FollowMe: An Indoor Blind Aid

Authors: George Chang, Jeffery Cao, Ging Luo
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable of detecting obstacles, providing voice assistance, and providing real-time road info for the visually impaired user.**

***Index Terms*—Blind Aid, Deep Learning, Object Detection, Robotic System, SLAM, YOLO, Edge Detection**

1 INTRODUCTION

Vision is oftentimes considered the most important sense of all senses (Maybe except Nociception – the sense of pain). Losing the sense of vision can lead to a wide variety of inconveniences. They will lose the ability to detect colors, depth, and motion visually.

In order to compensate for the loss of one sense, they frequently use auxiliary tools and technologies, such as text-to-speech, braille, and accessible apps that speak when interacting. Similarly, when blind people want to move from one place to another, they frequently use a walking stick, which they move the stick side to side to detect any impending obstacles. Blind who have extra budgets also adopt guide dogs, which can lead the way for the blind. Unfortunately, guide dogs are expensive to buy and care for, and they are not permitted in sensitive locations such as hospitals or labs. Due to financial and regulatory reasons, only 2% of the blind have a guide dog at all times, according to the Caring Eyes Foundation.

Walking sticks also have their own limitations. According to Jeamwatthanachai et al., Walking sticks are too short to give blind people confidence in indoor navigation, and they give too little information to the user [2]. The purpose of the walking sticks is to provide tactile feedback for the user to let the user create a mental map, but unfortunately, when the blind person is at a indoor environment, many of the cues are lost: The environment might be too noisy to determine the texture using a walking stick; randomly placed obstacles might hinder the blind's ability to create a mental map; Lack of obstacles that walking stick can respond will lead to a sense of insecurity for blind, as it has no cues for location [2]. Such unconfidence is one of the main reasons that blind people refuse to leave their homes frequently, which severely damages their work, study, and their social lives. As such, a blind aid that is able to give blind people more information than a walking stick in the indoor setting is needed.

In order to answer the need, our project will be aimed at indoor obstacle recognition and notification in planar ground. By using the system we developed in this project, the user should be able to move inside the Hamerschlag hallway with minimal guidance or external help such as a guide dog or volunteers. With the voice command provided by the system, the user should be able to gather information in a room or in the hallway compared to just using a walking stick.

The project itself was not meant to replace walking sticks, as tactile feedback was mostly trusted by blind people [2], and walking sticks can serve as second layer of detection if the system misses the obstacle. Instead, it serves as an extension to the walking stick, by giving the blind person more information to boost their confidence in an indoor setting, thus increasing their participation in work, study, and social events.

2 USE-CASE REQUIREMENTS

We define the scope of our system to work in flat indoor hallways without dangerous obstacles like cars/scooters dashing at the blind. The hallway contains no steps or doors. We also believe that other people will avoid the blind automatically, so the blind only need to avoid stationary obstacles by themselves. As such, we will focus on stationary objects. For convenience, we have organized the use-case requirement by each of their use case (or product requirement):

- The product shall detect obstacles accurately
  - Our system shall be able to detect large obstacles 5m away with 90-100% accuracy. This includes both moving and stationary obstacles.
  - This is based on past research on object detection for the blind, which achieved 80% accuracy in both indoor and outdoor environments [1].
- The product shall identify obstacles accurately
  - Being able to identify obstacles can boost users' confidence in walking and make their social lives easier [2]. Our system shall be able to identify a specific subset of interesting obstacles with greater than 85% accuracy. We include specific subsets of obstacles to be objects frequently appearing in an indoor environment, including desks, chairs, elevators, doors, etc.

  - Derived from past research is able to achieve 80% of accuracy in a variety of environments [5].

- The product shall notify the user of obstacles in real-time
  - Our system shall be able to notify obstacles with 500 ms latency. We believe other people will avoid the blind automatically, so the blind only needs to avoid stationary obstacles by themselves. Based on the 5-meter detection range and 1-second reaction time of the blind user, the blind user will have about 2-3 meters of space left with a walking speed of 1m/s. This is a reasonable reaction space, so we require the latency to be 500 ms.
  - Also the product should be able to notify the user of the closest obstacle. Since the information about an obstacle is dense enough, mentioning multiple objects will be overwhelming for the user, as they will share the same information processing capability with a non-blind person.

- The product shall have enough battery for 1 hour
  - Our system should be able to navigate the blind for about an hour. In most cases, blind people will not use the full battery because it's rare to have them walking for an hour in indoor hallways. The battery can be recharged once the blind people reach their destination.

- The product shall be light enough less than 5 pounds
  - Our system needs to be light enough so that the user can comfortably carry it for an hour. We estimate the weight should be less than 5 pounds because an empty backpack with a bottle of water is about 5 pounds on average.
  - In our use case, we consider the public welfare of blind people. The recognition system can replace guide dogs because it's cheaper and easier to carry around. So the weight needs to be minimized to maximize the utility of this product and maximize the public welfare.

- The product shall be economical
  - Our system should be affordable for blind users. The user's smartphone can work as a camera, sensors, and a processor. Other than the smartphone, the system should cost less than $100 dollars. Each guide dog costs about $50,000 annually, so our system can enhance blind people's welfare with a limited budget.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

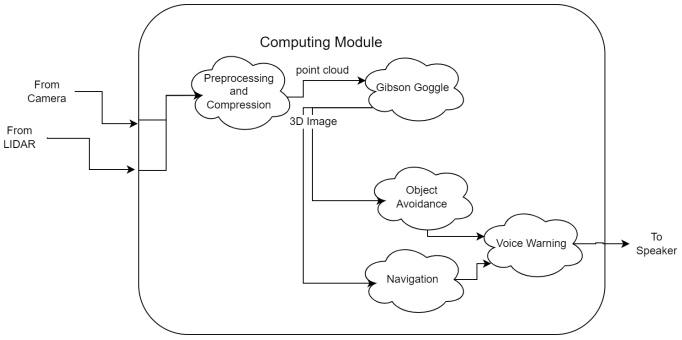

Figure 1: Software Implementation Pipeline. Input: Camera and LIDAR picture, Output: Strings inputted to Speaker controller code

The architecture of the design can be simplified using 2 block diagrams, one hardware, and one software.

