Flying Under the Radar

Linsey Szabo, Ayesha Gupta, and Angie Bu

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**We propose a universal drone attachment that uses mmWave radar and machine learning to accurately detect humans in fire and fog search and rescue (SAR) conditions. Our web application interfaces with the device to allow the user to drop a pin to mark a victim's location for subsequent rescue. From a hovering drone, our device–a lightweight, handheld attachment–will detect moving humans from 10 meters away in fire and fog conditions, providing a cheaper, automated SAR drone application.

***Index Terms*—**Chassis, drone attachment, GPS, mmWave radar, safety, SAR, sensor, 3D-CNN, web application

I. Introduction

RECENTLY, SAR applications have expanded by using drones [1]. Drones provide many advantages; they limit the exposure of first responders to extreme weather conditions and dangerous terrain, cover large swaths of area efficiently, and are compatible with high-definition cameras and sensors that enhance SAR missions. For these missions, the standard maximum flight altitude is 10 meters, and the duration is a maximum of 30 minutes [2], [3].

However, there are a few drawbacks to the current technology. Because they utilize a high-definition camera and infrared sensing, SAR drones are very expensive. Furthermore, they do not perform well in high temperature conditions and are subject to visible occlusions like fog and smoke. Lastly, they require manual identification of victims.

Our universal drone attachment overcomes these barriers to aid first responders. By using a millimeter-wave (mmWave) radar, we provide a more cost-effective solution, making our technology more accessible for public agencies like fire departments and the National Park Service. Our mmWave radar is also more robust to adverse weather conditions and fire rescue situations, broadening the utility of our SAR drone application. Specifically, we will focus on fire and fog situations. The radar functions by measuring the range (distance to target), Doppler (relative velocity of the target to the drone), and azimuth (angle to the target) data. The Doppler data in particular helps us locate moving targets, since they will create a different velocity relative to the drone compared to stationary targets. Therefore, we will be using this to our advantage. Using machine learning, we automate the manual identification of victims, increasing the efficiency of the mission where rescuing victims is extremely urgent.

To attach our device to the drone, we will encase it with a 3D-printed chassis that will go around the drone's rails. The first responder will fly the drone remotely and hover in certain areas that they choose to examine. The attached mmWave will transmit radar data to the machine learning architecture that is embedded in our web application. By viewing our web application interface, the first responder can see if a human is detected, and if so, they are sent the exact location of detection and can save that location for subsequent rescue.

II. Use-Case Requirements

Since we are building a SAR drone attachment, drone compatibility, increasing the efficiency of missions, safety of first responders, and cost are the most important factors of our application.

Our device must work well with a drone. Its size and weight cannot impede the drone, and it must easily attach. With our material we estimate our device will weigh less than 0.5 kg and have an area of 11 square inches. By encapsulating our device in a 3D-printed chassis, we can attach our device to any drone via its rails. While using plastic in high temperature situations may invoke environmental questions, we have taken this into consideration and will implement a high temperature warning system for the user, so that neither the plastic nor our device is ever damaged. Lastly, our device must work for the duration of a drone flight and work at the standard flight altitude. These last for roughly 30 minutes, so our device must maintain that functionality for at least that long; our device must work within 10 m above the ground.

Moving onto increasing the efficiency of missions, SAR conditions indicate people in crisis situations–it's paramount that these victims are rescued as quickly as possible. Our device facilitates this by automating the human detection process, pinning locations of victims, and having overall real-time functionality. Instead of first responders manually scanning for victims and not only wasting time but also having a more substantial environmental impact, we can point out where victims are using the mmWave radar and machine learning; pinning locations of victims allows first responders to mark spots of rescue, so that they can then systematically dispatch rescuers. Finally, our system must accomplish all of this in a timely manner under the pressure of a crisis situation.

The safety of first responders is also very important. By making a drone attachment, we help limit their exposure to extreme weather and dangerous terrain conditions.

Finally, since our application would be used by fire departments, it needs to be affordable to enable accessibility. Currently, drones are very expensive and employ a high-definition camera and thermal imaging to find victims. Our mmWave application overcomes this cost barrier, because mmWave radars are substantially cheaper. We can make it easier for fire departments to use our product and improve the overall process of rescuing people from wilderness fires.

