ECM3401 - Final Project Report

Drivers tiredness, distraction symptom detection and warning system

by

Toluwanimi Abolude

May 1, 2019

College of Engineering, Mathematics and Physical Sciences

I certify that all material in this dissertation which is not my own work has been identified.

Abstract

This paper will provide a Computer Vision-based driver drowsiness system. Driver drowsiness is a significant cause of road accidents which include severe injuries and deaths. Measures have been suggested to determine driver drowsiness such as (1) vehicle-based measures; (2) behavioural measures and (3) physiological measures but in this paper behavioural measures are the main focus because they are accurate methods for measuring drowsiness and are less intrusive to drivers compared to the other two. In this paper, I will talk about the various technologies, either already made or newly inspired methods used only for this project. A literature review and other supports for my design choice. The result of testing my system to see how accurate my system is, as well as the evaluation of the results compared to what I expected. This will be used to determine if my system is a success. The overall outcome is a robust and accurate solution to drowsiness detection, which can be applied to the real world with further improvements.

Keywords: driver drowsiness detection, Facial detection, eye closure, yawning, head pose,computer vision.

Contents

1 Introduction

Driving entails controlling the movement, direction and operation of an autonomous vehicle. Driving a vehicle is quite tasking because it has different aspects that require the full attention of cognitive and conscious assets to keep up execution over a period. Any shortage of these assets, i.e. drowsiness, drunkenness or distraction could have fatal outcomes, including accidents. Drowsiness is a state between being sleepy and vigilant. It is crucial for drowsiness to be measured in order to find a robust way of countering the effect of drowsiness on drivers primarily because accidents that occur on the road due to drowsiness is up to 30% of road accidents and in 2013, 0.8% of those accidents leads to death (i.e. 800 deaths in 100,000 road accidents caused by driver drowsiness)[1].

There are several estimation techniques used in the real world already; for example, Ford Driver Assistant uses a vehicle-based method such as analysing rapid steering movement, irregular movement in the acceleration or brake pedal. Whereas, another more intrusive but accurate detection measures are physiological (relating to the branch of biology that deals with the normal functions of living organisms and their parts) which includes electroencephalogram (EEG) to monitor the drivers brain wave changes, Electrocardiograms (ECG) and Electrocardiograms(EMG) to find the relation between physiological sense and drivers drowsiness. The basic, less intrusive yet accurate and easily reproducible means of measuring drowsiness involves measuring the drivers' facial components like the eyes, mouth or even the pose of the driver's head (the direction the head is facing, i.e. left or right), this is known as behavioural measures. I believe my project is vital to aid in observing and warning driver when they show the symptoms of drowsiness that was mentioned above.

A drowsy person will show a lot of slower and unnecessary facial movement like nodding or hanging their head either side (left or right), frequent yawning and blinking. There are available methods to measure these behaviours, and I will mention the ones I used for this project and other factors that aided in my decision to do so. This project aims to produce a detection and warning system for drivers who are show characteristics of drowsiness and distraction, this consists of yawning, head pose looking left or right (not paying much attention to the road), and finally eyes closed for more than 0.5 seconds (blinking). This is to improve the safety of road transport and reduce casualties caused by sleepy drivers on the road with the help of computer vision(Computer vision-based driver monitoring is mainly used for detecting drowsiness and distraction using a camera placed in front of the driver [2]).

The report first shows a literature review about the eye closure measurement, yawning measurement, head pose measurement, methods used for precise facial detection. Detailed design of the outline of the system is presented with detailed justification as to why they were chosen. Following this, testing and the result of the system performance is compared to already existing detection systems, then the evaluation and conclusion of this project.

2 Summary of Literature Review

The literature review will initially provide critical analysis of possible methods that could be used in this project and give a summary of their advantages and disadvantages. The focus on the plans will include methods based on facial detection, methods based on eye parameter, methods based on mouth parameter and methods based on a head pose. Then I will give a verbose description of the problems I might encounter during system development and possible solutions.

2.1 Typical detection methods including their pros and cons

In this section, I will talk about the most common or second best option for measuring and detecting the face, eye, mouth and head pose. I will also mention why I did not use them in my system.

2.1.1 Methods Based on Facial Detection

Hidden Markov Models (HMM): HMM is a set of statistical model that is mainly used for grouping features of a signal [3] and it has two related processes; a Markov chain (it is underlying and unobservable) with an initial state probability distribution [4], state transition probability matrix, set of probability density functions (linked to each state), and a fixed number of states [4] as shown in Fig 1.

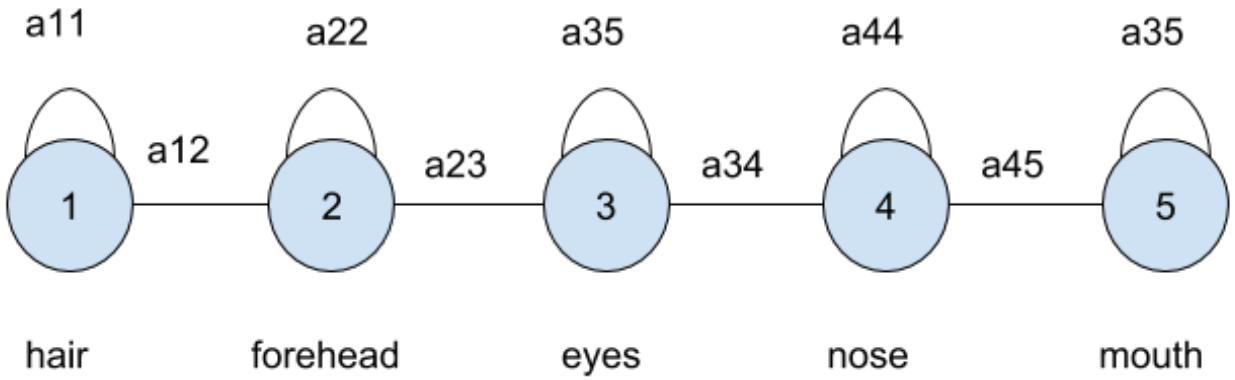

Figure 1: Left to right HMM for face recognition. Image taken from [5].

If the input has an image of a frontal, upright face position, then the properties of this face can be anticipated as the structure is the same, i.e. eye (above the nose and mouth), nose (above the mouth) and mouth (between the nose and chin). This proposes the use of a top to bottom model, this involves development between neighbouring states starting from the top and ending at the bottom [5]. The HMM model relates with vital facial features like eyes, nose, etc. In Fig 2 X,Y,L,M are labelled and used to calculate sequence O (The sequence of going top to bottom of an image with frontal face) by using the whole image i.e. X*Y and the window in the image of X*L for testing with window X*M. Where every perception vector equals a square of the L where there is an overlap with M [5].

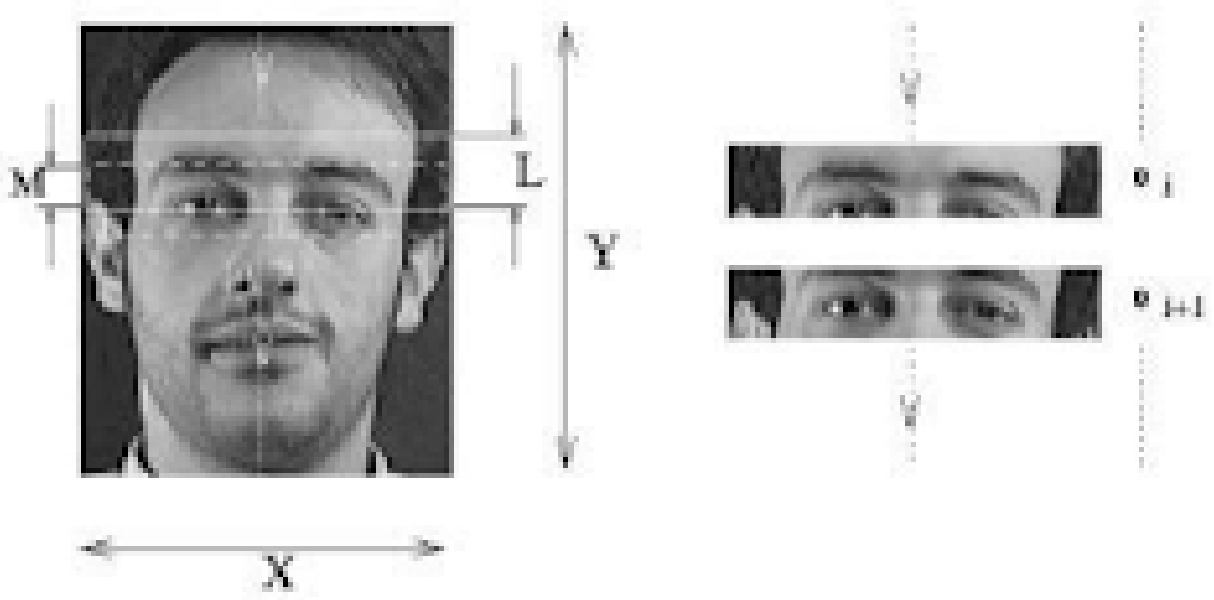

Figure 2: Image sampling method for one-dimensional HMM. Image taken from [6].

