xWALK

Authors: Jeanette de La Torre - Duran, Shayan Gupta, Yasaswini Doguparthi; Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—To increase accessibility to the visually impaired, many intersections have certain features to assist these individuals in knowing when to cross. These features include auditory cues with crossing signals and high traffic noise to detect the flow of traffic. However, in many intersections that do not have this external assistance, it can be challenging for a visually impaired individual who is training to navigate these intersections through the use of hearing and a cane. To remedy this problem, xWALK, a wearable device, notifies the user when it is safe to cross an intersection. The targeted intersections lack crossing signals and have low traffic flow. The product is a belt with a camera, raspberry pi, battery pack, and speaker attached.**

***Index Terms*—Visually impaired, navigation, pedestrian, traffic intersections, assistive device**

I. Introduction

Many intersections lack the necessary features to facilitate crossing for the visually impaired. For example, many intersections in Pittsburgh only provide the traffic signal as an indication to cross. Therefore, in these intersections, without crossing signals with auditory cues or with low traffic noise, it may be difficult for an visually impaired person, who is training to recognize traffic flow and crossing cues, to feel secure to cross. Therefore, xWALK is a wearable device that uses OpenCV and TensorFlow to guide users to cross an intersection. The device is a nylon belt connected to a raspberry pi, camera, speaker, and portable battery charger. The user should already be standing at an intersection in the direction of crossing. Once the user is ready, they will click a button to allow the device to take a picture of the intersection. In .5 seconds, the device will detect and crop the correct stoplight and describe the state of the stoplight. Given the state, it will decide whether the user should cross or wait. The device will be used for around 2 hours per day with a battery life of around 6 hours. In order to make sure the device would be usable and relevant, a visually impaired person was interviewed to take into account her concerns with the device. Therefore, the device is for training use only in order to assure a user's decision to cross. There has currently been research in crosswalk detection, but there is no wearable device for a visually impaired person to use specifically for training.

II. Design Requirements

The product has a hardware component for the physical device to take a picture of the surroundings and a software component that contains the training model and image processing. The physical device needs to be wearable and not distracting so the user is comfortable using it in public. Therefore, the device will be a belt with 812.8mm to 863.6mm in circumference. At the front of the belt, we will be using a Raspberry Pi Camera V2 due to its affordability and sufficient quality. Connected on the side will be the Raspberry Pi Model B 4 which will have a wireless speaker connected to the headphone jack and will also be connected to a Raspberry Pi Portable Battery. The speaker will be rechargeable with a battery that lasts 8 hours. On the Raspberry Pi, there will be an enable button for the user to press when they are ready to hear the command. This is the complete hardware that the user will be wearing at the crosswalk. In addition, the device is expected to have a battery life of more than 9 hours given the specification of the Raspberry Pi Portable Battery component. In order to measure this, the equation will be used:

[1] $$Life\ (hr) = Battery\ pack\ life\ (mAh) \,/\, current\ (mA)$$

As for the software requirements for the device, the device needs to ensure the user has enough time to cross the crosswalk and does not delay the user to cross. Therefore, we require the device to have a latency of less than 0.5 seconds. In addition, the device should take into account the yellow light status in order the users have a consistent starting point and do not cross towards the end of a green light. So, the user should only cross once they have seen a red light and then a green light. Another risk is the impeding objects or no traffic light in view. Detecting impeding objects is out of the scope for the project, so the device will only notify the user if it is unable to see the traffic light in the frame.

Safety is an important requirement in the system. False positives are one of the biggest risk factors which assumes the user crosses the street when the light is red. To calculate the FPRs, the equation below calculates the number of injuries which include deaths that the device has caused:

[2] $$injuries/year = \#\ times\ used/year * FPR$$

The risk uses the expected number of times the device will be used per year and outputs the expected deaths per year given our FPR. For the expectations given the scope of the project, 2 deaths per year is achievable. Given an expectation that the device will be used 2600 times per year, or 50 times per month, the FPR is .0769%.

In addition, the device must ensure it is looking at the correct traffic light. The intersections that the device will expect contain only traffic signals and perhaps a stop light for pedestrians to use. The training model must detect the traffic light facing the pedestrian. The device is expected to have more than 90% accuracy. The accuracy is a percentage of the mislabelled pictures/total picturess in a given testing session. Due to the scope of the project, the device will not orient the user. The device expects the user to at least be facing the correct crossing direction.

III. Architecture and/or Principle of Operation

As described in the introduction, our device is an adjustive assistive tool to help visually impaired individuals navigate traffic intersections. Typically, visually impaired individuals rely on their sense of hearing to navigate intersections - by listening to and focusing on the directionality of car engines as they move spatially. With this perception, visually impaired persons determine when to cross by listening to when vehicles that are parallel to their crossing direction start moving (i.e. the sound of engines accelerating those vehicles) as well as the sounds of cross-traffic that is perpendicular to the user's intended crossing direction. The perception of sound is used in combination with the perception of touch, through the use of a cane, which is used to orient the user to where the crosswalk starts, allow the user to feel where the crosswalk continues as he/she crosses the street, and to serve as a visual cue to drivers that someone who is visually impaired is crossing the street.

With these techniques, there are still challenges that remain when crossing these intersections. First, as described earlier, many intersections lack any auditory cues that can help indicate when to cross. Furthermore, one cannot use his/her sense of hearing if there is no nearby traffic - therefore no cars to hear. Also, there is a couple week to month long training period in which a visually impaired person learns to navigate using a cane and hearing, where external assistance (usually another person) is required to prevent training accidents from becoming fatal. Therefore, in the cases of training and/or the lack of traffic, our device will be a useful *adjunctive* assistant to help the user determine when to cross an intersection.

