

Final Project

Federick Gonzalez, Tian Low, Bryanna Geiger

May 18, 2020

Abstract

The application can be described as a visual interface due to its interactivity with a user's gestures and its ability to deduce information from a visual input. A user interacts with the program by inputting a video stream of their face. The application responds to the users by determining the percentage of time that the user has been inattentive to the screen based on the input from the webcam. Possible expansions to this visual interface to provide possible "real life" implementations include simultaneously measuring the attention of multiple users, and verbally identifying which users are not paying attention in real time. The application matched the results of a human interpreter 91.7% of the time with single images, and 60% of the time with a video.

1 Introduction

The goal of the investigation is to determine what components of an individual have the greatest effect on detecting levels of fixation. Components are defined as head pose, which is the direction in which the user's head is facing, and eye direction, characterized as the direction the user's pupil is facing. The application will detect the amount of fixation that individuals in a prerecorded video presentation are giving to the presenter or presentation on the screen based on the aforementioned components. Based on the weights of importance given to each component detected by the application and their respective accuracy rates in detecting whether an individual is fixated on the screen, this investigation aims to explore which components provide the most information about perceived attention, and the effects of combining multiple components to detect fixation.

2 Methods

In order to accurately gauge attention, it is important to clearly locate the face and body. That is difficult to do when different users have completely separate background conditions. Take figure 1 for example, There are lights, flags, and walls that are roughly similar to skin color in the specific lighting circumstances, and the lighting is strongly coming from the rightmost side of

the image, distorting facial color and possibly affecting attempts to find a face through color thresholds alone. It is therefore simpler to remove the background of an image before performing any sort of computation, in order to ensure that any interpretation of position or direction will be performed on a person, rather than a background object.



Figure 1: Example Video input frame

Initially, the ViBE technique, which can be found in source [1], was intended to be utilised. Essentially, for each pixel, there are a set of previously calculated pixel values that serve as a sample set. The current pixel value is compared to the sample set of previous pixels, and if there is a significant deviation, then this is no longer part of the background, through use of the Euclidean Distance formula. This set of samples to be compared against is known as the background model, and can be initialized from a single frame of input. Since there is no previous data to work off of, it can be assumed that there is a temporal distribution, and the background model can be made of pixel values close to the one being analyzed. The samples are to be randomly selected, as to allow for a statistically significant comparison to the surroundings, allowing a detection of whether the current location is a background space or not. The update method "update method incorporates three important components: (1) a memoryless update policy, which ensures a smooth decaying lifespan for the samples stored in the background pixel models, (2) a random time subsampling to extend the time windows covered by the background pixel models, and (3) a mechanism that propagates background pixel samples spatially to ensure spatial consistency and to allow the adaptation of the background pixel models that are masked by the foreground." This proved to be too intensive of a process to be handled by a laptop. The amount of computation required to initialize the background from a single frame took over 10 minutes when reducing image sizes to approximately a quarter of their original size. The actual process of removing the background took too long for a short video to reach completion. The few frames that were correctly analyzed, however, did not function as hoped, leaving vague outlines similar to those that can be seen in figure 2

Instead of using this technique, it was discovered that OpenCV contained its own background removal method, BackgroundSubtractorMOG2, which is sup-

posed to take an image, create a mask of areas determined to be the foreground, and then return said mask. The foreground and background differentiation were determined by movement within a frame, essentially calculating the pixels that were not exactly the same value between two frames and then marking those as foreground, and all others as background. The image generated from this subtraction method is included below, and did not provide information that would assist in a recognition of attention, as it was impossible to determine precisely where the body was from the collection of seemingly unrelated points.

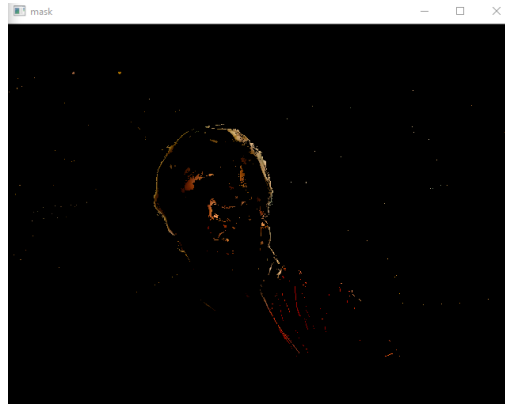


Figure 2: Example Video input frame with application of BackgroundSubtractorMOG2

Knowing what the original image is, it is rather clear that this is a vague outline of a head shape, however the outline is not complete, and it proved difficult to find a point or group of points where one could binarize the image as Head and Not head. After trying multiple techniques, such as finding out which point had the most non-black points within a certain radius, proved unsuccessful across varying media, as it could never be certain how large the radius should be and whether or not the background subtractor would be accurate enough to discover a head. Additionally, if a user had a sudden dip in framerate and was not moving significantly between frames, then there would be no dots at all, as the BackgroundSubtractorMOG2 is based on frame by frame differences rather than comparing to a background model.

After much experimentation, it was discovered that background and foreground differentiation could be achieved using semantic segmentation (source [11]). Using torchvision, the image was cropped and centered, and Deeplab, a semantic segmentation architecture, was used to segment the image into background and foreground. Semantic segmentation is the process of partitioning a digital image into multiple segments, essentially dividing the image effectively between person and not person. The image was again reshaped and resized, and applied as a mask over the original image, resulting in the background removal that can be seen in figure 3.

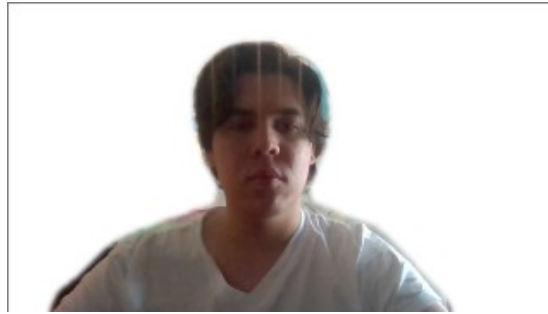


Figure 3: Example Video Frame with removed Background

Though this background removal method is clearly not perfect, as can be seen through the visibility of some portions of the background in the above image, a majority of confounding elements in the reference frame could be used, allowing for contour detection. Semantic segmentation proved to be the most accurate and useful background removing algorithm, and was used to separate the background from approximately one frame every three seconds.

The initial method for face detection was to detect contours in the using the OpenCV library, then detecting the children contours within the face contour based on the hierarchies. The images' backgrounds had to be inverted in order to prevent the contours from drawing around the background instead. However, this method proved to be unreliable, as the program would sometimes detect erroneous contours that were not part of the face.

Next, Haar Cascade Classifiers were used as a way to detect the general area of the face and eyes. Because of limited access to different faces due to quarantine, the Haar Cascade Classifiers provided in the OpenCV library were used, which are based on the work from source [6]. They have been trained on more than 1000 images, some that are used to represent what constitutes a face, and some that are used to represent what does not constitute a face, which allows the program to have increased generalizability. Blue, red, and green rectangles were used to outline the bounded area for the detected front-facing faces, side profile faces, and eyes, respectively. However, sometimes the classifier would over-detect, as shown in figure 4, where the individual's neck is also detected as its own face.

However, the Haar Cascade Classifier function very accurately detected the side profile faces as shown in figure 5 using a red bounding box. Consequently, the function was kept as the first part of the face detection process.

