Real-Time Driver Drowsiness Detection by Analysing Facial Features

Oskar Figura 915070

May 2019

Project Dissertation submitted to Swansea University
in Partial Fulfilment for the Degree of Bachelor of Science

Department of Computer Science
Swansea University

Abstract

Sleep-related vehicle accidents occur on a daily basis, many of which are fatal, with several resulting in multiple casualties. In this project, I have proposed real-time driver drowsiness and sleep detection system based on the analysis of facial features. The system analyses facial features and uses them to detect the early stages of drowsiness, by monitoring head-pose and the percentage of the eye closure over time. Data from both measures is combined every 3 seconds to produce accurate drowsiness and sleep detection system, that is capable of issuing an audible warning when a drowsy or a sleeping driver is detected. It has been estimated, that a driver whose eyes have been shut for more than 40% of the time, is drowsy. However, if the driver's eyes have been shut for more than 70% of the time, the driver is said to be sleeping. Experimental results have proven that this system is capable of achieving satisfactory performance for sleep and drowsiness detection in real-time. In addition, the results have shown that in order to achieve accurate blink detection it is necessary to recalculate the threshold of an eye blink every 6 seconds and that sleep and drowsiness detection can be performed by monitoring the number of eye blinks for as short as 3 seconds.

Declaration

This work has not previously been accepted in substance for any degree and is not being currently submitted for any degree.

May 13, 2019

Signed: Oskar Figura

Statement 1

This dissertation is being submitted in partial fulfilment of the requirements for the degree of a BSc in Software Engineering.

May 13, 2019

Signed: Oskar Figura

Statement 2

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are specifically acknowledged by clear cross referencing to author, work, and pages using the bibliography/references. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure of this dissertation and the degree examination as a whole.

May 13, 2019

Signed: Oskar Figura

Statement 3

I hereby give consent for my dissertation to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

May 13, 2019

Signed: Oskar Figura

Contents

1 Introduction

1.1 Motivation

The majority of road accidents that occur on monotonous roads, such as motorways, are caused by drowsiness or distraction. Drowsiness will affect a sleep-deprived driver after a long period of driving, which will lead to drowsy driving during which the driver is half-asleep [1].

A driver who has been awake for 17 hours, encounters a decrease in psychomotor performance equivalent to a person with a blood alcohol concentration of 0.05%. If the driver continues to stay awake for 24 hours, then the decrease in psychomotor performance will be equivalent to a person with a blood alcohol concentration of 0.10% [2].

In 2017, the Department for Transport reported that on average, there are 5 fatalities and 468 reported road casualties in Great Britain per day [3].

An in-depth study, by Horne and Reyner 1995 [1], established that sleep-related vehicle accidents are responsible for, 16% to 20% of all vehicle accidents in Great Britain.

Sleep-related vehicle accidents are known to cause severe injuries and often fatalities, this is because during most of these accidents the driver is unaware of his current situation, resulting in late or no braking.

The human body shows signs when it is entering a state of drowsiness, most of these signs can be captured using biometric data such as heart and breathing rate variability [4]. Also, a substantial number of these signs can be captured using yawn detection, percentage of eye closure (PERCLOS), gaze direction, blink frequency and head pose estimation [5].

Premium car manufacturers such as Volkswagen [6] are continually equipping their latest cars with systems that attempt to detect drowsy driving to reduce sleep-related vehicle accidents. However, due to the nature of this problem, it is impossible to say whether their systems are capable of detecting drowsiness in all cases.

My study will attempt to prove that it is possible to detect driver drowsiness using data collected by facial detection. A system capable of detecting when the driver is sleeping, and early stages of drowsiness could significantly reduce the number of casualties which are a result of sleep-related vehicle accidents.

1.2 Project Aim and Objectives

The aim of this project is:

> *To develop a system that is capable of real-time driver drowsiness and sleep detection by analysing facial features.*

This aim can be broken down into a number of objectives, that once met, can be used to measure whether the project aim has been achieved.

O.1) The system should be capable of detecting facial landmarks in good and poor light conditions. Since, it is necessary for the system to be able to detect when a driver is sleeping, no matter what time of day and the lighting conditions.

O.2) The system should be capable of detecting when a driver is drowsy and it must issue an audible warning before he falls asleep.

O.3) It will be necessary for the system to recognise when a driver is sleeping. Sleep recognition will allow the system to proceed with an audible warning in an attempt to waken a sleeping driver.

O.4) The system must be capable of detecting when the driver blinks. Real-time blink detection will be used when calculating the percentage of eye closure over time, to determine if the driver is drowsy or sleeping.