The software pipeline explains how is the output created from pictures of the real world. From the input picture captured using a visible-light camera and LIDAR, the processing pipeline will create a glb formatted mesh file that can be fed into the neural network called Gibson Goggle (a perceptual and physics Simulator developed at Stanford University), which will spit out a 3D image encoded with depth information. Afterward, the 3D image will be sent to the object recognition and navigation algorithm, where features of the environment will be detected. At the end, the features will be analyzed by the postprocessing algorithm developed by ourselves, and output a voiced warning based on the information provided.

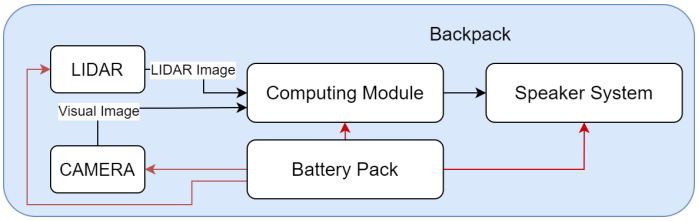

Figure 2: Hardware Implementation Block Diagram

The Hardware block diagram describes all electrical parts that can be used to perform to meet such requirements. We used lidar and camera as the input sensor, which the data will be sent to the computing module, which is local to the user to process data on time. Then the speaker system will receive the voice command from the computing module and eventually play the voice. Since the project is mobile, a battery back is used to support the power.

4 DESIGN REQUIREMENTS

With the architecture and principles of operation in mind, we then translate the use-case requirements to design requirements from the perspective of engineers:

- Sensors Detection Range
  Since we need to detect moving objects within 5 meters, e.g. a running kid with roughly 3m/s speed, we need to account for the 500ms latency and at least 1 second reaction time for the blind user. By the equation

$$d = 5m + 3m/s \cdot (1s + 500ms) = 9.5m \quad (1)$$

  the detection range needs to be at least 9.5 meters, about 10 meters.

- Camera
  For the camera, assume the camera's zoom lens was set to 50mm and the center-to-center pixel spacing is 3.89 µm, which is very common for camera lens[6], then we can get that we need 3214 pixels if each object constitutes exactly one pixel based on our calculation.For the deep learning model we are using, the input channel needs at least 16*16 pixels as input data, so the camera needs at least about 0.8 million pixels[6]. The frame per second needs to be at least 20 fps because the model needs at least 10 pictures in the 500ms latency period based on our estimation of the model[8].

- Processor
  We prioritize the user's safety, so we use a local computing machine such that the computation can be carried out in a reliable and fast way. The computation can't be hosted on the cloud to reduce cost, because cloud computing may not be reliable enough to ensure the integrity of data transmission and real-time data processing.
  By inspecting representative YOLO (Deep learning object detection) models, We estimate that most of them need to run on GPU with a minimum of 8GB memory requirement. NVIDIA GPU with CUDA support should be ideal[11]. A model with a desirable mAP(mean Average Precision) > 50% often requires more than 500 Billion FLOPS[8].

- Model
  Based on the use case, we need to achieve a 90% accuracy model with less than 500ms latency. The model needs to recognize common indoor objects such as elevators, desks, etc. The model also needs to recognize moving obstacles. One choice is YOLOv3-spp model. It can achieve 60.6% mAP though it requires > 100 Billion FLOPS. Another choice is a smaller YOLO model such as TinyYOLO with a lower mAP of 23% but it only requires only one fourth of computation power than YOLOv3-spp[8].

- Battery
  We will use a 4000 mAH lithium battery power bank. The use-case requires the battery to power the system for about an hour. Based on the common processor and sensor's spec, we estimate that the average power consumption is about 20W with regular voltage of 5V.

$$I = P/U = 20W/5V = 4000mA \quad (2)$$

  Based on the calculation, the battery needs to output 4000mA for one hour, so we think the 4000mAH battery should be enough.

- Speaker
  For the speaker, it should be loud enough so the blind can clearly hear. Also it needs to be relatively low cost, less than $20. It also needs to be lightweight, less than 0.5lb. This is to give more mass quotas to the battery bank and the microcomputer.

- Tripod Head
  For the stand or the tripod head that will hold the sensors, we require it to be strong and lightweight. Since the entire system needs to be less than 5 pounds in use-case, we expect the stand to weight less than 2 pounds and the material should be strong enough to hold 2 pounds, which is an upper-bound estimate of the sensors' weight.

5 DESIGN TRADE STUDIES

To satisfy our use case requirements, we have researched on four different software design algorithms and respective hardware components available to us online and considered on the advantages and disadvantages of them. The four designs are very different in terms of implementation and how they solve the same object recognition problem. [Table 1] So instead of putting them on the same metrics, the tradeoffs of the designs are more analyzed in a pair-wise manner where the difficulties of implementation, whether we can accommodate these algorithms to fit our use case, or whether the technique is too powerful and would lead to a waste of computing power in our use case is discussed. The software algorithms are VISUAL SLAM, LIDAR SLAM, YOLO, EDGE DETECTION respectively.

Before the in-depth discussion about the pros and cons of each algorithm, a "Too Long, Didn't Read" table of the summary is included to give an overview of different algorithms. Based on the TLDR table, we will choose to implement YOLO v3 SPP as our main focus, and then Edge Detection as a backup for reliability. Then we will explore other models if time allows.

5.1 Visual SLAM

5.1.1 What is Visual SLAM

Visual SLAM is the technology/process of determining the object's relation with its surroundings while mapping

Table 1: Tradeoff Summary Table

	Complexity	Detection	Preprocessing	Overall
Visual SLAM	-4	3	-1	-2
LIDAR SLAM	-4	3.5	-1	-1.5
YOLO v3 SPP	-2	2	0	0
YOLO v7	-3	2	0	-1
Edge Detection	-1	1	0	0

Side Note: Relative rating per column, the higher the better

its environment at the same time with the camera being the only sensor.

Most Visual SLAM uses point cloud algorithm which compares the position of points in successive photo frames and uses the relative distance change across all the point groups (same point position change across different frames) to calculate (through geometry and deep learning algorithms) the distance of the data point in relation to the camera. The object will then be extracted based on the distance calculated and a 3D map will be generated based on the location of the object.