Another issue stemmed from the difficulty in exactly pinpointing the location of the individual's eye, and subsequently, the direction of the individual's gaze. The initial idea was to locate the pupil by using the hough circles method in the OpenCV library to detect circular objects in the bounds provided by the Haar Cascade Classifier. However, as shown in figure 6, the circles detected as indicated by the green circles were not representative of the pupil; the circle

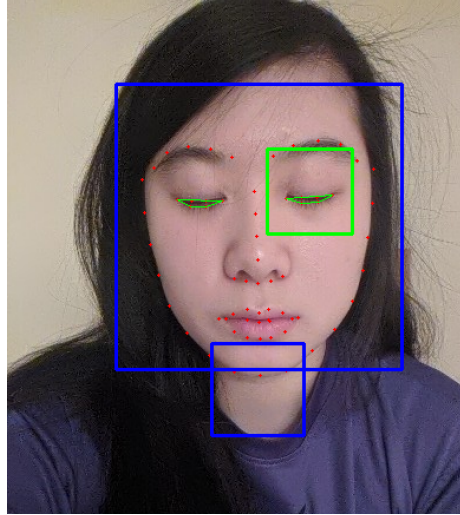


Figure 4: Example of inaccurate face and eye detection with Haar Cascade Classifier

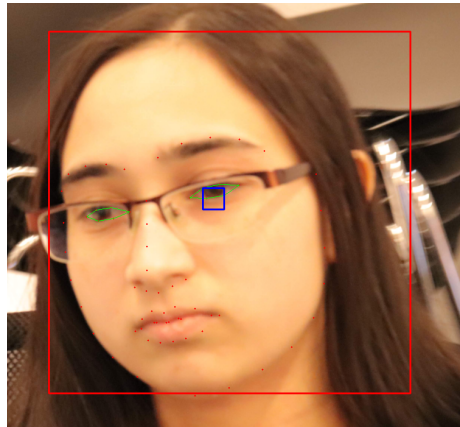


Figure 5: Example of side head pose and improved pupil detection

typically went around the entire eye and sometimes had extra circles in other areas. Despite numerous attempts at tuning the parameters for the circle detection, the program was never able to detect any of the individuals' pupils. This may be because while the pupil is circular, the part that is actually seen in the image can have a point when looking to the side, or is more of an oval shape rather than a perfect circular shape. Furthermore, the bound provided by that Haar Cascade Classifier contains a lot of skin that likely confuses the circle detector.

Consequently, to improve eye detection, the dlib shape predictor was used to

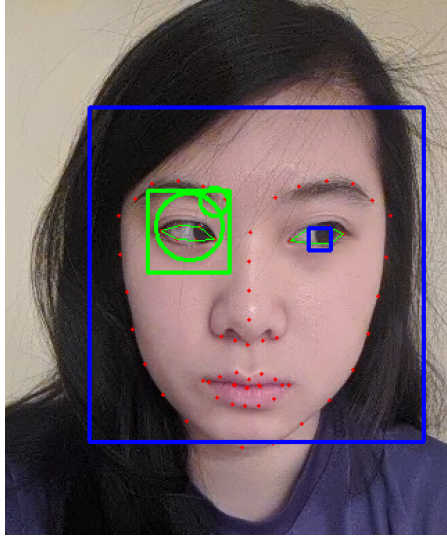


Figure 6: Face and eye detection with hough circles vs. pupil detection

provide a more specific mapping of the face. The shape predictor is trained on the iBUG 300-W dataset, as referenced from sources [7-9], using the template shown in figure 8. One can see the increased detail that the dlib shape predictor provides in comparison to all alternative methods, as outlined by the small red dots on the face and green outlines of the eyes in figures 4, 5, 6, and 7. Once again, generalizability was ensured, as the shape predictor was trained on 300 images.

After having a reliable method of tracking the eye location, eye aspect ratio calculations and pupil tracking were made possible. The eye aspect ratio formula, from source [10], is

$$\frac{||p_2 - p_6|| + ||p_3 - p_5||}{2||p_1 - p_4||}, \quad (1)$$

where $p_1 - p_6$ is 37-42 and 43-48 for the left and right eyes, respectively, as labelled in figure 8. This formula allows for open and closed eye detection, although its general use is for determining blinking across a period of time. Because static images were used, a simple limit for the eye aspect ratio was used instead. An accurate limit for the eye aspect ratio was tested to be about 0.12; any eye aspect ratio below this was considered a closed eye, and any eye aspect ratio above this was considered an open eye. If at least one eye was determined to be closed, the program immediately marks the individual as not paying attention. This is because the accuracy of this calculation was found to be one of the most reliable metrics that were tested.

If the eye was considered open, the pupil was then tracked to see if it was located in the center, or not in the center to determine eye gaze. The idea was

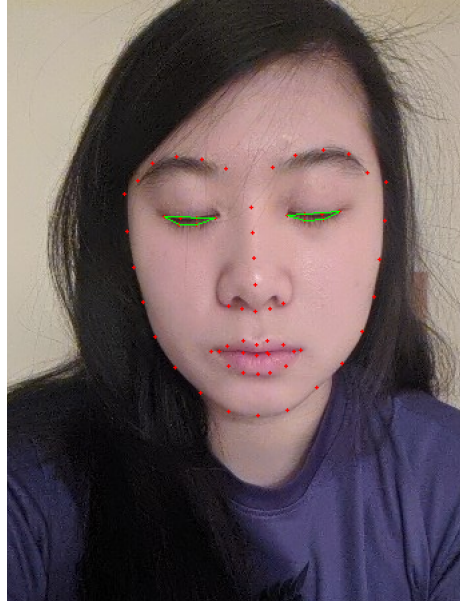


Figure 7: Example of Closed Eye

to take the bounding around the center of the eye as represented in figure 8, and calculate the percent of the area that was not the whites of the eyes. Examples of the bounding box in an image are shown as blue squares in figure 5 and 6. If the pupil area percentage was considered to be at least a certain minimum number that was tested, then the pupil was in the center; otherwise the pupil is considered to be not in the center. The minimum area required was measured to be about 80%. This tolerance is required, because the bounding is a box, which could cause the whites around the circular pupil or a small amount of skin to be detected.

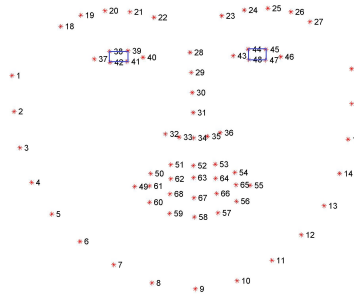


Figure 8: iBUG 300-W Template, with indication of the blue bounding box used for pupil calculations

Using all of the above metrics, the program first determines if the individual's eyes are closed, looking sideways, or looking forward with respect to the individual's head. As stated previously, closed eyes are an immediate indication of not paying attention. The program does not need to check anything else, because if someone's eyes are closed, they are definitely not paying attention, regardless of which way their head is facing. Next, the person's face is checked to see if it is side facing with the Haar Cascade Classifier. This step is done before checking if the individual is front facing, because if the individual's face is slightly looking to the side, then the face can still be found with the Haar Cascade Classifier. However, even a small degree of head turning indicates that the individual is likely looking past the computer screen, as the most "natural" position to look is if one is looking directly in front of them. However, by combining the pupil tracking into the program, if the individual is side facing and looking sideways with respect to their head, then they are marked as paying attention. The individual is then checked if they are forward facing. If they are forward facing, and their eyes are also looking forward with respect to their head, then they are paying attention. This additional analysis allows for a more robust program. If no face is detected, the individual is not paying attention, as they are not present, at least according to the laptop camera's point of view. To simplify, a diagram is provided below in figure 9 with the steps that the program takes to reach its decision.

User Testing: The final part of the project involved evaluating the system against our evaluation metrics which were comparing the results against a database of ground truth and consistency across results. This step was defined as the user testing process. In order to compile a database of ground truth, human evaluators were used to determine the ground truth, which was deemed to be an array of binary values inputted.

The array the inputted values were compared against were generated by iterating over each frame. The generated array was first iterated over to determine the percentage of time an individual was focused during the entirety of the sample size. After which the values of zeros and ones of the generated array were compared with the values of the inputted array. This step was intended to determine the accuracy of the system's results with what was deemed to be the "correct" output as determined by our human evaluators. The results generated included a.) the percentage of time an individual was focused and b.) the accuracy of the system's results.

3 Results & Analysis

The expected results are a textual representation of whether or not a user is paying attention, detailing where a user is looking in relation to their head position, if their eye was visible, and whether or not they were paying attention. The analysis revolves around the metrics of success. Our metric of success is that the program, when compared to a database of ground truth, similarly matches what a human deems to be focused.

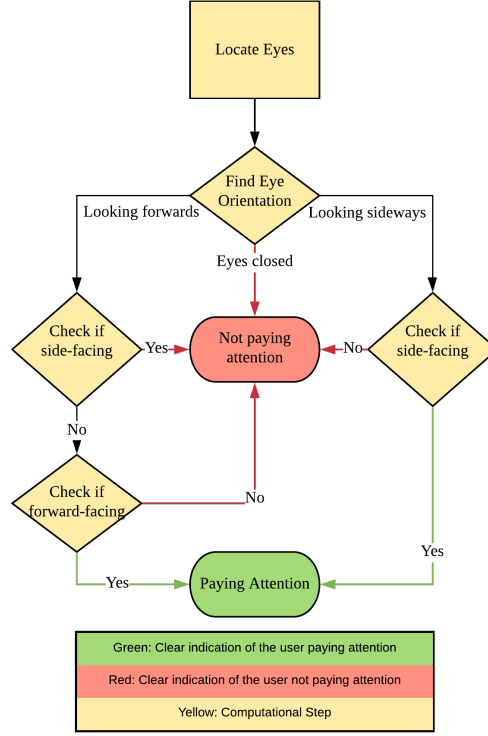


Figure 9: A diagram of the decision process made by the application

The database of ground truth was compiled based on human evaluators. Our group members served as the human evaluators. For our purposes, focus was deemed to be binary so our database of ground truth consisted of an array of values of 0 for not paying attention and values of 1 for paying attention. Our database of ground truth was compared against an array of values of 0 for unfocused and values of 1 for focused. This array was generated by looping over each frame.

Our database of ground truth or our inputted array was compared against the generated array. Through this comparison, we were able to ascertain how much the user was focused and the accuracy of our program in comparison to our database of ground truth.

We tested our program on 12 images over the course of 60 trials, tuning each parameter in the process. Based on these trials, the average accuracy of our program in comparison with the ground truth, was 91.7% percent. A 91.7% percent result in accuracy, for determining whether an individual appears focused, exceeds the expected results. A sample of the output is shown below in figures 10 and 11.

```
positionTracker x
C:\Users\chiyo\Anaconda3\python.exe C:/User
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
looking sideways
Head is front-facing.
Eyes are looking sideways.
The user is not paying attention.
closed
Head is front-facing.
Eyes are closed.
The user is not paying attention.
looking forward
Head is facing to the side.
Eyes are looking forward.
The user is not paying attention.
looking sideways
Head is facing to the side.
Eyes are looking sideways.
The user is paying attention.
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
closed
Head is front-facing.
Eyes are closed.
The user is not paying attention.
looking forward
Head is facing to the side.
```

Figure 10: Output for test images pt. 1

```
Eyes are looking forward.
The user is not paying attention.
None
None
user is focused 41.66666666666667 % of the time
accuracy of program output: 0.9166666666666666
```

Figure 11: Output for test images pt. 2

However, we then tested our program on 10 images retrieved from a test video over the course of 50 trials, tuning each parameter in the process, in an effort to expand the applications of our project. Based on these trials, the average accuracy of our program in comparison with the ground truth, was 60% percent. A 60% percent result in accuracy, for determining whether an individual appears focused, was a significant drop compared to the previous testing on the singular images. This is likely due to the limitations on the resources that the investigation had. When images of high quality were used as in the initial test image phase, the accuracy level of the machine was extremely high. However, due to the need to parse a video, some of the images' quality was sacrificed in order to allow the program to run; when parsing the video, the computers oftentimes overheated due to the amount of processing it needed to finish.

Curiously, only one individual was marked as nonexistent by the program multiple times. At first, it was deemed that it was because the individual was facing in different directions, thus being an error in the side face detection. However, because the individual was not looking near the screen in either, it was assumed that the face was turned so far up or to the side that the face was unable to be detected, and in these cases, the individual was definitely not paying attention.

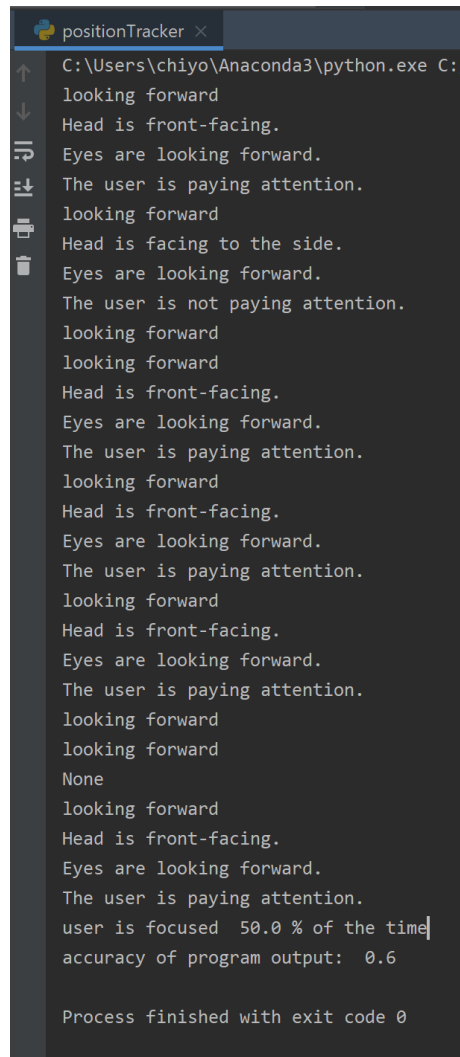
However, when testing frames from the video, it was the same individual whose face was unrecognizable, even while looking straight ahead. Yet, that same individual was also detectable with his hand in front of his face.

Based on the aforementioned faulty individual, it can be deduced that factors such as headphone presence and distinct shadows across the face greatly decrease the ability of the program to actually recognize it. The clear line across the center of the face could influence contour detection, therefore splitting the face in half and not allowing the Haar Cascade Classifier or the Dlib shape predictor to recognize face shape or the presence of facial features, therefore automatically returning that the user is not paying attention simply because they aren't detected.

We can conclude, that while there are limitations, based on our evaluation metrics, the accuracy and consistency of the results across each trial are within reason and thus the results of the system can be deemed acceptable.

4 Conclusion

It is concluded that, overall, head pose and eye direction are both equally important in determining user focus, and it is imperative they be used together. It is difficult to determine focus from eye position or head orientation alone, other than the idea that a closed eye implies a loss of focus. After removing the background from an image and then detecting facial features using contours from remaining image components, it was possible to differentiate between a head that was facing forward and one that was facing off to the side using a Haar Cascade Classifier function. Using the Classifier function in conjunction



```
positionTracker x
C:\Users\chiyo\Anaconda3\python.exe C:/
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
looking forward
Head is facing to the side.
Eyes are looking forward.
The user is not paying attention.
looking forward
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
looking forward
looking forward
None
looking forward
Head is front-facing.
Eyes are looking forward.
The user is paying attention.
user is focused 50.0 % of the time
accuracy of program output: 0.6

Process finished with exit code 0
```

