Journal:

09/25/21

We signed up for the Synopsys fair. Our first step to start making our prototype is to learn OpenCV in python. We're using the following YouTube Video to learn it: "LEARN OPENCV in 3 HOURS with Python | Including 3xProjects | Computer Vision" by Murtaza's Workshop - Robotics and AI. We plan to properly finish the video within the next two weeks. We also plan on making and drawing out our plan/layout for our project once we have completed the video.

10/12/21

We started watching the OpenCV video listed in the previous entry. Unfortunately, we got a little busy with school work so we weren't able to finish it but we plan to get back on track within the next week.

10/18/21

We have finished the CV video and we are currently in progress in making our plan. We plan on dividing it into different parts and working on it. Two major parts are coding it to properly identify a person's face and then creating the timer portion.

10/24/21

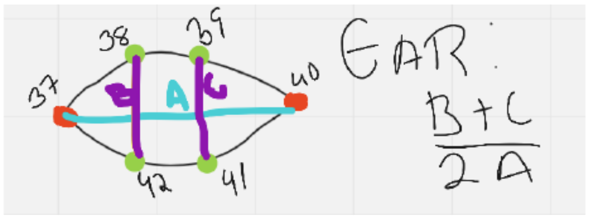
We have made our plan and we are ready to begin development for our prototype. However, with some difficulties, we have been doing some more research into this area and found and will use some resources from places like Github.

12/29/21

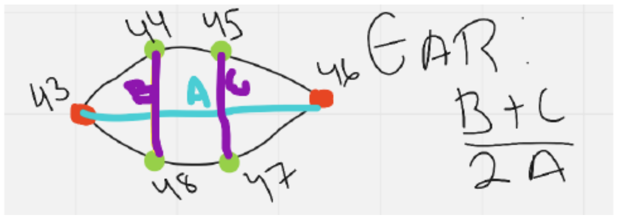
Winter break has started and we took a week break to relax and recharge ourselves after finals but we are ready to start working on the project again. We researched these python libraries to use in our timer, **time**, **playsound**, **pygame** (import **mixer**). We needed to use the time to get a start time for our timer to start counting, **mixer** to load the mp3 file for our alarm sound, and **playsound** to play our audio file. This took about two hours as we also had to understand how to use these libraries and what commands we have to use. Development for the timer has begun. Along with the timer, we have started our research on DLIB and how the face tracking software behaves in certain scenarios.

12/30/21

We have started programming the DLIB section of the code. Today we were able to create a flowchart of how the program will work. We have also discovered that the EAR for the left and right eye are very similar.



(left eye)



(right eye)

We just need to use different landmarks corresponding to each eye and have slightly different fields of vision. After making our flowchart, we have started creating our loops and functions for the ratio. Because we want our data to be real-time rather than every few seconds, we must incorporate this into our main while loop. First, we must find the coordinates that correspond with each landmark on the face. Later, we will be using these coordinates to calculate the EAR. To calculate these coordinates, we run a for loop from 36 to 42 for the left eye (landmark numbers), and a for loop from 42 to 48. Inside these loops, we use the .part(n).x and .part(n).y to calculate the x and y coordinates. Finally, we append these as pairs (x, y) into separate arrays for both left and right eyes.

12/31/22
Yesterday, we found the coordinates of each landmark around the user's eye. Today, since it is new year's, we only have a simple goal. To find the distance between the eyes and find the EAR. To find this distance, we have created a function named __RATIO__ and passed the array as our parameters to be used. In this function, we use a for loop to find 2 values x1 and x2 where x2 is the index after x1. In a loop, we subtract xn+2 to form a triangular pattern. Through the calculations, we get the two side lengths of the triangle which we can use to calculate the horizontal distance (A [from the image shown above]). After this, we return our EAR as leftEAR or rightEAR depending on the data passed. Then, in our loop, we use the formula (B+C)/2A as shown above. Since we both had other plans for the day, we decided to stop here and test the app tomorrow.

01/01/22
Today, after conducting multiple tests, we found that while our app does work, the EAR values are not very accurate and vary a lot over different uses. We realized this was because the program did not have a method of finding the angle and was assuming the angle between the diagonal and horizontal line was 45 which is not right. To fix this, we decided to scrap this method and implement a new method. In this method, we use the distance.euclidean() command to find the distance between 2 points. Since we already have the points in pairs in an array, we run the command 3 times and find the length of A, B, and C separately and then combine them at the end into one using our formula (B+C)/2A.

01/02/22
We have finished the timer code and are ready to implement it with the main code and run our trial. Unfortunately, we ran into multiple errors in multiple spots in our program crashing as well as in other attempts, the camera kept freezing or slowing down (lagging). We looked into this, turns out, the code we made for our timer, we had created another infinite while loop, so when the main while loop was running, every time it had to run the timer code, the main

while loop that was running the cv code kept stopping to run the timer's while loop.

01/04/22
We went on stack overflow and attempted to run two infinite loops to be run at the same time by using the threading python library. This would solve the problem of one while loop stopping to run another while loop. This did not work as well. It created more errors, even with the thread. In the end, we decided to scrap the timer code we made and then implement new code in the main while loop so it wouldn't freeze. This code was integrated without face recognition code to have the core of the app working perfectly. Now we just needed to integrate our user recognition software into the app.

01/05/22
Initially, we had no idea how to implement this, we were not sure how we would scan for users and remember their EAR. Although, we did know that we will be storing their names and their EAR in a file. So to make sure everything worked before we messed up our core, we began working on this in a different file. We began by entering data manually and iterating through it using a for-loop. While we first found the length of the file and traversed the loop that many times, we later realized we could just use the for line in line loop command. We also decided that we would only store data in the form [name: EAR] so that we can split our line at the space and extract our EAR. However, during our first try, we spent more than twenty minutes trying to find out why it was not working. We forgot to convert our extracted number to a float. After converting the number, half of our file work was complete. Now, we needed to make sure the data was coming from the program because the user will obviously not write their own data in a file. That defeats the whole purpose. So we decided to ask the user for their name and EAR, and add it into the file using the format. This way, their information is stored already when the user reopens the file next time.

01/15/22
With everything complete, we just need to test the app one final time. Through every step, we have been testing the app to confirm our steps work, but this was the final test. The test was passed with flying colors. Whenever we would squint (pretend to be drowsy) for too long, the app would play a timer, and as soon as we opened our eyes to the control EAR, the alarm would stop immediately. We also tested the file system with different caps-lock combinations which was also no problem because, on input, we used the .lower() function to translate everything into lowercase.

02/21/22
Today, we have received instructions that along with this notebook and abstract, we will also be needing a presentation and an optional video. We have planned to complete the presentation by the 23rd and finish the video on the 24th, so we can upload materials into judging folders on the 25th.

02/25/22
See you on the other side!