B1: FocusEd

Authors: Heidi Batres, Vaheeshta Mehrshahi, Danielle Kakish: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—FocusEd is a daytime driving aid that will alert a driver if it is determined that the driver is engaging in distracted or drowsy behavior. Based on eye classification and head pose estimation through a video stream, FocusEd uses several software algorithms to detect a drowsy or distracted driver. The main hardware components of FocusEd are a Jetson Xavier NX and a Raspberry Pi Camera Module V2, in which the Jetson is responsible for the computing power of the system and the Raspberry Pi Camera intakes the video of the driver. FocusEd then outputs an audio alert in order to alert the driver that they are showing a distracted behavior, allowing the driver to refocus on the road before another alert is given. The project focuses on re-educating the driver on the skills that would prevent distracted driving.**

***Index Terms*—distracted driving, Eye Aspect Ratio (EAR), eye classification, facial detection, facial landmarking, focus timer, head pose estimation, Histogram of Oriented Gradients (HOG), Support Vector Machine (SVM)**

1 INTRODUCTION

FocusEd serves as a way for drivers to curb their distracted day driving while simultaneously improving road safety and their own driver education. With the increase in distracted driving-related deaths in recent years, it is imperative that a solution be found to help drivers fix these habits. Much of this increase has to do with the increased reliance on smartphones as technology evolves and people feel the need to send a text or email immediately rather than waiting. According to the NHTSA, 3,142 people died in vehicular accident due to distracted driving [17].

Current technologies are mostly focused on the idea of lane detection—– alerting the driver if they get too close to another car. However, this solution focuses more on the car than on the driver themself. If a driver is drifting into another lane without meaning to, they can already be posing a danger to other cars around them. Thus, we want to correct that behavior before the driver even begins to drift and potentially cause an accident. In order to do this, FocusEd must detect that the driver is not focusing on the road and output an audio alert within a 3 second interval in order for the driver to swiftly correct their behavior. Users require a driver safety system to not hinder their driving by not obstructing their view of the road with cumbersome hardware and not further distracting them with excessive alerts. Thus, our system not only determines driver behavior but length of behaviors to better classify movements as distracted versus normal.

2 DESIGN REQUIREMENTS

2.1 Face Detection

The first design requirement for this project is face detection. Specifically, we narrowed our scope to strictly facial detection in daylight conditions where the individual is not wearing any sunglasses or other eye or face obstructions, except for clear eyeglasses. This requirement is vital to the success of our project as the remainder of the software algorithms depend on the detected image of the driver's face. Thus, we would like to have at least 90% detection accuracy. We also chose this threshold because the face detection method we chose, Histogram of Oriented Gradients and Support Vector Machine (HOG and SVM), was proven to be 96% accurate in a trial with 40 distinct subjects[13] and 94.43% accurate in a trial on 26,416 images[22]. Since we are detecting with a real time video stream where the user is moving their face in any possible direction, we are lowering our accuracy rate. To test the HOG and SVM facial detection algorithm, we use the datasets Labeled Faces in the Wild (LFW)[12], FACES[6], Gourier and Crowley[10], and Closed Eyes in the Wild (CEW)[8], in addition to testing on sampled drivers inside a vehicle.

2.2 Head Pose Estimation

We require accurate head pose estimation to determine if the driver is performing a regular driving movement or a distracted driving movement. We require 85% head pose estimation accuracy. Since we are creating this algorithm on our own, its accuracy relies heavily on that of the facial detection. Thus, we needed to make the accuracy slightly smaller than that of the detection in order to account for this. This benchmark is also derived from two research papers — one that arrived at above 80% accuracy for driving head poses [33] and another that reached an average of 93% for specified pose accuracy[2]. By averaging the two and accounting for real-time estimation, we determined 85% as the estimation accuracy appropriate for our use case. In order to test this, we compare the actual direction of the driver's head to the head pose estimation output. We use the Gourier and Crowley head pose image database for this.

2.3 Eye Classifier

Additionally, we require accurate eye classification to determine whether the driver is falling asleep at the wheel or not. We require our classifier to achieve a 90% accuracy. We chose 90% because the eye landmark calculations we hope to use, specifically the Eye Aspect Ratio (EAR), has

been proven to be 90% accurate [28]. We use the Specs on Faces [1], CEW, and LFW datasets to train a baseline eye classifier as well as test it.

2.4 Focus Timers

Since there are normal movements that would make it hard to define distracted vs normal driving movements, we require a focus timer for both head pose and eye classification. For example, if the driver's eyes are closed for an extended period of time, they are not blinking and possibly drowsy. Similarly, if a driver is looking down too long, they are likely looking at their phone. The focus timer for eye classification is 1 second as this can separate normal blinking from possible drowsiness. According to the Rules of Drowsiness for driving, a person is considered drowsy if their eyes are closed for longer than 1 second[15]. The focus timer for distracted driving is 2 seconds since the National Highway Traffic and Safety Administration states that drivers should never take their eyes off the road for more than two seconds at a time[31]. These focus timers determine the difference between normal versus distracted behaviors at least 90% of the time. We tested these time cutoffs on driving individuals. These in combination with the eye classifier and head pose estimator shall take into consideration both positioning and timing.

2.5 Power Supply

Since the system is located in the vehicle during the day, the power supply for the device needs to last for the driver's average commute. FocusEd uses a portable power supply, and thus we test that it is powered between 8 to 10 hours to account for a driver's commute to and from work on an average work week.

2.6 Audio Alert

To notify the driver that they are distracted, we require an audio alert triggered 99% of the time when a distracted movement is detected. We aim for a 99% accuracy since misreads could occur. We also require that the driver responds to the alert by refocusing their head position and eyes to the road, and thus we test that our system no longer produces an audio alert once the driver refocuses within 3 iterations of the system.

2.7 System Latency

Finally, because we want to correct the driver in real time to prevent accidents, we need an efficient and fast system to alert the driver. We require our system's full iteration to complete within 3 seconds, and we test this by timing full iterations of our working FocusEd system installed in a parked vehicle. The iteration begins when the driver performs a distracted behavior and ends when the system provides the audio alert to refocus on the road.

3 ARCHITECTURE OVERVIEW

To meet the system requirements discussed in the previous section, FocusEd is an enclosed camera system, as depicted in Fig. 1, to be placed on the dashboard of the car. The Raspberry Pi V2 Camera Module is secured directly in front of the driver, specifically attached behind the wheel either directly on the dashboard or on the windshield. Placing the camera directly in front of the driver ensures the highest accuracy in our software algorithms. The camera is plugged into the Jetson Xavier NX via a 50 cm ribbon cable. The Jetson Xavier NX is placed inside a case secured to the dashboard to ensure that the Jetson is not damaged while driving. The positioning of this case is near the passenger side of the dashboard, so as to not obstruct the view the driver has on the road.

In addition, the USB Speaker that outputs an audio alert is attached on top of the Jetson's case. To power the Jetson, the TalentCell 12V Battery is plugged into the power jack of the Jetson and sits in the driver's cup holder. We did not secure the battery pack to the dashboard as to allow the user to safely unplug the battery pack for charging.