The main problems with HMM are that this method requires registered images to obtain a proper sequence obtained. The analysis is not precise, because the *raster scan* will not consider the notion that various components of the human face could possibly have a different information quantity [6, 7] and the computational requirement is increased because the whole image is processed,but every information about the image is considered [6].

2.1.2 Methods Based on Eye Parameter

Dinges and Grace introduced the most reliable method to measure a person's level of alertness called percentage of eyelid closure (PERCLOS) according to the dataset of the Federal Highway Administration with 98% accuracy [8]. However, a downside would be that it requires a considerable training time to get the essential features, i.e. the eyes from a set of training images when using principle component analysis (PCA) to classify the eyes. Another commonly used method for measuring eye parameter involves a 2D Gabor wavelet transform to extract the textural features of the person's eye and the neural network to classify if the eyes are open or closed [9], introduced by Rong-Ben. One of the main cons of using Gabor wavelet is that you have to use it many times from different angles, to form a relevant rendition of an image by merging all gathered images into one image to gain the most accuracy, and it is unsupervised [9].

2.1.3 Methods Based on Mouth Parameter

Rongben mentioned three mouth states for his method namely; normal, yawning and talking. He then used neural network - back propagation for classifying them [10]. A disadvantage for this method is that it takes time to train the dataset and it will be slow to implement in a real time scenario. Another prominent method for detecting yawning is mentioned by Yufeng and Zengcai who used the Euclidean distance between the tip of the nose and the chin to determine if the person is yawning [11]. I did not use this method because I feel that measuring the distance between the top and bottom lip makes more sense.

2.1.4 Methods Based on Head Pose

For head pose detection two famous methods involve using ellipsoidal head model (model methods include modelling the 3D structure of the human head, and the yaw angle is estimated by mapping the face feature points with the model to determine the yaw angle [12]. The ellipsoidal model is better than the cylindrical model because the human head is ellipsoidal in shape and is not cylindrical [12]) which was introduced by Lee et al [13]. The other is using the cylindrical model (similar to the ellipsoidal model but the head is modelled as a cylinder instead of an ellipsoid) to determine the driver's yaw, pitch and roll. The cons of these methods are;

1. They are sensitive to the misalignment of face feature points.
2. You can not represent all heads with a single head model.
3. They need high image quality and resolution.
4. They are easily affected by partial occlusions because some of the facial feature points are lost [12].

2.2 Problems I might encounter during system development

1. Illumination: It is said that Illumination is used to improve the quality of an image or video and can have a massive difference on how precise face detection can be, for example, if the surrounding of a video is dark with low lighting the face of the driver will not be picked up clearly because some features would be missing. For the 68 points dlib facial landmark to work, all the facial properties have to be clear to map those 68 points on the face. A possible way around this is using a camera with IR Illuminator which is not affected by less lighting in an image or video, but I used a regular camera because it is cheap to reproduce the result for my system and the accuracy for facial detection is around 98% for eye blink detection in [8].
2. Head Pose: Roll, Pitch and Yaw are three of the main head movements that cause a problem for the 68 points dlib facial landmark because it is in 2 dimensions (i.e. they are x-y coordinates) whereas the human face is a three-dimensional object and head movements are not limited to a set direction. In this case, this is a problem because we need to know where the driver is facing to tell if their attention is away from the road and if so warn them. A possible solution is to train a dataset where roll, pitch and yaw are labelled in the dataset by Amazons face detection API.
3. Occlusion: This occurs when a portion of the facial features, for example, eyes or nose might seem incomplete or obstructed [14] by other objects. This might be due to angle of the light is reflected or refracted on the face and other objects.

For detection, a pattern has to be established for it to be successful. To detect an object a trend that is synonymous to the object you are looking for has to be found to classify that object as a positive image. Haar-like features have been generally utilised in various boosting algorithms and object recognition, particularly face detection [15]. The strategy contains three principle thoughts to accomplish ongoing object recognition execution: compelling images for fast computation, AdaBoost for feature determination, and classifier cascades for the immediate dismissal of non-face windows. Detection speed has moved towards practical use because

of ease and the adequacy of the Haar properties [16]. Be that as it may, the rate of deception cost a lot at the expense of false positives, i.e. (wrong acknowledgement of objects said to be human) however the false negatives (i.e. dismissal of human appearance where applicable) is fundamentally low [17]. This may be because generally, *Viola-Jones dismissal cascade of classifiers* are joined with AdaBoost to frame solid classifiers where every node (i.e. an arrangement of weaker classifiers) are bundled to utilise the Haar properties [18]. The source [16] expressly states that a cascade or waterfall method classifiers are directed where multiple windows (any window that passes every hub will be considered a candidate human face) containing different objectives for an input image are successively tried for every hub in the course. With Viola -Jones course, every node has a precise recognition rate with less false negatives and false positive (close half, somewhat superior to this figure), this implies higher false positive rates [16]. A choice of dismissal ends the process of identification at any node, whereas the procedure for identification continues where temporary acceptance is made [16]. Along this line, time for computation is extraordinarily spared because different windows that have human faces will be less than windows that do not have human faces in an image.

3 Design

I will use this section to describes the high-level design that allows for building such a system - the modules that encompass the drowsiness detection system. Finally, it presents the work-flow of how those modules are bound to interact.

3.1 Initial Specification and Design

3.1.1 Overall System Description

The system is a python based system which helps determine if the driver being examined is showing signs of drowsiness which have been mentioned above and include yawning, frequent eye blinks, head tilting, and so on. This outcome of the detection is a visual/audio warning message to inform the subject of the drowsiness that has been detected. This system should be easy to access and run on Windows, Mac OS or Linux system as long as the python libraries required to run them. The following are the methods I proposed to be used in the measurement of each facial components:

1. Yawning: The recurrence of yawning can be measured over a specified period. To identify a yawn the distance from the top lips to the bottom (D) will need to be regulated. D will vary from a different facial expression like talking, smiling, laughing, and so forth. After the facial landmarks have been generated, measure the openness of the mouth you have to measure D (the distance between the top and bottom lips). There has to be a threshold distance T where if $D \geqslant T$ then it is recognised as yawning.

2. Eye Closure Analysis: A vital measure of tiredness is the state of the subjects eye which is popular to determine the level of fatigue. PERCLOS (which stands for the percentage of eye closure) is mainly used for measuring eyes state over time[19]. It has a boolean value to determine eyes state, i.e. open or closed. Another significant way for measuring eyes stat is EAR [20]. EAR stands for eye aspect ratio, and it measures the ratio of the Euclidean distance of the height and width of the human eyes [20].

3. Head Pose: Representing a 2D (x,y coordinate) representation of the human facial landmark (68x2 matrix) to be translated to get a 3D (x,y and z coordinates which represent the pitch, yaw and roll respectively) rotation angle of a head is tasking and necessary for this project. Since the face of a person is a 3D object, it can rotate over all three axis - but with some limitations. In a head pose estimation problem, the movements are called the roll, pitch, and yaw.

   The plan is to use deep learning to estimate the head pose. Arnaldo Gualberto, a deep learning engineer, proposed a method for applying deep learning to find the three rotations [21]. This method used a dataset which was created by him. It has 6,288 images from various face datasets. He then applied the dlib facial landmark on them which give 68x2 matrix and computes the pair the Euclidean distance between all points to get 2,278 features (68*67)/2. He also used Keras (open-source neural-network library written in Python) to aid in training his model. His model architecture includes strong regularisation (prevent over-fitting and to avoid the curse of dimensionality by ensuring that all layers have L2-regularisation: forces the weights to be small but does not make them zero, therefore giving higher accuracy of prediction when the output variable is a function of all input features [22]). It also has no dropout (not dropping out units/nodes in a neuron in the neural network).

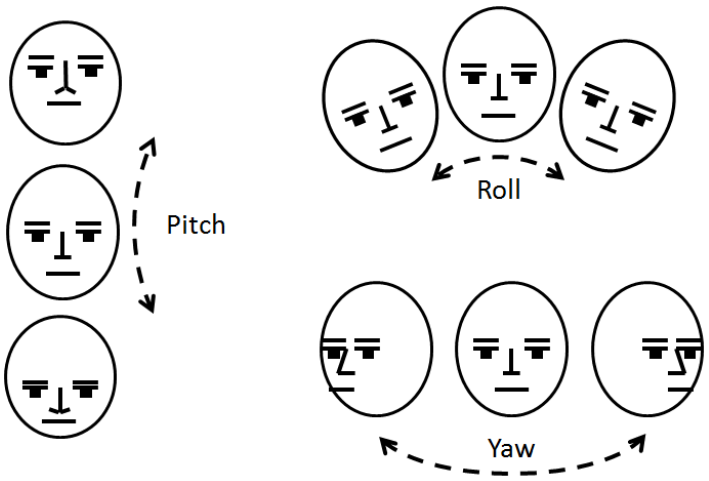

Figure 3: This is a visual representation of the three main head poses. This image is taken from [21]

3.1.2 Non-functional requirement

1. Reliability: The system will run correctly every time needed. Failure (per unit time) will not be acceptable it should work at least 90% of the time [23].