In an 2013 research paper on 3D mapping from visual SLAM,[3] it demonstrates how they achieve 3D mapping with feature extraction and point cloud algorithm: use feature extraction which forms a point cloud, mapping of points at different image frame to calculate the 3D positions which is then optimized and accumulated to form the 3D map they want. They have feature extraction → descriptor (feature points) → match of points → transformation validation with point cloud they get from distance measurement to compose a graph.

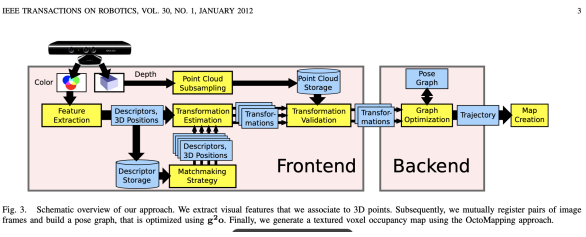

Figure 3: Visual SLAM Architecture

5.1.2 Pros and Cons of Visual SLAM

Recall that the goal of the software system for our project is to recognize close-by objects in real time, Visual SLAM implementation will surely satisfy this requirement.

Our design is that we will use Camera (currently we have Intel RealSense Lidar Camera L515 at hand) to capture visual image which will send picture streams to our "computer" (currently we have Nvidia Jetson Xavier NX) which will run our SLAM model (based on Gibson model) to produce a 3D image. The 3D image will then be processed (maybe compressed) to run object avoidance/navigation algorithms developed by us.

The Intel RealSense Lidar Camera L515 supports both camera and lidar measurements and has a rgb resolution of 1920 x 1080 with 30 fps rgb frame rate.[6] These would satisfy our use case since in our experimentation with the camera, the clarity of videos being shot is good with unrecognizable delay in frames. By looking at the architecture design of the camera, the frames taken by cameras will be sent to drivers/USB ports, then to RealSenseId library and then to user applications which we can communicate with our computing device via APIs. Gibson's Goggle Model can be a helpful module to simplify the API callings [10]: It is an open-source CV model that uses a visual sensor (camera) to output real-time 3D images. We will use it for training/testing data generation and for easier visual slam 3D map generation.

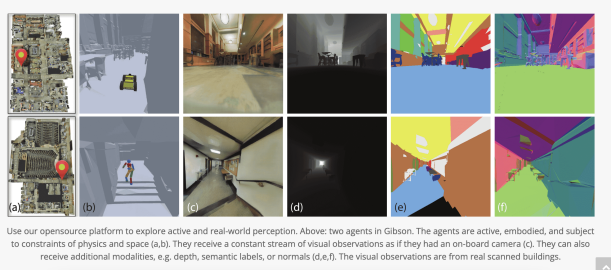

Figure 4: Gibson Goggle Outputs

Compared to the latter model choices we have, Visual SLAM provides us with the most amount of information and better accuracy: constructing a 3D map will allow us to know not only the distance from the object but also the angle and the shape of the object which can be helpful for us to do more detail analysis like object identification (if the object presents a threat to our client or not) or more detail information giving (not just in front of you, but also identify obstacles to the left of our client).

However, the Visual SLAM does have many limitations and disadvantages such that we would not put this as our primary/secondary design implementation for this project:

1. Visual SLAM is hard to implement: a typical Visual SLAM system requires thousands of lines of code for its

core model and subsequent training to tune its parameters. As shown in figure 4 and 5, open source models tend to be trained on a specific indoor environment such that their parameters are tuned to produce accurate result in their settings. Even if we choose to develop our own Visual SLAM based on their model, it could be tricky to tune their parameter because our use case (indoor hallway settings) are different from theirs in terms of objects available and people movement.

2. Visual SLAM is an overkill for our purpose: a typical Visual SLAM model as mentioned before will produce a 3D map of the environment in the end and requires several rounds of data collection under the same environment. We do not need a 3D map in order for us to detect objects and give navigation help to blind persons. Whatever behind or to the side of the client may not matter if we just want to go straight.

3. This is a common technical challenge for any visual-based object recognition algorithm: Visual SLAM is impacted by the quality of the image. There are many scenarios where even from human's perspective would be different to judge the distance. For example, if there is a strong light source in the environment, the surrounding area of the light source would be dim and more difficult for object recognition algorithm to recognize objects (this is common in indoor settings where we have sunlight shines through windows.). We saw in our testing video shots that under strong sunlight, objects will reflect sunlight and lead to difficulties in distinguishing objects.

5.2 Lidar SLAM

5.2.1 What is LIDAR SLAM

Similar to Visual SLAM, Lidar SLAM's goal is also to generate a 3D map of its surroundings.[4] Different from Visual SLAM, the Lidar is now the sensor and only source of input. Lidar sensors can generate dense data clouds where points in 3D space represent the surface of objects. With the surface of objects generated and the distance to the surfaces known, the algorithm will then be able to generate a 3D map of our surroundings which consist of all the surfaces we identified.

An example of how Lidar SLAM is trained can be found in the image below:

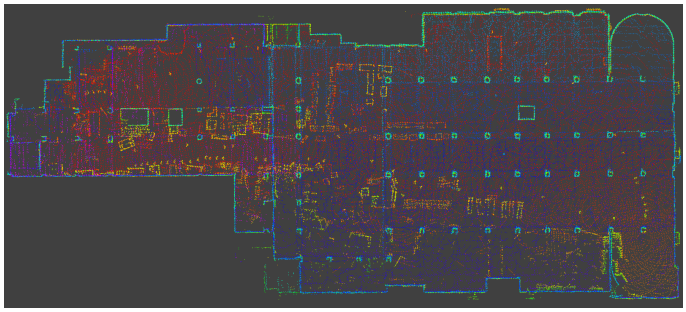

Figure 5: Sample Lidar SLAM Test Map

As shown in the image, the red points are points in which they use Lidar to collect data points for its surrounding and the lighter lines denote the boundary/objects detected by the Lidar.

For our project, similarly to the visual SLAM, the resulting 3D map will be processed (maybe some data processing algorithm to reduce the size of map for better bandwidth and latency). Then it will be sent to our object recognition/navigation algorithms which result will be sent to the speaker to give out directions.