Our user story is depicted in Fig. A4. First, the user presses the power button on FocusEd to start up the system. This begins our 1-minute welcome and calibration sequence. In this sequence, the user is welcomed with an audio message explaining FocusEd, which includes a warning to not perform the following calibration steps while they are driving. Then, they are asked to look forward at the road for 10 seconds, and then close their eyes for 10 seconds. This is done to calibrate our eye classifier for drowsiness, as each individual's eye sizes are slightly different. Specifically, we train an SVM model, and also have a baseline in case the calibration step is performed incorrectly. Then, the system informs the driver that they may begin driving, at which point it continuously processes the following video stream. For each inputted frame, the Jetson performs the HOG and SVM facial detection algorithm in order to detect the driver's face. Following this, the algorithm for facial landmarking runs to landmark the driver's face for further assessment. The system then simultaneously runs both the eye classification and head pose estimation algorithms to determine whether the driver of the vehicle is exhibiting behavior of sleeping or distracted driving, respectively.

During this, the focus timers is running to actually make this determination. We have two separate focus timers, one for drowsiness and one for head pose. The drowsiness focus timer runs in conjunction with our eye classification algorithm, and if the eye classification determines the driver's eyes are closed for more than 1 second, then a signal is communicated that the driver must be alerted. The head pose focus timer works in conjunction with our head pose algorithm. If it is determined that the driver's head pose is not facing forwards towards the road for more than 2 seconds, then a signal is communicated that the driver must be alerted.

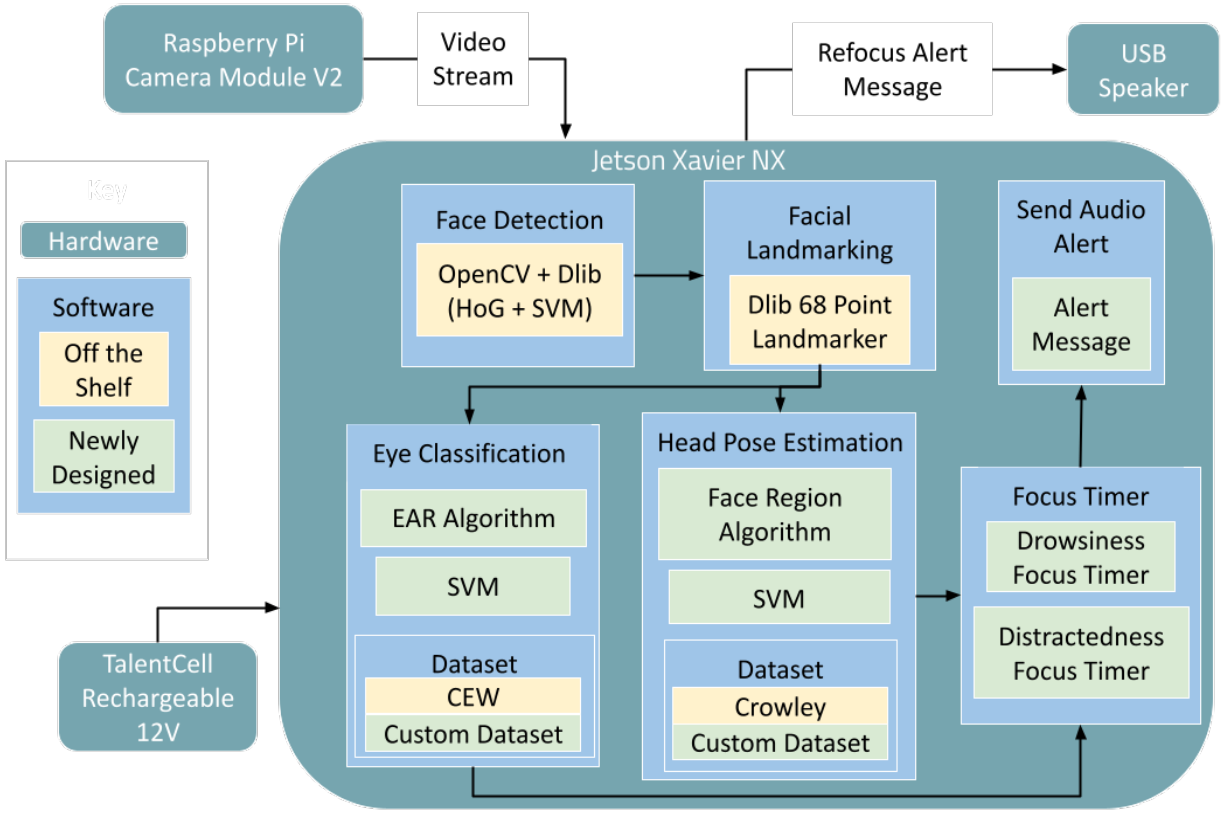

Figure 1: Overall System Block Diagram

To sound an alert, the USB speaker that is connected to FocusEd's Jetson outputs a 1 second vocal audio alert stating "Please refocus on the road." If the driver does not correct their behavior within the next 3 iterations of the system, the driver will once again be alerted with the audio message. If the driver is not determined to have been distracted, the algorithms will continually run with the video stream of the driver's face so any distracted driving will be alerted in real-time.

Since submitting the design report, we made several changes to our system. For one, eye classification now involves calibration with audio prompts in order to train an SVM. Additionally, our head pose estimation calculations have shifted and head pose estimation is also using an SVM. Moreover, since we did not want to obstruct the driver's view, we chose to have our camera system separate from the rest of FocusEd, and connected via a 50 cm ribbon cable.

4 DESIGN TRADE STUDIES

4.1 Face Detection and Landmarking Algorithm

For this specification of our system, we opted to use HOG and SVM over another facial detection algorithm such as Haar Cascades. Our reasoning for this choice is that HOG works well for frontal and slightly non-frontal and works under small occlusion. Also, since we are looking at just the driver's face in the image, we don't need to worry about too small of a face for the detector. The HOG and SVM detector is from Dlib's Python libraries[11]. Then, for facial landmarking, we use Dlib's default 68 points landmarks from their shape predictor[14][5]. The shape predictor includes landmarks of the face and around the eyes which was beneficial for us to use in both our eye and head pose models, instead of building a shape predictor from scratch.

4.2 Head Pose Estimation

Initially, we thought following examples that convert the 2D facial landmark points to 3D points would yield the best results for head pose estimation. However, since we are running multiple algorithms and our landmarks might not be fully precise, we tried to use an imaginary axis over the driver's face, with the origin being the tip of the nose. This was effective in determining when the driver was not facing forward, but unfortunately this method was too sensitive, as drivers need to have some flexibility from looking directly forward. Thus, following a similar approach to the EAR algorithm used for eye classification, a ratio based on triangular areas of the face was developed. The first iteration was to use triangles over the left and right cheeks

to determine whether the driver was looking right, left or forward. Then, to include the down direction, a third triangular area over the chin was added to create a ratio. This ratio subtracted the cheek areas and divided by the lower triangle. This approach was not as effective because there was difficulties determining forward and down or right and down. To solve this, we realized that when the driver is looking down, their eyes appear closed, and thus this behavior is caught by the eye classifier. Therefore, we removed the lower triangle. In beginning stages of testing, it was seen that the ratio of the difference between the two cheek areas was relative to how close the driver's face was to the Raspberry Pi camera module. Thus, the ratio was adjusted to be left cheek area divided by right cheek area. This approach is the final approach we use in our final MVP.

Additionally, we had to decided whether to calibrate the model in real time or use a pre-trained model for head pose. Asking the driver to hold their face in certain positions could likely come with errors, and the harder to distinguish the poses, meant the model would take longer to train. Thus we opt to use a pre-trained SVM model for head pose as there is fewer differences in people face positions as there are in their eye shapes. This reduces the calibration time at the beginning of our system.