Our expected device is therefore meant to be a portable, adjunctive navigation assistant for a visually impaired individual. To further portability, the device is designed to be worn by said individual as a belt around the head, hips, or torso. Figure 1 shows the final product - a wearable device that is housed on an adjustable belt strap while Figure 2 depicts all of the hardware components that will be assembled into said final product. All the components will be placed on the outside of the belt. This placement by the buckle allows the user to align the camera to face the same direction as the user's body is facing by feeling and rotating the buckle so that it faces forward. In order to turn on the device, the user must turn

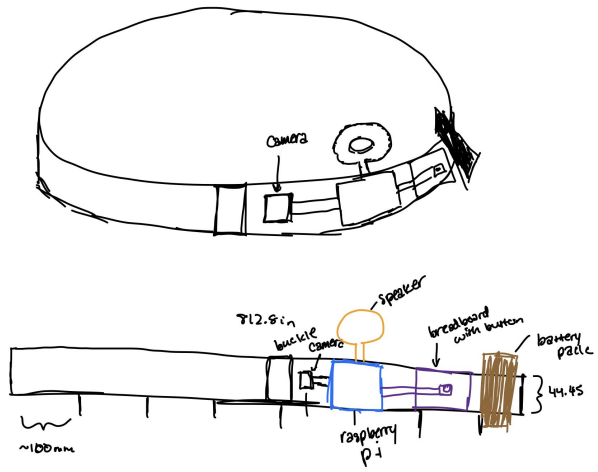

Fig. 1. The envisioned final product - outside (top) and inside (bottom)

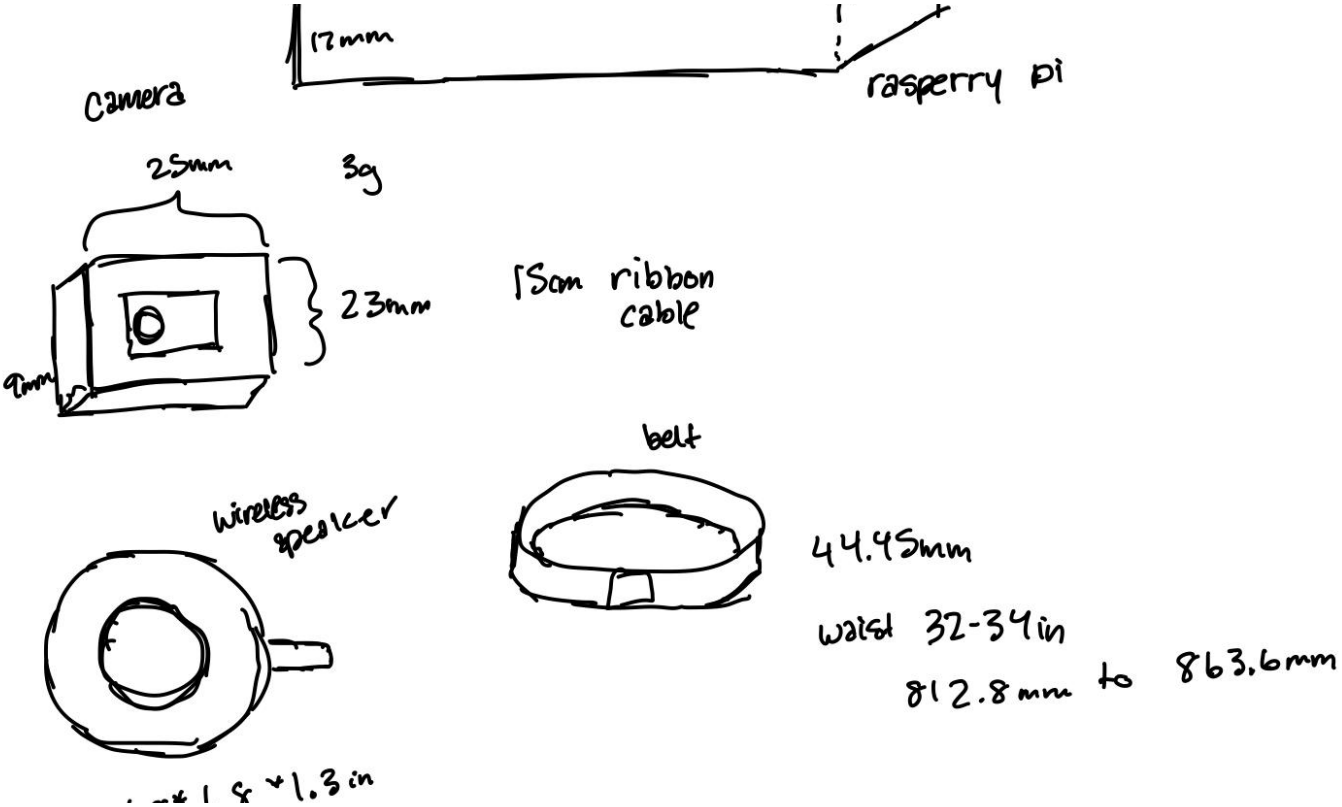

Fig. 2. Diagram of all the components comprising the final system

on the battery pack, speaker, and wire button. Therefore, it is advised for the user to turn on the device before approaching a stoplight. The user will hear a beep when the device is turned on. To use the device, the user first navigates to an intersection that they desire to cross. When the user desires to cross, they press a button on the device, which is housed next to the camera on a breadboard. The user will hear another beep when the button is pressed. When the button is pressed, the camera takes an image of the intersection directly in front of it. The device takes this image and determines, based on the state of the relevant traffic light in the intersection, when to alert the user to cross. While the button is pressed, the system, through an attached mini speaker, emits a voiceover of "green", "yellow", or "red" that indicates the state of the light or a beep if the device cannot locate a stoplight.. This process is depicted in Figure 3 below.

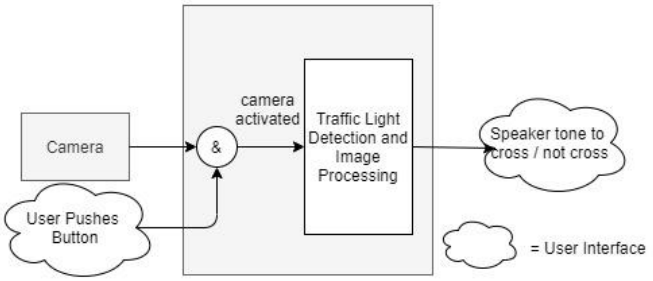

Fig. 3. Functional diagram of system (boxes) and user interaction

IV. Design Trade Studies

To test the given system, the algorithm is to be run on both a validation data set and a test data set of unseen photos. This testing will be done using OpenCV, which is different from the library TensorFlow which was used to build the training mode. There are a number of factors that need to be considered in measuring whether this system holds up to its expectations. These include identifying the targeted traffic light, assessing the state of the traffic light, and measuring the time between the change in traffic lights. For the method that was decided upon, there were numerous other options, but each with their own tradeoffs, which will be discussed further below in detail.

A. Identification of the Correct Traffic Signal

As shown in Fig. 4, that specific picture frame is an instance of when there are multiple traffic signals in the view of an individual. In order for the algorithm to work properly in this case, the pictures that are being tagged in training need to tag all of the stop lights seen, as it is done in Fig. 4. This allows for the algorithm to recognize that there are multiple stop lights in the frame and choose which one to focus on for the individual. The algorithm needs to be able to only recognize the traffic light facing the individual (the one tagged in green) over the ones tagged in pink and yellow. By feeding the algorithm fully tagged images like this, it allows it to discredit the traffic lights that are not in the same look direction desired. This accounts for the most accurate real time situation and allows the system to work and function even when given conflicting information such as in the Fig. 4, which is very likely to happen. The way the algorithm would recognize which traffic light is facing the correct direction is by having the algorithm attempt to identify the light with three visible circles. Usually the other traffic lights will not be facing the correct angle in order for this to occur, as it can also be seen in Fig. 4. Also the distance to the traffic light can be calculated by using the number of pixels between the centers of the green and red circles of the traffic light. This method was preferred to simply trying to identify the color of all the traffic lights in the image, which some other models implemented. Those simply processed all the visible traffic lights in the frame, which is more costly in terms of time and efficiency. The accuracy aimed for is shown in Table 1, as greater than 90% would be considered close enough.

B. Traffic Light State

Identifying the correct state of the traffic light relies on separating and analyzing the colors present. The method used by xWalk includes first extracting the green, yellow, and red parts of the image through RGB filters. Then, the image is searched for a combination of three circles in a row in the correct colors. The FPR is the main concern since that is the only situation in which the pedestrian's safety is threatened. The FPR is calculated through the number of deaths projected per year based on the number of times the device is used. Based on those calculations, as it can be referred to in previous sections as well, the FPR comes out to about .0769%, which can be seen in Table 1. Other methods include converting the entire image to grayscale, applying top hat morphology, applying a watershed algorithm and selecting the bright spots. This type of method is a bit more intensive and is less efficient. It also involves more rigorous testing because there are more steps which could easily not output the correct results along the way.

C. Changing Traffic Light State

The one other factor to be taken into account is the situation in which the traffic light is changing colors. The pedestrian needs to be informed to cross when the light turns yellow to red, not when the light is turning from red to green. In both these situations, there is the chance of this traffic light being identified as red. Thus, the proposed system needs to identify multiple frames when the light is changing so the system can detect the change. This involves taking a video, which is a composition of multiple images, documenting the number of frames it takes for the light to change. This is also the same method in which the latency for this system is to be measured. However, in the case of measuring the latency, the time it takes for the system to respond after the change after a light change is tracked, which is targeted for less than 0.5 seconds as seen in Table 1. Instead of simply telling the pedestrian to cross when there is a red traffic light detected in sight, xWalk's solution is to only inform the pedestrian when the light has been detected to change from yellow to red. While this may take longer for the individual to cross the intersection, the accuracy of the system is improved through this method. The

safety and thus the accuracy of the system is being prioritized while compromising on speed.