O.5) The system must be capable of head pose estimation in real-time. The estimated head pose must be accurate, as it will improve the accuracy of facial landmark detection.

O.6) The system must be capable of analysing drivers facial features to detect sleep and drowsiness for a time period of at least 100 minutes, with an accuracy of 98%.

1.3 Review of the Literature

Academic staff from all over the world have attempted to develop a system capable of detecting drowsiness using various tools and techniques. This section will describe and analyse the different techniques used to detect sleep and drowsiness.

1.3.1 Doze Detection

Dozing occurs without warning [7] and therefore it increases the chances of a road collision due to drowsy driving or sleeping. Doze detection can be used to alert a sleep-deprived driver that he is about to fall into a light sleep unintentionally [8]. Dozing significantly reduces reaction time, vigilance, alertness and concentration. Therefore, it is impossible to safely perform attention-based activities such as driving [8].

A sleep-deprived individual will start to experience a significant reduction in consciousness level, this is a result of our body trying to achieve a calm mental state, gradually reducing the heart rate in combination with consciousness, slowly entering a rest phase which is a sign of dozing [7].

Research has shown that it is entirely possible to perform real-time doze detection based on closed eye time [8], as effectively as it is to use a heart rate peak sensor to detect the first moments of doze [7].

1.3.2 Gaze Direction

During the process of eye tracking, it's possible to determine where the driver is looking by measuring gaze direction, this measure can be used to identify driver drowsiness [5].

Different stages of sleep can be distinguished by observing eye movements. During REM sleep, we can observe rapid eye movement, this is the lightest form of sleep. However, during the transition of being awake to falling asleep, it can be observed that the eye movement becomes noticeably slower. Therefore, it's possible to determine the driver's alertness level based on eye movement characteristics, specifically, slow rolling eye movements are a sign of drowsiness [9].

An alert driver, will have quick eye movements that he uses to constantly scan the road and its surroundings, to identify possible risks, as well as occasionally check his mirrors without taking his eyes off the road, for more than 2 seconds.

However, a sleep-deprived driver will focus his eyes in one direction and any change of that direction will be slow and almost certainly longer than 2 seconds. This will greatly increase the risk of a collision due to the inability to cover a wide enough area in an adequate amount of time, this is an indicator of fatigue and that the driver needs to take a break [10].

Studies have identified that during driving, the risk of a collision is highly increased if the driver takes his eyes off the road for longer than 2 seconds. A duration of less than 2 seconds does not have any impact on road safety. Therefore, it's necessary to distinguish between quick eye movements to check mirrors and slow rolling eye movements that are caused by fatigue [11].

1.3.3 Blink Frequency

It is common for people to have a unique blink frequency and speed [12]. On average an adult will blink 20 times in a minute, with each blink lasting from 0.2 to 0.3 seconds [8].

A study on doze detection method, based on blink characteristics, has identified that a drowsy person is more likely to blink slower. Specifically, it has been concluded that a person is drowsy, if the closed eye time intervening blink burst exceeded one second [8].

A blink can be identified by applying facial landmark detection to localize the eye region of the face. Each eye can be represented by six coordinates as shown in Figure 1.

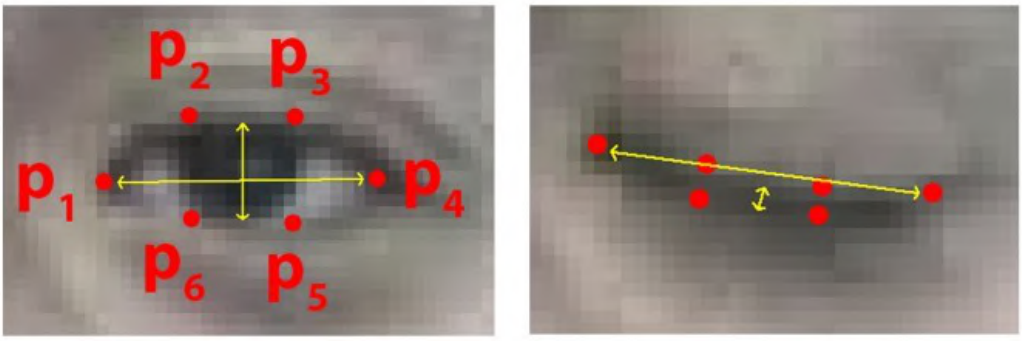

Figure 1: Open and closed eye with marked landmarks p_n [13].

A real-time blink can be detected by computing the ratio between height and width of the eye for each frame, using the following formula and identified landmarks p_n.