4.3 Eye Classifier

For classifying if the driver's eyes are closed or not, we are using the Eye Aspect Ratio (EAR) method [28] and training a linear kernel SVM. Other methods include matching an open eye template to the user's eye to determine if the eyes are closed or not[18]. Other eye classifiers rely on thresholding to determine the whites of the eyes and if these white regions disappear or not. However, we wanted an efficient eye classifier that relied on calculations using 2D landmarks that are already outputted by our facial landmark. Thus, we chose to use EAR because of the reduced computation time required.

4.4 Focus Timer

Initially, when we were thinking of creating a focus timer, we struggled with determining the proper timing to wait before alerting the driver because we would have to distinguish between distracted driving and normal driving. During our initial design, we determined that we would have a singular focus timer for a set period of time that was yet to be decided upon. However, following our research we determined that having two focus timers——one for texting and another for the sleeping portions of our scope——would suit the project best because the timing is different. As for alerting distracted driving, we made the determination that 2 seconds would be a sufficient time before alerting a driver because it is generally known that a driver must keep approximately 3-4 seconds behind the car in front of them. So within that time of 2 seconds, alerting the driver would still keep them a couple seconds behind the car in front. Thus, we have two focus timers in our design, one for drowsiness alerts and another for distracted driving alerts.

4.5 Dlib

Dlib provides various open source machine learning and image processing algorithms. Dlib also supports GPU acceleration which pairs well with the Jetson Xavier NX. This includes the GPU accelerated HOG and SVM face detection library and the GPU accelerated 68 point shape predictor for facial landmarking [3]. Additionally, Dlib has good documentation to reference. We opted to use Dlib over piecing together detectors from different libraries to ensure compatibility throughout our system.

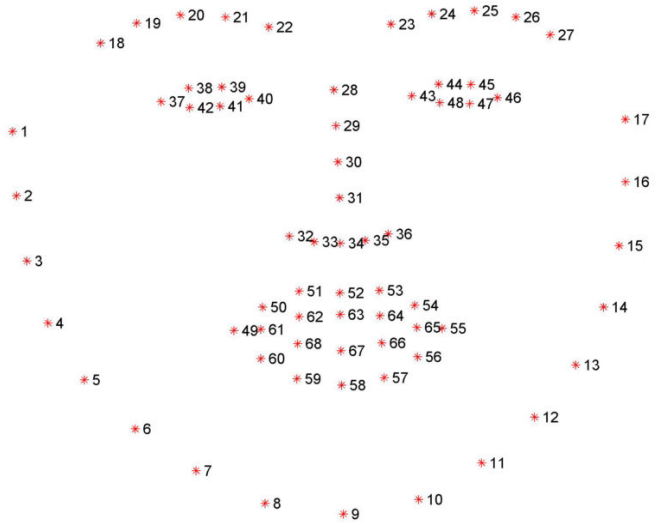

Figure 2: Diagram of 68 facial landmarks[25]

4.6 OpenCV

OpenCV provides computer vision and machine learning libraries for Python and is used in projects in combination with Dlib for image processing of video and images. The other library we were considering was TensorFlow. However from our initial research, OpenCV had more documentation and examples that matched our use case, which facilitated debugging[29][30].

4.7 Jetson Xavier NX

During our initial design, we believed that NVIDIA Jetson Nano would be sufficient for the computing power that we needed for our project and that the size would be smaller and better suited for placement directly in front of the driver. Following our proposal presentation, our instructors expressed concern that the Jetson Nano would simply not be powerful enough for the computations that we intended to perform. Following this, we researched further into the NVIDIA Jetson Xavier NX and determined that its size was not much different than the Jetson Nano and was

much more powerful, thus increasing the frames per second which would give us a much closer "real-time" alert. Specifically, the Jetson Nano averages about 15 fps while the Jetson Xavier NX averages about 30 fps [32].

4.8 Raspberry Pi Camera Module V2

The Raspberry Pi Camera Module is compatible with various NVIDIA products including the Jetson Xavier NX and its default packages (JetPack SDK) and is recommended in NVIDIA forums. Additionally, there are various examples online for reference to consult. Most importantly, it records live camera feeds[16].

5 SYSTEM DESCRIPTION

This project consists of both hardware and software. The project's hardware system consists of a Jetson Xavier NX, Raspberry Pi Camera Module V2, TalentCell Rechargeable 12V Battery Pack, and Mini External USB Stereo Speaker. Fig. A5 presents the hardware block diagram. The software system, which runs on the Jetson Xavier NX, consists of the processing of the Raspberry Pi Camera Module V2's video stream in order to determine of the driver behaviour and signaling subsequent alerts, as well as ensuring the system runs upon boot. This includes facial detection, facial landmarking, eye classification, head pose estimation, focus timing, audio alert, and the init system. Fig. A6 presents the software block diagram.

5.1 Jetson Xavier NX

The Jetson Xavier NX serves as the processor for the video stream it receives from the Raspberry Pi Camera Module V2, which is plugged into port J1 on the Jetson Xavier NX. Upon receiving power, the Jetson begins outputting the FocusEd welcome message and audio prompts for calibration. It simultaneously processes the video stream input and runs the facial detection and landmarking for training the eye classifier. Following calibration, the Jetson continues processing the video stream and performs facial detection and landmarking, eye classification, and head pose estimation algorithms on each frame. These algorithms are described in more detail below. Additionally, the focus timer and any subsequent audio alerts run as well.

5.2 Raspberry Pi Camera Module V2

Our camera of choice is the Raspberry Pi Camera Module V2, which has a 50 cm ribbon connector that is connected to the Jetson Xavier NX via port J1, which enables use of CSI cameras. This camera has a Sony IMX219 8-megapixel sensor and GStreamer is used to interface with the camera using the Jetson [23]. The camera is enclosed with a acrylic cookie wheel case that allows for flexibility in adjusting the angle of the camera. To secure the camera to the dashboard or windshield while also allowing for easy removal, 3M heavy duty command strips are utilized.

5.3 TalentCell Rechargeable 12V Battery Pack

To power the Jetson Xavier NX, we use the TalentCell Rechargeable Battery Pack. The battery pack delivers 12V, which satisfies the 9-20V power requirement of the Jetson Xavier NX. The battery pack connects to the Jetson's DC power jack with a 2.5 mm jack pin DC connector[19].

5.4 Mini External USB Stereo Speaker

For our audio alert, we use Adafruit's Mini External USB Stereo Speaker. This is a USB-only speaker, and thus is both powered and receives audio via one of the Jetson Xavier NX's USB ports. Because the speaker has a 46 inch cable, we have flexibility in securing the speaker either onto the Jetson case or anywhere else near the driver.

5.5 Facial Detection and Landmarking

Facial detection of the driver is performed using the Histogram of Oriented Gradients and Support Vector Machine (HOG and SVM) algorithm. HOG is a feature representation method. A HOG descriptor is first extracted from our image frame from our video stream. Then, a SVM, which is a supervised machine learning model, is used to train a model to detect a face. Dlib is the library that is used for this. Facial landmarks are sequentially obtained using Dlib's facial landmark detector. This estimates the location of 68 2D coordinates[14][5].