5.2.2 Pros and Cons of LIDAR SLAM

Unlike visual SLAM where we need to calculate the distance from the image/3D map, Lidar SLAM can get a more accurate distance measure with high consistency. It would not be impacted by the shaking of camera when our client walk which is a common technical challenge for other vision-based algorithms. As our use case is to detect obstacles/objects in our way, Lidar detection is sufficient for distance measuring (in terms of it can detect objects' distance accurately within its detection range) and further processing to give instructions.

However, similar to the Visual SLAM approach, Lidar SLAM has disadvantages which contain some of the same disadvantages that any SLAM has when applied to our project which are:

1. Lidar is impacted by the brightness of the environment quite heavily: in our testing of the Lidar available in Intel RealSense 515 Sensor, it can detect as far as about 9 meters in dark room but only about 4-5 meters in bright room (room with strong sunshine). The worst detection range is less than what we have in the design requirement which is "at least 9.5m".

2. Like visual SLAM, Lidar SLAM is an overkill for our purpose and hard to implement: we do not need a 3D map to satisfy our use case and it would be hard for us to implement a Lidar SLAM from scratch in 6 weeks. We have thoughts about only using the Lidar-computed distance graph like the one below, but the tricky part is that we would then need to implement an object recognition

algorithm from the distance graph. Despite that the detection range is short for our use case, having Lidar to get the distance of the objects seem to take an extra step since we can get if there is obstacle in one step in Yolo which is based on vision.

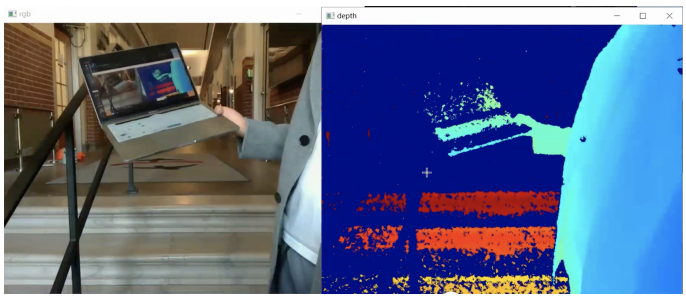

Figure 6: Lidar Data Sample

5.3 Yolo

The model we choose to use is YOLO s3 SPP. The details of it are further discussed in System Implementation. YOLO is a real-time object detection model that can be used to detect obstacles such as desks and identify objects such as elevators, doors, etc.

Compared to SLAM, YOLO is smaller and easier to implement. It doesn't involve localizing the user and memorizing a map of past trajectories. As a result, its accuracy (mAP) might not be as good as SLAM because the detection is not continuous. However, based on our use case where we only need to detect stationary obstacles, we think this tradeoff is acceptable.

Compared to edge detection models, YOLO has additional capabilities to recognize objects. This is important for blind users because based on our research, it is helpful to point out important objects to the blind as well as detect obstacles. These objects can serve as Visual Landmarks to boost confidence in walking. [2] However, it is not the focus of our project, and we will only point out important landmarks such as a large desk in the hallway, an elevator on the left, etc.

The details of the YOLO algorithm will be omitted as it will be thoroughly discussed in the system implementation section to avoid repetition.

5.4 Edge Detection

5.4.1 What is Edge Detection Algorithm

Edge detection algorithms are image processing techniques that used to sharpen the edges of objects shown in pictures where edges are defined to be lines of points that shows discontinuity in brightness/color within the image. After an ideal edge detection algorithm is run on an image, the contour lines of objects shown in the image will be shown whereas other more detailed feature will be obscured.

5.4.2 Pros and Cons of Edge Detection Algorithm

Edge detection is the secondary approach we plan to go for our project (after the Yolo approach).

A popular edge detection algorithm implementation that is available online is the OpenCV implementation.[9]To incorporate the edge detection algorithm in our model, the workflow will go as follows: the camera will pipeline images to the edge detection algorithm which detect the edges of objects. The result of contour line pictures will then be processed for object recognition (a script by us to recognize object from edges). The object recognition result will then be sent to a navigation instruction computation algorithm for instruction generation and speaker to speak out the instruction.

There are many advantages of this approach:

1. The work we need to do in training/tuning the algorithm is less compared to the SLAM approach: edge detection in openCV already contains multiple edge detection algorithms that we could test our model on. We do not have to test our model in videos in order to generate a SLAM, images would suffice for edge detection testing.

2. We are more controllable about what object we want to recognize: all other approaches will detect all objects that are in the image frame. This would lead to excess calculations and waste of battery life and computing power which have constrained in design requirement session (we do not need to call out a painting on the wall as a potential source of danger for example). We could specify on things we want to detect in this approach: human, tables, chairs, doors covers all the objects that pose a potential danger to a blind person in most safe flat indoor hallway settings we identified in our use case requirements.

However, incorporating the edge detection algorithm to our use case can be time-consuming: We would have to implement an object recognition algorithm from edge detection which can take lots of time to ensure 90-100% detection accuracy in our requirements. Edge detection method is prone to noise in image where detected edge can include minor details that affect accuracy. For example, in the image shown below, with the focus of the image drawn close to the viewer, the detail of flower and the hair of the girl is detected as edges. Thus we decide to implement the Yolo approach first before attempting the edge detection algorithm.

Figure 7: Edge Detection Sample

6 SYSTEM IMPLEMENTATION

The system can be divided into several subsystems as follows:

6.1 Sensors

This is the sensor we will be using. The camera has 2 million pixels and 30fps. The lidar can detect 9-meter range with 25-centimeter depth accuracy. We will be mainly using the camera for the YOLO model and edge detection model. The Lidar sensor is for SLAM model which we had explored in design tradeoff analysis part.

Figure 8: Intel Realsense 515 Sensor

6.2 Electrical Parts

Figure 9: Electrical Parts

Figure 10: Electrical Parts

For the electrical part, the speaker, battery, sensors, and processor will be connected as indicated in Figure 2: Hardware Implementation Block Diagram.
The speaker and battery we bought are listed here. This part doesn't involve many complexities because we are simply connecting the parts. Our design choice for the electrical part is based on the 1-hour battery life use case requirement.