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

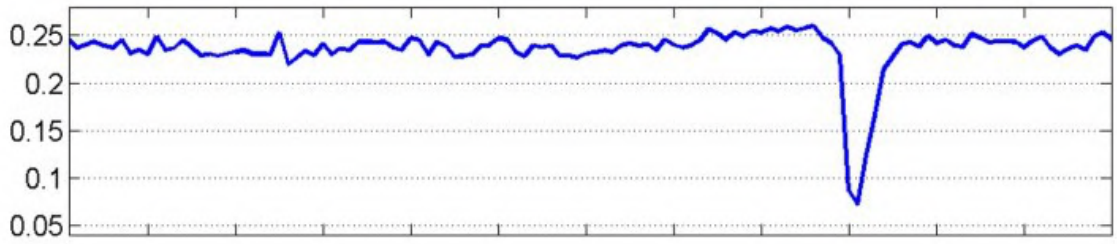

Figure 2: The eye aspect ratio (EAR) for several frames of a video sequence with a single blink [13].

Eye aspect ratio will remain approximately constant during the whole time the eye is open. However, the EAR falls close to zero as the eye closes as shown in Figure 2. This method is insensitive to the difference in the head pose as well as facial characteristics. Both eyes blink synchronously, therefore the EAR of both eyes is averaged [13].

1.3.4 Use of PERCLOS to Detect Drowsiness

An indicator called, PERCLOS (percentage of the closure of eyelid) is used as a drowsiness detection measure [14]. This measure focuses on the percentage of eyelid closure over the pupil, over time and focuses on slow eyelid closures instead of blinks [12].

PERCLOS is an effective measure when it comes to drowsiness detection, as it's proven to be very accurate. The accuracy of this measure comes from the capability of distinguishing between blinks which suggest that a driver is tired and blinks that are caused by other factors and are not linked to tiredness [12].

During driving, various situations can occur that cause the driver to suddenly blink more frequently or to briefly close his eyes for a slightly longer period. Such blinks could potentially be registered by a drowsiness detection system as blinks that suggest driver drowsiness, resulting in falsely suggesting that the driver is drowsy [12].

PERCLOS is potentially the best real-time measure for detecting driver fatigue, it has three evaluation indicators P70, P80 and EM. P80 which equates to eyes being closed at least 80% of the time and this is the usual state of a drowsy driver's eyes, during the early period of drowsiness.

Any results obtained using PERCLOS equating to the eyes being closed more than 80% of the time, can be used as a clear indication that the driver is very likely to be drowsy and should take a break from driving [12] [15].

An accurate drowsiness detection system based on PERCLOS, will need to be adjusted to the unique blink frequency and speed of every person, to ensure that the measured time proportion is correct [12].

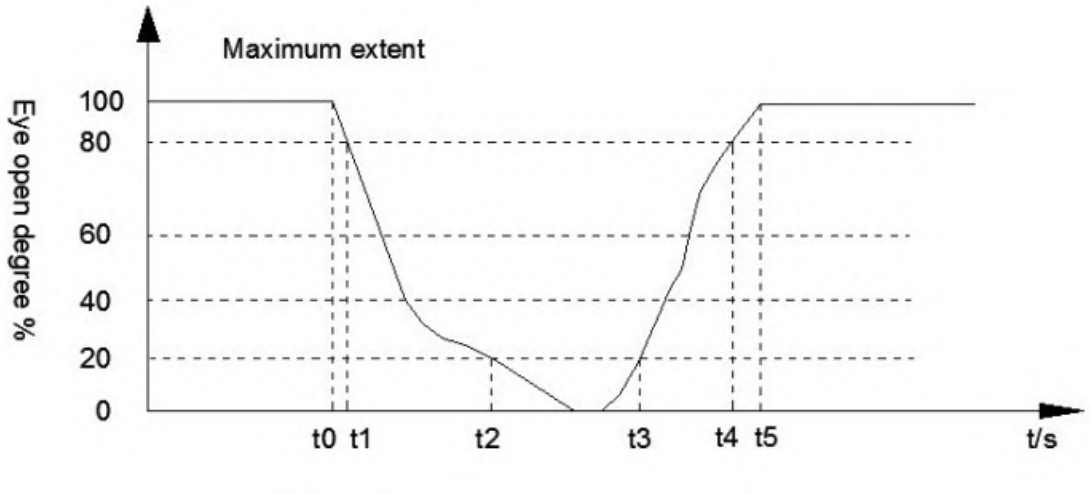

Figure 3: PERCLOS measurement principle graph[16].