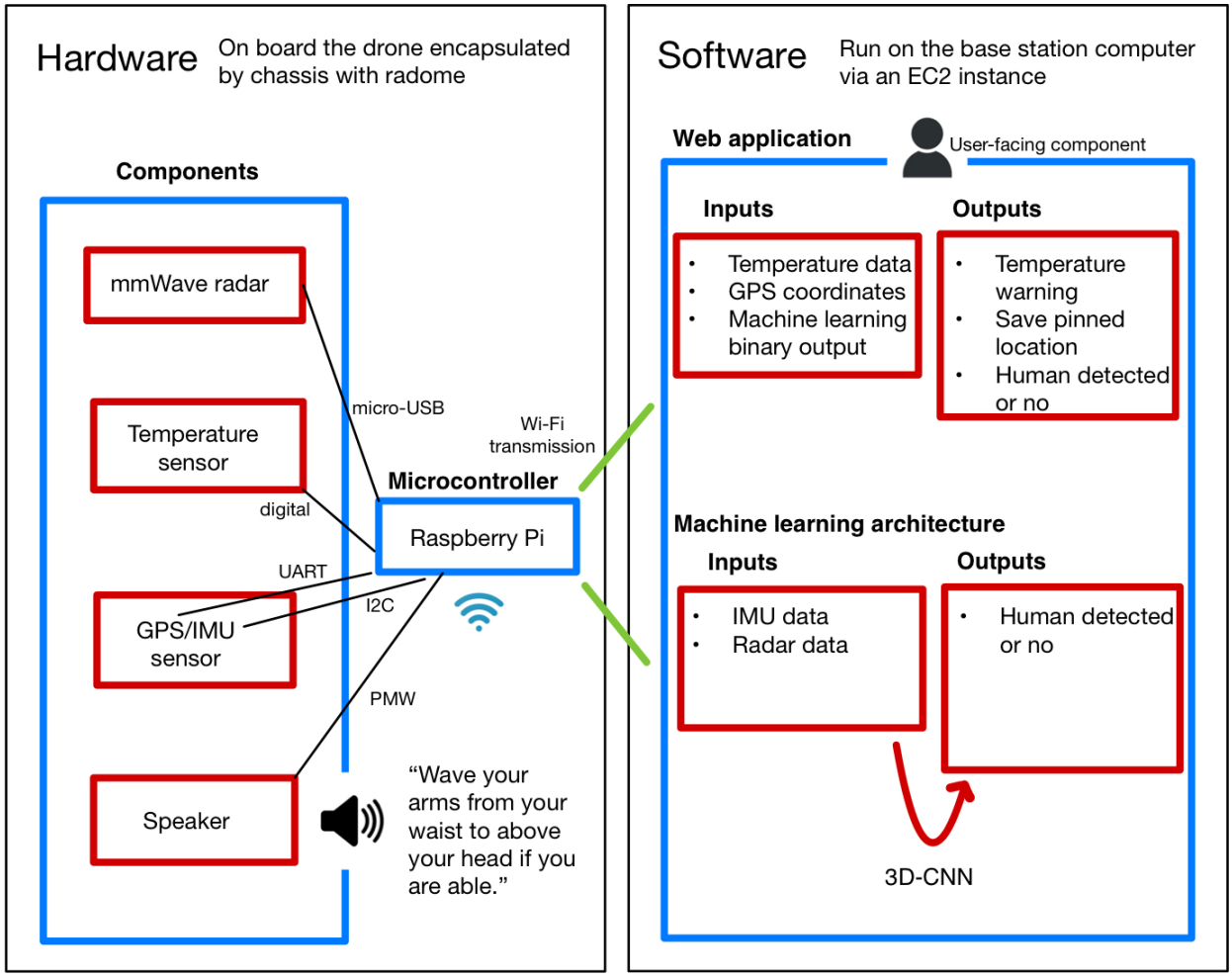

Fig 1: System block diagram

III. Architecture and/or Principle of Operation

Enclosed in a 3D-printed chassis and attached to the drone, the device collects range-Doppler and range-azimuth coordinates through the mmWave radar and temperature, Global Positioning System (GPS), and inertial measurement unit (IMU) data through their respective sensors. The device also includes a Raspberry Pi (RPi), which is key for data transmission. Lastly, on board the drone, there is a speaker, which is a safety addition.

We implemented an important high temperature feature to ensure the functionality of our device during its deployment in fire and fog missions. To prevent the first responder from flying the drone into temperatures that can cause functionality degradation, the temperature sensor data is captured on board the drone and sent via the RPi to the web application. Once the measured temperature exceeds 100° C, the web application indicates to the first responder that the current environment is dangerous for the drone and device, so the first responder can safely fly the drone out of harm's way.

The speaker also enhances the usability of our system. By playing a message like "wave your arms from your waist to above your head if you are able" repeatedly, it effectively alerts nearby victims to wave their arms. This movement will enhance their ability to be detected by the Doppler shift that we observe with the mmWave radar.

Embedded in the web application in the base station computer, the machine learning architecture is connected to the web application via the Django REST API. The range-Doppler and range-azimuth coordinates are sent via the RPi to the machine learning architecture. Inference is performed when the IMU data–specifically the drone's velocity and horizontal acceleration–indicates that the drone is upright and stationary. The 3D-CNN (convolutional neural network) architecture then runs and determines whether a human has been detected by outputting either valid range, Doppler, azimuth coordinates indicating a successful detection or invalid coordinates for no detection. That binary value is forwarded to the web application.

If a human has been detected, the web application alerts the first responder of the detection. The web application uses the GPS data that has the same timestamp as the radar data to determine the location of the detected human. The web application will then provide the user with the ability to save this location on the map and drop a marker to track this location, using the Google Maps API.

IV. Design Requirements

Table I: Design Requirement Metrics

Requirement	*Metric*
mmWave radar detection range	< 5 m
GPS localization accuracy	0.5 m
Temperature warning point	100 ºC
F1 score	0.7
Web application latency	100 ms
System latency	3 s

A mmWave radar must be able to produce data from 5 m away such that the machine learning architecture can detect humans from the radar returns. This range is safely within 10 m, which is the standard flight altitude for a SAR drone. The human may be obscured by obstacles such as fog, fire, and smoke. A temperature sensor that can function between -20◦ and 120◦ C and output whether the temperature exceeds 100◦C is to be included to inform the user that the drone and radar system are operating in high temperatures that may damage the drone or the radar and shorten its service lifetime. A speaker is included to output audio to alert potential victims of its presence. By encouraging the victims to wave their arms, this movement will help our system detect them and increase the efficiency of the SAR mission. GPS and IMU sensors are included to determine the location, speed, and orientation of the drone. A RPi controller is included to collect data from the sensors and transmit the data to a base station computer, which runs our web application. The assembly is powered by a 9V battery, enabling it to function for the 30-minute flight time. It is encased in a 3D-printed enclosure including a radome to protect the electronic components from damage caused by hostile environments the drone may encounter such as smoke and fire.

The machine learning architecture for detecting humans will have an F1 score of at least .7; similar architectures have achieved F1 scores in the range .6-.7 when predicting over multiple classes (i.e. identifying many different objects), so we will beat that metric when predicting over just two classes–there is a human present versus there is not. This ties back to the user requirement of increasing the efficiency of the SAR mission. We need to accurately detect humans in order to successfully automate the victim searching process. It is also important to clarify that this metric will be achieved on our own radar data. In terms of latency, similar architectures ([4]) take approximately 40 ms for inference time. Therefore, to achieve real-time functionality, the speed of the web application is the more important component.

Given that our use case for search and rescue missions can occur outdoors where there may be slow service, we are allowing for a maximum latency of 3 seconds end-to-end of our entire system. This means that for our web application, when strong signals are available, we expect the latency of the web application functions, such as loading pages, to be 100 milliseconds, since this is perceived as instantaneous to users. The purpose of the web application ties back to our use case requirement of keeping first responders safe by being able to track detected humans and knowing their exact location without having to search the entire area first.

V. Design Trade Studies

A. Radar

A 120 GHz radar module was used due to its availability from CyLab and ability to stream raw ADC samples in real time and an API for easy control via RPi. In the beginning of the project, we had used a 77 GHz TI AWR1843Boost radar module due to its availability, small size, lower power usage (380 mW vs 2000 mW). However, while this radar had superior penetration, it required an additional DCA1000EVM module to stream real-time raw radar data which was not available and alone cost more than the $600 budget. Therefore, we chose the 120 GHz radar module.

B. Temperature Sensor

A temperature sensor with analog output was chosen due to its immediate availability and low cost ($1) compared to a temperature sensor with digital output (> $5). Since the design requirements for this system only prescribe notification of temperature beyond a threshold, it is easier to implement this functionality with analog output than digital.

C. Machine Learning Architecture

For the machine learning architecture, the tradeoffs considered pertained to the preprocessing and the architecture itself. For preprocessing, we weighed the difference between reconstructing the 3D data on the RPi versus in the architecture. However, reconstruction on the RPi requires sending higher dimensional data over Wi-Fi, which would consume bandwidth that may be very sparse in the wilderness. Therefore, we chose to perform this preprocessing in the machine learning architecture.

Next, for the architecture, we needed a design that would be able to learn complex relationships among the 3D data. Traditionally, CNNs are 2D, so the kernel only slides in two dimensions–this is what Linsey has worked with many times before. However, after conducting research, having a kernel that slides in all three dimensions is better able to learn relationships for 3D data. Therefore, we chose a 3D-CNN architecture. To support this approach, [4] and [5] both used 3D-CNNs to detect targets using radar data. Additionally, [4] provided relevant metrics for F1 scores, leading to the benchmark .6-.7 range that we mentioned.

D. Web Application

There were not too many tradeoffs we considered. To create the web application, we used Django because Ayesha has experience with this, and it is simple to set up a base site with the simple pages that we wanted to lay out. In order to fulfill the design requirements, we knew we wanted map functionality. We considered both GeoDjango and Google Maps API. The GeoDjango library is much more limited, and since we want our web application to allow us to add pins to the map and receive GPS data, Google Maps API provides that capability. We chose to deploy our web application on Amazon Elastic Cloud Compute (EC2) because we are given full control of EC2 instances, and it allows for fast deployment. There were not any significant differences between EC2 and Azure since Google

Compute Engine does not have all of the functionalities that EC2 and Azure do. Since Ayesha has previously done deployment with Amazon EC2 and it was efficient, we chose this route.

VI. System Implementation

Our system has both hardware and software components. The software is split into machine learning and frontend.

A. Hardware

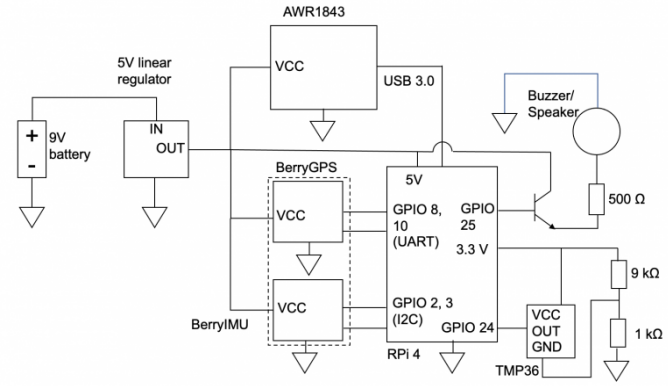

Fig 2: Hardware block diagram

The sensor suite consists of a mmWave radar module, IMU, GPS receiver, and a temperature sensor. A speaker is also included to broadcast audio to potential humans in the search area and is connected to the RPi. All the aforementioned components are controlled with a RPi which transmits the data via Wi-Fi to a base station computer, which runs the web application. The unit is powered by a single 9V battery and is enclosed in a chassis (including the radome) additively manufactured from polylactic acid (PLA). The Silicon Radar TRX_120 module transmits FMCW chirps from 122 to 123 GHz from a 2 by 2 patch antenna array and receives from another 2 by 2 patch antenna array and streams raw ADC samples to the RPi through a micro-USB connection. The GPS data is streamed through UART, and the IMU data is streamed through I2C to the RPi. The temperature alert is streamed along with the radar, GPS, and IMU data through Wi-Fi included on the RPi.

B. Machine Learning Architecture

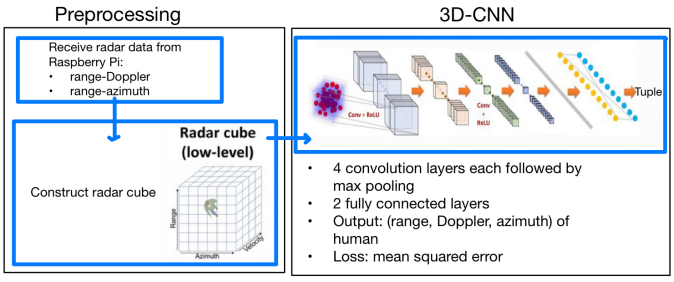

Fig 3: Machine learning architecture block diagram

The machine learning architecture is embedded within the web application. The radar data–both the range-Doppler and range-azimuth data–is received by the base station computer from the RPi through Wi-Fi. First, the drone needs to be hovering in a stationary position in order for the input to be valid. The drone is considered stationary if the drone's velocity and horizontal acceleration, calculated from IMU data that has been damped with a FIR filter on the base station computer, do not exceed 0.1 m/s and 1 m/s^2.

From there, by multiplying the range-Doppler and range-azimuth data along the corresponding axes, we can create a low-fidelity 3D tomographic reconstruction of the scene. This radar cube is fed into the 3D-CNN network, which consists of the following in this order: 4 convolution layers each followed by max-pooling and then 2 fully connected layers at the end. The convolution layers work to learn relationships along each of the range, Doppler, and azimuth axes. By following each convolutional layer with a max-pooling layer, we prevent overfitting and reduce the output size between layers. The fully connected layers do the final reduction in output size step by step, until we eventually output a tuple with 3 elements– (range, Doppler, azimuth). After each of these layers, we employ a ReLU activation function to prevent the vanishing gradient problem during training and introduce nonlinearities into the network to better learn the potential location of a human. By employing this 3D-CNN architecture, we will detect micro-Doppler features produced by moving objects in the radar's frame. From these features, we can then deduce which correspond to a moving human. If no human is detected, the range value will be -1 (an invalid value). Otherwise, there will be valid range, Doppler, and azimuth values in the returned tuple.

During training, mean squared error will be the loss used to tune the network. This loss is necessary, because we are concerned with real-valued, continuous outputs; mean squared error will effectively compute the difference between our output and our ground truth target, while penalizing outliers. Once detection is complete, the web application will receive a yes or no output. Then, we must create the machine learning architecture-web application interface. This will be done using the Django REST API, which will allow our machine learning model to deploy within our web application, so that the output of the machine learning model is accessible to the application.

C. Web Application

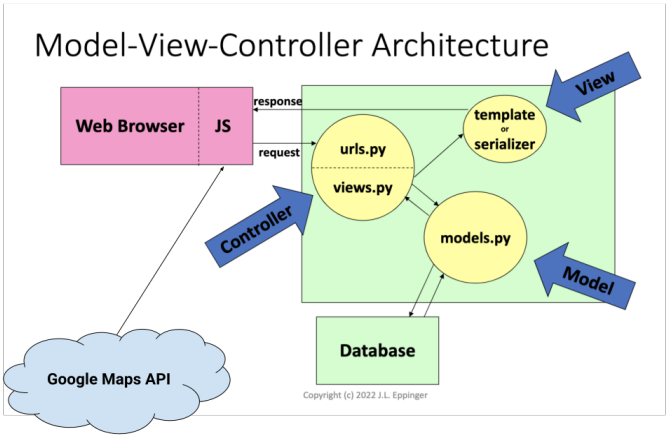

Fig 4: Diagram of web application architecture and interactions.

The web application architecture will be implemented using

the Django framework in conjunction with AJAX and the Google Maps API. Django allows for easy website creation and maintenance and is implemented in python. From the browser, an HTTP Request will be sent to the server, and using Django, we will be able to connect that request to a specific URL, retrieve the associated data, and display the associated page. AJAX will be used in conjunction with Django to send and receive data such that the browser display and behavior is not interrupted. This is where we will utilize the Google Maps API in order to display a dynamically updating map. The Google Maps API will interact with the JavaScript files, as well as the templated HTML files, to display a map and add marker functionality. GPS data will be sent from the RPi to the web application containing the location of the device when a human was detected from the captured image so that the location can be marked on a map, displayed by the maps API. Finally, we will be deploying our web application on an SDK to increase latency. We will deploy our site on Amazon EC2.

D. Integration

In order to connect our hardware and software components, we are using an RPi. This RPi will send the radar data to the machine learning architecture to begin the image processing and running the human detection algorithms. The RPi will also collect the sensor data, as outlined in figure 2, and send that to our base station computer through Wi-Fi. This base station computer is what runs the web application. This is important because the GPS data needs to be sent from the RPi in order for users to be able to drop pins on the map which is displayed on the web application.

The machine learning model is run from a Python file, which allows it to be deployed within the web application using the Django REST API. Since our web application uses Django, we can use the Django REST Framework, which is free. This framework is easily installable within our application and will work with our web application being deployed on EC2.

VII. Test, Verification and Validation

Our goal for testing all our components is to achieve the metrics outlined in our design requirements summary table (see Table I).

A. Tests for mmWave Radar

To test the radar functionality, data of different scenes will be captured from a stationary location: scenes with moving humans waving their arms, no humans, and humans obstructed by fog at different distances from 1 to 10 m away in increments of 3 m. To simulate fog, we will use a fog machine. Additionally, because it is risky to deploy our device in high temperature conditions, we will use our device's performance in fog to generalize to smoke, since fog and smoke have the same properties with respect to the radar's penetration ability.

B. Tests for Temperature Sensor

To test the functionality of the temperature sensor, the output of the heat alert is monitored as the temperature sensor and a separate thermometer are heated from room temperature to 125^{o}C using a heat gun. System integration testing and latency measurement is achieved by measuring the difference between the timestamp sent from the RPi and the time upon receival by the base station computer.

C. Tests for Machine Learning Architecture

For initially training the machine learning architecture alone, we will use the drone radar dataset from Ghent University [6] with a 70-30 training-validation split. It is composed of range-Doppler, range-azimuth, and position data taken from a hovering, upright drone which is taking images of a corner reflector–our model human. This preliminary training is to ensure that the network integrates with the radar data with respect to dimensions and sensible metrics, i.e., accuracy increases with training (we do not have a specific metric in mind, we are merely observing trends). Once we have collected more radar data (~1000 samples), we can use this for further training the model to better generalize to our real-world scenarios as mentioned in section A. During these tests using our own collected radar data, we will be comparing the F1 score against our .7 requirement.

D. Tests for Web Application

We will manually test the web application by trying out the functionality ourselves and having others try it out and getting feedback. This will be how we test the qualitative aspects of the site, such as how easy it is to navigate and how visually appealing it is. We will test the latency of the web application functions itself, such as the timing to add a marker for a specific geolocation and the timing between simple actions like switching pages and scrolling. As mentioned in the design requirements, when strong service is present, we expect these functionalities to seem instantaneous to users, so we will be testing for a latency of 100 ms.

E. Integration Tests

For integrating the machine learning architecture with the web application, we can return a binary value and ensure that the web application correctly displays to the first responder whether a human has been detected. For integrating the radar and sensors with the web application and machine learning architecture, we will ensure that the data is transmitted by the RPi to the software end.

VIII. Project Management

A. Schedule

We will all be working on our individual sections in parallel for the first half of the semester. We plan to begin integrating during the second half of the semester, so around mid-March. See Figure 5 on the next page for the schedule, shown in a Gantt chart. Linsey is blue, Ayesha is brown, Angie is green, and the remaining colors involve multiple people, if not all of us.

Our major tasks include the following:

1. Acquire radar
2. Find dataset
3. Set up web application
4. Capture radar images
5. Train ML architecture
6. Validate ML architecture
7. Test ML on radar images
8. Send images to web application

9. Test sensors
10. Integrate Google Maps API
11. Add marker functionality
12. Send sensor data to web application
13. Test ML output and temperature warning display on web application
14. Test entire system

B. Team Member Responsibilities

We have both hardware and software components in this project, but we have split it up into three specific concentrations—hardware, machine learning, and web application.

Angie has a lot of experience with hardware and signal processing, and she had a specific interest in using the radar we procured. Angie will be working on capturing the images with the radar, connecting the sensors (GPS/IMU and temperature) and the speaker, and using a RPi to store and send the images to our software system.

Linsey is minoring in machine learning, so she will be working on the image processing portion of our project.

Ayesha has experience with building web applications, so she will be creating the frontend portion of our project. Ayesha's secondary responsibility will be to help Linsey with the machine learning architecture as needed.

All three members will test their individual portions on their own and test the integrated system parts together.

C. Bill of Materials and Budget

Our total budget for this project is $122.45. This is because we were able to borrow our most expensive items from labs such as CyLab. If we were to have purchased each item, the total price would have been $1,175.58. Our bill of materials is located on page 8 (see Table II).

D. Risk Mitigation Plans

In the event that we do not meet our desired test outputs, we have some plans to adjust each component in order to overcome any issues.

If our hardware accuracy is low, we plan to increase the data rate of the radar. If our hardware speed is low, we plan to decrease the data rate of the radar. This will affect the two most important metrics of the hardware components.

For the machine learning architecture, if we do not meet our desired F1 score of 0.7, we will tune the hyperparameters of the CNN. Specifically, this involves changing the filter size and the number of layers in the CNN.

For the web application, we will already be using an SDK to improve latency. Therefore, if latency is still an issue in areas with strong service, we will reduce the number of HTTP requests, as well as simplify the templates, which include the HTML and CSS files, to increase the latency of the web application.