TABLE 1. MEASUREMENTS FOR SYSTEM ACCURACY

	Metrics		
	Correct Detection	***Light State***	***Time for Change***
Accuracy	>90%	~0.0769% (false positive rate)	<0.5 sec

Fig. 1. Training Image (*Intersection with tagged traffic lights*)

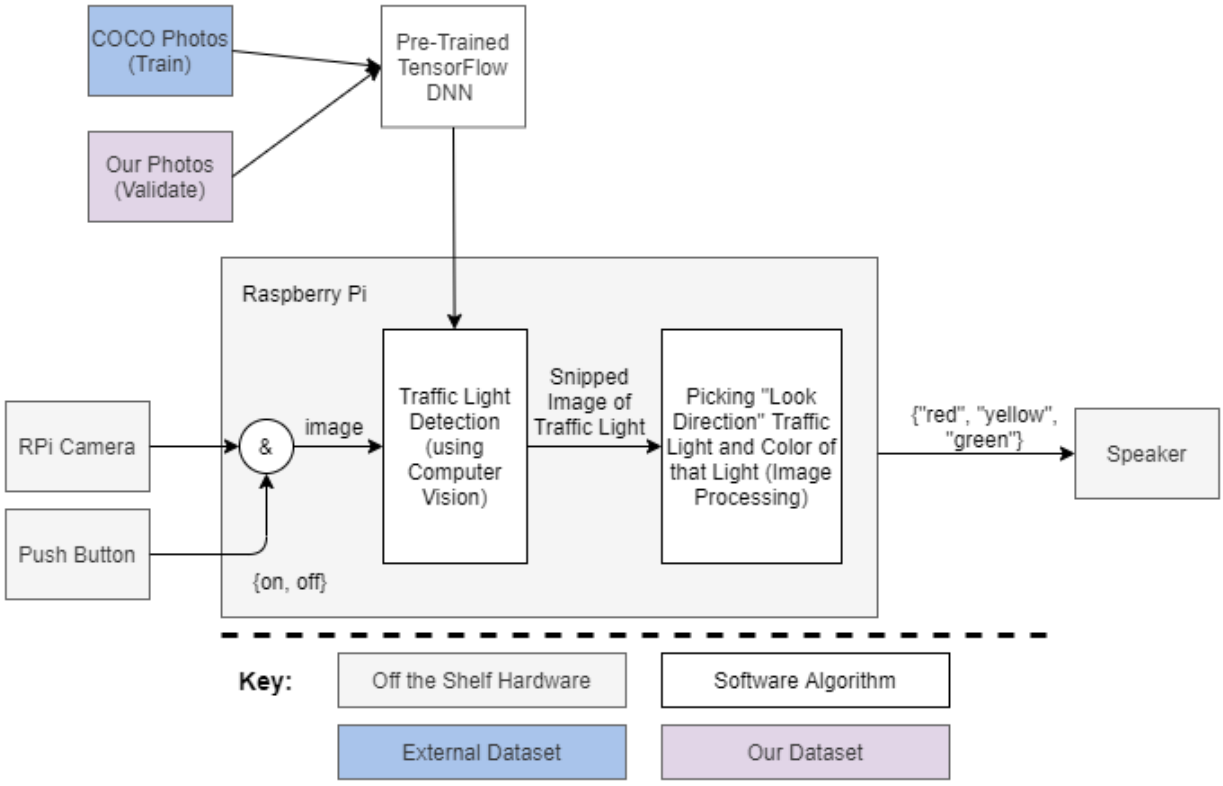

Fig. 2. High-level full system diagram

V. SYSTEM DESCRIPTION

As described in Section III, the overall system, from the user's perspective, consists of a camera and push button for input and a mini speaker for auditory cue output. In this section, more detail will be provided with regards to how the inputs are processed to obtain this auditory output. Figure 5, below, depicts a high level of the entire system's processing.

The system diagram shows how the push button input from the user, when indicating that it has been pushed, allows for the image captured by the Raspberry Pi camera to be passed into the software algorithms (white boxes) to determine (1) detect traffic lights in the scene, (2) determine which of the traffic lights, if any, are the lights the user is facing, and (3) the state of said traffic light which the user is facing. The output of this second software subsystem is the overall system output of {"red", "yellow", "green"} sound signals that are converted into auditory tones driven from the mini speaker. The following subsections describe the software algorithms in more detail.

A. Traffic Light Detection

As described in Section IV.A., the objective of this algorithm is to detect all the traffic lights in a scene, regardless of orientation. This algorithm therefore takes an image, passed in from the Raspberry Pi camera and enabled by the push button, as input and outputs the locations of detected traffic lights, as depicted with the green, hot pink, and yellow boundary shapes in Figure 4. This algorithm includes a trained deep neural network (DNN) model into which the pixels of the input image are passed in and from which the locations of detected traffic lights are outputted. The training procedure, described in the following section, is represented in Figure 5 with the training dataset (blue box) used to train the DNN (white box) within an AWS environment (orange box). The validation procedure is further depicted with the validation

dataset (purple box).

1. Training the DNN

For the model, we are using the TensorFlow library, which includes a DNN, whose specific architecture is optimized specifically for computer vision (CV) applications. To train the model, we need images with tagged traffic lights. The COCO dataset has a couple hundred such images, in which traffic lights, among other classes of objects are tagged just as shown in Figure 4. It is worth noting that COCO has standardized their boundaries to be more specific polygons as opposed to a rectangle, as shown by the hot pink boundary box in Figure 4. At training time, the model takes in the vectorized pixel values of the image, runs a series of calculations as these values are passed through the DNN, and estimates pixel locations for each polygon boundary box for each traffic light in the scene. These estimates are compared with tagged boxes from the COCO dataset by computing an error metric through backpropagation and iterating through the hundreds of tagged images until this error converges.

The computational complexity of training the DNN model over hundreds of images is too large for an individual machine. Therefore, the training is exported to an Amazon Web Services (AWS) graphic processing unit (GPU), which performs these calculations faster. Validation, as described in the next subsection, is also performed in AWS.