6.3 Mechanical Parts

Figure 11: Mechanical Parts

For the mechanical parts, we will use the tripod head as shown above to hold the sensors and the speaker. The body of the tripod head will be inserted into the frontback. The tripod head shown here looks larger than it really is and we ensure its weight is less than 2 pounds. In this way, we have considered the welfare of blind people of waling with ease and comfort.

The reason why we want to use a frontpack instead other ways to connect the recognition system to the blind is its simplicity. Compared to a drone or a guide dog, it's easier to carry and cheaper to purchase and maintain. Compared to a backpack, the tripod head ensures the view sight won't be blocked.

6.4 Data Preprocessing

In our system, data preprocessing is broken down into 2 steps.

The first step is the control pipeline of the sensors. We will not utilize full fps of the sensors because 30 frame-per-second is both an overkill and a threat to our computational power. So we will sample only 10 fps and transmit the images into a raw data folder on NVIDIA Jetson. Then, python codes will truncate the image into squares for ease of the model. We may also want to filter out malformed images such as overexposure at this step.

Then, the input data can be converted to 3D simulated point cloud by utilizing Gibson Goggle Network as shown below. The conversion to simulation environment might not be needed for classical object detection model such as YOLO though it might improve the stability and conver-

gence of training by eliminating non-ideal factors of real-world settings.

```
path = rospack.get_path('gibson-ros')
assets_file_dir = os.path.dirname(assets.__file__)

class Goggle:
    def __init__(self):
        #self.rgb = None
        rospy.init_node('gibson-goggle')

        self.image_pub = rospy.Publisher("/gibson_ros/camera_goggle/rgb/image", Image, queue_size=10)
        self.depth_pub = rospy.Publisher("/gibson_ros/camera_goggle/depth/image", Image, queue_size=10)

        goggle_img = (recon.data.clamp(0, 1).cpu().numpy()[0].transpose(1, 2, 0) * 255).astype(np.uint8)
        goggle_msg = self.bridge.cv2_to_imgmsg(goggle_img, encoding="rgb8")
        self.image_pub.publish(goggle_msg)

goggle = Goggle()
goggle.run()
```

Figure 12: Gibson Goggle Code

6.5 Object Detection Model

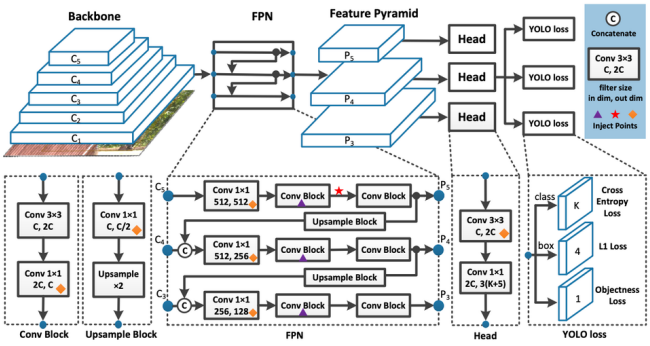

Figure 13: YOLO Architecture

The model we will implement is YOLO v3 SPP. We will primarily focus on implementing the object detection model first, and then further exploring edge detection models and SLAM. So this section will mainly talk about YOLO models, since other potential candidates are discussed in design tradeoff section.

The model we choose is YOLO v3 SPP. Compared to normal YOLO v3, it has a high mAP due to the feature selection of its spatial pyramid pooling layer. There are many variants of YOLO model, and the latest one is YOLO v8 released by Ultralytics. However, although YOLO v3 seems a little bit archaic, the progresses YOLO made is mainly on resolution and recognizing small objects. In terms of mAP (mean average precision), YOLO v3 SPP is comparable to latest v8. Also, according to our use case, the objects and obstacles in hallways are mainly large objects like desks and elevators. So the special focus on detecting small objects is not only hardly useful but also computationally expensive. Finally, I'm more familiar with the classic architecture of object detection using anchors and bounding boxes (state-of-art YOLOs use anchor-free technology). Therefore, we chose to use YOLO v3 SPP model to fulfill the task.

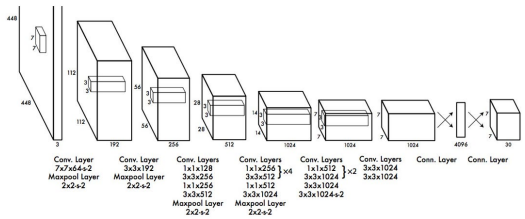

Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Figure 14: YOLO Backbone - Convolution Layers

Object detection models like YOLO can be divided into a backbone, a neck and a head. Darknet-53 is a convolutional neural network that acts as a backbone for YOLOv3. It made some advancements such as using residual layers (skip connections) to avoid the issue of vanishing gradient. The backbone will extract high-level features and feed to the neck.

The neck is an FPN (Feature Pyramid Network), as shown in figure 9. It has multiple layers of downsampling to combine and mix features in various dimensions. It can anchor features to bounding boxes at different scale so it can detect objects in varying sizes.

Then, the features will be passed to the head (as shown in figure 9), which is another convolution layer that can predict the label and location of the object. It will draw bounding boxes with regression and predict labels with classification. Then, after training, the head can output the most accurate/confident detected objects with NMS (Non-Naximum Suppression) algorithm [7].

6.6 Instructions Control Pipeline

In this subsystem, the input is the label and location of detected objects identified by YOLO model. The goal of the subsystem is to drive the speaker to output auditory signals.

Based on our research, it is helpful to point out important objects to the blind as well as detect obstacles. These objects can serve as Visual Landmarks to boost confidence in walking [2]. However, it is not the focus of our project, and we will only point out important landmarks such as a large desk in the hallway, an elevator on the left, etc. This can be easily achieved with labelled objects from the output of YOLO.

Since our use case focuses on the scope with only stationary obstacles, we can directly estimate the distance of the obstacle based on the depth map of the lidar.

The control code will also provide basic road information for the user such as how far is the nearest intersection.

Since our object detection model won't directly point out intersections, the control needs to figure out the location of intersections based on the depth map. An alternative approach is to train an edge detection model in addition to the object detection model that focuses on detecting intersections. But the details are still in plan and we may choose to cut down this road info feature in the end.