2. Security: Test subject image will not be released without consent.

3. Maintainability: My code will be well documented to allow a third party understand the implementation of my code. I will also use good naming conventions to increase the readability of my code for future improvement.

4. Availability: The system will be set up so that possibility of the system crashing will be minimal (ideally zero). My whole system will be recovered at a reasonable time.

5. Portability: The Python system shall be able to run on multiple operating systems.

3.2 Update on Initial project specification

Section 3.1 mentions the main goals and ways to achieve them, but over the implementation of the system, there have been some changes. Initially, the plan was to use images or image sequences as the data being processed and report if the image showed a drowsy test subject, but now it is using webcam feeds which give live feedback when the test subject shows any sign of tiredness. Furthermore, I added a menu system where the user could choose what they want to detect or measure.

3.3 Drowsiness detection system workflow

The workflow expresses what the layout of the system is and how it interacts with one another to function. Figure 4 reflects the way that the system will work and interact with other modules that make up the entire system. Arrows introduced in figure 4 show the flow of data or action. After initialisation, the live video stream thread with the camera is opened and active. This can then capture frames from the video stream and begins to load up the facial landmark indexes. With this facial landmark, extraction is carried out on the face in the live video stream. Following this, the facial landmark points (which are a 68x2 matrix of x and y coordinates) are displayed on the human face show on the live video stream. This can now be used to measure different region on the test subject's face. This would include the eye region (detect eye closure using the EAR method), the mouth region (detect yawning) and the entire face to measure head pose. To conclude the detection process, if any signs of drowsiness are detected then an audio and visual warning while the result.

3.4 System Input

The system takes webcam feeds (live feeds) and images to operate on, depending on the users choice in the menu as to what they want to detect using the system. There is also a requirement for the system to run like the 68 landmark algorithm built in which requires a pre-built shape predictor to execute the logic. Other packages are necessary to run this system like OpenCV and Imutils which helps to predict the face shape with the input data given to the system. Another package required would be **playsound** which enable the warning alarm played to gain the attention of the test subject.

3.5 Pre-process

The pre-processing stage takes place just after the image or live stream acquisition process and before the image processing stage. The first thing the system will do is start up the video stream thread (webcam) or ask for the file name of the image to ensure that there is an input data to operate on. Next is to initialise dlib's face detector (HOG-based) and then create the facial landmark predictor. This may take a bit of time, so the idea is to let the users know that the webcam and face detector algorithm is loading rather than them looking at a blank screen without knowing what is happening.

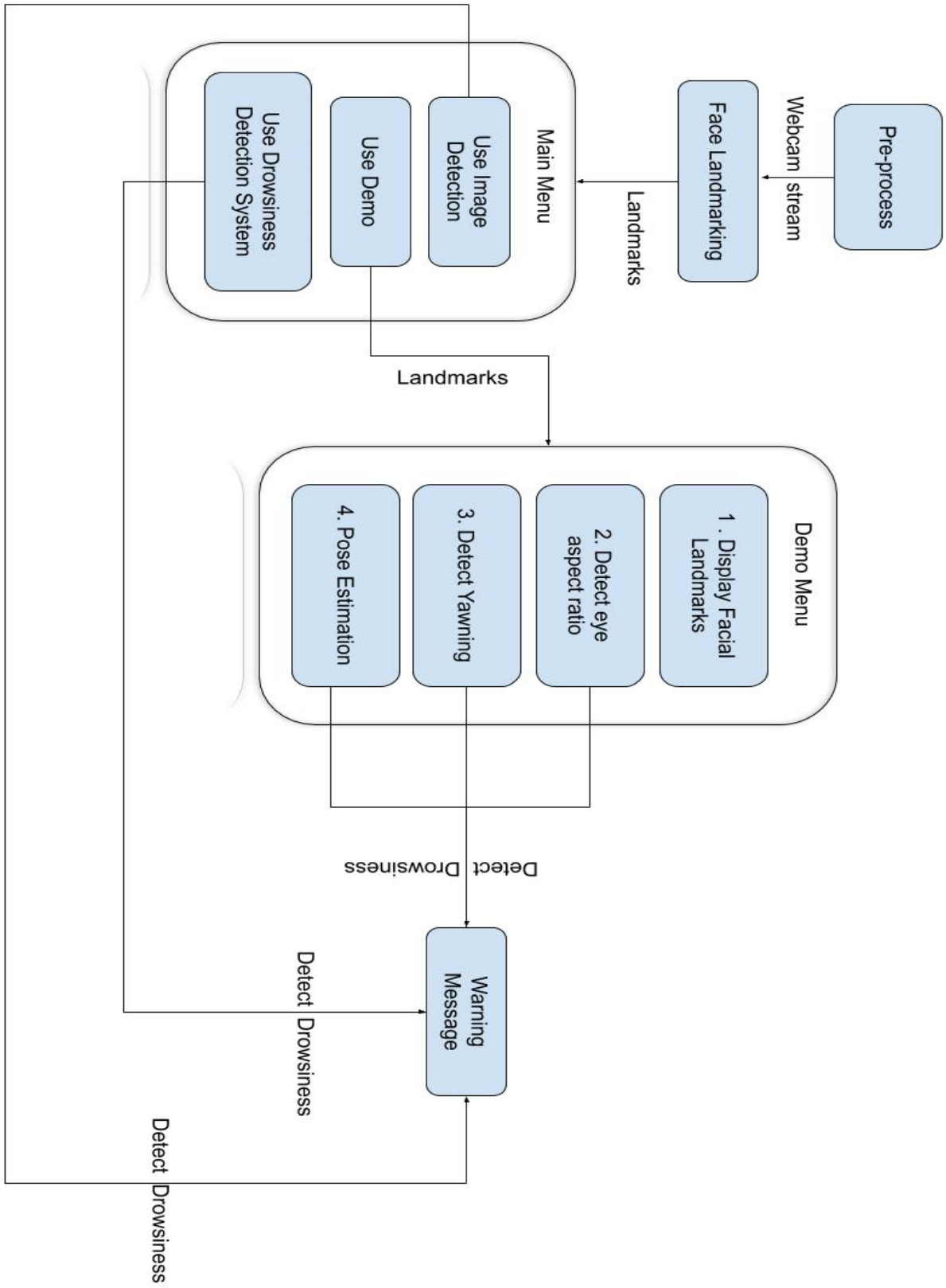

Figure 4: High level workflow of the module of the drowsiness detection system.

3.6 Facial Landmark

The most important part of my project is detecting the face and the facial properties, like the location of the nose, mouth and eyes to measure the changes that occur to them while drowsy to warn the driver. To achieve these landmark points are essential as the system requires the points on these facial features to extract information about specific parts of measurements. According to Adrian Rosebrock, facial landmark detection and extraction is a subset of the shape prediction problem where an attempt to localise critical points of interest (which in this case is essential facial features on a human face) along a given image or shape [24]. My system will need to use a landmark extraction algorithm for the live feed from the webcam footage. This extraction is the foundation on which all of the measurement is done like the eye aspect ratio. It was recommended that I got robust and already existing facial algorithms that are in the structure specified by IBUG benchmark [18]. With this in mind, I chose Dlib library which was based on Kazemi and Sulivan's work. This landmark detector has already been trained to pinpoint 68 specific points on a human face. The use of a pre-trained facial landmark detector is needed preferably the iBuG 300-W dataset 68 point annotation (it has 68 x-y coordinates that maps the facial structure on a face) which the dlib facial landmark is trained on because it is the most commonly used. Another choice would be the 194 point annotation trained on HELEN dataset. In context with my project, the dlib face landmark was an excellent tool as it provided accuracy and speed (for real-time application) and made it possible to observe specific parts of the human face I wanted to measure. For more information on why I chose this library, please refer to section 4.

3.7 Eye Closure Detection

This module aims to detect eye closure over a period to ensure the test subject isn't just blinking but have their eyes closed. The input of this module is the output of section 3.5 - the facial landmarks which contain a 68x2 matrix, i.e. x and y coordinates of the points in the human face as seen in figure 2. I used this to extract the eye regions to compute the eye aspect ratio to decide if the eye is closed (This was introduced in section 2.3.1). To obtain the eyes region I decided to use the available facial landmark indexes for face regions. Dlib has put in place already been encoded inside a dictionary to ensure access to the different areas on the face, i.e. the left and right eyes can be accessed through points (42,48),(36,42) respectively. This dictionary is called "FACIAL_LANDMARKS_IDXS". It can be used to extract the indexes into the facial landmarks array easily and extract various facial features simply by supplying a string as a key - right_eye", "left_eye".