2. Validating the DNN and Validation Data Collection

To ensure that our trained model has the highest possible accuracy with relevant data, we validate our model with pictures gathered at various intersections in Pittsburgh with traffic lights then tagged by hand. These images were gathered at different types of intersections (just traffic lights versus traffic lights with crossing signals), varying amounts of cars and pedestrians, varying weather conditions (sunny, cloudy, overcast, rain), varying light conditions (day, dusk, night), varying angles, and varying traffic light colors. These images will be input and their resulting classification error will be calculated as described in the previous section. The objective of this validation procedure is to reduce the classification error in this validation set. This validation error metric will be used to reparameterize and rerun the DNN training to reduce the error as much as possible, making sure to not include the validation photos at training time as to not overtrain the DNN model.

3. Running the DNN

A new image, one that is neither part of the COCO dataset / training set nor part of the validation set, will be input into the trained and validated DNN. The DNN outputs boundaries of all detected traffic lights from the new image. In Figure 5, this process is represented within the gray box labeled “Raspberry Pi”, with the arrow labelled “image” going into the white box depicting the traffic light detection algorithm and the arrow labeled “image of traffic light” (i.e. the traffic light boundaries) leading out of the detection algorithm. The boundaries are used as input to the “look direction” detection algorithm, as discussed in the next section.

B. Look Direction Light (“Look Light”) Detection

As shown in Figure 5, the traffic light detection algorithm passes on the boundaries of the traffic lights, if any, in a photo to the Look Light algorithm. These boundary boxes are then used to “snip” the images of just these detected traffic lights from the overall image. The objective of the look direction light detection algorithm is to identify which of the detected lights from the previous algorithm is the light for the user’s desired crossing direction.

Assuming that the user is directly facing the crosswalk he/she wishes to cross (let’s call this a look angle of 0°), the traffic lights in the user’s crossing direction will be facing the user, albeit likely not necessarily directly in front of the user. From the user’s perspective, the traffic light in his/her crossing direction will appear to be more of a circle than lights facing other directions. Figure 6, below, shows a pair of such lights from a photo in our validation set.

Fig. 1. Look (green) v. non-look (red) lights

As shown in Figure 6, the green light appears to be a circular shape unlike the red light (even if it’s not a perfect circle). This intuition of the look light being more circular than a non-look like is the basis of the image processing behind the Look Light Algorithm.

1. The Circle Hough Transform (CHT)

CHT is a standard feature extraction algorithm used in Computer Vision applications. Its objective is to highlight circles present in an image. CHT is based on the cartesian equation for a circle, reproduced below, where (a,b) is the x,y center of the circle and r is the radius.

$$[3]\ (x-a)^2 + (y-b)^2 = r^2$$

The CHT algorithm, in two stages, searches an image for all possible circles with centers (a,b) (Stage 1) and with radii r (Stage 2). The OpenCV HoughCircles() function efficiently implements this using gradient descent to speed up the search process.

2. Using the Circle Hough Transform (CHT) to pick the Look Light

A convenient feature of HoughCircles() is that it will return a probability that a detected circular object in an image is a circle. In other words, if the algorithm detects any closed shape with curved edges, it will return the coordinates along with the probability that said shape is a circle. This feature allows the Look Light algorithm to assess how close to circles the lights in each traffic light is. That way, the traffic light with lights that most closely resemble a circle, as outputted from the CHT algorithm, based on our intuition described above, must be the look like. This algorithm then passes along the boundary box / snipped image of the predicted look light onto the next algorithm as shown in Figure 5.

C. Light State Detection and Crossing State Decision

As described in the section before and depicted in Figure 5,

the look like boundary box is passed into this algorithm. The objective of the Light State detection algorithm, as the name suggests, is to detect the state of the look light from {“red”, “yellow”, “green”}. This algorithm exploits the fact that the traffic light outputs a colored light. Therefore, the pixels in the red light in an RGB format will have a high value for the R content. Similarly, a green light has a high G content and a yellow light has high R *and* G content (as yellow light is the sum of red and green light.

The OpenCV library has functions to (1) apply a specific RGB filter to an image and (2) assess the saturation/intensity of an image. Our algorithm applies RGB filters and saturation assessment in that order. Therefore, for a red look light, for example, the red filtered image of the red light will have a circle of high intensity where the light is and relatively low intensity elsewhere in the snipped picture of the look light. This intensity will then be assessed (by comparing to an empirically-derived threshold) to derive the conclusion that the look light is red. A red light, through a green filter, will not have such intensity in the light and therefore will not be classified as green. Table 2, below, shows the combination of red-filtered and green-filtered intensities (middle, right columns) and their respective state outputs (left column).

Table 2. Light State Detection Classification

State	Red Filter	Green Filter
Red Light	Y	N
Green Light	N	Y
Yellow Light	Y	Y

The classification output from {“red”, “yellow”, “green”} is stored and updated in memory. We reciprocate the status to the user with a voiceover.

VI. Test and Validation

To assess how closely our device delivers on the metrics described, various subsystems as well as the overall system were tested and results compared to the desired metrics. The respective tests that were performed are reported in the table below.

Table 3. Validation testing

Metrics	Testing
System Latency	Use RPi time functions to report time from image capture to overall system output.
Traffic Light and State Detection Accuracy	Count how many images in which traffic lights are mislabelled. Accuracy (%) = mislabelled frames / total frames
Light State Detection Accuracy and False Positive Rate (FPR)	For each image, calculate proportion of light states labelled correctly (Accuracy) and proportion of red lights label;ed as green lights (False Positive)
Battery Life	Running the device and time the duration until it powers down.

The following subsections report the findings from said testing, as well as how many times the tests were performed and any constraints during said testing.

A. Results for Traffic Light Detection

Our entire device was tested through both systems tests and real time tests. The system tests were conducted using the validation data set of 50 images on our desktop environment instead of on the RPI. By running those through the model, we gained an 89.7% accuracy rate as it can be seen in Table 7. This rate refers to the images in which there was a traffic light and the algorithm outputted the correct color of the traffic light.. During our real time testing, we picked two traffic lights to use our device at, which were both smaller intersections without a pedestrian crossing light. A total of 20 tests were conducted, 10 of which were done on a rainy day, and 10 of which were done on a sunny day. The accuracy of these real time tests turned out to be 80%, as it can be seen in Table 7. The validation tests simply required for the software to run, as the files were after integration. However, for real time tests, the user needed to start up the battery pack and the speaker. Then the user would press the button for the camera to take a picture of the traffic light they were facing. The latency for the speaker to output the result took about 26 seconds, which was significantly longer than the system tests. Both of these times were longer than what was aimed for, which was supposed to be less than 0.5 second, as it can be seen in Table 4.

The battery life was tested in real time by wearing the device throughout the day and occasionally using it. With this, it lasted about six hours, when it was aimed to be about nine hours, as seen in Table 4. However, 6 hours still provided a large enough lifetime for the user to use throughout the day, as the pack could be turned off as well. The battery pack was able to be charged overnight, which also matched the specifications that were set forth in the design.

Another metric that was measured was the false positive rate. This metric was around 3.4% when originally specified to reach less than 1%. It was measured through the number of images in our validation data set that were labeled as red when the light was actually green. The FPR is concerning, as it brings the danger of using this device up incredibly. Telling the user to cross during a green light actually puts the user’s life at risk, and might increase the risk calculations that were done previously.

B. Results for Traffic Light State Detection ("Look Light" Algorithm")

This subsystem was routinely tested using the following tests: Traffic Light State Detection Accuracy, Traffic Light False Positive Rate, and Subsystem Latency. These tests were carried out on the system processing all 43 validation images. The results of each of the tests, described in the following paragraph, are reported in the Tables 4 through 6, below.

Output from Subsystem latency, as with the overall system latency, was calculated using the processor's clock via the python call of time.time(). The average latency of this subsystem across all photos, when run on a desktop, was 0.0623 seconds. When run on a Raspberry Pi, this latency was prolonged to 0.1253 seconds due to slower processing speeds. State accuracy and false positive rates were calculated as described in Table 3, above, across all of the validation set photos. While the overall state accuracy of 89.7% was near our desired accuracy of 90%, the false positive rate of 3.4% was much higher than desired as such a false positive rate could lead to a high incidence of collisions when the device is used. It is also worth noting that the system was able to return None when a non-Look Light was presented 100% of the time, or all 14 of them in the data set, so any error was entirely from any misclassification of the remaining 29 Look Lights.

C. Results for Overall System

Following integration, further testing was conducted to test battery life and overall system latency and accuracy. Battery life was first estimated on the bench by measuring the current (in mA) and dividing from the battery manufacturer's reported capacity (in mAh). This estimate deduced a battery life of 4000 mAh/ 600 mA = 6 hours 40 minutes. We then tested this battery life in real time by running the system from full charge to automatic shut down, which lasted closer to 6 hours. Overall system accuracy in traffic light and state detection as well as full latency was obtained using the same methods mentioned for other subsystems above. The results are reported in Table 4 below.

Table 4. Overall Results

Metric	Requirement	Result
System Latency	< 0.5 sec	26 seconds
Traffic Light and State Detection Accuracy	> 90 %	89.7%
Light State False Positive Rate	< 1 %	3.4%
Battery Life	9 hours	6 hours

Table 5. Light State Detection: Look vs. Non-Look

	Amount	Accuracy
Total Lights	50	N/A
Look Lights	29	100 %
Non-Look Lights	21	100 %

Table 6. Light State Detection: Color Detection Rates

Rate	Value
Correctly Detected	0.8966
False Positive	0.0345
False Negative	0.0690

Table 7. Overall Testing Results over N Trials

Type of Testing	Number (N)	Accuracy
System	50	89.7%
Real Time	20	80%

VII. Project Management

VIII. Schedule

Our schedule is divided into five different sections which illustrate the core tasks of the project which are further divided into subtasks. These main sections, which can also be seen in Fig. 8, include Project Proposal and Planning, Design and Implementation, Basic Integration, Full Implementation, and Performance Testing and Integration. The first phase involves gathering the data and setting up the environments primarily for both the training and hardware aspects. The Basic Integration Stage involves having the image classifying algorithm functioning and connecting all the external parts to the Raspberry Pi. The Full Integration stage is when the auditory feedback is to be implemented and the detection algorithm's accuracy rate needs to match up with the design requirements. The Performance Testing and Integration Stage primarily focuses on having the hardware components work in all sorts of conditions. Phases 3 and 4 involve a lot more tasks being done by everyone together, as these are heavily focused on integration between the individual platforms. The Final Report Phase is simply putting the project together and by then it requires a fully functional device.