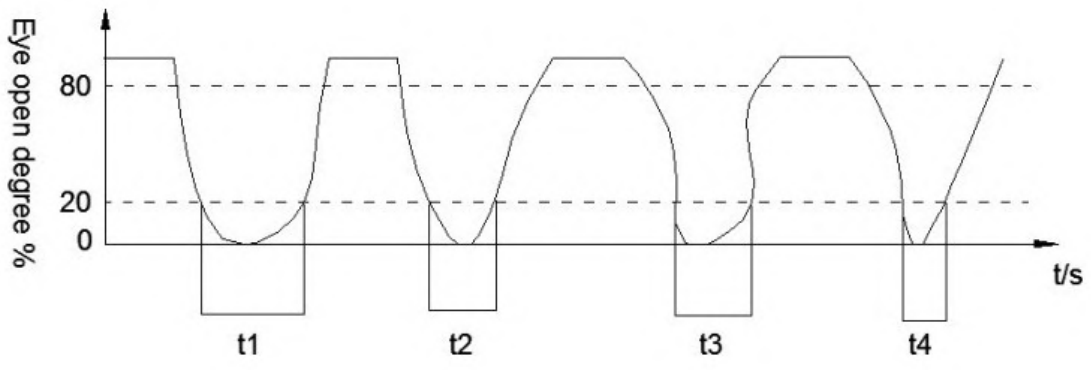

Figure 4: PERCLOS actual process graph [16].

The measuring of PERCLOS is as shown in Figure 3, the actual process graph is shown by Figure 4. The PERCLOS value for a single blink can be calculated using the following formula.

$$F = \frac{t_3 - t_2}{t_4 - t_1} \times 100$$

PERCLOS value can be obtained within the time it takes to blink once, by using values from t_0 to t_5. In the above formula, F represents the PERCLOS value that is the percentage of the time eyes have been closed; t_0 the time eyes are fully open; t_1 the time it takes for open eyes to close only to 20%; t_2 the time for open eyes to close to 80%; t_3 the time it takes for open eyes to close and then to open to 20%; t_4 the time it takes for open eyes to close and then open only to 80%; t_5 the time it takes for open eyes to close and then open [16].

The above method can only be used to calculate the PERCLOS value for a one-time eye blink, to calculate PERCLOS in real-time it is necessary to use the following formula.

$$PERCLOS = \sum_{k=1}^{n} \frac{t_k}{T}$$

In the above formula, it is assumed that n is the amount of time eyes are closed and the time an eyelid covered 80% of the pupil is referred to as t_k on the k time, with a total sampling time represented as T [16].

1.3.5 Yawn and Eyelid Closure Combined in the Detection of Drowsiness

Eyelid closure has proven to be one of the most reliable measures in predicting the beginning of sleep and a reduction in the ability to safely perform attention-based activities such as driving [8]. It has achieved more accurate results than measures such as; respiration rate, skin electrical characteristics and heart rate variability [11].

Studies have verified that eyelid closure is linked to a reduced driving performance and the ability to keep a vehicle in the correct lane, yaw stability (skid evasion) and the ability to quickly steer [11]. Therefore, it is possible to develop a system that is purely dependent on eyelid closure and produces a very accurate result.

However, even though the measure of eyelid closure produces accurate results in a drowsiness detection system, the entire method is based purely on monitoring the state of a driver's eyes. Drivers are very likely to wear sunglasses therefore, it won't be possible to make any assumptions about the driver's tiredness levels or even check if the driver is sleeping. In addition, it is also very common for drivers to cover their mouth whilst yawning, hence it is nearly impossible to produce a solution that is based on a single measure [17].

The use of yawn and eyelid closure as a combination will create a reliable system that doesn't rely on a singular measure, ensuring that when one fails the other could be used to provide accurate data. However, if data from both measures is available, it can be analysed and compared to better understand the driver's tiredness level, producing a more accurate result and significantly reducing the number of incorrect assumptions [18] [19].

Regardless of the extensive research that has gone into yawn detection, it is still uncommon for premium car manufacturers to equip their luxury models with the latest systems that are capable of accurate drowsiness detection based on yawning. Car's that have been equipped with such systems sporadically perform well informing the driver of his current tiredness levels [20].

However, it is common for these systems to provide inaccurate results, because during development they are tested in a workshop environment, with perfect lighting conditions, as well as no facial obstruction that can occur at any time during driving [20].

Yawning is a reflex that will occur when the driver is drowsy and about to fall asleep. A yawn can be detected by monitoring the mouth region and measuring the amount of change and rate at which it occurs [20] [18] [17]. A yawn can be distinguished from a driver speaking because during a yawn, the mouth will stay open for approximately 4 and 7 seconds [19].

Several studies have shown that tiredness is linked to both, yawning and eyelid closure. