made here: a mount too high would potentially be too heavy, and more importantly, could knock the rover off-balance, as per general torque physics. Thus, having a mount raised too far would be impractical as well. Understanding the benefits and trade-offs of the mount height, we chose to have the mount be two feet tall, as this would be roughly twice the length of the rover. We observe the torque equation

$$\tau = rF\sin(\theta)$$

where τ is torque, r is the radius, F is force, and θ is the angle between the force and the lever arm. For the purposes of the following calculations, we assume that θ is roughly 90°; i.e. $\sin(\theta) = 1$. We see that the width of the rover (being 0.08m away from the base of the mount) would exert a torque roughly equal to

$$2kg * 9.8ms^{-2} * 0.1m = 1.96Nm$$

, assuming the weight of the rover is dispersed evenly across the rover. The camera, weighing 0.48kg, exerts a torque of

$$0.48kg * 9.8ms^{-2} * 0.6m = 2.82Nm$$

This means that the rover can withstand tilting by

$$cos^{-1}(\frac{1.96Nm}{2.82Nm}) = 45°$$

We find this to be a sufficient margin of error to prevent the rover from toppling over.

5.1.2 UGV and MCU Choice

Using the TUOPHONE UGV was a decision that stemmed from the capabilities of the rover. Using a rover that supported wireless communication over WiFi was crucial, as a significant amount of video data had to be shared with the other components of the system. Furthermore, support for an MCU that could collect video data and relay video feed to a Central Hub was crucial. Other rovers that had these capabilities were either too expensive, didn't have a software API that could control it, or didn't have support for a MCU. The TUOPHONE UGV has support for hosting multiple computers like RPi, Jetson Nano, and Jetson Orin Nano. Furthermore, these host computers could communicate with the ESP32 slave computer on the rover through the serial port, allowing for communication with the Central Hub. This UGV also had a wider base with six flexible rubber tires that had shock absorption and off-road capabilities, allowing for greater stability of the rover on uneven, slightly hilly terrains.

Using a RPi module stemmed from the MCU having more desirable capabilities than Jetson Nano and Jetson Orin Nano. The NVIDIA Jetson Nano is an MCU that has significant computing capabilities and could be used for GPU-accelerated machine learning tasks. However, we decided on off-loading this computationally intensive task due to the high power consumption required by the Jetson Nano. The Jetson Nano operates at a power consumption of higher than $10W$, whereas Rpi operates at a power consumption of approximately $5W$. Since using the Jetson Nano would significantly reduce the overall drive time of the UGV, RPi's power requirements are more appealing. Furthermore, Jetson Nano requires additional modules for wireless connectivity, whereas RPi comes with support for versatile networking options like WiFi. RPi is also significantly more user-friendly, as it requires Python. Due to these factors and the availability of numerous GPIO pins to connect the ArduCam PTZ Camera and the laser pointer, we decided that RPi would be the suitable MCU to use for our system.

5.2 Object Detection Algorithm Selection

Our project is highly dependent on using Machine Learning frameworks that would be suitable for object detection. We were considering using either YOLO object detection or a Region-based Convolutional Neural Network (R-CNN) in our system. Both of these models are open-source and great for the task of object detection; however, there are some significant different and tradeoffs to consider. R-CNNs are known for their precise detection and localization and are great at handling complex backgrounds, making it more desirable than YOLO in detecting humans in varying poses. However, R-CNNs are much more complex and computationally expensive than YOLO, as it uses a region proposal step instead of a single neural network like YOLO. YOLO also has a lower false positive rate than R-CNNs as it contextualizes the entire image instead of a particular region in an image.

YOLO also allows for real-time processing due to its speed compared to R-CNNs. YOLO is also much more user-friendly to implement than R-CNNs, since they do not require any additional training. Training a R-CNN would have been incredibly difficult due to a lack of sufficient training data. These factors, along with the need for

efficiency in our system, we decided that YOLOv5 was the best object detection algorithm to use in our system.

Using this object detection algorithm came with some concerns. We researched that YOLOv5 might not be as good as detecting objects that are small or objects that are in a cluster. This mandated a need for fetching multiple angles of objects. Hence, we determined that a creeping line search pattern would be the best way to gather video data, as multiple angles increases the accuracy of successfully detecting a human. A creeping line search pattern enables us to get closer to a potential human as well, exacerbating the concern the algorithm might not detect smaller objects. Therefore, this algorithm is ideal for the purposes of our system.

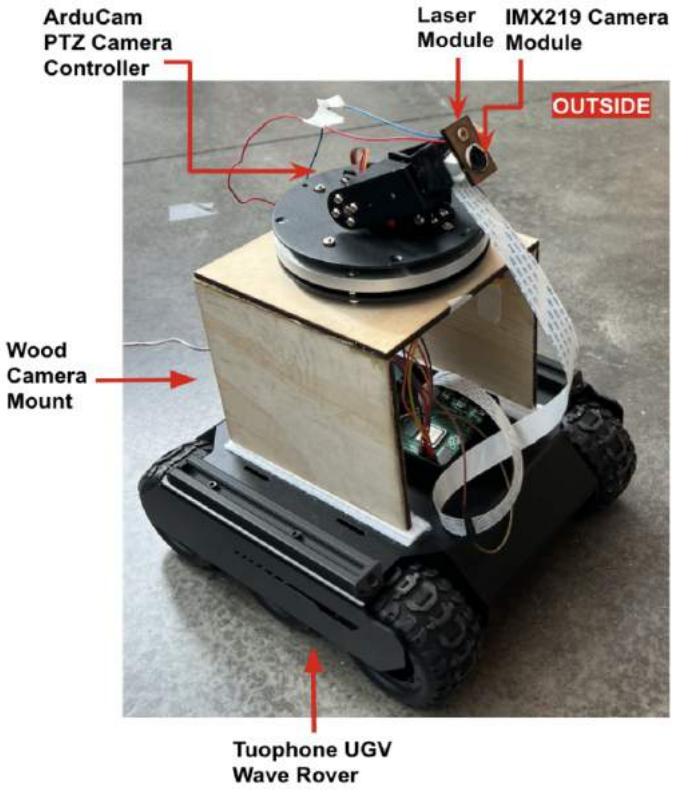
5.3 Load Balancer/ Distributed System

Our design requirements mandated an efficient CV server that is able to deal with computationally intensive tasks quickly. Through research, running YOLOv5 on single node would be impractical for quite a few reasons. Firstly, the ArduCam VU6112 PTZ Camera captures video feed 90 frames per second. YOLOv5 processes 45 frames per second, leading to a backlog of processing if a single server node was used. Secondly, in a real life rescue scenario, multiple rovers need video feed analysed. The CV server must be able to promptly execute the object detection algorithm for all of these concurrent video feeds, necessitating the need for parallelization. Additionally, using a distributed system proves to be energy efficient, as load is alleviated on a single computer. It also prevents the computer from being a bottleneck, as handling both communication and object detection would be inefficient. Hence, offloading the computation to a distributed server would allow for the computation time to become not as concerning.

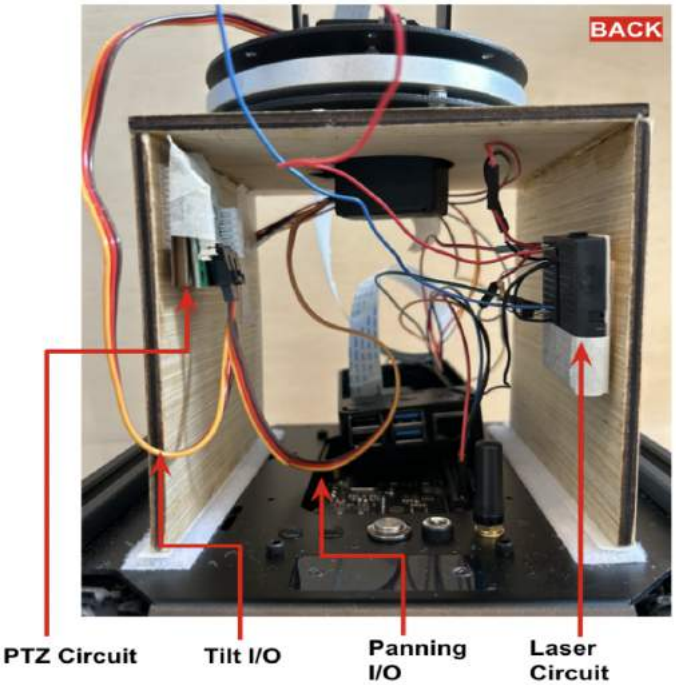
Due to these reasons, and the fact that processing video frames is highly parallelizable, we decided to use a Distributed server that handles concurrent batch processing of video frames. We decided to go for a leader-follower system, where the leader node is responsible for synchronization and the follower nodes are responsible for running the object detection algorithm. Through research, this kind of system proved to be great at load balancing to different follower nodes, improving resource utilization and response times. It would also support fault tolerance, as if one of the follower nodes goes down, work can easily be picked up by another one of the follower nodes.

For our system, we also decided to implement Round Robin load balancing, as it is one of the simplest and computationally inexpensive load balancing algorithms. It also allows for equal distribution of workload amongst children nodes, which is essential to speed up processing. Additionally, this algorithm is incredibly easy to troubleshoot, as the distribution of tasks amongst nodes is predictable.

6 SYSTEM IMPLEMENTATION



(a)



(b)

Figure 3: Physical rover implementation and circuitry. (a) *Top View.* (b) *Side View.*

Fig. 10 shows the block diagram of the overall system implementation. Our system consists of four major components: the rover, the computer vision servers, the website, and the Raspberry Pi. As aforementioned, a critical part of this implementation revolves around the ideas of offloading work from the rover to maintain a smaller compute body on the rover, and being able to establish a proper connection with the rover as it searches.

The Raspberry Pi exists to primarily assist towards this goal; it serves as a designated information-transfer point, helping to direct, redirect, and parse data (in the form of requests and replies) relating to the other three components. The rover performs the information gathering and executes the main action of the project (eg. the physical literal portion of searching and shining). The computer vision servers perform the key portion of the information processing, signifying the offloading of the heavy computation to away from the rover itself. Lastly, the website enables monitoring of the rover from a user's perspective, capturing the essence of the usability of the rover. The specific implementations of these four components will be discussed in greater detail, keeping in mind the overarching theme of distributed work and proper communication.

6.1 Autonomous Rover

Fig. 4 shows all of the various components of our autonomous rover. Our mode of transport, the TUOPHONE Waveshare WAVE UGV, is controlled through our custom-built controller program on the Raspberry Pi that sends JSON commands through UART communication. These commands allow the rover to perform creeping line search while gathering video data for analysis.

Mount on the rover are various components that help with stability and the general functionality of our system; these components are outlined in Fig. 3. A laser-cut wood camera mount is attached using velcro on the rover base and is used to elevate the camera's FOV and provide the PTZ mount stability. The PTZ mount has 2 degrees of freedom that are used to point the camera at detected individuals and pan the camera during search horizontally. The laser module and the IMX219 camera is attached to the top of the mount.

Underneath the wooden mount lies our Raspberry Pi and circuitry to control the camera, laser, and PTZ mount. The laser circuit involves a simple circuit that uses a MOSFET as a switch to turn on the laser from the Raspberry Pi. Tilt I/O and Panning I/O is connected to a PTZ circuit to actuate the mount's motors. Finally, a ribbon cable goes from the Raspberry Pi to the IMX219 camera to gather video data.

6.2 180°ArduCam

For object detection in a wide range, the ArduCam U6112 PTZ Camera allows the Wave Rover to visually peruse the area due to its pan-tilt-zoom feature [5]. To enable video data flow from the rover to the Central Hub, the ArduCam transmits photo frames of its data over Wifi through the Raspberry Pi. The Raspberry Pi is powered by the Wave Rover, interfacing with the GPIO pins and prebuilt circuitry from the rover. As mentioned above, the camera is placed on the top of the mount to support full range of 180°motion. In addition, since the rover itself is very low (33.70mm) and objects may look distorted from a lower angle, the high camera mount will also better mimic a person's eye level for clearer monitoring. However, attempting to have a mount that can completely mimic eye-level perspective would be ideal for image purposes, it also poses the risk of rover imbalance and instability while traversing terrain. Therefore, we decided mounting 2 feet off the ground would be a good compromise between the two concerns.

6.3 Laser Pointer

The HiLetgo Laser Head laser is used to point/shine a specified object in order to better outline where a person is for the rescue team to come in and rescue them. The laser is powered through a 5V power input, which is sourced from the Raspberry Pi. To control the laser, a RPi-controlled switch interfaced with the MOSFET circuit determines if the power source is connected or not. Due to how lightweight the laser is, we directly strap the laser onto the Arducam using a laser-cut wooden connector so the laser is at the same height as the camera. Doing so allows us to be able to control the direction of where the laser points by utilizing the camera's servos to rotate in all cardinal directions. This way, the camera simply directs its vision to centralize the object in question after receiving GPS coordinates sourced by the server to accurately target the laser at the object.

6.4 Computer Vision Server

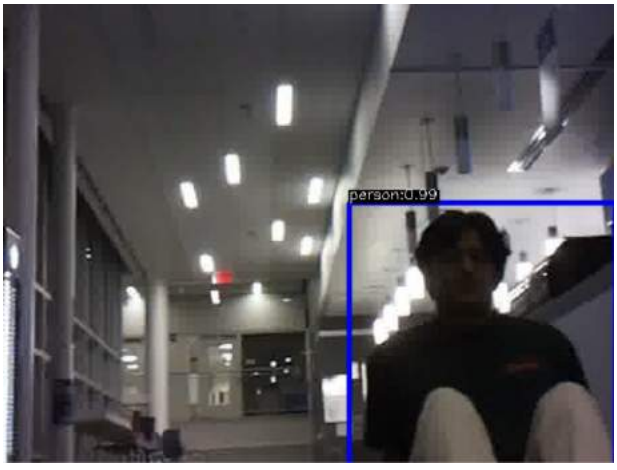


Figure 5: Example of YOLOv5 object detection running on a detected person

Computer vision is necessary to be able to perform the actual algorithmic calculation of the detection portion of

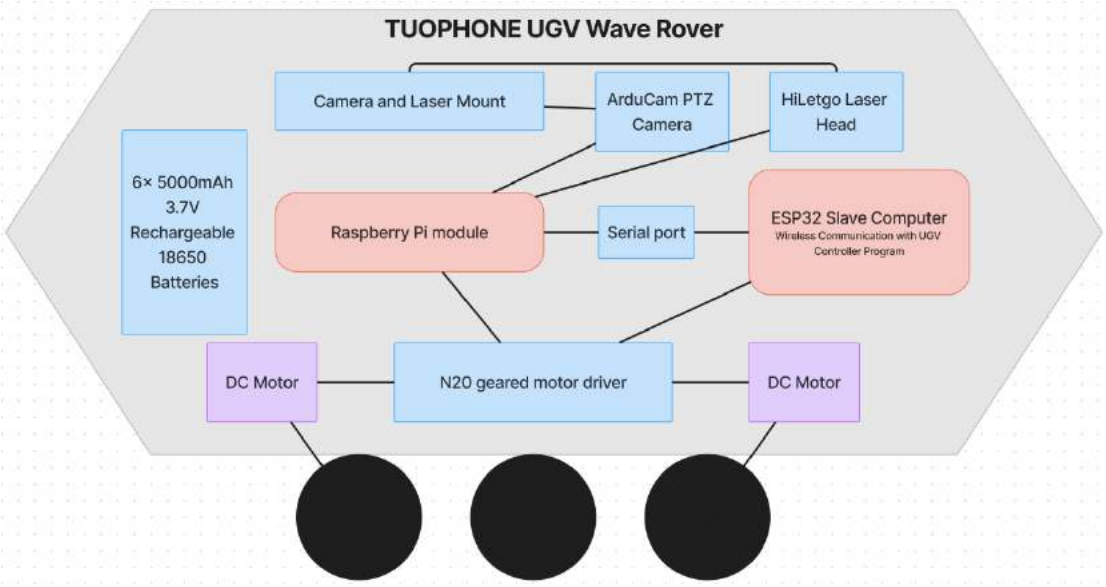
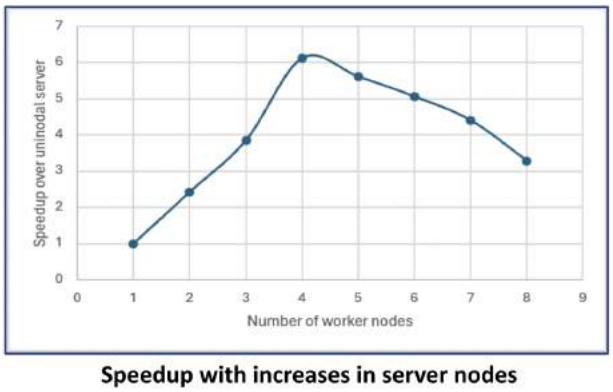


Figure 4: TUOPHONE UGV Layout

this project. Because this is the most computationally intensive part of the project, it is crucial to be able to perform this portion of the calculation as quickly as possible. Thus, the object detection algorithm used is YOLOv5 object detection, and work is distributed across several servers following a lead server. This distributed server is hosted on the GHC52 machine at Carnegie Mellon University, which contains NVIDIA GeForce RTX 2080 B GPUs. These GPUs are utilized by the object detection algorithm, accelerating this computationally intensive task to meet the latency requirements.



Speedup with increases in server nodes

Figure 6: Speedup analysis of speedup of object detection with the number of worker server nodes

6.4.1 Round Robin Load-Balancing

The CV servers receives the image frames from the Raspberry Pi using a TCP connection in the form of a byte stream. The leader server decodes the byte stream into video frames, and then performs Round Robin load-balancing to allocate a series of frames to children nodes. This load balancing algorithm is ideal for the server, as each CV processing unit has roughly identical computing capabilities and storage capacity. The algorithm involves rotating chunks of video frames in turn to each CV processing unit. For example, the first set of frames is assigned to CV processing unit A, the second set frames is assigned to CV processing unit B, and so on.

The number of CV processing units to deploy is determined from trade-off analysis, which is outline in the graph in Fig. 6. We determined from the analysis that the optimal number of worker nodes to spawn is 4. Fewer the 4 worker nodes leads to insufficient parallelism for accelerate the algorithm. Greater than 4 worker nodes puts the leader server in a bottleneck for communication, and high synchronization and communication costs significantly reduce speedup.

6.4.2 GPS Calculations

When a person is detected, the worker server nodes perform a series of calculations to correctly notify the Raspberry Pi of the angle the PTZ mount must turn, detailing the existence of the person as well. The calculations of these involve a set of image-processing logic code and trigonometry. Using these pieces of information allows the Raspberry Pi to be able to correctly determine the next set of actions for the rover, as detailed in the Raspberry Pi

section.

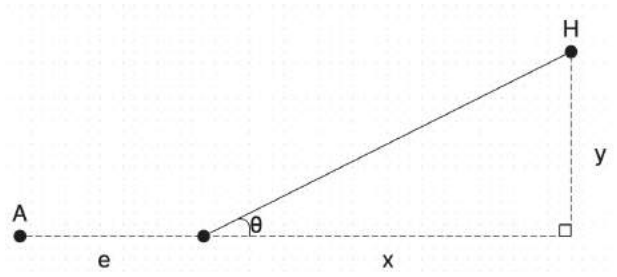


Figure 7: Trigonometric calculations to point rover towards human

Fig. 7 showcases the GPS calculations that will be made by the CV processing units if a human is detected, where A is the rover position at which the human is detected, H is the human, θ is the angle the rover will have to turn, x is the horizontal distance between A and H, y is the vertical distance between A and H, and e is the distance the rover would've travelled before the controller receives information on the human's coordinates. e could be calculated with the following equation:

$$e = t * v \tag{2}$$

where t is the latency of communication between the CV worker servers and the UGV controller and v is the speed of the rover. From "(2)" and Fig. 7, we can formulate an equation to determine how much the rover must turn:

$$\theta = \tan(\frac{x - (t * v)}{y}) \tag{3}$$

The calculation in "(3)" is performed by the CV child server that detected the human, and is sent to the leader CV server. The leader CV server then sends this information to the Raspberry Pi. A similar calculation is also made to determine how much the mount must turn vertically.

6.5 Monitoring Website

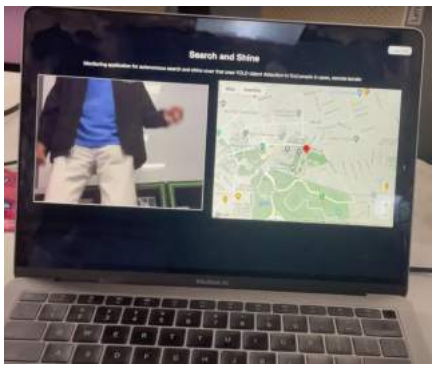


Figure 8: Website after login showing the video stream and GPS coordinates of the rover

The interface of the website after login is shown in Fig. 8. The website exists to provide the rescue team with a way to monitor the rover as it explores, displaying GPS location of the rover [6], where a person is spotted, and live video feed from the rover's ArduCam [7]. Layered on top of the Django web framework, GoogleMaps API will be used as the map base and will interface with WebSocket API that uses a TCP socket to receive GPS coordinates transmitted from the Raspberry Pi. Furthermore, the live video feed is sourced from the video frames sent over WiFi through the RPi.

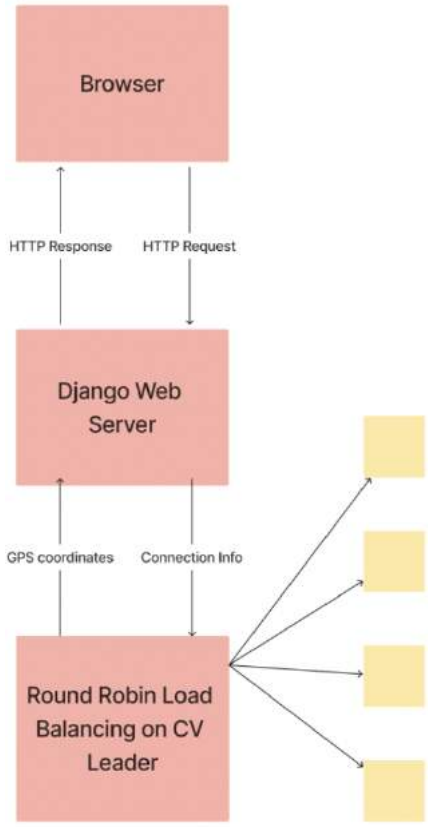


Figure 9: Communication between the CV Leader, Django server, and the web browser rescue workers use

Fig. 9 shows the communication that exists between the Django Web Server, the browser, and the CV leader. The Django Web Server initially sends connection information to the CV Leader server so that a WiFi connection via a TCP socket can be established to send data. If a human is detected, this information is sent in the form of a byte stream to the Django Web server to promptly display this information. The Web Server communicates with the browsers via a series of HTML requests and responses, and allows for numerous clients in geographically distributed locations to access rescue information.

6.6 Raspberry Pi

Due to the constant routing of information between each subsystem, a MCU is essential to organize and effi-

ciently send data to maintain low latency. The RPi perpetually runs several threads that handle all data transfers that occur within the rover and to other subsystems.

Since video data is vital for our CV server and website, the Raspberry Pi will be constantly receiving video data frames from the IMX219 camera through the ribbon cable on the rover. These exact video data frames are sent over WiFi to both the CV Servers and the Website for their appropriate usage.

The next important set of communication involves the control of the rover. Generally speaking, the rover's movement is controlled by commands sent through the Raspberry Pi. These commands are sent through UART communication in the form of JSON commands instructing the motors on the rover to perform a desired action. The preset search pattern starts as a creeping-line search pattern, implemented based on the size of the arena specified for the search. During search, commands to pan the mount are also given to gather a better FOV.

Once a person is detected, the CV server will ping the Raspberry Pi with a signal to disable the rover from further searching and remain idle until further instructions. The CV server's message contains information regarding the object's position (how much the camera must turn along the 2 degrees of freedom to center on the human) and whether a person is found. With this information, the RPi can parse and process the next set of directions for the rover.

Once it's reached its designated location, the PTZ mount will stop moving and the Raspberry Pi will activate the pin controlling the MOSFET on the laser circuit. Because of where the laser is mounted, it is necessary to tilt the laser slightly more upwards to properly hit the target at a center location. The amount of panning to do is calculated on the RPi using received angle instructions.

Because video data information will always be routed to and from the Raspberry Pi, the rover is always be able to make adjustments in its path, even after the object is detected. This essence of live-correction is crucial for target accuracy and adjustment.

7 TEST, VERIFICATION, & VALIDATION

To thoroughly to test out the system, we designed unit, module, and scenario tests to ensure that all of the design specifications and use case requirements are met. We will describe the procedure we used for testing each of our design and use-case requirements in the following sections.

7.1 Scenario Testing Setup

To test out the overall system, we designed a scenario test. This involved a 20ft by 20ft hard-surfaced arena with 5 feet high cardboard walls enclosing the space. The cardboard walls exist so that no humans beyond the actual arena interfere with the object detection. Furthermore, we mimiced an uneven terrain by having the rover go over small objects like books and small bumps. The uneven terrain is to test out whether the rover can remain stable despite having numerous components on it. We also have fixed lighting conditions as a fixed variable to ensure fairness amongst all trials conducted. In the arena, there is one human randomly placed amongst other objects like sports equipment, large plants, furniture, and human dummies. The other objects in the arena help to test the limits of the object detection algorithm.

We define a single trial by the time taken for the human to either detect a human or complete the creeping line search and return to the start point. We will test the following 4 different scenarios 10 times each:

- Human in a standing position randomly placed in the arena
- Human in a sitting position randomly placed in the arena
- Human in a sitting position with their face obscured with a mask randomly placed in the arena
- No human placed in the arena

As per our design requirements, the rover must complete the search in less than 5 minutes. Throughout our scenario tests, on average, the rover is able to complete the search in **2.6 minutes**. This was measured using a stop watch to evaluate the time it takes for the rover to start movement and eventually shine the laser at the detected human.

7.2 Results for Overall Testing

Testing overall was putting together all the components, and running the process entirely. In our 30 unit tests, we had the rover perform its search, and upon the CV servers detecting the person, the laser would be targeted directly onto the person. The web app should properly display the location and video feed of the rover at all times. Our goal was to hit 95% success rate with the rover, where success is defined as a laser pointing at the target, and we managed to hit an amazing **97% accuracy**. The rover was always able to find the person, and then after adjusting, hit the person on some part of the body. The laser sometimes was just slightly off, causing it to miss the person, but this was determined to be a hardware positioning issue over a software issue. Tighter connections helped remedy this problem for the future.

7.3 Results for Accurate Human Identification

During our scenario tests, we also heavily tested and tuned the angle calculation done by the children CV processing units in our distributed server. This primarily dealt with tuning the variable t in "(3)" in our calculations as we gather more data. We used comparison tests to verify

the offset in the actual person's location and the angle calculation by verifying whether the person is centered in the camera's video stream. This offset must be < 1ft.

When running our scenario test, we also tuned what confidence of the model we should use to successfully say a person is detected. We performed several iterations of tuning this threshold, and concluded that we should a successful classification occurs at $> 80\%$ confidence. Using a lesser threshold led to inaccurate results due to a higher false positive rate. Using a higher threshold led to more frequent misses of humans. Hence, using a confidence of 80% was optimal for our system.

We ran several tests to test out the accuracy of the YOLOv5 object detection algorithm on our Distributed Computer Vision Server. We gathered and used a validation dataset of 100 stock images containing one human to test the accuracy of the object detection. All of these stock images also contained other objects like bicycles, cars, dogs, chairs, and traffic lights to test the limits of our system. The images also had people in various positions and orientations, with some images having obscured faces. After gathering this dataset, we executed our sequential YOLOv5 algorithm with a confidence of 80% on the images to analyse the accuracy of our model.

Metric	Requirement	Result
Top-1 Accuracy	>80%	95%
Top-5 Accuracy	>90%	100%
False Positive Rate	<1%	0%

Table 1: Evaluation of the YOLOv5 object detection performance using a confidence threshold of 80% on the validation dataset.

Table 1 shows the results of running the sequential model on the validation dataset. The results show that all of our quantitative design requirements on accurate human identification are met with our system. The model failed on a few images where the person's face was not visible and where the photo had bad lighting; however, the model performed exceedingly well on all other images. The false positive rate achieved ensured that no non-human objects are classified as human.

7.4 Results for Autonomous Search Pattern

To test out whether the rover could autonomously comb through an area with the load of a Rasperry Pi, camera mount, laser, and PTZ camera, we inspected whether the rover is able to perform a creeping line search in the arena. The rover must not deviate from the creeping line search pattern unless a human is detected, which we thoroughly tested. Additionally, we tested the distances the robot moves in the creeping line search pattern, which must be 20ft for horizontal traversal and 1ft for vertical traversal.

A large portion of the initial search abilities of the rover involves the rover performing a creeping line search throughout the arena to find the target. As such, 20 unit tests were performed to see if the rover could be controlled properly (eg. move in the pre-specified pattern), and generally stay on course. We aimed to have an offset of 1 ft from the original course route, but in our tests, we found that the average distance off-course was nearly **8.2ft**. This was due to rather mysterious reasons, as the rover seemed to be incapable of moving the exact same pattern in a row, despite nothing software being different. The terrain was flattened and examined, and a thorough examination was given to find the cause, but nothing apparent stood out. As a result, our only solution was to reduce the amount of "predefined" searching that was done, and in order to cover for the lost ground, we implemented a scanning camera. The camera would now pan back and forth, enabling a much larger FOV while allowing for less rover movement (and ultimately less deviation from the pre-set path). Furthermore, we inspected whether the rover stayed stable while carrying all of the components on it, which it did during its search.

7.5 Results for Laser Control

We also heavily tested out the accuracy of the laser control. Unit testing laser accuracy is extremely important, as it is imperative that the laser is indeed accurate for our system to function. Testing the laser's accuracy involves examining three main geographic points: the person's actual location, the person's calculated location, and the laser-pointed location.

The offset between the person's actual location and the person's calculated location comes from the error in the bounding box provided by the CV servers. We chose the person's calculated location to be the center of the bounding box in both the x and y axes, and examined how far this was from the true center of the human. Our original requirement was to have this offset be within 0.5 ft, accounting for the irregularity of the human body while trying to give enough leeway for error. Our 20 unit tests, involving examining the bounded box provided by the CV algorithm, yielded an average offset of 0.3 ft, which is within our requirement; fortunately, it appeared our CV servers were accurate enough to place a good enough bounding box on the person.

The offset between the person's actual location and the laser-pointed location comes from two sources of potential error: the software precision and the hardware positioning. As our tests would show, ensuring a theoretically 100% accurate precision was impossible; the laser was not able to be tuned that finely by the software program. The hardware was also very finicky, as the laser is binded to the camera. Even the slightest tilt in the laser head meant a potential large offset due to how distance scales the further one is from the rover. As such, our requirement had this offset be within 1 ft, which was still reasonable due to the fact that a human "has width and height", and thus the

laser is still likely able to hit the person even though it is not perfectly centered. Fortunately, the offset from our 20 unit tests, involving having the rover adjust its laser at the target, had an average of **0.38 ft**. Achieving such high precision meant heavily fine-tuning, costing us time in latency, but we believed that the accuracy that resulted from it far outweighed the cost.

7.6 Tests for Power Consumption

To ensure that the lifetime of our rover would not be greatly impacted by all the paraphernalia attached, some unit test benchmarks were designed to verify the battery life. For our baseline, the rover was first left on until the battery died, and then as a comparison, the rover was run with all the paraphernalia running (eg. wheels turning, camera streaming, laser firing, etc.) until the battery died. We aimed for a power consumption loss of less than 5 minutes, which we believed to be a reasonable amount of power loss (about 10%), but our stress test ended up revealing a loss of around **8 minutes**, with a decrease from 45 minutes to 37 minutes. This failed requirement was fortunately workable, as it is important to understand that this was a stress test; our actual search procedure would not be as power-consuming. For example, during our search procedure, the laser would only turn on when the target was centralized, and no other time. For further optimization, we adjusted the targeting mechanism to be performed by moving the PTZ controller rather than the entire rover, enabling smaller actions overall and finer tuning capabilities.

7.7 Results for Efficient and Secure Website

Efficiency is crucial so that rescue workers receive accurate information promptly We thoroughly tested out the efficiency of data routing to the web application. This involved testing the latency between when the rover first starts recording video data and when the web application first displays the video feed. We used unit testing for this by recording time stamps, and we aimed to achieve a latency of < 50ms.

Latency is a critical feature to our project, as real-time CV analysis had to be as close to real-time as possible. We generously had a requirement of 5 seconds, as we had believed that with the data transfer and CV processing, we would suffer from high latency. For our 20 unit tests, we clocked the time it took for the video feed to be received by the CV servers and be processed with image detection, and then the results to be returned to the rover. This yielded an average latency of **1.44s-1.64s**, which is much faster than what we planned for; this was primarily accomplished due to using a distributed system for our CV servers. While resolving many issues that could arise by a slower latency, it is important to note that there was still a substantial amount of delay. To remedy and optimize this, our search and targeting algorithm had to act a bit slower so that our rover would not end up receiving "lagged" results from the CV servers, which would cause a general hysteresis effect across all actions.

Efficiency of calculating the human coordinates lies primarily on the distributed CV server. This was thoroughly tested with the validation dataset, and we aimed to achieve a speedup of 5x using a distributed server. For this, we used comparison testing with a YOLOv5 implementation running on one server node. The implementations we tested involved 1-8 children CV processing units, and we performed trade-off analysis between acceleration and synchronization costs to ensure an optimal number of children processing units are spawned. As per the results in Fig. 6 and previous discussions, we determined that spawning 4 nodes was optimal. This gave us a speedup of **6.11x**, which met our design requirement.

It is imperative that our website is invulnerable to security threats. Per our design requirement of having a secure and functional website, we used unit testing that incorporates a Selenium web driver to ensure the functionality of navigating through the website, particularly the login and register functionality and GPS coordinates of the rover. We also visually inspected whether the stream was up on the website.

8 PROJECT MANAGEMENT

8.1 Schedule

Our Gantt chart is shown in Fig. 11. This differed slightly from the Gantt chart is our design review report due to delays in implementing functionality as well as some hardware failures we encountered with the camera and the Raspberry Pi. However, all of the tasks we planned on completing were finished on time for our final demo, and we achieved all implementation and testing plans by our deadlines.