IX. Team Member Responsibilities

We split up our responsibilities between the hardware and

software components. Jeanette's main responsibility will be the hardware components with the raspberry pi, speaker, camera, and so on. She will be assembling the device and assisting and setting up real time usage. Yasaswini and Shayan will be in charge of the software. Yasaswini's main responsibility will be finding, annotating it, and training the data set. Jeanette will help assist her with any difficulties. Shayan's responsibility is to create the image processing algorithms and test those. All bench testing and integration testing will be done as a group.

X. Budget

The budget is located on page 8 in figure 7.

XI. Risk Management

Throughout the semester so far, we took into account the feedback and suggestions given by peers and professors for the project proposal, design presentations and weekly meetings. At first, we were unsure of the design, a week in our schedule was given to design exactly how our product would be. From this, we ended up changing our design completely into a belt which changed some of the hardware components and design that we originally planned. We created an extensive design drawing to ensure that the components are the correct size and weight to meet our requirements. This will ensure that the necessary components are ordered ahead of time. In addition, we made sure to investigate our software programs during the design week to ensure we are familiar with the technologies that we are using.

An important aspect of the schedule that we added is slack time. We added a week of slack time in case we get behind schedule or end up with some difficulties throughout the semester. In addition, most of our schedules are free towards the end of the week, so we created working sessions every Thursday to ensure that everyone has time to work on their part. Our schedule uses week long tasks to ensure that we are able to complete our tasks by the status report deadline. We ended up using the slack time when the image processing and the training model took more time than expected.

Early planning allowed us time to acquire any components or software that we did not account for. For example, Jeanette was in charge of ordering the components for the hardware. The components were ordered during design week so that after our design presentation, we are able to start on the hardware. Unfortunately, we were missing small cables that were necessary to set up the raspberry pi, so luckily, it was still early in the semester to acquire these components and continue working on schedule. In addition, we added a last minute addition of AWS in order to train on tensor flow. Immediately after the presentations, we acquired and downloaded any extra software that we did not take into account in our early planning.

XII. Ethical Issues

For a device that is used by vulnerable populations, it is important to think about ethical issues in our project planning. Visually impaired individuals are the most affected by the product. The greatest risk of our device for our users is injury or death from error in our calculation. Therefore, if this device were to be produced in the real world, we would need to make sure that the false positive rate is extremely low (to almost perfect) and the accuracy is extremely high. Testing and validations must be very detailed. Our device needs testing at every different type of stoplight in the United States. As of now, we created a training model that prioritized simple stoplights but a larger dataset will help mitigate this risk.

Another population affected could be pedestrians and background objects in our images. Since our device includes a camera, privacy issues can be an edge case for users to alter. Users might purposely use our product to record and take pictures of pedestrians and impeding objects. To help mitigate this, we may involve security measures so the user is unable to access the images and videos. In addition, something we included in our project is erasing all pictures and videos once the RPi resets. Therefore, when the RPi reboots, there will be no history of images and videos taken beforehand. We changed the program to take a picture in order to mitigate the risk of surveillance of pedestrians. A picture is less invasive than a video.

Another safety risk could be the weather wearability of our product. Since many cities in the United States face many different weather conditions, our device should withstand any event that it would come in contact with. This is very important for snow and rain if water were to come in contact with our board. A mitigation to this risk would be a waterproof camera and covering over the board to protect any vulnerable parts. The accuracy in any weather condition should be very high since we expect our users to be able to use the device at any time during the year. If it is raining or snowing heavily the camera may not be able to recognize stoplights.

Comfort in wearability of our product is very important. Our device should be inclusive to every different body type. As of now, we have one size that fits 32-34 in of the waist. If our product were to be expanded to the real world population, many sizes must be included such as plus size and petite. As of now, our product is very bulky and uncomfortable to wear for a visually impaired individual. Batteries and the speaker will only notify the users that it has been turned on with a light. This could be difficult for visually impaired users to recognize. Perhaps, exchanging the components for noise instead of visual cues when the battery and speaker turn on would be very helpful.

Our test group should be actually visually impaired individuals. Our group actually spoke with a visually impaired individual when designing our product. In the beginning, we changed our initial use case because we were not aware of how a visually impaired user would actually work with our device. We learned that visually impaired individuals would use this device to help assist them in training to cross streets and is not a permanent product in their routines. Therefore, an accurate representation of the users can help mitigate any issues that may arise.

XIII. Related Work

Traffic light detection and state recognition is a common problem for autonomous vehicles. Therefore, there is a robust body of prior work to develop such algorithms for the specific use case and resources at the disposal of autonomous vehicles. Such algorithms include convolutional neural network (CNN) based state recognition [1] and support vector machine (SVM) approach based on converting the traffic image from RGB to hue-saturation-value (HSV) space [2]. These algorithms are built upon editing robust frameworks for recognizing a traffic light, typically using deep neural networks (DNN) with the image input, trained with thousands of tagged traffic lights in photo databases. The scope of these software developments are biased in the advanced processing capabilities as well as stricter requirements of autonomous vehicles. Therefore, they are optimized for the lowest latency possible without much priority to processing power used.

In addition, there are many products that are similar in detecting traffic lights either for people or cars. For example, a traffic light detection system uses a camera attached to the head to videotape the user's view at a crosswalk. Using OpenCV for image processing, the device was able to detect the shape and color of traffic lights [3]. A traffic light pilot app for visually impaired people was developed by a working group at the Research Institute for Ophthalmology in Tubinge in 2017. They collected and classified about 3000 images from which about 1000 were selected to train using machine learning. The final "LightsCatcher" app allowed for the user to take a picture on their phone of their surroundings and have the app detect the red and green phases of a pedestrian light by translating them to acoustic or tactile signals [4]. The project ended up being able to detect a large part of the pedestrian traffic lights. They expected that the detection rate could improve even further with more images. However, the applicability of this project is limited currently by the distance to the traffic light and the light conditions.

XIV. Summary

Our system did not meet all the specifications but did achieve some measure. We were able to create a MVP of our device with a 90% accuracy in detecting the traffic light and its correct state. However, the FPR is 3% which is high for the requirement of our system. The latency is 26 seconds instead of <3 seconds.

A. Future work

I believe more experimentation with the processor might help improve the latency. RPi 4 Model B does not work well with Tensorflow and should not expect robustness when working with large training models. Next time, I would compare the Jetson Nano or an earlier version of RPi to see if the latency would improve with these processors. In addition, fine tuning parameters of the Look Light algorithm would decrease our FPR. Another attribute that would have been achievable with our device given more time is parallel processing. It would be very helpful to have the user press the button multiple times in order to detect a transitioning state at the crosswalk.

B. Lessons Learned

We should have done more research in the processor that we needed. By the time we set up the program and integrated our code, the latency was impacted by our design plan and should have been further research before we started the project. We, also, spent most of the time setting up the model on the AWS while in the end we did not use it. In addition, we spent most of our time integrating all our parts together. Perhaps, communicating what we need between each part would have made the process faster than before.

Glossary of Acronyms

CNN - Convolutional Neural Network
CV - Computer Vision
DNN - Deep Neural Network
FPR - False Positive Rate
HSV - Hue-Saturation-Value
RGB - Red-Green-Blue
RPi - Raspberry Pi

References

[1] S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj and N. Ganeshan, "An efficient vision-based traffic light detection and state recognition for autonomous vehicles," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 2017, pp. 606-611, doi: 10.1109/IVS.2017.7995785.

[2] Guo Mu, Zhang Xinyu, Li Deyi, Zhang Tianlei, An Lifeng, Traffic light detection and recognition for autonomous vehicles, The Journal of China Universities of Posts and Telecommunications, Volume 22, Issue 1, 2015, Pages 50-56, ISSN 1005-8885, https://doi.org/10.1016/S1005-8885(15)60624-0. (https://www.sciencedirect.com/science/article/pii/S1005888515606240)

[3] Tamgadge, Suraj, Rohit Bhise, et al.. "Traffic Light Detection System for Visually Impaired Person with Voice System." *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 8, no. 3, Mar. 2019, pp. 744–760., doi:10.15662.

[4] "The Gateway to European Vision Research." *Vision Research*, ROCHE, 2017, www.vision-research.eu/index.php?id=1122#:~:text=According%20to%20federal%20laws%2C%20most,recognize%20red%20and%20green%20phases.

Component	Cost+Shipping		Software
Raspberry Pi Model B 4	$88.50		Tensor Flow
Power USB Switch Type-C Cable	$5.89		OpenCV
Battery Pack	$23.95		
Adafruit Mini Speaker	$8.57		
Amplifier	Inventory		
Belt	$39.99		
Raspberry Pi Camera V2	$24.99		
Micros SD Card with Adapter	$29.99		
Ethernet Adapter	$12.99		
AWS Credit	$50		
Total	$284.87		

Fig. 7. Budget and Parts list

	TASK TITLE	TASK OWNER	3/1 - 3/8	3/8-3/12	3/15-3/19	3/22-3/26	3/29-4/2	4/5-4/9	4/12-4/16	4/19-4/23	4/26-4/30	5/3-5/7
	Design and Implementation											
2	Proof of concept and hardware prep											
	Order critical components	Everyone										
	Take pictures at designated stoplights	Shayan										
	Organize data set to use in code	Yasaswini										
	Research opencv code examples	Yasaswini										
	Refresh how to code on raspberry pi	Jeanette										
	Design headband to hold raspberry pi and camera and speaker	Jeanette										
3	**Basic Integration**											
	Connect speaker to raspberrypi	Jeanette										
	Connect audio for reboot											
	Connect camera to raspberrypi	Jeanette										
	Set up AWS Network	Yas										
	Code look direction algorithm	Yas										
	Test look direction algorithm	Yas										
	Design Report	Everyone										
	Connect powerbank to raspberrypi	Jeanette										
	Code for the state of the stoplight	Shayan										
	Test state of the stoplight algorithm	Shayan										
	Bench test	Everyone										
	Sew camera and raspberry pi into headband	Jeanette										
4	**Full Implementation**											
	Connect raspberry pi and Ricky's algorithm	Jeanette										
	Clean up User interface with the controls	Jeanette										
	Code auditory feedback	Jeanette										
	Recognize stoplights	Yasaswini										
	Give correct auditory feedback for stoplights	Shayan										
	Integration test at stoplight	Everyone										
	Slack	Everyone										
5	**Performance Testing and Integration**											
	Test battery life	Yasaswini										
	Test durability and comfort (running, weather, etc)	Shayan										
	Tweak parameters (speaker volume)	Jeanette										
6	**Final Report**											
	Record Video	Everyone										
	Final Presentation	Everyone										
	Edit Video	Everyone										
	Project Due (May 10)	Everyone										

Fig. 8. Team Schedule