7 TEST & VALIDATION

7.1 Hardware: Camera

To meet the design requirement of a camera, we will test its behavior on various lighting conditions: Sunny day, Overcast day, and Fluorescent Light (Which simulates an isolated hallway or a hallway at night). With those three lighting conditions, we then test the Average Noise of the picture, using Scipy's sigma function to estimate the noise value. We expect a noise value of less than 10. With a noise value > 10, it will be rather difficult for the pipeline to process.

7.2 Hardware: Computing Module

To test the power usage and computing power of the computing module, we will then evaluate the following metrics with our implementations of YOLO algorithm under 10W, 15W, and 20W of power. For each algorithm, we will insert timing functions inside the image fetching functions of the pipeline to time the average time (in seconds) to fetch one frame, and that value will be multiplied by the wattage (resulting Joule) used by the computing module. Ideally, we want a value less to 2J, which corresponds to a 10-frame-per-second image fetch rate at 20W.

7.3 Hardware: Battery Pack

We will perform integration testing for the battery bank, which the battery bank will be connected to the computing module. Then, the module will operate with 20W of power running the YOLO detection model. Then, we time its time of operation until the system dies out. We are expecting an operation time greater than one hour (60 minutes)

7.4 Software: Object Detection Methods

Depend on the algorithm we had implemented will conduct different testings to determine the object detection accuracy of each algorithm.

7.4.1 Edge Detection

For edge detection algorithm, we will use a marked set of 30 pictures taken in the main hallways of the Hamerschlag Hall's first floor to serve as the oracles of the testing. Then, we will feed the image to the system, and read its output. The accuracy of the detection will be determined by the percent of object matches between the algorithm's output and the oracle. (We are not looking for pixel-to-pixel correspondence of the two set of pictures, but will only test on logical). We are looking for a accuracy of greater than 90%.

7.4.2 YOLO, SLAM

For other advanced algorithms, other than the detection measurement stated in the Edge Detection section of the measurement, we will also test the object identification. With the same 30 pictures taken in the main hallways, we label the interested location with a name, such as "Chair" or "Garbage Bin". Then the same picture will be sent to the YOLO algorithm, which the system will not only spit out an identification but also a classification. As such, we can compare the detection similarly using the testing method in the Edge Detection section, and for the identification section, we will manually compare the classification of the YOLO model and the oracle, which the classification should be close enough. When two words are not exactly synonyms (Synonyms include: Pavement vs Sidewalk), a wordnet WUP sematic distance of greater than 0.5 should be considered good (Identical words are scored 1.0).

For SLAM models, we also need to test the distance accuracy of the algorithm. In this case, our oracle also needs to include distance information by either tape measure or iOS's Measure App. We are looking for the accuracy of 20% of oracle or 1 meter, whenever is smaller.

7.5 Software: Text-to-speech

For the text-to-speech system, we want to make sure that every output string to the module will get processed. In order to test the TTS system, we will test on all target objects the algorithm is trained to detect, and make sure 100% of the object names can be correctly pronounced. Additionally, we will test a select sample of full voice warnings, calculate the play time and number of syllables, and verify the talking speed is less than 25 syllables per second, which is the maximum speed a blind can comprehend.

7.6 System: Latency

For the latency, we will include timers at the image intake stage and the voice output stage. Then, we will take in multiple images with a detectable feature, which due to the previous section the system will be able to detect it. Then, we will time the time used from the image being fed into the pipeline to the time when the TTS system stopped playing the warning notification. The algorithm we will use will be the SLAM model, which is the heaviest algorithm that is expected to cost the most time. Testing will be marked successful after the maximum processing speed of all example images is less than 500ms.

7.7 System: Weight and Cost

We measure the weight by a scale, which includes every component we will use for this project. The test is passed if the overall mass is less than 5 pounds.

7.8 System: User acceptance

We will conduct two rounds of user acceptance. First, one of our team members will be blindfolded, and move from one side of the baker hall to the other side of the baker hall with protection from other teammates. If there are urgent need for help, other teammates will help, and the first phase of the test is passed only if 0 help is needed to navigate around the baker hall hallway. The second phase will be tested by a blind user with a walking stick. The test participant will navigate from one end of the Hamerschlag hallway to the other end. Then the test participant will score the project on a Likert scale from 1 to 5, The project will be a success if the scale is greater than 3.

8 PROJECT MANAGEMENT

8.1 Schedule

The schedule is shown in Fig. 16.

8.2 Team Member Responsibilities

As we have three people in the team for this project, we will split the work between three of us evenly as shown in the schedule chart mentioned in section A. As a summary, the responsibilities for three of us respectively are:

Jeffery Cao: All hardware-related tasks - Making sure video taken by the camera can be received by our Yolo algorithm in Nvidia Javier NX correctly and on time and implement the speaker which deliver the final navigation instructions.

George Chang: Part of software-related tasks which includes: 1. Implementing script to give instructions based on object detection result; 2. Implement Edge Detection Script if we choose the edge detection approach; 3. Help make sure data transmitted between Camera to Javier, Javier to Speaker is correct and within time bound requirement, develop mitigation plans if it fails. 4. Team Schedule and Team Management

Ging Luo: Part of software-related tasks which includes: 1. Implement Yolo-based object recognition algorithm; 2. Help on developing object detection script.

8.3 Bill of Materials and Budget

The materials needed for our project is listed in Table 1. We have made sure that the total cost is less than $600 budget limitation.

8.4 Risk Mitigation Plans

The major risks for our project are two-fold: first is about YOLO system and second is about the mechanical part.

Regarding the Yolo system, there are chances that the Yolo system may not actually be able to detect object that are 9 meters away (which is required by our use case requirement) in a dim setting or calling the Yolo algorithm costs lots of time to run for our use case and hardware design such that we will miss the latency requirement. When such cases take place, we will mitigate by replacing the Yolo system with an edge detection system as mentioned in section 5.4. As edge detection system is local and resistant to brightness, it can unblock us from the limitations of the Yolo system.