IX. Related Work

To obtain our dataset, we examined this study [6] that collected FMCW data with a mmWave radar mounted on a stationary drone. While this study collects data in several scenarios, we focused on the one where a corner reflector, an

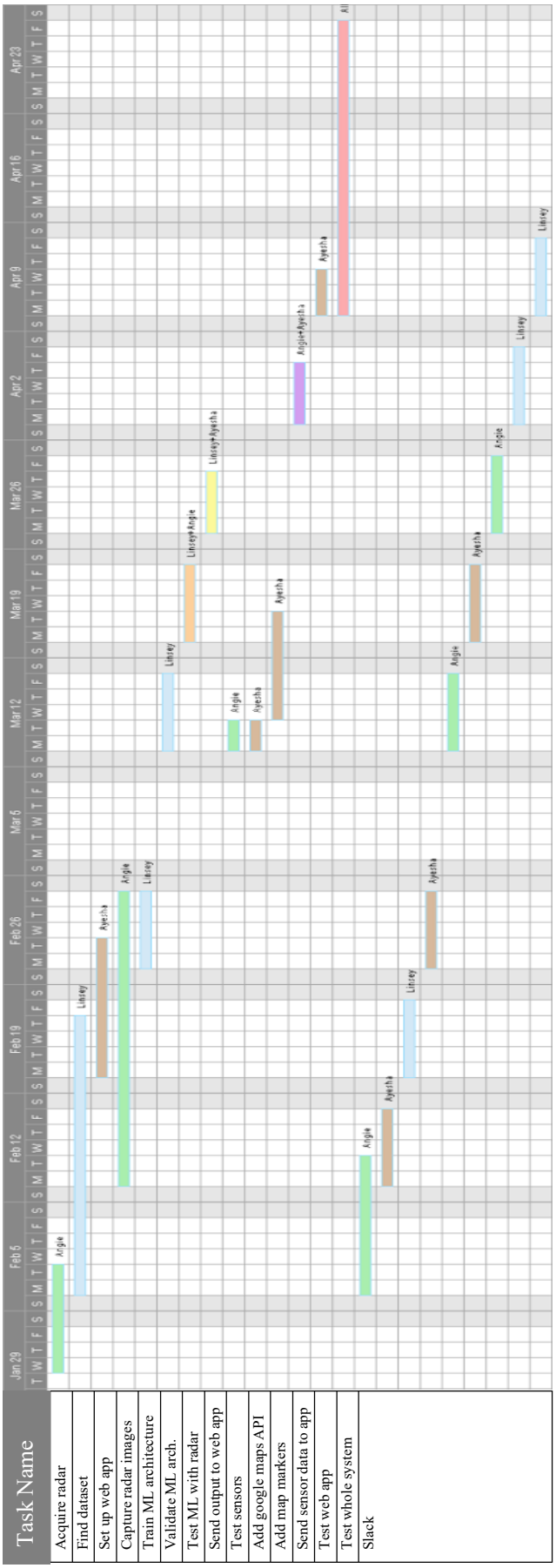

Fig 5: Gantt chart with milestones for the semester.

aluminum foil pyramidal reflector, is placed at the center of the open space and the drone hovers in front of it; this scenario provides us with 2869 training examples. The corner reflector is our model human as it ensures strong returns for the radar signal. We acknowledge that this corner reflector isn't moving like our intended human target would; it will only mimic a stationary human with much smaller Doppler shifts like e.g., breathing. Therefore, it will only be used for initial training of the machine learning architecture.

For the system implementation, we examined two studies [4] and [5] that used radar to classify targets on the road. Both construct 3D cubelets to accurately represent the range, Doppler, and azimuth data. They also use 3D-CNNs to classify their targets. Therefore, for our architecture, we reconstruct the 3D representation using the range-Doppler and range-azimuth data and employ a 3D-CNN architecture. Although these papers classify over multiple classes (they focus on cars, bikes, and pedestrians), we adopted their methods for our two-class prediction problem.

For the architecture code, we followed the construction of the network in [8]. It details a 3D-CNN network for classifying CT scans. Their initial code contained only 3 convolution layers. Linsey added an additional layer to more closely follow the RTCNet architecture mentioned in [4]. Therefore, [4] and [8] were key to building the machine learning architecture.

X. Summary

Our universal drone attachment serves to aid and improve SAR missions by providing a more cost-effective and robust solution. Our device will use mmWave radar to overcome barriers that are typically faced with infrared and cameras, such as seeing through fog and smoke. Our device will also be lower cost than what is currently available with our total being $1,175.58, and SAR drones currently in use are ~$3,300 [2]. Additionally, because our attachment is independent of drone peripherals and weighs less than 0.5 kg, it will be compatible with most drones used for SAR missions (the average weight a drone can carry is 0.5-2 kg) [7]. Our device will capture images during SAR missions and send them to be processed by our machine learning architecture. Finally, our device will communicate with a web application that will receive location data and allow users to pin specific locations on the map based on where humans are detected; to ensure the safety and usability of our system, we included a temperature sensor which in turn allows us to alert the user when the drone is flying in conditions that are too hot, and we included a speaker to instruct victims to move their arms, making it easier for our device to detect their presence.