3.8 Yawning Detection

This module aims to recognise when the test subject is yawning compared to when they are talking or smiling. To achieve this the output of section 3.5 is used as the input - the facial landmarks which contains a 68x2 matrix, i.e. x and y coordinates of the points in the human face as seen in figure 2. Like section 3.6 I used the built-in dlib dictionary mentioned in section 3.7 to access the facial landmark indexes for the mouth, i.e. (48,68). I used these indexes to find four points in the mouth for the top and bottom lip. The idea is to use the top and bottom lip to find the distance between them. The higher the distance, the more likely it is classified as yawning. To ensure that the test subject is yawning I made a threshold value, and if the gap between the lips is greater than or equal to this value, then it is classified as yawning.

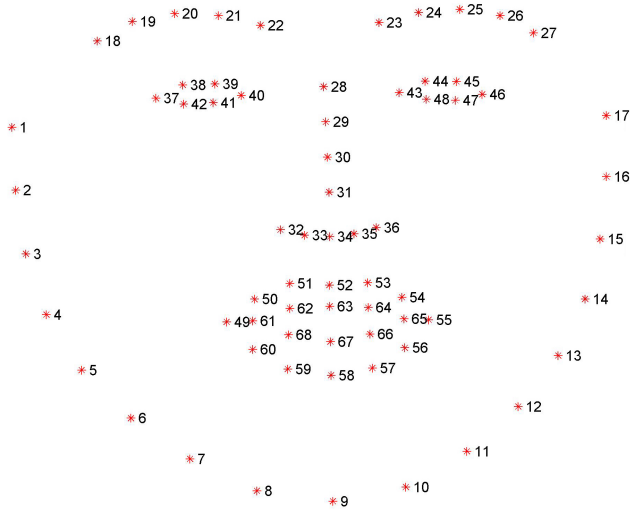

Figure 5: The 68 facial landmark coordinates from the iBUG 300-W dataset. Image taken from [25]

3.9 Pose Estimation

This module aims to estimate the pose by calculating the relative position of the test subject's face and the camera. I have made a change to this module form the original design where I proposed using the deep learning to estimate at the head pose. The main reason for doing this was because the time it took to compute the pitch, yaw and roll for each frame was a lot because it had to compute a pairwise Euclidean distance between all 68 points (making an additional 2,210 features points), for more information see section 2.3.1. I then decided that the best idea was to use the built-in OpenCV ***solvePnP()*** function to aid in finding the Euler angle. The Euler angles are three angles introduced by Leonhard Euler to describe the orientation of an object concerning a fixed coordinate system [26]. SolvePnP finds an object (in this case a human face) pose from a 3D-2D point correspondences [27]. This function is a subset of the Perspective-n-Point (PnP) problem were given a set of n 3D points in the world, and their corresponding 2D projections in the image(in this case live video stream) produce the pose of a calibrated camera [28]. This function achieves this given a set of object points, their corresponding image projections, the camera matrix and the distortion coefficients [27]. This returns the rotation vector (which is converted into a rotation matrix using built-in OpenCV function - Rodrigues) and translation vector combined to form the pose matrix. To get the Euler angle I used the ***decomposeProjectionMatrix()*** function in OpenCV. It takes the pose matrix as an argument and returns the Euler angle. The X, Y and Z coordinate of the Euler angle represents the pitch, yaw and roll respectively.

3.10 Warning Message

This module aims to produce an appropriate visual and audio message to let the test subject know about any symptom of drowsiness detected. Warnings are meant to alert the test subject of an imminent risk to their safety. Whenever a notification is used, the test subject is at risk of distracted so to ensure that it is efficient the warning message as to be concise and accurate. If too long, overly technical, repetitive, inaccurate, or ambiguous, the warning will be discarded, and its purpose will be lost. Offer meaningful options: Warnings should present clear choices

like "Warning! Eye closure Detected. Please be on alert!". The audio I used were short, but semi-loud attention-grabbing siren or messages, not long musical melodies. The reason for sound is to stimulate the driver's sense of hearing for warning purpose of assisting drivers for safe driving.

4 System Development

This section gives more detailed information on how I made my system. There is a mention of development and research methodologies recommended by my already existing similar projects. There is a comprehensive view of how I implemented my design mentioned in section 3 and all the improvements along the way and the various reasons for them.

4.1 Research Methodology

While the development of this system was going on, all the areas in the project that required research to guide me to what the next best method or perspective for a fast and robust drowsiness detection system are detailed in my literature review. I researched commonly used eye detection method for eye closure, yawning and pose detection which was briefly mentioned in section 2.3.1. The research had been used as inspiration and also in some cases already made it easy to use a method that shaped up what my project looks like now. Some of the research material and guidance were provided by my supervisor to help ease through the different stages of system development as I have no background experience or knowledge of the topic. The other research materials used in my project was a stack-overflow platform, digital libraries like IEEE Xplore, ACN Online, Sensors - open access journal, and so on. These were places I got information on the different ways to measure drowsiness in drivers and what the best and most reproducible methods, face detection, PERCLOS, EAR, and so on. The research methodology used for this project cannot be specified because my project is less research-based and more development based. The only form of research I did was for the literature review and methods to better serve different functionality in my system that has been proven to be robust and fast. This was also because I was unable to do questionnaires or interviews, but I was able to look at different methods and see which was the best or the time I was given and also the scope I was meant to cover for a successful system.

4.2 Development Methodology

Development is the major part of my project because I am required to make a system which detects drowsiness and distraction and sends a warning message. There have been multiple methods for ensuring timely progress throughout the project to develop this system. The most commonly recommended method of development was an agile scrum. Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organising cross-functional teams. In my project, I worked closely with my supervisor to achieve a complete system. Agile methods generally promote a disciplined project management process. That encourages frequent inspection and adaptation, self-organisation and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals (in this case instead of customers you have a supervisor and project specification instead of company

goals). Scrum is a subset of Agile, which is a lightweight process framework for agile development [29]. Scrum uses sprint which is a development cycle - where work is divided over a set time - generally about an average of two weeks long for this project.

My development methodology for the project was agile development. Like I mentioned earlier there were an average of two weeks sprints for my project which I have described in details in Table 1. Furthermore, for the sprints, I had scrum meetings with my supervisor every week to discuss what progress I have made as well as directions on what to do and how to improve my current approach. For each sprint cycle, I did sprint planning, weekly scrum and sprint review with my supervisor. The main difference between my scrum development and a typical agile scrum development is that I was the only developer as it is an individual project. So instead of a team where there are scrum master (the facilitator for the agile development team) and developers, I was doing all of that during the development of this project and my supervisor was the scrum product owner. Additionally, this project's complexity was mostly based on the mathematical orientation for measurement and other back end parts of the project or understanding the use of python libraries like OpenCV, dlib and so on. Due to these facts, I would like to say that the majority of the sprint cycle was taken to understand and apply that understanding to the project.

Start Date	**Sprint Objective**	**Completed?**	**Issues**	**Notes**
25-09-2018	Learn what my project style is i.e. software engineering or research. Select a topic for and supervisor for my project.	Yes, I selected Drowsiness and distraction detection system project and my supervisor is Dr Luo Chunbo.		I learned about the meeting time and the different deadlines for different terms in the academic year.
02-10-2018	Get preparing literature sources for the Project statement and plan deliverable.	Yes, I got a major journal from my supervisor that will direct my project statement as well as plan.	Found it difficult to know what part of the recommended book I was meant to take inspiration from for my project statement.	Since there was a draft of this deliverable from the previous year, I was able to know roughly what the end product will look like.
17-10-2018	Finish the Project statement and plan. In addition to that, I want to get a better Idea of the task set for me.	Yes, I was able to finish my work and read up on the recommended book for better understanding on what to do. I also briefly used a Gantt chart to illustrate the time planning for the remainder of the year.		The Project statement was just 4 pages but took more time than I expected. I will be good idea to start earlier to reduce rushing my work.

Start Date	Sprint Objective	Completed?	Issues	Notes
27-10-2018	Review the lecture slides on Individual Literature review and find new related literature for my project and analyse them.	Yes, I have a significant number of references to help with my Literature review deliverable.	Not entirely sure of the reference and quotation mark part of the lectures.	
21-11-2018	Finish my Literature Review and start looking at how to implement my work.	Yes, I have finished and handed in my work.		It was my first summative essay and it was tasking.
25-11-2018	Install Conda and crate an environment to add all the packages need for my system. Also decide what programming language to use.	Yes, I have installed Conda and created an environment called project where have install OpenCV for my system so far.		I have also decided to use python 3 because there are support available for development in this language.
07-12-2018	Take a video of myself carrying out drowsy behaviours for my system to detect.	Yes, I was able to make a 2 minutes mp3 format video that show me carrying out drowsy behaviours for my system to detect.		
09-12-2018	Convert the video into image sequences.	Yes, I wrote a python code to convert mp3 video in to a list of images at a set number of frames.		
21-12-2018	Start my code for my system. Looked at detecting drowsiness in an image. Write a code to accept an image as the argument and write a code to allow a video to be passed as an argument.	Yes, I was able to successfully get both image and video to pass as argument for my drowsiness detection system.	Just getting used to the new ideas of OpenCV and it built in functionality to make it happen.	

Start Date	Sprint Objective	Completed?	Issues	Notes
07-01-2019	Prepare for Oral presentation of my system and set future goals for them.	Yes, I made a 5 pages power point slide to aid in my presentation.		
15-01-2019		I made my presentation to my supervisor and other students.	My time management was poorly planned.	
16-01-2019	Started research on the use of Neural networks for drowsiness detection in python.	Yes, I discovered the deep learning python tool called Keras and Convolution Neural Network (CNN) and how it is used for image classification.	I was not finding it easy to understand as I learned about the epoch and loss values and other key terms which were new to me.	
23-01-2019	Implement the neural network method in images.	Yes, I successfully implemented the neural network method on an image.	I realised that I wanted to move from only images to live video stream detection.	
04-02-2019	Learn how/write a code to start up a live video feed using python and OpenCV.	Yes, I wrote python code to take the webcam as argument and run the video stream thread using OpenCV.		
11-02-2019	Write code to implement the neural network for face recognition.	I managed to make it work with CNN.	It was complex and extremely slow and defeated the purpose of real time response.	After a long time of research, I found a library called face_alignment which can find 2D or 3D facial landmark of a human face.
27-02-2019	Write code to implement the face_alignment library which I found on the last sprint.	Yes, I was able to do it and make it work.	It was also slow and but accurate and efficient.	I found the dlib 68 shape predictor and also how to use it for facial recognition and landmark extraction

Start Date	Sprint Objective	Completed?	Issues	Notes
04-03-2019	Write a python code that uses dlib for facial land mark and try to implement the eye closure measurement.	Yes, I wrote a system to detect eye closure using EAR in real time.		
11-03-2019	Write a python code that implements the yawning detection.	I made a python system to measure yawning by finding the distance between the top and bottom lips.		I was still looking for a better way of getting facial landmark extraction because dlib is limited to frontal face and not extreme rotation which is useful for head pose detection.
22-03-2019	Write a code for pose detection using the 4D face reconstruction code provided by Dr Tassoss to aid my project.	Yes, I accomplished this.	I was difficult because I had to change things in his code and use wxpython GUI. I was challenging.	I found a new way of detecting head pose using the dlib facial landmark format, which was called solvePnP.
01-04-2019	Write code to detect head pose in python using the method I found in the last sprint.			
08-04-2019	Start my Implement section of the dissertation while testing my system and reviewing my code.	I have started my dissertation.	25-30 page seems like a lot of page to write and I dont know if I have a lot to say in this dissertation.	
18-04-2019	Carry out unit testing, and other technical testing to ensure that the standards of my system is good.		I am not sure if there are any standardised test to check against false positives and false negatives which are important for my system.	

Table 1: My Sprint notes over the course of the development of my system.

4.3 Design Implementation

This section will highlight in details technical implementations of each module of the design in section 3. The is details information on the external libraries used as well as a component-level design process that follows a step-by-step improvement process choices to implement the system's logic.

The program workflow is similar to what has already been proposed in figure 4. There will be actual logic on how they are implemented for each component from the pre-process to the warning message. In a nutshell this is the technical overview of the system; Initially the pre-processing happens only on the system start-up. The next step is for a loop that includes face landmark extraction, detecting eye closure, detecting yawning and then sending a warning message. These modules are in separate files and a central controller file called ***main.py***. This is where all of these modules are combined to function together. This main controller has a specific role which is to help manage the logistics of how different module comes together and manages the processing and sending data between modules. From a code point of view, the pre-processing stage is done in the ***main.py*** file.

4.3.1 Pre-process

There is not much work to be done in pre-processing only to set the webcam using the argument parser to let the computer system know to get the built-in webcam ready. For this to be possible, you have to set up the parser object and tell it to expect the webcam argument. The parser can then be used to process the command-line arguments when the program runs, but in my case, I ensured that the I defined the webcam argument was a default option - so that the user does not have to enter a verbose command on the command line.

The code is encapsulated in the main controller file under the function name ***loading()***. It has a print statement after each set up (webcam and facial landmark predictor) letting the user know what is happening. This function returned the face detector and predictor as well as the video stream thread.

4.3.2 Face Landmark

In the original specification, I mentioned the chosen face landmark and also in section 3.5 I vaguely wrote about the dlib library in use for frontal face recognition, but I did not mention the algorithm I chose. First of all, dlib is a C++ library, and many tools can be used from python applications. It contains a machine learning algorithm and other tools for solving complex real-life problems. The algorithm used to detect faces is Histograms of Gradients (HOG) plus a Linear Support Vector Machine (LinearSVM). Subsequently, dlib also has a Neural Network implementation. It performs better than HOG to detect non-frontal faces which would be useful for observing and measuring head pose but unfortunately it is computationally heavy which will not work for real-time use. To avoid accuracy for run-time efficiency, I used HOG. HOG algorithm iterates over every pixel of a given image - in this case, each frame in the live video - and checks around it to find out which pixel is darker compared to its surrounding pixels. This aids in its rendering of arrows to suggest the direction getting darker. This process is repeated for every available pixel to break down how many gradients point to each significant direction. Then using many faces, I could have used the LinearSVM model to train my systems for detecting faces in an image - frame in a live video stream - but I used the Dlib pre-trained model using HOG through the get_frontal_face_detector method. The class shape_predictor was

also used because it is a tool that takes in an image region containing some object and outputs a set of point locations that define the pose of the object. An example of this is human face prediction, you take an image of a human face as input, and the expected output would be to identify the locations of important facial landmarks such as the corners of the mouth and eyes, the tip of the nose, etc. [30]. It takes the "shape_predictor_68_face_landmark.dat" file as an argument to aid in predicting face landmarks.

Figure 6: This show the dlib 68 points facial landmark used in the system.

This part of the system is simple to use because it integrates already existing solution. I used the OpenCV library to access the webcam on my laptop and get the current video frame which is then converted into a matrix of corresponding pixels in the image. This is then sent as an input to the dlib face landmark extractor as I mentioned above and turned to a 68x2 matrix if a face has been found in the image. However, this extraction will not work if the face was rotated at an extreme angle or the was partial obstruction of the face because of the facial extraction heavily focuses on computational efficiency and this because it only recognises a frontal face and not an extreme face rotation.

The program logic for this module (in a code perspective) is kept in the ***facial_landmarks.py*** file and is called in the central controller python file to show the landmarks and send the 68x2 matrix to other modules to use as input for computation.

4.3.3 Eye Closure Detection

As I mentioned in the section 4.3.2 where I applied facial landmarks to localise regions of interest like the eye, mouth, jaw, nose and so on. This means that I can use get specified facial structures by their indexes as mentioned in section 3.6. Let us focus on the eye region

of the face and its corresponding indexes. 6 x and y coordinates represent each eye, typically starting from the left corner of the eye, into a clockwise direction around the outline of the eye. The equation used to measure real-time eye blinking, and eye closure is called Eye Aspect Ratio(EAR):

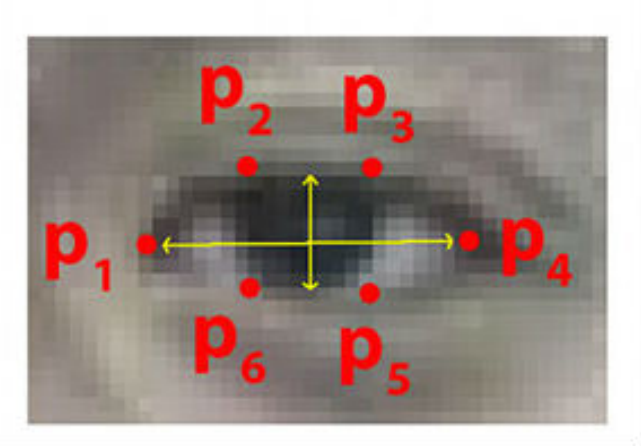

Figure 7: The annotated points on the eyes. [24]

$$EAR = \frac{\|P_2 - P_6\| + \|P_3 - P_5\|}{2 \times \|P_1 - P_4\|} \tag{1}$$

1. *EAR Formula* [24], where p1,p2,p3,p4,p5 and p6 are 2D facial landmark locations.