5.6 Eye Classification

Using landmarks retrieved from facial landmarking, we then classify whether an eye is opened or closed. This is done by finding the proportion between the width and height of the eye based on 6 eye landmarks. This proportion is called the Eye Aspect Ratio, or EAR, which is found using the following formula[28]:

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\,\|p_1 - p_4\|} \tag{1}$$

where $p_1, ..., p_6$ are 2D landmark locations provided by Dlib's 68-point facial landmark detector as shown in Fig. 3. The smaller the ratio, the closer the driver's eyelids are together, signalling that their eyes are closing. The threshold for the EAR is determined through calibration on the user and training of an SVM model. The SVM module used is the linear kernel SVM from the Scikit-learn library[21]. During calibration, an audio prompt is played asking the driver to look ahead at the road for 10 s. The prompt counts to 10 while the user keeps their eyes steadily open. This obtains 3-dimension feature vectors associated being awake. Then, an audio prompt is played asking the user

to keep their eyes closed for 10 s. The prompt counts to 10 while the user keeps their eyes steadily closed. This obtains 3-dimension feature vectors associated with being asleep. Then, the linear kernel SVM is trained with the recorded feature vectors along with data obtained from the CEW dataset. Thus, if an individual does not correctly follow the audio prompts and is not in the frame to participate in the calibration process, then we still have a baseline SVM to make predictions with.

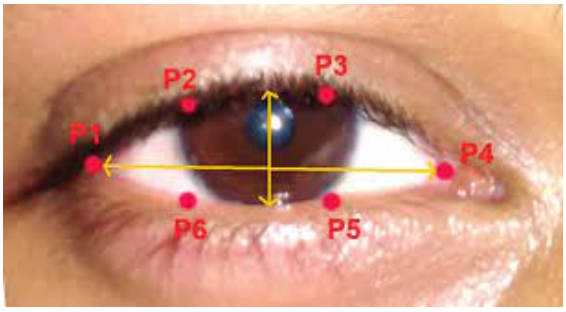

Figure 3: Depiction of 2D landmark points used for EAR[4]

5.7 Head Pose Estimation

From the landmarks retrieved from the facial landmarking phase, we calculate triangular facial regions of the cheeks and chin. From the Dlib landmark vector, index 30 refers to the tip of the nose, indices 16 and 11 refer to bottom right and left jaw points respectively, and indices 0 and 5 refer to the points connecting the face to ear on the right and left sides (see Fig. 2)[25]. With these points, triangular regions are drawn. For head pose estimation, a ratio is calculated by dividing the area of the left cheek by the area of the right cheek. Similar to the eye classification method, a SVM model was pre-trained with pictures of faces looking forward, left and right. Using the calculated ratio and Scikit-learn Python library, the SVM model is built using an 80/20 train and test split. The model is trained with the Crowley database as well as photos taken of ourselves being distracted and not distracted in Danielle's car. If the driver is looking left or right, we could determine that the driver is possibly distracted.

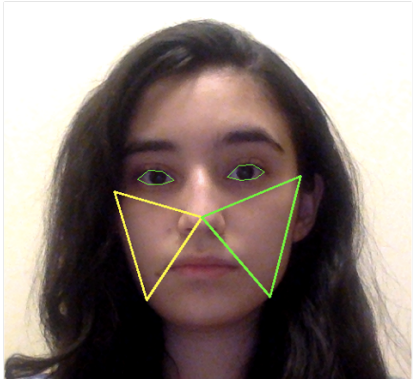

Figure 4: Example of non-distracted driving pose

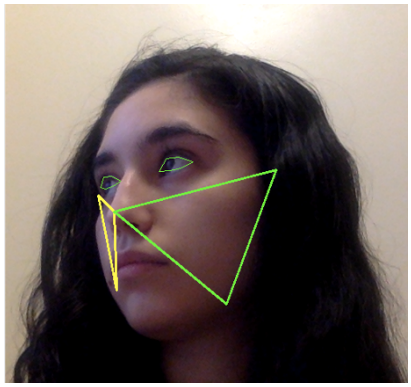

Figure 5: Example of distracted pose (right)

5.8 Focus Timer

The focus timers for head pose and eye classification help determine whether a driver is distracted or not. There are normal driver movements that require the driver to not face forward, such as checking their side and rear view mirrors. The timers account for these brief glances, as long as they are less than 2 seconds, and thus limits the alerts as to not inconvenience the driver. Additionally, the timers check if the driver has corrected their behaviour. If the driver has not corrected their behavior, the driver will once again be alerted with an audio alert.

5.9 Audio Alert

To ensure that the driver is woken from their drowsiness, our audio alert is a clear 1-second vocal instruction to refocus on the road. Specifically, the alert states "Please refocus on the road." In order to ensure that there is no overlap in the audio alerts if triggered more than once, threading using Python's Thread library was used to run the audio alerts. The library used for playing audio is Pydub. Additionally, we wanted optimal volume experienced by the driver, so our speaker is situated facing our driver. The volume of the alert is also loud enough to be heard over a car radio. Furthermore, we created several other audio prompts specifically for the calibration steps that also use the threading and optimized volume to make the experience for the driver better.

5.10 Init System

To ensure that FocusEd runs upon pressing the system's power button, we used systemd, an init system and system manager on Linux distributions. We created a systemd service that initializes the FocusEd components that must be started after the Linux kernel is booted [7].

6 TEST & VALIDATION

6.1 Results for Face Detection

We tested our face detection algorithm of HOG and SVM on 4 databases, covering a total of 4,355 photos. We exceeded our goal of a 90% accuracy by achieving a facial detection accuracy of 94%. The databases used were Labeled Faces in the Wild (LFW), FACES, Gourier and Crowley, and Closed Eyes in the Wild (CEW).

We noticed that the pictures where faces were not detected were those where either the face was too small, the face was partially covered such as with long bangs or sunglasses, or the photo was taken at extreme non-frontal angles such as side or upward angles that do not include both eyes in the frame. These significant results point us towards the drawbacks of HOG and SVM, specifically being that we trade better efficiency for higher inaccuracy for non-frontal photos. Additionally, as stated before in 4.1, obstructions to the face such as long bangs or sunglasses are out of scope for this project. Finally, because our camera is placed directly in front of the driver on their dashboard, specifically 15.5 in - 20 in away from the driver's face, we do not have to worry about detecting small faces in our frames.

6.2 Results for Head Pose Estimation

Head pose estimation was tested with 2 databases, Crowley and FACES, for a total of 1,451 photos. The SVM model used 80/20 and 50/50 train/split using sklearn train and test split function[27]. Additionally, we took advantage of the random state variable in the sklearn function. This variable allows us to shuffle the photos in our dataset to test our trained model for robustness. The random state variable iterations were 0, 15, and 42, as these were the most common values[26]. The average of all 6 tests was 93% average with the lowest percentages occurring when the random state variable was set to 42 as this created the greatest difference in training versus testing images. Our result exceeded our design specification of 85% which improves the reliability of our system. This ensure that we are able to classify driver's head position's well enough as to not inconvenience the driver with unnecessary alerts.

6.3 Results for Eye Classifier

The eye classifier was tested on 3,403 photos across with 3 databases— Specs on Faces, Closed Eyes in the Wild, and Labelled Faces in the Wild. We achieved 92% correct classifications of eyes open or closed, which surpassed our desired accuracy of 90%. We ensured to test upon photos of individuals of various races and ages to obtain a holistic accuracy that best estimated the variety of individuals that would be using our system.

6.4 Results for Focus Timers

The focus timer was tested by running 100 trials in a parked car on two individuals. It was tested whether or not the system accurately detected a drowsy or distracted individual in the corresponding time frame—2s or 1s respectively. The driver performed normal driving movements, such as looking at the rear view and side mirrors for the recommended less than 2 seconds [31], as well as distracted movements— such as talking with a passenger and looking at a phone at various angles— and the drowsy behaviour of closing one's eyes. After running 100 trials with various driver behaviours, we achieved accurate detections 87% of the time. Significant instances where we noticed the driver abiding to the 2-second rule but still being classified as distracted is when the driver makes a turn on the road or is looking backwards when in reverse.