8.2 Team Member Responsibilities

1. Nina - Help design and laser-cut print fixed camera/laser mount, transmitting GPS and live video feed data, development of web application with real-time tracking and display, help prototype and test rover system

2. David - Program rover with preset road path, parallelize and synchronize rover processes involving data and communication, autonomous control of rover, PTZ controller, and laser with live inputs, help prototype and test rover system

3. Ronit - Help design and laser-cut print fixed camera/laser mount, development of distributed object detection servers to identify humans, data routing of person's angular coordinates back to rover program, help prototype and test rover system

8.3 Bill of Materials and Budget

The bill of materials is summarized in Table 2. All of the components that we bought were used, with additional components like wiring, wood pieces for laser cutting, and circuit components provided through the grace of TechSpark and IDeATe at Carnegie Mellon University. AWS credits were also provided free of cost from some of the classes we have taken during past semesters. This was used to deploy our website as as AWS EC2 instance.

8.4 Risk Management

We encountered several challenges that threatened to hinder with our progress in completing this project. This ranged from accounting for hardware failures, latency issues, and more.

Hardware failures were much more frequent than we expected. This threatened to increase the time it took to debug our overall system and gives us inconsistent results. We mitigated this risk by frequently testing out hardware components. For example, we tested out all major pins on our RPi regularly to see if they were functional in case we had unexpected behavior like the rover not responding to commands sent. We also kept some extra hardware components like extra laser heads to ensure that we had sufficient parts to replace circuitry. This proved to be incredibly helpful when we burned out one of the laser diodes and MOSFETs.

We also encountered a significant risk of delays in data routing giving us inaccurate laser pointing functionality. Delayed data routing between rover and object detection server caused rover to move incorrectly and run into the person it has detected. We mitigated this risk by allowing the rover to take pauses (taking breaks in between movement steps) to allow for ample response time to adjust itself according to where the person was seen. While the rover searches slower overall from this, it also mitigated the issue of race conditions during movement. Furthermore, to make sure that the laser pointing was accurate, we slowed rover movement and adjusted speeds to improve calculation accuracy between human's actual location and where in the video frames the person was detected.

During our demo, we also mitigated the risk of our rover running out of batteries. We did this by keeping charged backup batteries for the rover to ensure that the drive time remains consistent throughout. Additionally, we had a contingency plan to charge the RPi through an external power supply instead of the rover if the rover could not power all the circuitry. However, through repeated tests, this didn't prove to be an issue significant enough to implement this system change.

9 ETHICAL ISSUES

Although search and detection rovers would aid rescue workers as an extra eye, improper usage may cause social distrust and cross ethical boundaries. As a surveillance system, it can impose on certain social and cultural liberties as it partakes in harvesting video data that may not always be consensual from those who are being filmed. Since privacy is essential to a person's rights, a rover traipsing through land and streaming what it's seeing to remote viewers can definitely pose as a concern to modern social ethics. In addition, security is also essential to a person's welfare and safety. If the video stream and GPS tracking of the rover were to be hacked, it can definitely endanger those whose location and video data is shown through the web application. Public health and global concern would also be impacted by this matter as governments or foreign groups could make use of this object detection software and use it in wartime scenarios that can lead to deadly situations. Security and privacy of the general public would be bartered for autonomous tracking of people and expose them to dangers such as being targeted, stalked, and other malicious activities.

Not only does the application of the rover pose a concern, the method of human detection raises questions about biased training on datasets. In the scenario that the machine learning dataset being used had discriminatory, the search and shine rover might only detect people of a certain race, gender, ethnicity, etc and overstep on social and cultural ethics. Search and rescue operations in which it favors finding people of one subgroup would only further exacerbate current social inequalities. Additionally, environmental factors would need to be accounted for in the case that the rover dies while on a trek in the wild and becomes harmful litter.

It is also important to note that our project at its core merely provides a method of autonomously locating a person, and then accurately targeting them. Such a framework is prone to misuse, as it is relatively open to adaptation; the targeting system does not actually imply a specific use ability. It would be critical to manage how this project would be used by anyone, including government agencies or other interested individuals. Hence, this rover should only be provided to organizations that would not misuse this targeting system to bring about societal harm.

10 RELATED WORK

Several projects exist that are similar to our rover. Rovers have long been used as alternative methods of early reconnaissance in rescue missions due to their robustness in disaster scenarios and increased functionality to locate survivors. One example was the deployment of a chassis that shined beacons to help rescue workers more easily locate the person if detected [8]. In addition, real-time visual data of the environment the rover was searching through was transmitted to the team to view safely from a distance. By separating the search and rescue operations, human labor and PPE can be used more efficiently by equipping themselves for a singular rescue mission.

Furthermore, rovers were introduced due to their abil-

Description	Model #	Manufacturer	Quantity	Cost	Total
6 Wheels 4WD Mobile Robot Chassis	0001	TUOPUONE	1	$250.99	$250.99
18650 Rechargeable Battery 5000mAh	0022	Tokeyla	6	$29.98	$29.98
PTZ Camera	U6112	UCTRONICS	1	$0	$0
Raspberry Pi	4	Raspberry Pi	1	$0	$0
Breadboard	-	-	2	$0	$0
HiLetgo Red Dot Laser Head	0001	HiLetGo	1	$6.79	$6.79
					$287.76

Table 2: Bill of materials

ity to traverse and operate in toxic or unknown environments to address the need of mitigating risk to rescue teams in dangerous SAR missions. Due to their customizability, rovers can be used to overcome areas that are impossible for a normal rescue team to explore such as the amphibious rover that can search and deliver across water and land surfaces [9]. Through combinations of wheels and propulsion fins, this rover can easily glide through aquatic environments and perform tasks to the same ability as if it were on land. In contrast, a team of all people might introduce superfluous costs such as waterproof gear and methods of transport for water and land.

11 SUMMARY

Our rover project effectively met its design specifications, particularly in the areas of object detection accuracy and system response times. The system achieved a Top-1 Accuracy of 95% and a Top-5 Accuracy of 100%, surpassing the project's accuracy goals. Additionally, it maintained a false positive rate of 0%, ensuring high reliability in human detection. The autonomous navigation capabilities of the rover allowed it to operate within the required parameters, with positional adjustments kept within the acceptable range of $\pm$ 1 foot, demonstrating effective control and precision in target pointing.

However, the system faced limitations in stability and hardware precision. During testing, the rover exhibited deviations from its set navigation patterns, which could impact its effectiveness in real-world search scenarios. The mechanical stability and hardware alignment also posed challenges, occasionally affecting the precision of the laser targeting system used to mark human locations.

Potential improvements to the system could include enhancing the rover's navigation algorithms and incorporating advanced sensory technology to better handle environmental variability and terrain challenges. Upgrading the mechanical components could provide greater stability and improve the durability of the system under operational conditions. Further software enhancements aimed at robustness against environmental disturbances could also enhance the system's overall performance and reliability.

Rescue workers are the backbone of our society who serve the general public in moments of emergency, which is why it's imperative we provide a highly accurate and efficient system in which they operate. Not only do they have to operate with the utmost urgency, they also have to risk their own lives to search for survivors in dangerous situations. With the burden of PPE costs that increase with more intense scenarios, rescue workers have to be exceedingly efficient in their method of looking. Thus, being able to incorporate a search and spotlight rover will immediately reduce all costs of external PPE in toxic zones and allow rescue workers to operate remotely and from a safe distance. By being able to mitigate both costs and risks, disaster scenarios will be much less formidable for both taxpayers and rescue agencies. Furthermore, by having a methodical way of detecting survivors through computer vision and a myriad of search patterns, more lives are likely to be saved in the long run.