This equation computes the Euclidean distance between the top and bottom eyelashes. The numerator and denominator play their part in this equation where - according to Adrian Rosebrock - it calculates the gap between the vertical eye landmarks and computes the distance between horizontal eye landmarks respectively. As visual annotated in Figure 3 p1 and p4 make up the horizontal eye landmark whereas (P2, P6) and (P3, P5), make up the vertical eye landmark. An important thing to note about this equation is that the EAR value is approximately constant when the eye is open but will fall instantaneously when the eyes are closed.

Soukupov and ech introduced the EAR equation in 2016 which is unlike any other image processing method for detecting eye closure. Conventional methods for detecting eye closure would generally involve a combination of eye localisation, a threshold to find the whites of the eyes and to decide whether the white in the eyes disappear for some time. As you can tell, the difference in ease, speed and efficiency the EAR equation is compared to the standard methods used. With this equation I have avoided image processing techniques which takes time and rely on the ratio of eye landmark distances, this saves time and allows for real-time response.

From a code perspective I have written two python functions; ***eye_aspect_ratio()*** and ***main()***. It is written in a python file called **eye_closure.py**. The ***eye_aspect_ratio()*** function is where the equation is expressed and computation to find the EAR and return the EAR value. The ***main()*** function opens the live video stream and then applies the ***eye_aspect_ratio()*** function on each frame to calculate the EAR for the current frame. So, to highlight the steps for eye closure detection;

- setup all pre-processes needed for this to work i.e. a camera that monitors a stream for faces,

Figure 8: This show the head pose detection at work in real time.

- The system applies a facial landmark detection and extract the eye regions when a face is detected.
- Since the eye regions has been extracted, the eye aspect ratio can be computed in order to decide if the eyes are closed.
- If the EAR shows the eyes closed for long enough period of time, an alarm to wake up the driver is played as well as a text message.

4.3.4 Yawning Detection

In section 3.8 I briefly discuss my method for detecting yawning in a test subject. The method involves the top and bottom lips, this because they need to find the vertical distance between them. The method of finding this is quite similar to the EAR formula; however, just the numerator of the equation is applicable. It will be used to find the Euclidean distance from point 52 and 58 in figure 5. The function ***lipDistance()*** is where this calculation is carried out and the result is returned from this function. The ***main()*** function uses this value to calculate the lip distance for every frame, and when the criteria are met(if the lip distance is greater than or equal to the threshold distance), it will alert the test subject that they are yawning.

4.3.5 Pose Estimation

In section 3.9 I highlight the change in design for detecting head pose where I introduced the idea of using the built-in OpenCV ***solvePnP()*** function. This module takes the output of

Figure 9: This show the dlib 68 points facial landmark used in the system.

section 4.3.2. This subsection is going to show the complete implementation of the algorithm. This module takes in 2D extracted landmark of the test subject's face and makes a 3D coordinates to represent head pose (pitch, yaw and roll). To achieve this Camera Calibration and 3D Reconstruction will be needed. This is also called the *pinhole camera model*. In this model, a scene view is created by displaying 3D points into an image plane using *perspective transformation*.

$$s\, m' = A[R|t]M'$$

$$or$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where:

- (X, Y, Z) are the coordinates of a 3D point in the world coordinate space.
- (u, v) are the coordinates of the projection point in pixels.
- A is a camera matrix.
- (cx, cy) is a principal point at the image center.
- fx, fy are the focal lengths in pixel units [31].

- $[R|t]$ is called the rotation-translation matrix [31].

The joint transformation matrix $[R|t]$ is used to describe the camera motion around a static scene/object, or a static camera around a moving scene/object. That is, $[R|t]$ (in equation 2) translates coordinates of a point (x, y, z) to a coordinate system (a method for identifying the location of a point on the camera scene), fixed with respect to the camera [31].

There are three main steps:

- Face detection. A face detector is adopted to provide a face box containing a human face. Then the face box is expanded and transformed to a square to suit the needs of later steps. Reference Figure 9.
- Facial landmark detection. The output from section 4.3.2 output 68 facial landmarks.
- Pose estimation. Once we got the 68 facial landmarks, the solvePnP algorithm is used to compute the pose.

Figure 10: This show the head pose detection at work in real time.

The pose is detected frame by frame, which results in small variance between adjacent frames. It is not that noticeable in the system. From a code point of view, it is written in a python file called **head_pose.py** also, has a ***main()*** function as the others and this is where the Euler angle (which contains the pitch, yaw and roll) is computed.

4.3.6 Warning Message

After the monitoring the test subject for symptoms of drowsiness, if there are signs of either yawning, eye closure or distraction from the driver, i.e. yaw in head pose highlight that the test subject is looking left, or right then a warning message will be displayed on the screen as well as a short audio message alerting the test subject of what the problem is.

- Text Warning Message. In the system, the warning message is not verbose and straight to the point, for example for eye closure it is "Eyes closed alert". The warning message is located at the top left to get the test's subjects attention. It only appears when drowsiness is detected over a set period but disappears when the test subject stops shows casing drowsy symptoms.

- Audio Warning Message. This system also has an audio warning message, so when it is used in a driving environment it will not throw off the driver but calmly let the driver know about the issue. The message is short and straight to the point, and it will not play unless drowsiness is detected. It is not a siren but a message that is played.

From a code perspective the package **playsound** is imported for the sounds to work in all files that measure different drowsiness symptoms. The **playsound** module is a cross-platform module that can play audio files like wav or mp3 files [32].

4.4 System Improvement

The main improvement I made in my system was to move the design from only image drowsiness detection to real-time video and image drowsiness analysis and warning. Like I said earlier, the initial plan was to incorporate a neural network for facial recognition and head pose computation on image-based inputs. However, because I prioritised real-time video analysis, I had to use methods best suited for that and also apply that test images.

When I applied the deep learning method to the live video stream to determine head pose, I worked and displayed the pitch, yaw and roll, but I had two major problems.

1. The dlib 68 facial landmark points and deep learning implementation of head pose computation. This combination works but has its limitation (It does not work correctly when head rotation is extreme, i.e. looking left or right). This was a problem because when the test subject turns their head left/right the point in the jaw are missing because either the nose is obstructing the view or it is gone out of the camera. The dlib 68 facial landmark points can only work if all 68 spots are seen to plot each point on to the human face. Therefore because the side of the jaw has disappeared it no longer consider the object on the screen as a human face. Besides, dlib only recognises frontal human faces, and head rotation is not supported. A way I look to solve it was to us the dlib five facial landmark points. The dlib 5-points facial landmark detector only has two points on the left and right eye and one point for the nose. The most appropriate use for it is to attempt to obtain a facial alignment based on rotation, translation and scale. However, this did not work because as mentioned in section 2.3.1 the deep learning approach I planned to use needed 68-points to make more features or nodes in the neural network to make prediction more accurate.

2. The frame by frame processing speed. Even though it works, it is extremely slow. It freezes each frame and then applies the deep learning method to get the Euler angle. The reason could be because there is a method called *compute_feature* requires that to be the 68x2 matrix containing face points is provide so that it could return an array containing the calculated pairwise Euclidean distance (the straight-line distance between two points in a metric space) between all 68 locations. This requires two for loops that iterate over 68 points 67 times and computes the pairwise Euclidean distance. Then I look at the trained model and uses it as a standard to measure pose in the frame. This also takes time as the trained model has to be loaded and compute the relative pose for the frame, all of this in less than a second for each frame in the live video is hard to accomplish except with the aid of a more powerful processor compared to the one available on my personal computer.

Another improvement to my system would be as a result of testing the separate modules. I noticed that there were a lot of false negatives when the distance of the test subject's head and then the camera is long. The EAR value reduces when the driver is far away from the camera compared to being closer. This was a significant problem that could disrupt the driver when in a real driving environment. To reduce the effect of distance, I decided to find the size of the person's face, i.e. the Euclidean distance between the eyebrow and chin. The idea is that the ratio of the face size and the eye will always be the same closeup or far away from the camera. Therefore I used that ratio and then divided by the initial EAR value to get the new EAR value in proportion to the face size.

5 System Testing

Since this is a software development project, my system is approximately 665 lines of code. To perform unit testing, I have tested each method separately with white box testing before they are integrated into the central system. This was achieved with the aid of many print statements where the output is compared to the desired output, if they are the same, then it works, and I move on else find out what and where the problem is coming from using more print statements. Additionally, there also have integration testing to make the modules come together into one system. Outside this, Ad-Hoc (testing with no set-out plan or documentation) was also used to test methods interaction between themselves as well as at the end of each sprint. There has also been a test to check the performance (false negatives and false positives). More details about the test and the results are mentioned in section 7.1.

6 System Evaluation

6.1 System Behaviour

In this section, a verbose description of my system in use is written. The following happens for the video version of the system called the ***drowsiness_detection.py***:

1. The user runs go to the correct directory where the python system is.

2. The user activates a conda environment called *project* which has all the basic packages that allow the system to run properly.