Regarding the mechanical part, it is possible that the way we place camera is uncomfortable from users' personal experience and may need to adjust the design. If this happen, we can replace the tripod head with strips attachment or place the camera in the back with a support of a backpack. We will then tested on these mechanical designs and find the best way that could be more user-friendly yet not reducing too much video quality and detection accuracy.

9 RELATED WORK

During research, our team found out there aren't a lot of technologies that are ready for full blindness, as many of them are aimed at the legal blind, which are people who can see but are not clear enough. Most of the time, fully blind people still rely on traditional techniques such as regular walking sticks guide dogs. In this section, some example project that is similar to our project will be explained.

9.1 WeWalk Smart Cane

WeWalk smart cane is a tool with similar factors to a regular walking stick, which the device will use ultrasonic to scan the obstacles and report back obstacles using a phone app. The app will also store frequently visited places such as restaurants, cafes, and shops. WeWalk is also connected to the public transportation system, where the app can find bus stations which can be then informed to the user.

In general, although it is a blind aid, the stick itself serves more like a control remote to the phone app, where the phone app will do all of the heavy lifting such as navigation, searching, and bus-finding. The ultrasonic obstacle finder only works under a limited range, and will only provide a boolean feedback using the vibration of the walking stick.

In the end, it is dramatically different from our project. Our project has a fixed scope so we can concentrate on good

Table 2: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Jetson Xavier NX	812674024318	NVIDIA	1	$0	$0
RealSense L515 LIDAR	82638L515G1PRQ	Intel	1	$0	$0
Talentcell Rechargeable 12V 6000mAh	YB1206000-USB	Talentcell	1	$39.99	$39.99
USB Mini Speaker	H002	HONKYOB	1	$12.99	$12.99
Tripod Head Stand	14043CM18INCH	QISHI YUHUA	1	$34.98	$34.98
					$87.96

quality navigation only in a hallway setting, where WeWalk acts more like a phone controller for its phone app. The walking stick itself only provides limited object detection, and its detection is only limited to low obstacles, which a walking stick can do equally well. Additionally, due to the time constraint of the project, we did not build a dedicated PCB nor plan to minimize our form factor to fit inside a walking stick.

9.2 CU Boulder – CAIRO LAB: Seat Locater and Object Finder

The research project developed at the University of Colorado Boulder and Collaborative Artificial Intelligence and Robotics (CAIRO) Lab provided blind aid from another point of view, which the aid will act as a walking stick and uses a camera to understand the environment, find a seat, and even read off cereal packages. This system uses a sophisticated computer vision algorithm to map the surrounding, detect the target of interest, and gives a score of confidence which will be used to determine the best seat or where is the cereal box the user want to buy.

One big downside of this project is that the computing power needed to use the AI model developed at the institute requires a full-fledged laptop, instead of a microcomputer that can be stored anywhere. Additionally, the research team was not able to combine all of its functionality into a convenient package, so technically, three different AI model is used for three different scenarios: Navigation, Seatfinder, and Boxfinder. Although this shows the versatility of its algorithm and hardware design, the question of integrability still remains.

Lastly, it is too dramatically different from our project. Our project utilized a microcomputer that have a weaker computing power compared to a full-fledged, expensive laptop, which allows for a long operation time with battery installed. Additionally, we only have one use case topic, where that project has multiple use case topics and needs to be changed manually. In the end, it is more similar to a research demo than a full engineering system that is able to provide end-to-end capability.

9.3 UltraCane

UltraCane is one of the oldest blind aid technologies. It uses ultrasonic to scan the environment in front of the user and vibrate based on distance. The device is relatively simple as it has no other capabilities and no phone apps.

Because of its limited technology, it was not able to have a far range greater than 4 meters. Additionally, it has no object identification capability, and the user will have a hard time understanding what the obstacle is and changing its route.

This system is simpler than our project and has a smaller effective range. With only ultrasonic sensors, the information the stick receives is limited. On the other hand, UltrasCane did not cost much for the components, and a simple micro-controller is suffice to manipulate the ultrasonic distance data.

10 SUMMARY

Overall, the project is very challenging. In order to meet the design requirement use-case requirement, our team has to tackle the challenges of training the neural networks, reaching the appropriate accuracy, managing the power consumption and battery life, and solving many edge cases discussed above such as different lighting conditions. Additionally, our team also has to find the sweet spot between "giving too much information" and "giving too little information", as only giving just the right amount of information will help the blind.

Losing the sense of sight is already a challenging obstacle blind people have to overcome. As blind aid technologies are still vastly overlooked, our project will not only serve as a convenient tool for the blind to use when traveling indoors, but it will also serve as one of the first steps of accessible technology developments. Our project will help by helping blind students who are moving between classrooms, warning obstacles for blind workers, notifying elevators for blind veterans, and many other scenarios involving indoor blind safety.

Glossary of Acronyms

- LIDAR – Light Detection and Ranging
- SLAM – Simultaneous Localization and Mapping
- YOLO – "You Only Look Once", is the name of the neural network object detection algorithm

References

[1] Kedar Potdar et al. "A Convolutional Neural Network based Live Object Recognition System as Blind Aid". In: *ArXiv* (Nov. 2018).

[2] Watthanasak Jeamwatthanachai et al. "Indoor navigation by blind people: Behaviors and challenges in unfamiliar spaces and buildings". In: *British Journal of Visual Impairment* 37.3 (Feb. 2020).

[3] Hyunggi Chang. "Visual SLAM RoadMap". In: (). URL: https://github.com/changh95/visual-slam-roadmap.

[4] Kenny Chen. "Direct Lidar Odometry". In: (). URL: https://github.com/vectr-ucla/direct_lidar_odometry.

[5] Jain Anuj Kumar Nitin. "A Deep Learning Based Model to Assist Blind People in Their Navigation". In: *Journal of Information Technology Education: Innovations in Practice* 21 (2022).

[6] Alan Marcus. "How to calculate the size of object in pixels, knowing the camera properties and distance?" In: (). URL: https://photo.stackexchange.com/questions/90059/how-to-calculate-the-size-of-object-in-pixels-knowing-the-camera-properties-ande.