11.1 Future Work

For next steps, we would like to address the concern that our rover does not actually have any object collision prevention in place. As a result, this restricted our usage to be over generally flat terrain. Future work would look into reacting to the environment in several ways, such as including ultrasonic sensors or augmenting the CV to also detect obstacles (eg. trees or rocks). From there, the rover would be able to avoid obstacles autonomously and could extend its use case to an all terrain search.

Furthermore, future work could involve the use of a PID controller to account for latency to improve the accuracy of laser pointing. This would allow the rover to speed up the process of pointing the laser once a human is detected. Such a technology, whilst increasing computation on the RPi, would be excellent for providing self-correcting feedback to improve our system.

Due to the low cost and scalability of our rover, we can adapt it to work in hazardous conditions by increasing its durability with weather-resistant hardware and using fire detection software to navigate in wildfire zones for missing people, further exemplifying our mission of improving public safety.

11.2 Lessons Learned

Through the development of our project, we ran into many challenges and have learned major takeaways after overcoming them. First, it is important to start research

and development of the project early. Due to unknown but likely issues in the future, it is imperative to have enough time to either diverge from or fix those issues, so thorough research and proper planning is essential to a less stressful engineering process. In addition, issues from all areas may arise, whether it may be software or hardware, it's important to gauge and prepare accordingly depending on the project's subsystems. We, unfortunately, faced many software and hardware issues involving the hardware we ordered and spent many hours just deciphering what type of issue we were having. Never expect it to just be one type of issue, let alone be just one.

This also raises the lesson of not being afraid to diverge from the original plan. Because our original mode of transport was a drone, we were hesitant to move forward with a search and detect rover since all our previous research on data flow and communication was dependent on the drone. However, we weighed our pros and cons of transitioning to a rover and had a much more pleasant time of integrating our subsystems compared to worrying about the possibility of damaging the drone. Concerns spurred preparation and comprehensive testing as we firmly believed in Murphy's Law: "Anything that can go wrong will go wrong." Widely acknowledged during our entire process, we sought to never expect effortless or perfect results during development which proved to be correct as we were constantly trudging through problem after problem. This supported us in reaching our design requirements as we managed our project effectively enough as we allocated enough time to deal with issues and surmounted each one meticulously.

Glossary of Acronyms

- API - Application Programming Interface
- AWS - Amazon Web Services
- CV - Computer Vision
- EC2 - Elastic Cloud Compute
- FOV - Field of view
- GPU - Graphics Processing Unit
- JSON - JavaScript Object Notation
- MCU - Microcontroller Unit
- PPE – Personal Protective Equipment
- PTZ - Pan-Tilt-Zoom
- RPi – Raspberry Pi
- SAR – Search and Rescue
- TCP - Transmission Control Protocol
- UGV - Unmanned Ground Vehicle
- YOLOv5 - *You Only Look Once* algorithm: version 5

References

[1] Sevil CENGİZ. Possible hazards and risks that search and rescue (sar) dogs may face in cbrn incidents. *Gümüşhane University Journal of Health Sciences*, 11(1):300 – 306, 10 2022.

[2] Daniella Diaz, Geneva Sands, and Cristina Alesci. Protective equipment costs increase over 1,000% amid competition and surge in demand. https://www.cnn.com/2020/04/16/politics/ppe-price-costs-rising-economy-personal-protective-equipment/index.html, 4 2020.

[3] Anh T. Dang. Accuracy and loss: Things to know about the top 1 and top 5 accuracy. https://towardsdatascience.com/accuracy-and-loss-things-to-know-about-the-top-1-and-top-5-accuracy-1d6beb8f6df3.

[4] Ivan Ralašić. A better map for object detection. https://towardsdatascience.com/a-better-map-for-object-detection-32662767d424#:~:text=False%2Dpositive%20(FP)%20%E2%80%94,not%20be%20detected%20as%20objects., 10 2021.

[5] Arducam. Focuser example. https://github.com/ArduCAM/PTZ-Camera-Controller/blob/master/FocuserExample.py, 2019.

[6] GoogleMaps. Geolocation: Displaying user or device position on maps. https://developers.google.com/maps/documentation/javascript/geolocation, 2019.

[7] Chris Ruk and David Plowman. Mjpeg server. https://github.com/raspberrypi/picamera2/blob/main/examples/mjpeg_server.py, 2022.

[8] Zachary Agustin, Charles Lewis, Elizabeth McMahon, Cameron Pierce, Pranav Pradhan, and Michael Tamshen. Safer: Search and find emergency rover. https://scholarcommons.scu.edu/cgi/viewcontent.cgi?article=1062&context=mech_senior, 6 2016.

[9] Mustafa Ayad, Nana Kofi, Ryan Wiszniewski, Nick Curinga, Khaled Ayad, Suleiman Erateb, and Adel Saad Emhemmed. Amphibious rover for search and rescue applications. *IEEE*, pages 64 – 69, 7 2022.