3. The user runs the system.

4. Two messages notify the user what is happening in the background - this roughly takes 3-5 seconds to load.

5. The camera opens, and the 68-point facial landmark is plotted on the users face.

6. After this, the system detects and warn users of any drowsiness symptom.

7. The user can quit the system by pressing 'q' when this happens the camera closes.

Next is the flow of events the occurs in the demo version of my system called ***demo.py***. Note the first two steps of the main system is the same as the demo:

1. The user boots up the system.

2. A menu option is displayed where **1** is to display the facial landmark, **2** is to detect eye closure, **3** is to detect yawning, **4** is to detect head pose and 'x' is to exit the system.

3. After the user picks an option; two messages notify the user what is happening in the background - this roughly takes 3-5 seconds to load.

4. The camera opens, and the 68-point facial landmark is plotted on the users face.

5. After this, the system detects and warn users of any drowsiness symptom.

6. The user can quit the system by pressing 'q' when the menu is shown for the user to choose other options. I the user want to exit the menu then they need to press 'x', upon which the camera is closed, and the menu is cleared.

Another flow of events the occurs in the image version of my system called ***image_main.py***:

1. The user boots up the system.

2. The user is prompted to enter the file name of the image.

3. The user runs the system.

4. Two messages notify the user what is happening in the background - this roughly takes 3-5 seconds to load.

5. The camera opens, and the 68-point facial landmark is plotted on the users face.

6. After this, the system detects and warn users of any drowsiness symptom.

7. The user can close the system by selecting the x on the top right of the frame when this happens, the camera closes.

6.2 Test Description

6.2.1 Ground Truth Data

In this section, I will be talking about a functional test that will be carried out to find out the accuracy of my system. The system's accuracy can only be determined by the rate of false negatives and false positive the system gives. A false negative is a test result that indicates a person does not display any symptom of drowsiness when the person does whereas false positive is a test result which wrongly indicates that the person is sleepy or show any signs of drowsiness when they do not. I decided to test my system with these in mind because they are crucial. For example if when using this system when driving and the system is constantly warning the driver that they are drowsy when they are not then that can cause distraction which could lead to an accident(which the system is trying to prevent) or if the driver is showing drowsiness symptoms and the system is not picking it up and warning the driver, they could eventual sleep while driving and cause an accident (which is what we are trying to prevent). To judge the accuracy of my system, I decided to directly observe the system at work and let other people use the system rather than asking them to use the system without any direction and get their feedback. This method seemed to be a good choice as it is an objective assessment method that is used a lot in the computer vision field. Therefore the system accuracy is evaluated by using boolean values (True or False); these values are determined from observation. There are two main test branches; the image testing and the live video test as input. For the image testing, after I enter the input image, I note what the expected result should be, and then I observe the actual result. I carried this test out for 30 images; you can see the result in section 6.3.1.

6.3 Test Result

The following result is from testing my chosen methods on both images and live video. They both give a clear indication of the accuracy and what lighting does to that accuracy in the system.

6.3.1 Unit Testing and User Testing

1. **The image test**: As I previously mentioned in section 6.2.1, I have used 30 images to test a segment of my system. I created a table for each module, i.e. EAR, Yawning and Head Pose. Basic information to understand the test is that the expected and actual result is based on the outcome of the system after it runs on the input image, i.e. Does it detected eye closure, yawning or a specific head pose - Yes or No. In my test, I found out that some of the images used displayed more than one symptom of drowsiness and they were all detected. From my observation, i.e. the result from the tables I was able to calculate the false negative and positive percentage, and with this, I was able to calculate the accuracy rate on images for each method. The overall a percentage for false negative was 8.65% whereas for the false positive it was 1.92%. Also, the accuracy for all the methods was as follows;

   EAR - 92.3% compared to PERCLOS's 98%, Yawning - 80.2%, Head Pose - 88.5%. Figure 11 show proof of the test.

   To find the accuracy, I counted the number of correct *actual result* compared to the *expected result* so then found the overall percentage of that comparison.

Figure 11: This is a demonstration of each method applied to images. The first row is for yawning detection, the second row is for eye closure detection, and the third is for head pose detection. Notice I removed the 3d box in the head pose.

EAR TEST										
Images	1	2	3	4	5	6	7	8	9	10
Expected Result	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Actual Result	Y	Y	Y	N	Y	Y	Y	Y	Y	N

Table 2: The followings is a sample result from the 30 tests carried out.

Head Pose TEST										
Images	1	2	3	4	5	6	7	8	9	10
Expected Result	N	N	N	N	N	Y	Y	Y	Y	Y
Actual Result	N	Y	N	N	N	Y	Y	N	Y	Y

Table 3: This is the first 10 results from 30 of the head pose/distraction test.

Yawning TEST										
Images	1	2	3	4	5	6	7	8	9	10
Expected Result	N	Y	N	Y	N	Y	Y	Y	Y	Y
Actual Result	N	N	N	N	N	Y	Y	Y	N	N

Table 4: This is the first 10 results from 30 of the yawning test.

2. **The video test**: For this test, I made python files that encapsulate the different method of detecting drowsiness and then run these methods on a live video. In this test I had five people test my system and give their feedback on the system as a whole. I assisted them with activities to carry out which will be presented below. Note; the following instructions are after starting up the system - when they are in the demo menu page. The test was divided into two groups; experiment with adequate lighting in the background and inadequate lighting.

   (a) Select the EAR option.

   (b) Close your eye for 10 seconds.

   (c) If there were an audio alarm and warning message to get your attention record yes, else no.

   (d) Press 'q' to exit back to the menu.

   (e) Select the Yawning detection option.

   (f) Open your mouth and Yawn.

   (g) If there were an audio alarm and warning message to get your attention record yes, else no.

   (h) Press 'q' to exit back to the menu.

   (i) Select the head pose option.

   (j) Turn your head around 30 to 40 degrees to the left then to the right(same angle) for 15 seconds.

   (k) If there were an audio alarm and warning message to get your attention record yes, else no.

   (l) Press 'q' to exit back to the menu.

   (m) Select 'exit' to exit the demo system.

   (n) Press 'x' to exit the system.

EAR TEST					
	Adequate Lighting			Inadequate Lighting	
	Test 1	Test 2		Test 1	Test 2
Person 1	Y	Y		Y	N
Person 2	Y	Y		Y	N
Person 3	Y	N		Y	Y
Person 4	Y	Y		N	N
Person 5	N	Y		Y	N

Table 5: This is the first 10 results from the eye closure test.

6.3.2 Integration Test

In this section, I combined all the methods for drowsiness detection into one file called ***main.py***. In this file, I managed to ensure that the system will identify only one of the three main drowsiness and distraction symptoms at a time and alert the test subject about it. The test

Yawn TEST					
	Adequate Lighting			Inadequate Lighting	
	Test 1	Test 2		Test 1	Test 2
Person 1	Y	Y		Y	N
Person 2	Y	Y		N	N
Person 3	Y	N		Y	Y
Person 4	Y	N		Y	N
Person 5	N	Y		N	N

Table 6: This is the first 10 results from the yawning test.

Head Pose TEST					
	Adequate Lighting			Inadequate Lighting	
	Test 1	Test 2		Test 1	Test 2
Person 1	Y	Y		Y	N
Person 2	Y	Y		N	N
Person 3	N	Y		Y	Y
Person 4	N	N		N	N
Person 5	Y	N		Y	N

Table 7: This is the first 10 results from the head pose test.

is, therefore, to ensure that if the user displayed any form of drowsiness it is picked up and a warning message is sent to the user. The accuracy is therefore determined by how consistent the error message is to the drowsiness symptom displayed and also the false negatives and positives. The accuracy for each method dropped massively, view the table below.

	EAR	Yawn	Head Pose
AL	80%	70%	60%
IL	50%	40%	40%

Table 8: This table show the effect of adequate lighting on my drowsiness detection system. Note; AL - Adequate Lighting, IL - Inadequate Lighting

The technical reason behind this fall inaccuracy is because the dlib facial landmark (as mentioned in earlier sections) is sensitive to light, partial occlusion and does not work well with extreme head rotation. Compared to the image accuracy it is lower because for each frames in the video the values of measurement i.e EAR, yaw and lip distance changes (sometimes the range in change is very notable), and also because the threshold value is compared with the measurement value of various consecutive frame over a period of time it is more likely to miss a sign of drowsiness compared to testing these methods against images which has just one frame.

7 Critical Assessment of the project

This section shows a detailed assessment of the approach taken to achieve this project and how successful the project has been.