Figure 12: Output for video

with the Dlib shape predictor, it was possible to determine a mapping of facial features in order to precisely determine where the eye was within the space and allow for calculations from pupil tracking. The eye aspect ratio was calculated using formula 1. The eye aspect ratio allowed for detection of whether or not the eye was open in the current frame. If one eye is closed, then the user is not paying attention. Otherwise, the pupil was located and tracked, and its position labeled as either centered or non-centered. Then, using these results in conjunction with the facial orientation from the Haar Cascade Classifier, it was possible to determine whether the user was paying attention or not, the process



Figure 13: Example of an individual with no face detected



Figure 14: Example of the same individual with no face detected



Figure 15: Example of the same individual with no face detected



Figure 16: Example of the same individual with face detected, but with hand slightly obscuring face

of which is detailed in figure 9.

The application was determined to be 60% accurate when tested with a video, and 91.7% accurate when tested with several predetermined images. The accuracy of the program was calculated by comparing the binary output of the program, where 1 means the user is focused and 0 means the user is unfocused, with a user inputted binary array for the same images. Though not perfect, the application performed within acceptable means. This program could potentially have been improved with greater computing power, allowing the ViBe technique to be used, and higher quality images. Possible expansions for this application could be utilizing the framework across multiple faces at the same time, recognizing certain individuals, and potentially determining what a user is focused on when gazing at the computer screen.

References

- [1] Barnich, O., & Van Droogenbroeck, M. (2011). ViBe: A Universal Background Subtraction Algorithm for Video Sequences. *IEEE Transactions on Image Processing*, 20(6), 1709–1724. <https://doi.org/10.1109/TIP.2010.2101613>
- [2] Engelke, U., Zepernick, H.-J., & Maeder, A. (2009). Visual attention modeling: Region-of-interest versus fixation patterns. 2009 Picture Coding Symposium, 1–4. <https://doi.org/10.1109/PCS.2009.5167393>
- [3] R. Oyini Mbouna, S. G. Kong and M. Chun, "Visual Analysis of Eye State and Head Pose for Driver Alertness Monitoring," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1462-1469, Sept. 2013.
- [4] R. Stiefelhagen, "Tracking focus of attention in meetings," *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*, Pittsburgh, PA, USA, 2002, pp. 273-280.
- [5] S. O. Ba and J. Odobez, "Recognizing Visual Focus of Attention From Head Pose in Natural Meetings," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 16-33, Feb. 2009.
- [6] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
- [7] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, M. Pantic. 300 faces In-the-wild challenge: Database and results. *Image and Vision Computing (IMAVIS), Special Issue on Facial Landmark Localisation "In-The-Wild"*. 2016.
- [8] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, M. Pantic. A semi-automatic methodology for facial landmark annotation. *Proceedings of IEEE Int'l Conf. Computer Vision and Pattern Recognition (CVPR-W), 5th Workshop on*

Analysis and Modeling of Faces and Gestures (AMFG 2013). Oregon, USA, June 2013.

- [9] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, M. Pantic. 300 Faces in-the-Wild Challenge: The first facial landmark localization Challenge. Proceedings of IEEE Int'l Conf. on Computer Vision (ICCV-W), 300 Faces in-the-Wild Challenge (300-W). Sydney, Australia, December 2013.
- [10] Soukupova, T., & Cech, J. (2016). Real-Time Eye Blink Detection using Facial Landmarks. 21st Computer Vision Winter Workshop. <http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf>
- [11] Ahmed, Z. (2019, July 23). Applications of Foreground-Background separation with Semantic Segmentation. Retrieved April 25, 2020, from <https://www.learnopencv.com/applications-of-foreground-background-separation-with-semantic-segmentation/>