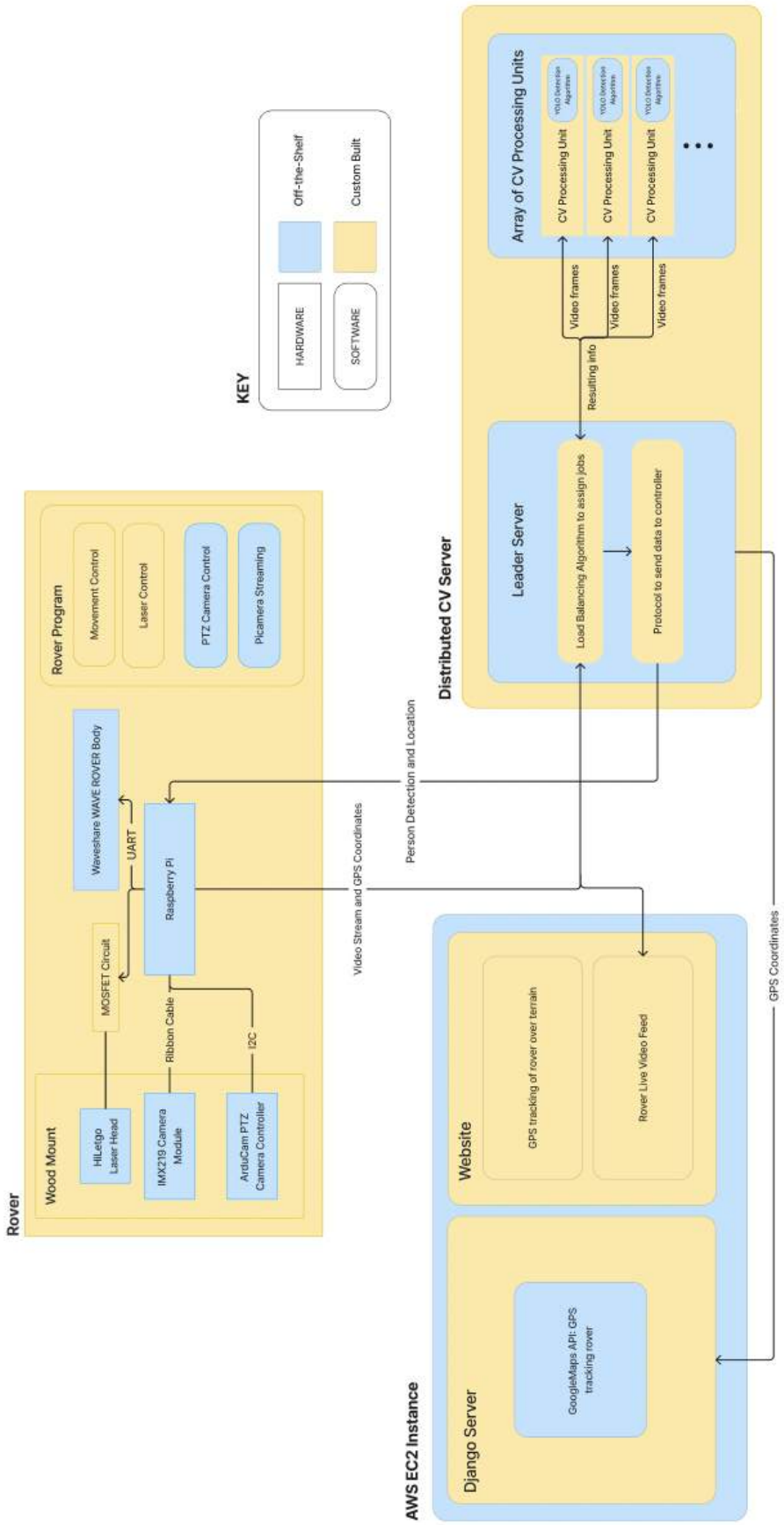


Figure 10: Overall system implementation of the Search and Shine rover

task/milestone	description	person/people with primary responsibility	start date	finish date	Wed 2/7	Wed 2/14	Sun 2/18	Wed 2/21	Wed 2/28	Fri 3/1	Wed 3/6	Wed 3/13	Wed 3/20	Wed 3/27	Wed 4/3	Wed 4/10	Wed 4/17	Sun 4/21	Wed 4/24	Wed 5/1	Sat 5/4
Perform material research	Perform research on correct materials to buy (motors, Arduino components, etc.), especially finding the right drone	David	2/7	2/14	x	x															
Load Balancing Algorithm (1/2)	Perform research on optimal load balancing algorithms and implement in Go	Ronit	2/7	2/14	x	x															
General Website Setup	Create base template for Django website, do front/backend setup of site, and research hosting live video streaming	Nina	2/7	2/14	x	x															
Design Presentation Slides	Design Presentation Slides	All	2/14	2/18		x	x														
Find methods for obtaining drones	Find where and how to purchase/obtain drones to use that are suitable for our needs - talk with robotics professors (like Basti)	David	2/14	2/21		x	x	x													
Load Balancing Algorithm (2/2)	Implement the load balancing algorithm in Go, and finish unit testing. Research fetching the video stream from the DCC	Ronit	2/14	2/21		x	x	x													
Implement GPS Tracking	Research and implement GPS tracking of drone using GoogleMaps API on top of Django website	Nina	2/14	2/21		x	x	x													
Talk with Basti, Work on rover pivot by re-examing parts and redesign project	Talk with Prof. Basti about drones. Talk with Tamal and Kim about failure, redesign project to be accomadating for rover over drone usage	David	2/21	2/28				x	x												
Rover Camera & Communication Research	Research camera modules with moving feature that can be attached to rover, order, research new communication methods of camera data to Ronit and monitoring site since drone fly app is removed	Nina	2/21	2/28				x	x												
Computer Vision Algorithm on the Distributed Server	Implement the Computer Vision Algorithm on the CV processing units. Integrate with video frames and unit tests	Ronit	2/21	2/28				x	x												
Obtain rover project materials, initial work, spring break	Order rover and PTZ camera, examine rover code	David	2/28	3/13					x	x	x	x									
Implement communication from Raspberry Pi to the Distributed CV server	Use FFMPEG to receive a uploaded video stream and break it into several frames for batch processing in Go	Ronit	2/28	3/13					x	x	x	x									
Implement working PTZ camera on RPi desktop	OS conversion from Buster to Bullseye. Firmware updates and changes to fix stale external trigger issue with camera, order new ribbon cables in order to fix no camera display issue, implement PTZ on RPi with nondeprecated libraries that OpenCV can support	Nina	2/28	3/13					x	x	x	x									
Obtain more mats, switch from rover wifi to UART comms	Purchase smaller batteries and laser heads, test with rover web app, work out the correct JSON commands to send over UART	David	3/13	3/20								x	x								
PTZ refinement and camera debugging	Implement keyboard press control of PTZ gimbal using Arducam libraries and continue debugging camera firmware/potentially hardware issue of stream going stale and no camera display showing	Nina	3/13	3/20								x	x								
Integrate frame breakdown with CV server code/ build laser pointing circuit	Build a laser pointing circuit using the Raspberry Pi pins and GPIO. Integrate frame breakdown with the Distributed CV server and write unit tests	Ronit	3/13	3/20								x	x								
Rover movement with UART comms	Work out the correct JSON commands to send over UART and/or debugging rover movement, set up new RPi for concurrent work (VNC, ssh, Bullseye)	David	3/20	3/27									x	x							
Camera Frankenstein	Order new imx219 camera and attach to camera serial data of RPi along with PTZ portion of broken camera and disable communication between the components to allow for separate movement and live stream, research live streaming from RPi to personal computer	Nina	3/20	3/27									x	x							
Communication from CV server to rover	Write scripts to use TCP to communicate from the CV server to the rover to turn on the laser pointer	Ronit	3/20	3/27									x	x							
Finalize rover movement and tuning	Get controlled movement for rover through extensive debugging, and tune it to do exact turns and movement	David	3/27	4/3										x	x						
Real time communication of video data	Implement RPi streaming communication through use of OpenCV and picamera2 libraries to send video data to Ronit and monitoring site and embed stream into site through iframe	Nina	3/27	4/3										x	x						
Monitoring Site Refinement & Scripting	Create script that will have preset camera movement pattern to look around and also further flesh out web application with manual movement controls of camera	Ronit	3/27	4/3										x	x						
Threading	Have the code running on one RPi with threading between video streaming and code logic, or by using two RPis	All	4/3	4/10											x	x					
Laser pointer accuracy	Coordinate math to be able to point laser correctly, accounting for angle and positioning; Work with CV server to handle location and pointing logic	Ronit & David	4/3	4/10											x	x					
Integrate physically together	Integrate everything, make mount	All	4/10	4/17												x	x				
Final Presentation Slides	Final Presentation Slides	All	4/17	4/21													x	x			
Test entirety of setup	Make sure everything is integrated and working with server and website	All	4/17	4/24													x	x	x		
Prepare for final demo	Fine-tune anything that is needed	All	4/24	5/4															x	x	x
Final Report	Final Report	All	4/24	5/4															x	x	x

Figure 11: Gantt Chart