The Drowsiness detection system should be considered a success from a technical point of view. It was able to pass the evaluation and specification test with minimum error in the real-time video, and it had better accuracy in with image tests. The main problem with the video testing was lighting (the system, i.e. facial landmark software is sensitive to poor illumination

[33]). As you can see in the result from the test, I found that there are less false negatives and positive when there is adequate lighting compared to when there is not (the system is more accurate) - see section 6.3.2. There are areas that the operation can be improved, with face alignment software the facial landmarks can be more precise on the face hence making the measurements more accurate. However, it used the neural network which takes time to process each frame and give back a response(face with facial points on them), which stops the system from being classified as responding in real-time. To speed the process of getting responses faster, a GPU can be used to process the frame and introduction of concurrency and parallelism could also be used. Although it was not a requirement for my project, this was a real drawback to the methods I choose and the implementation of my system.

Also, the project plan also requires evaluation. In section 3.2 I mentioned that I introduced real-time video as my additional input in order to detect drowsiness and give real-time response which took more time than I had initially anticipated. This was because I originally planned on using a neural network for estimating the yaw of the head pose but after implementation and testing. I realised from observation that the system was freezing in order to carry out the computation on the background which took quite a bit of time. This findings contradicts the ideas of having a real-time response, so I had to cancel the whole idea of a neural network and find a better method (solvePnP in OpenCV) which also took time to understand an implement. However considering that there was a bit of deviation from the original plan and time allocated to that plan, I would say it has been successful in terms of time management.

This project should be considered a success bearing in mind that I have no prior knowledge or experience with computer vision and facial detection. The whole concept was complex and quite challenging for me so I gained more information and expertise through research as well as trying out different methods to see their accuracy and how they can be used to achieve my specification which has helped to improve my research skills. After research had been completed I made a decision with support from my supervisor to start the implementation of my system using Python because I had previous experience with this programming language, their library put in place for effective use of computer vision in detecting drowsiness and most importantly it has not been overly used to make a similar system. My environment for development was Windows because it was the OS my personal computer was running on. It is commonly used OS and works well with scripted programming. This was easier for me when developing and testing new methods as I could implement them any time I wanted without having to book a machine to work with, this made my programming sprints in the development entirely stress-free and convenient. There are a lot of external software and packages that are needed for this program to work like the face landmark software, algebraic functions and visualisation software (OpenCV) and their specific versions to allow the system to run correctly. The novelty of my project is that I have a system that detects drowsiness from images and real-time videos - which is one of a few available ones.

8 Conclusion and Further Work

This project aimed to create a drowsiness and distraction detection and warning system that was efficient robust and accurate. I have talked about the possibilities available methods to achieve this (different methods for different parameters like eye, mouth, face, head pose), their advantages and disadvantages. I have also briefly introduced problem mainly involved with computer vision based drowsiness detection system and their possible solutions. Further, into

the report, I talk about the design of the system, where I highlighted what the initial design is and then what I changed and why. Following from this, I then mention about the methods I used to measure my main drowsiness symptoms (eyes closed, yawning, hanging head or looking away - left or right) and the way they are fit for my system. In addition to this, I talked about my research approach as well as my development methodology which paved the way for how I implemented every one of these methods and included the proof of them in use in the system. In the later course of this paper, I have detailed my testing and evaluation of my system and my project in general.

Potential future work is endless, mainly to do with the real-time video drowsiness detection system as there are many obstacles like illumination, occlusion and soon on. They will have viable solutions but due to the processing power of my personal computer, additional hardware needed and the time constraint on this project I was not able to go for those solutions which would have been more accurate and robust. Another way to improve this system is to make it Ubiquitous by creating an application in Android, and iOS that would render the system available to people to use when driving. Another improvement could be the integration of the system with Google maps to provide the net rest spot to recommend to the driver when drowsiness is detected to prevent road accidents.

References

[1] Tami Toroyan, Margaret Peden, and Kacem Iaych. Who launches second global status report on road safety. *Injury prevention : journal of the International Society for Child and Adolescent Injury Prevention*, 19:150, 04 2013.

[2] Kun Ju, Bok-Suk Shin, and Reinhard Klette. Novel backprojection method for monocular head pose estimation. *International Journal of Fuzzy Logic and Intelligent Systems*, 13(1):5058, 2013.

[3] B.C. Tefft. Prevalence of motor vehicle crashes involving drowsy drivers, u.s. 2009-2013. *AAA Foundation for Traffic Safety*, pages 1–8, 2014.

[4] Yangsheng Wang Xiaoxu Zhou, Xiangsheng Huang. Real-time facial expression recognition in the interactive game based on embedded hidden markov model. *Proceedings. International Conference on Computer Graphics, Imaging and Visualization, 2004. CGIV 2004*, 2004.

[5] Ferdinando Samaria and F. Fallside. Face identification and feature extraction using hidden markov models. 1993.

[6] Ara Nefian and Monson. Hayes. Hidden markov models for face recognition. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 1999.

[7] Ferdinando Silvestro Samaria. *Face recognition using hidden Markov models*. PhD thesis, University of Cambridge Cambridge, UK, 1994.

[8] Tapan Pradhan, Ashutosh Nandan Bagaria, and Aurobinda Routray. Measurement of perclos using eigen-eyes. *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, 2012.

[9] `https://ieeexplore.ieee.org/document/1212893/metrics#metrics`. Accessed on 27.05.2019.

[10] `https://ieeexplore.ieee.org/document/1398917`. Accessed on 04.05.2019.

[11] `https://ieeexplore.ieee.org/document/4272636`. Accessed on 01.05.2019.

[12] A. Narayanan, R. M. Kaimal, and Kamal Bijlani. Yaw estimation using cylindrical and ellipsoidal face models. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):23082320, 2014.

[13] Gabor-based dynamic representation for human fatigue monitoring in facial image sequences. `https://www.sciencedirect.com/science/article/pii/S0167865509002268?via=ihub`, journal=Pattern Recognition Letters, Sep 2009.

[14] Yasaman Anisi. Ali Sharifara, Mohd Shafry Mohd Rahim. A general review of human face detection including a study of neural networks and haar feature-based cascade classifier in face detection. *International Symposium on Biometrics and Security Technologies (ISBAST)*, 2014.

[15] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision - IJCV*, page 57, 2001.

[16] Jia Nan Wu Jianhua Li Cuimei, Qi Zhiliang. Human face detection algorithm via haar cascade classifier combined with three additional classifiers. *13th IEEE International Conference on Electronic Measurement and Instruments (ICEMI)*, 2017.

[17] Phillip Ian Wilson and John Fernandez. Facial feature detection using haar classifiers. *J. Comput. Sci. Coll.*, 21(4):127–133, April 2006.

[18] Facial point annotations. `https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/`.

[19] P. Rau R. Knipling. Perclos: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance. *Proceedings of the 2nd International Workshop on Education Technology and Computer Science; Wuhan, China.*, 31:1237–1252, 1998.

[20] T. Soukupova J. Cech. Real-time eye blink detection using facial landmarks. *21st Comput. Vis. Winter Work*, 2016.

[21] Arnaldo Gualberto and Arnaldo Gualberto. Real-time face pose estimation with deep learning. `https://medium.com/analytics-vidhya/face-pose-estimation-with-deep-learning-eebd0e62dbaf`, Sep 2018. Accessed on 10.05.2019.

[22] Renu Khandelwal and Renu Khandelwal. L1 l2 regularization. `https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2`, Nov 2018. Accessed on 10.05.2019.

[23] Jesus Nuevo Luis M. Bergasa, Member. Real-time system for monitoring driver vigilance. *IEEE Transactions on Intelligent Transportation Systems*, pages 63–77, 2006.

[24] Eye blink detection with opencv, python, and dlib. `https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/`, Apr 2017. Accessed on 03.05.2019.

[25] Drowsiness detection with opencv. `www.pyimagesearch.com`, May 2017. Accessed on 03.04.2019.

[26] Euler angles. `https://en.wikipedia.org/wiki/Euler_angles`, Apr 2019. Accessed on 10.05.2019.

[27] Camera calibration and 3d reconstruction. `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#boolsolvePnP(InputArrayobjectPoints,InputArrayimagePoints,InputArraycameraMatrix,InputArraydistCoeffs,OutputArrayrvec,OutputArraytvec,booluseExtrinsicGuess,intflags)`. Accessed on 05.05.2019.

[28] Perspective-n-point. `https://en.wikipedia.org/wiki/Perspective-n-Point`, Mar 2019. Accessed on 20.05.2019.

[29] Krista Trapani. What is agile? — what is scrum? — agile faq's. https://www.cprime.com/resources/what-is-agile-what-is-scrum/, Apr 2019. Accessed on 25.04.2019.

[30] Classes. http://dlib.net/python/index.html#dlib.shape_predictor. Accessed on 12.05.2019.

[31] Camera calibration and 3d reconstruction. https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. Accessed on 05.05.2019.

[32] Play sound in python. https://pythonbasics.org/python-play-sound/. Accessed on 11.05.2019.

[33] Yue Wu and Qiang Ji. Facial landmark detection: a literature survey. *International Journal on Computer Vision*, page 12, May 2018.