We foresee that image capture with our radar will be a challenging task because the data is relatively low resolution. Integration is always challenging. We have allotted a lot of time for this by planning to start it mid-March.

Glossary of Acronyms

EC2 – Elastic Cloud Compute
GPS - Global Positioning System
IMU - Inertial measurement unit
mmWave - millimeter-wave
RPi - Raspberry Pi
SAR - search and rescue
SDK - Software Development Kit

References

[1] R. Tariq, M. Rahim, N. Aslam, N. Bawany and U. Faseeha, "DronAID : A Smart Human Detection Drone for Rescue," 2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT), Islamabad, Pakistan, 2018, pp. 33-37, doi: 10.1109/HONET.2018.8551326.

[2] "Inspire 2 - DJI." *DJI Official*, DJI, https://www.dji.com/inspire-2.

[3] Vision Aerial, Vision Aerial. "How to Use Drones for Search and Rescue." *Vision Aerial*, Vision Aerial, Inc., 15 June 2021, https://visionaerial.com/how-to-use-drones-for-search-and-rescue/.

[4] Palffy, Andras, et al. IEEE ROBOTICS AND AUTOMATION LETTERS, 2020, *CNN Based Road User Detection Using the 3D Radar Cube*, https://arxiv.org/pdf/2004.12165.pdf.

[5] K. Aziz, E. De Greef, M. Rykunov, A. Bourdoux and H. Sahli, "Radar-camera Fusion for Road Target Classification," 2020 IEEE Radar Conference (RadarConf20), Florence, Italy, 2020, pp. 1-6, doi: 10.1109/RadarConf2043947.2020.9266510.

[6] Safa, Ali, et al. IDLab, Ghent University, Leuven, Belgium, 2023, *FMCW Radar Sensing for Indoor Drones Using Learned Representations*, https://arxiv.org/pdf/2301.02451.pdf. Accessed 2 Mar. 2023.

[7] Dronedek. "How Much Weight Can a Delivery Drone Carry?" *Dronedek The Mailbox Of The Future*, Dronedek, 10 May 2021, https://www.dronedek.com/news/how-much-weight-can-a-delivery-drone-carry/.

[8] Zunair, Hasib. "Keras Documentation: 3D Image Classification from CT Scans." *Keras*, Keras, 23 Sept. 2020, https://keras.io/examples/vision/3D_image_classification/.

TABLE II. BILL OF MATERIALS

Total budget: $122.45

Description: mmWave radar **Name:** TRX_120_067 **Manufacturer:** Silicon Radar **Cost:** $0 ($1000 on the market) **Notes:** Radar and evaluation board borrowed from CyLab
Description: Controller for the sensors **Name:** Raspberry Pi 4 Model B with 8GB RAM **Manufacturer:** Element14 **Cost:** $0 ($75 on the market) **Notes:**
Description: Temperature sensor **Name:** TMP36 **Manufacturer:** Adafruit **Cost:** $1 **Notes:** Bought from TechSpark
Description: GPS and IMU sensor **Name:** BerryGPS-IMUv4 **Manufacturer:** OzzMaker **Cost:** $71.20 **Notes:** Single board with separate power supplies and communications
Description: Speaker **Name:** AS02008MR-LW152-R **Manufacturer:** Digi-Key **Cost:** $0 ($3.13 for a single speaker, $1.81 in bulk) **Notes:** Part from previous coursework
Description: 5V Linear regulator **Name:** L7085 **Manufacturer:** STMicroelectronics **Cost:** $0.25 **Notes:** From TechSpark
Description: 9V batteries **Cost:** $0 **Notes:** From previous coursework
Description: NPN transistor **Cost:** $0 **Notes:** From previous coursework
Description: Google Maps API **Name:** Google Maps API **Manufacturer:** Google **Cost:** $50 ($100 without coupon) **Notes:** Using educational coupon
Description: Resistors **Name:** 500 Ω, 9 kΩ, 1 kΩ resistors **Cost:** $0 **Notes:** From previous coursework
Description: 3-D printed chassis **Name:** 3-D printed PLA **Manufacturer:** TechSpark **Cost:** $0 **Notes:** Self-designed
Description: Wires **Manufacturer:** **Cost:** $0 **Notes:** From previous coursework