[7] Manika Nagpal. "What is YOLOv3 Architecture ?" In: (). URL: https://www.projectpro.io/article/yolov3-architecture/836#:~:text=YOLOv3%20Architecture%20Explained&text=YOLOv3%20uses%20a%20convolutional%20neural,features%20from%20the%20input%20image..

[8] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *CoRR* abs/1804.02767 (2018). arXiv: 1804.02767. URL: http://arxiv.org/abs/1804.02767.

[9] OpenCV Team. "Edge Detection Using OpenCV". In: (). URL: https://learnopencv.com/edge-detection-using-opencv/.

[10] Stanford Gibson Team. "Gibson Environment Home Page". In: (). URL: http://gibsonenv.stanford.edu.

[11] Ultralytics. "Ultralytics YOLO Frequently Asked Questions (FAQ)". In: (). URL: https://docs.ultralytics.com/help/FAQ/.

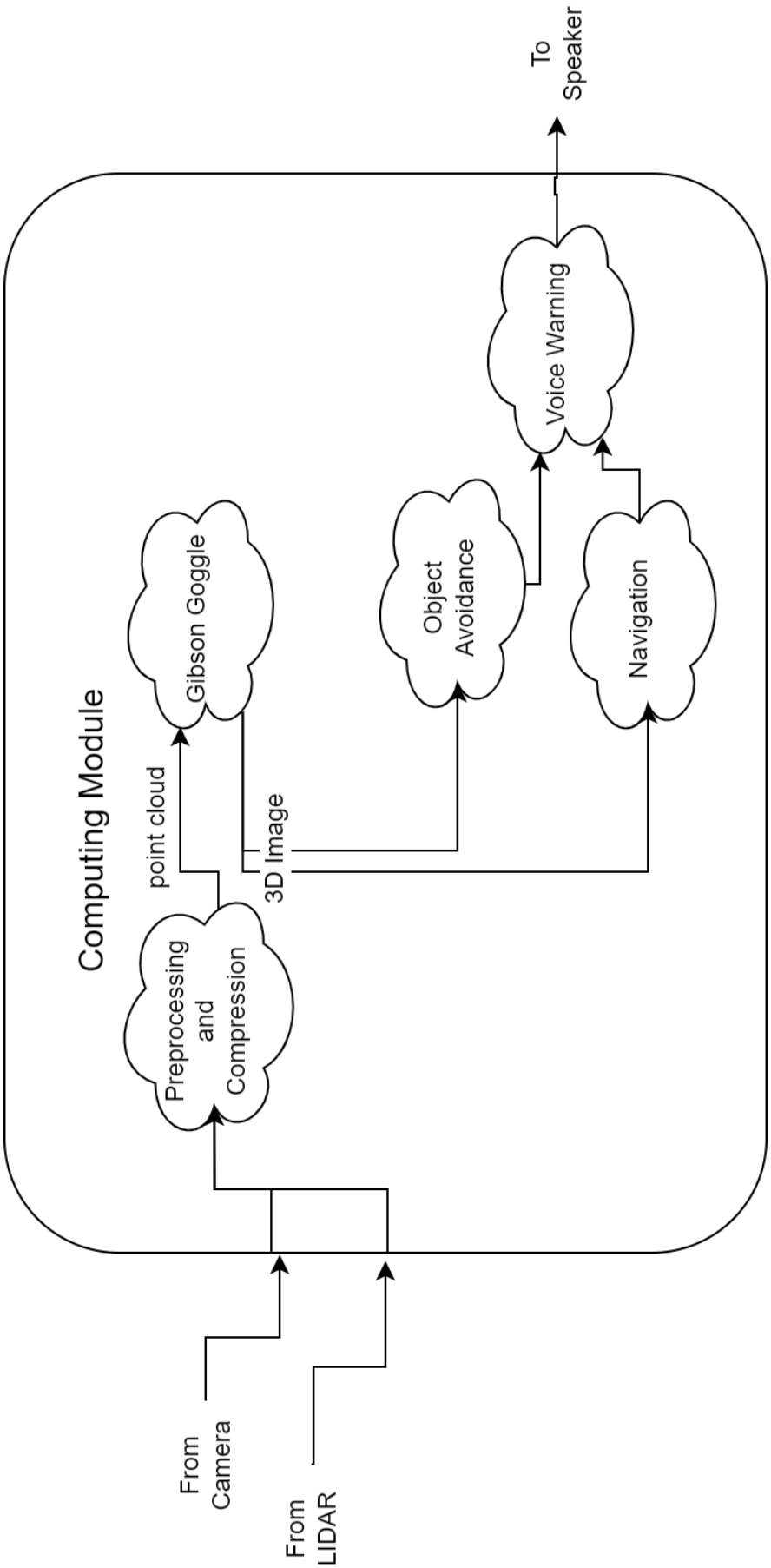

Figure 15: A full-page version of the same system block diagram as depicted earlier.

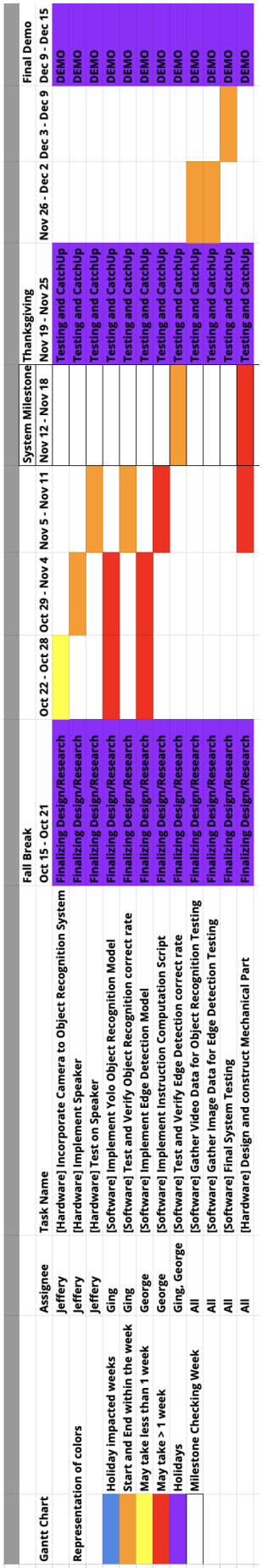

Figure 16: Gantt Chart