6.5 Results for Power Supply

To test the power supply, the TalentCell power bank was left running with our system for as long as it could. The power bank was able to power the Jetson with our algorithms running for 8.5 hours. This was within the our goal, which was a range of 8 to 10 hours.

6.6 Results for Audio Alert

The audio alerts were tested in 30 trials. The driver would perform a distracted behavior, whether closing their eyes for more than 1 second or not looking forward for more than 2 seconds to trigger the distracted alert. We determined that our system triggers the audio alert 100% of the time.

6.7 Results for System Latency

For system latency, we wanted our system to detect and output alerts within 3 seconds, After 30 trials, the system detected and outputted within an average of 3.9 seconds. This includes the 1-2 seconds that the driver is distracted or drowsy. This was tested by timing how long our system took to alert the driver that they were distracted. The start time began when the user would close their eyes or look to the side. The end time was when FocusEd would trigger the audio alert. However, upon further testing, by reducing the size of the OpenCV window, our system latency average reduced to 2.5 seconds.

7 PROJECT MANAGEMENT

7.1 Schedule

Fig. A1 and A2 present this project's Gantt Chart. This schedule has faced significant revisions since presenting the design stages due to the changes in our system, such as having various iterations of our head pose model and adding GPU acceleration, both of which had bugs that set us back in our schedule.

7.2 Team Member Responsibilities

All team members worked together throughout the design, development, and testing of this project. We assigned each team member to be in charge of specific aspects of the design. Heidi focused on implementing and testing the facial detection, facial landmarking, and head pose estimation algorithms on the Jetson Xavier. Vaheeshta focused on implementing and testing the eye classification algorithm, training the eye classification algorithm, and creating the system calibration. Danielle focused on implementing the focus timer, creating the audio alert system, and helping with training.

7.3 Budget

The table represented by Fig. A3 presents all materials that were purchased for the development of this project. Out of our $600 budget, we used $509.47. This is a $50.90 increase from our proposed budget of $455.57. In the figure, items that we did not plan for in the design report but ended up requiring are labelled as "No" in the "Planned?" column, and items that we obtained but did not use are labelled as "No" in the "Used?" column.

Most of our budget is used by the NVIDIA Jetson Xavier NX Developer Kit, which totalled $399.00. We were able to have $90.53 unused in our budget largely due to us having 12 materials either on hand or provided by the 18-500 inventory.

7.4 Risk Management

There are several risks we considered while planning the project. For one, there was significant concern about our system latency. Specifically, detection of drowsiness and alerting the driver might not happen in the desired time frame. While we have metrics discussed previously to determine how fast we would like our system to be, until we began our first tests of our system, we were unsure if we can meet these metrics. Therefore, our risk mitigation was in the form of a fallback design. Our backup plan included performing face detection on the Jetson Xavier, then crop out the face and send this to the cloud using AWS to perform the remainder of our computation. Then, the Jetson Xavier would receive the signal from AWS of whether or not to trigger the audio alert. Thankfully, this backup plan did not need to implemented. With our algorithms running we were seeing up to 8 fps. To improve this number, we added GPU acceleration and lowered the camera resolution and decreased the size of the image the Jetson intakes. This resulted in our system receiving up to 17 fps.

Additionally, another risk we considered was distinguishing between normal movements and distracted movements, and thus not detecting all distracted instances. Our mitigation was to slightly overcorrect to ensure that we at least detect the distracted cases. We would rather have more false positives than false negatives, while still keeping both within reason. We went through several iteration for our head pose estimation and ensuring to test on the Jetson as often as possible. The final head pose ratio is independent of the distance between the driver's face and the camera module on the car dashboard which improves the reliability of the algorithm. To help create a more inclusive model, we incorporated photos from established datasets in addition to our custom dataset.

Moreover, another risk we anticipated involves our TalentCell power bank not being able to power the Jetson Xavier for 8-10 hours as our metric hoped. To mitigate this risk we researched power specifications for the Jetson Xavier NX, chose the 10W and 4 CPU core mode, and ran the Jetson with the power bank for as long as it could. Additionally we had a backup plan to use the car's power outlet in the event the power bank failed to perform.

A risk that we did not anticipate but ran into during the project were dependency issues of additional libraries when implementing audio, GPU acceleration, and threading. To resolve this, we reflashed the SD card of the Jetson, and lost little time due to our documentation of the installation processes that we had taken previously.

We were also concerned about the possibility of going over budget. However, we made use of any resources we already had, and limited our budget to purchasing products that were essential and could not be replicated with other resources already at our disposal.

Lastly, our final risk involved varying lighting conditions affecting our drowsiness detection accuracy. We avoided this risk by limiting our scope to only direct lighting conditions. If we faced testing conditions where we did not have daylight, we used a ring light instead.

8 ETHICAL ISSUES

The main ethical issue of our system is the privacy concern of what we do with the video stream after the driver turns off our system. The safety of the users privacy is important in any system that involves recording. Since we are observing the driver in real-time, we do not store any video or driver data. On the physical box of our system we include a blurb that describes how the video is used while the driver is on the road. A possible case would be that some adversarial subject could manipulate the code to save the video files of the system and retrieve information about an individual's facial movements or driving habits. An approach to mitigate this risk would be to revoke writing permissions to the specific scripts and physically protecting the hardware by removing it from the vehicle when the driver is not using it. Also, blocking or removing additional unused USB or HDMI ports could protect the system. Another approach could be to add a software case check to examine if any additional system is plugged into the Jetson, and if so, to not power on.

Another ethical issue would be ensuring that users are not driving during the calibration step of our system as this would pose a danger to the driver and those around them. We currently have an audio warning in our system to warn

the driver to not perform calibration while driving. A future mitigation technique for this issue would be to include an accelerometer to ensure that the car is at 0 mph during calibration. If the car is not at 0 mph, then the system will stop asking the driver to calibrate and instead proceed to detection while using the baseline eye classification model.

9 RELATED WORK

Many solutions for distracted driving tend to focus on the vehicle itself, such as the use of lane detection and automatic breaking, or on the phone with cell phone blocking. However, there exists several solutions to distracted driving that are similar to our project, FocusEd, and use cameras to assess the driver's distracted behavior.

One such commercial solution is the device, GoFleet. GoFleet's scope is rather larger than ours and uses audio alerts when the driver is "texting, eating, micro-sleeping, holding a phone, yawning" and also uses infrared LEDs for night detection [9]. Another solution that is slightly different than ours was created at the University of Texas at Austin by a group of researchers that used deep learning to create a distracted driving solution. Their project trained using the State Farm distracted driving data and combined that data with image augmentation to effectively predict the distracted behavior [20]. Finally, another solution uses similar hardware – a Raspberry Pi Camera Module along with a Raspberry Pi – and HAAR Cascades to detect yawning and the likes, as well as alcohol consumption [24].

10 SUMMARY

Overall, our system met the majority of our design specifications. The design specification that we did not meet was our focus timer detecting drowsy or distracted driving at least 90% of the time. Our system is limited to being used in ideal lighting conditions and does not account for other distracted behaviors besides falling asleep or looking away from the road that is directly in front of the driver. Additionally, we attempted to account for the driver looking at their side mirrors by adding the focus timer. However, on the road we observed that when the driver is making turns or reversing, they may be turned away from the forward position for longer, but this is not a distracted movement as their eyes are still on the road. To improve this we could have an additional measurement that works within our system to determine if the driver is making a turn. This could be implemented with an internal vehicle compass and if the driver's direction changes more than 45 degrees then this would indicate a turn and the system would not trigger a distracted alert if the driver is facing the direction of the turn. Additionally, in our welcome message we advise the user to not calibrate while driving, but to ensure this we could add an accelerometer to our hardware to design thus ensuring that the vehicle is stationary during calibration.

10.1 Lessons Learned

To groups who want to explore distracted driving in the future we recommend that they add additional cases to better categorize distracted versus normal behavior. Additionally, we would recommend to utilize fine-tuning if the driver does not properly perform calibration. Moreover, we would recommend to ensure that files required for the systemd service are not located in a directory that require sudo permissions for editing access. Also, initially we wanted to have a vibration motor attached to the driver's seat to alert that they were distracted rather than using an audio alert that could possibly be grouped with other audio alerts such as GPS notifications. We recommend to explore this option.

References

[1] Mahmoud Afifi and Abdelrahman Abdelhamed. "AFIF4: Deep Gender Classification based on AdaBoost-based Fusion of Isolated Facial Features and Foggy Faces". In: *Journal of Visual Communication and Image Representation* (2019).

[2] L.M. Brown and Ying-Li Tian. "Comparative study of coarse head pose estimation". In: *Workshop on Motion and Video Computing, 2002. Proceedings.* 2002, pp. 125–130. DOI: 10.1109/MOTION.2002.1182224.

[3] *Compile: Using dlib from Python.* URL: http://dlib.net/compile.html.

[4] Datahacker.rs. *011 How to detect eye blinking in videos using dlib and OpenCV in Python.* 2020. URL: http://datahacker.rs/011-how-to-detect-eye-blinking-in-videos-using-dlib-and-opencv-in-python/.

[5] *Dlib Shape Predictor.* URL: http://dlib.net/python/index.html#dlib.shape_predictor.

[6] Riediger M. Lindenberger U. Ebner N. C. *FACES - A database of facial expressions in young, middle-aged, and older women and men: Development and validation.* Tech. rep. 2010.

[7] Justin Ellingwood. *How To Use Systemctl to Manage Systemd Services and Units.* Nov. 2020. URL: https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units.

[8] X.Liu F.Song X.Tan and S.Chen. "Eyes Closeness Detection from Still Images with Multi-scale Histograms of Principal Oriented Gradients". In: *Pattern Recognitio* (2014).

[9] GoFleet. *Driver Distraction Camera.* URL: https://www.gofleet.com/Driver-Distraction-Camera.

[10] Nicolas Gourier and James Crowley. "Estimating Face orientation from Robust Detection of Salient Facial Structures". In: *FG Net Workshop on Visual Observation of Deictic Gestures* (Jan. 2004).

[11] Vikas Gupta. *Face Detection – OpenCV, Dlib and Deep Learning*. Oct. 2018. URL: https://learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/.

[12] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, 2007.

[13] Kh Tohidul Islam, Ram Raj, and Abdullah Al-Murad. "Performance of SVM, CNN, and ANN with BoW, HOG, and Image Pixels in Face Recognition". In: Dec. 2017. DOI: 10.1109/CEEE.2017.8412925.

[14] Davis King. *GitHub: Face Landmark Detection*. URL: https://github.com/davisking/dlib/blob/master/python_examples/face_landmark_detection.py.

[15] Caio Bezerra Souto Maior et al. "Real-time classification for autonomous drowsiness detection using eye aspect ratio". In: *Expert Systems with Applications* 158 (Nov. 2020), p. 113505.

[16] Paul McWhorter. *JETSON XAVIER NX LESSON 3: USING A WEB CAM OR RASPBERRY PI CAMERA IN OPENCV WITH GSTREAMER*. June 2020. URL: https://toptechboy.com/jetson-xavier-nx-lesson-3-using-a-web-cam-or-raspberry-pi-camera-in-opencv-with-gstreamer/.

[17] NHTSA. *TRAFFIC SAFETY FACTS Research Note*. Dec. 2020. URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813060.

[18] Talal Bin Noman. "Mobile-Based Eye-Blink Detection Performance Analysis on Android Platform". In: *Front* (Feb. 2018).

[19] NVIDIA. *NVIDIA Jetson Xavier NX Developer Kit Carrier Board P3509$_A$01*. May 2020. URL: https://download.kamami.pl/p579520-Jetson_Xavier_NX_DevKit_Carrier_Board_Specification_v1.0.pdf.

[20] Satya Naren Pachigolla. *Distracted Driver Detection using Deep Learning*. 2019. URL: https://towardsdatascience.com/distracted-driver-detection-using-deep-learning-e893715e02a4.

[21] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[22] Michael Reynaldo Phangtriastu, Jeklin Harefa, and Dian Felita Tanoto. "Comparison Between Neural Network and Support Vector Machine in Optical Character Recognition". In: *Procedia Computer Science* 116 (2017), 351–357. DOI: 10.1016/j.procs.2017.10.061.

[23] Raspberry Pi. *Camera Module*. 2020. URL: https://www.raspberrypi.org/documentation/hardware/camera/.

[24] Vaibhav Rathod and Ranjana Agrawal. "Camera Based Driver Distraction System Using Image Processing". In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, pp. 1–6. DOI: 10.1109/ICCUBEA.2018.8697463.

[25] Adrian Rosebrock. *Training a custom dlib shape predictor*.

[26] sklearn. *Glossary of Common Terms and API Elements*. URL: https://scikit-learn.org/stable/glossary.html#term-random_state.

[27] sklearn. *sklearn.model_selection.train_test_split*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[28] Tereza Soukupová. "Real-Time Eye Blink Detection using Facial Landmarks". In: *21st Computer Vision Winter Workshop* (Feb. 2016).

[29] StackOverflow. *Tensorflow vs OpenCV*. 2018. URL: https://stackoverflow.com/questions/49919300/tensorflow-vs-opencv.

[30] StackShare. *OpenCV vs Tensorflow*. URL: https://stackshare.io/stackups/opencv-vs-tensorflow.

[31] Eric Taub. "2-Second Rule for Distracted Driving Can Mean Life or Death". In: *The New York Times* (2018). URL: https://www.nytimes.com/2018/09/27/business/distracted-driving-auto-industry.html.

[32] Elaine Wu. *NVIDIA Jetson Nano and Jetson Xavier NX Comparison: Specifications, Benchmarking, Container Demos, and Custom Model Inference*. June 2020. URL: https://www.seeedstudio.com/blog/2020/06/04/nvidia-jetson-nano-and-jetson-xavier-nx-comparison-specifications-benchmarking-container-demos-and-custom-model-inference/.

[33] "Chapter 5 - Driver Behavior Recognition in Driver Intention Inference Systems". In: *Advanced Driver Intention Inference*. Ed. by Yang Xing, Chen Lv, and Dongpu Cao. Elsevier, 2020, pp. 99–134. ISBN: 978-0-12-819113-2. DOI: https://doi.org/10.1016/B978-0-12-819113-2.00005-1. URL: https://www.sciencedirect.com/science/article/pii/B9780128191132000051.

Appendix A

TASK	ASSIGNI	START	END
Proposal Presentation	All	2/15	2/21
Research design specifications	All	2/22	3/1
Research face detection, landmarking, headpose examples	Heidi	2/22	2/24
Research eye classification examples	Vaheeshta	2/22	2/25
Write eye classification algorithm (on laptop)	Vaheeshta	2/26	3/9
Order hardware components	Vaheeshta	3/1	3/1
Write simple face detection (on laptop)	Heidi	3/2	3/4
Design Presentation	All	3/2	3/7
Write simple face landmarking (on laptop)	Heidi	3/3	3/5
Write face detection algorithm (on Xavier)	Heidi	3/7	3/10
Write simple head pose (on laptop)	Heidi	3/9	3/12
Design Report	All	3/10	3/17
Test eye classification algorithm (on laptop)	Vaheeshta	3/10	3/16
Test face detection algorithm (on laptop)	Heidi	3/12	3/16
Write face landmarking algorithm (on Xavier)	Heidi	3/16	3/19
Find and download datasets for open/closed (CEW & DROZY)	Vaheeshta	3/17	3/20
Test eye classification on self, partners, and datasets	Vaheeshta	3/20	3/22
Test face landmarking algorithm	Heidi	3/21	3/24
Write head pose estimation algorithm	Heidi	3/24	3/26
Set up Jetson/Camera/Static IP	Danielle	3/25	3/27
Test head pose estimation with different cutoffs	Heidi	3/27	3/31
Write calibration and training system for eye classification	Vaheeshta	3/29	4/3

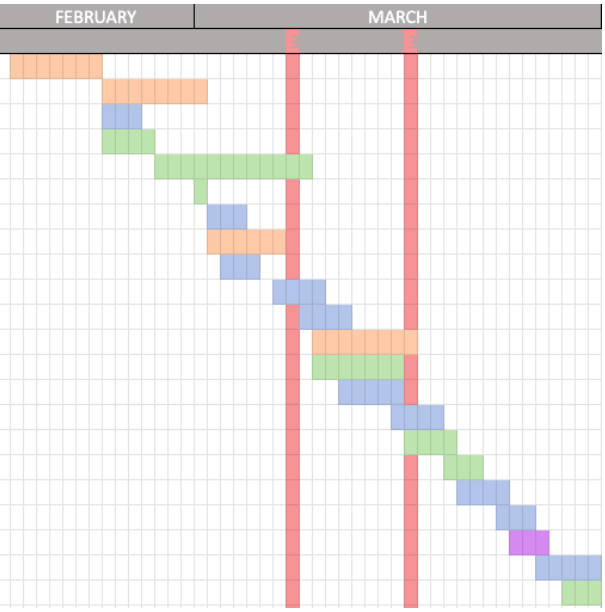

Figure A1: Gantt Chart, February - March

TASK	ASSIGNI	START	END
Write calibration and training system for eye classification	Vaheeshta	3/29	4/3
Integrate head pose estimation with landmarking	Heidi	4/1	4/2
Create custom dataset for face detection	Danielle	4/1	4/3
Write focus timer (2) classification based on time not facing forward	Danielle	4/2	4/7
Integrate eye classification + head pose estimation v1	Vaheeshta	4/3	4/9
Write focus timer (1) classification based on time eyes closed	Danielle	4/7	4/10
Write calibration and training system for head pose	Heidi	4/7	4/11
Integrate and test eye classification on Xavier	Vaheeshta	4/8	4/9
Prepare for interim demo	All	4/10	4/12
Integrate eye classification + head pose estimation v2	Heidi	4/10	4/11
Integrate head pose + eye clasification with focus timer	Danielle	4/10	4/11
Create audio prompts for calibration	Danielle	4/10	4/14
Create audio alert system	Danielle	4/12	4/18
Implement feedback from interim demo	All	4/13	4/17
Integrate custom data set with researched datasets	Danielle	4/15	4/20
Implement GPU acceleration	Danielle	4/18	4/23
Implement threading	Danielle	4/18	4/23
Integrate all parts	All	4/19	4/22
Slack	All	4/20	4/27
Test in controlled environment and gather metrics	All	4/22	4/30
Test audio alert and audio prompts on Xavier with speaker	Danielle	4/22	4/26
Test Xavier with power bank	Vaheeshta	4/25	5/1
Final Presentation	All	4/27	5/3
Test in car and gather metrics	All	4/29	5/1
Final Video + Poster	All	5/2	5/10
Final Report	All	5/7	5/14
Public Demo	All	5/13	5/13

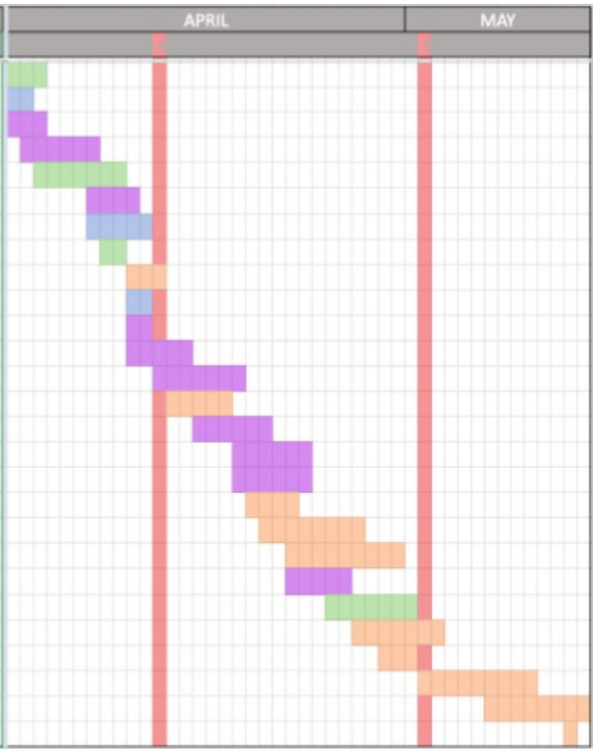

Figure A2: Gantt Chart, April - May

Item	Supplier	Planned?	Used?	Quantity	Price per Unit	Cost
AWS Credits	18-500 Inventory	Yes	No	1	$0.00	$0.00
Ethernet Cable	Personal Inventory	No	Yes	1	$0.00	$0.00
HDMI Cable	Personal Inventory	Yes	Yes	1	$0.00	$0.00
Jetson Xavier NX Case	Geekworm	No	Yes	1	$25.43	$25.43
Keyboard	Personal Inventory	Yes	Yes	1	$0.00	$0.00
Male to Male 5.5 x 2.5mm DC Connector	CableCC	No	Yes	1	$8.35	$8.35
Micro USB Charging Cable	Personal Inventory	Yes	Yes	1	$0.00	$0.00
MicroSD Card 64 GB	SanDisk	Yes	Yes	1	$13.15	$13.15
MicroSD Card 8 GB	SanDisk	Yes	No	1	$5.93	$5.93
Mini External USB Stereo Speaker	Adafruit	Yes	Yes	1	$12.50	$12.50
Monitor	Personal Inventory	Yes	Yes	1	$0.00	$0.00
NVIDIA Jetson Xavier NX Developer Kit	Nvidia	Yes	Yes	1	$399.00	$399.00
Picture Hanging Command Strips	Personal Inventory	No	Yes	1	$0.00	$0.00
Precision Screwdriver Set	Personal Inventory	No	Yes	1	$0.00	$0.00
Raspberry Pi 3 Model B v1.2	18-500 Inventory	Yes	No	1	$0.00	$0.00
Raspberry Pi Camera Cables (6pack)	Pastall	No	Yes	1	$7.41	$7.41
Raspberry Pi Camera Cookie Wheel Case	C4Labs	No	Yes	1	$12.71	$12.71
Raspberry Pi Camera Module V2	Raspberry Pi	Yes	Yes	1	$24.99	$24.99
Ring Light	Personal Inventory	Yes	Yes	1	$0.00	$0.00
TalentCell Rechargeable 12V Battery	18-500 Inventory	Yes	Yes	1	$0.00	$0.00
Wireless Mouse	Personal Inventory	Yes	Yes	1	$0.00	$0.00
Total						**$509.47**

Figure A3: Bill of Materials

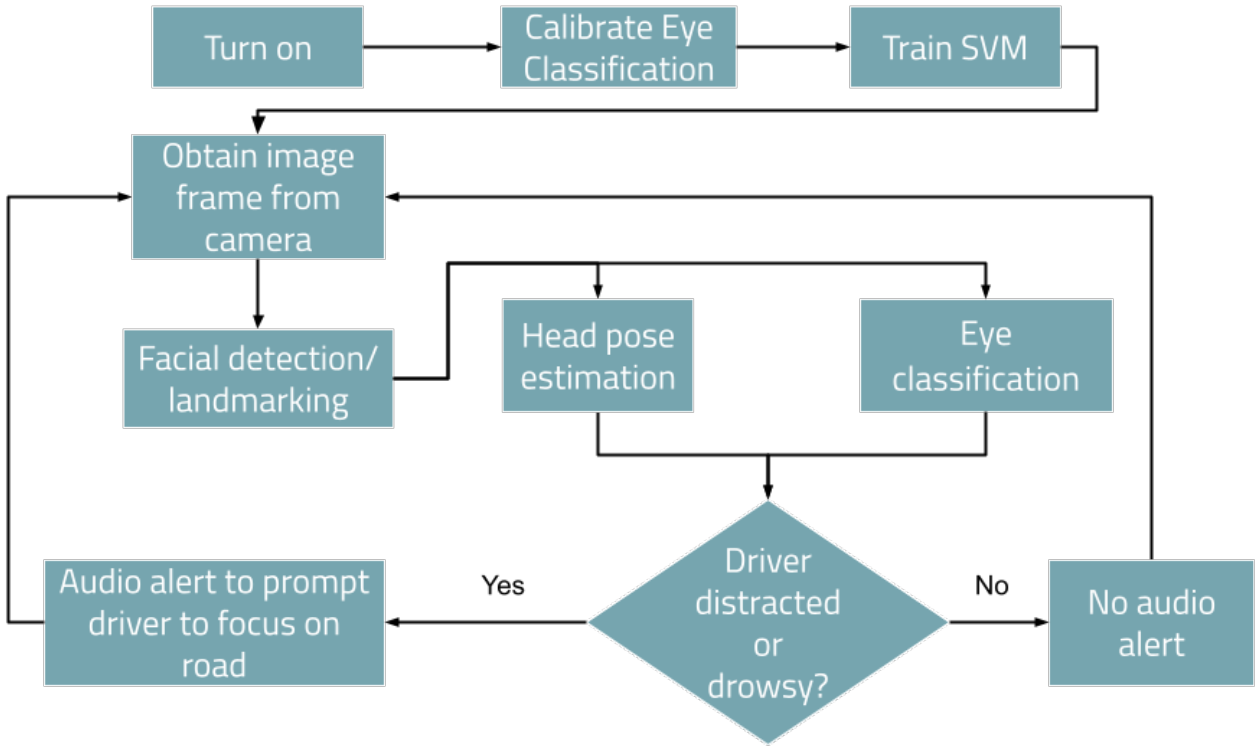

Figure A4: User Story

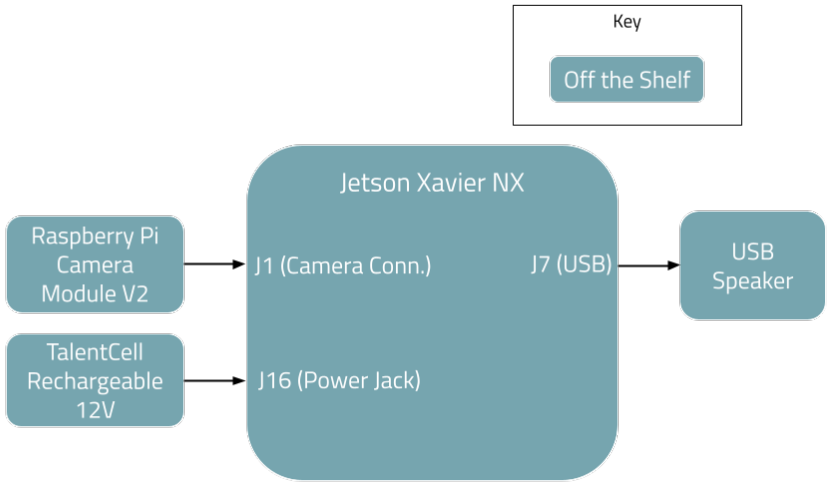

Figure A5: Hardware Block Diagram

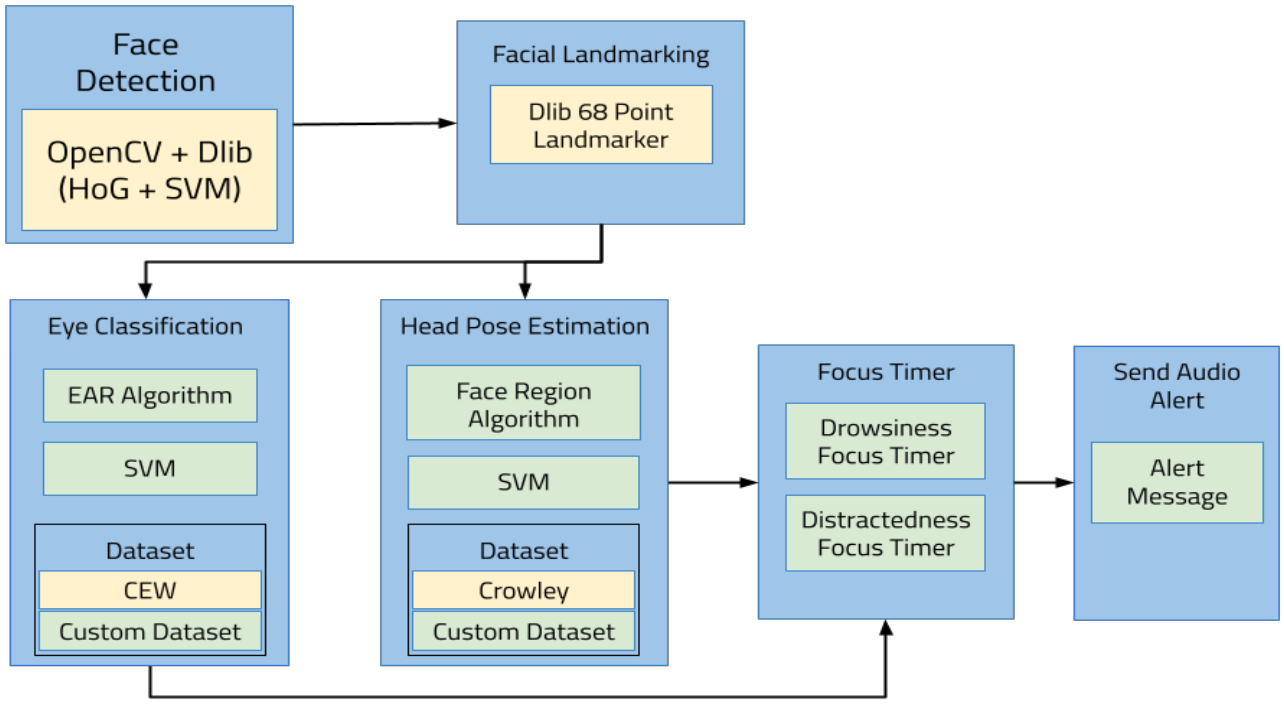

Figure A6: Software Block Diagram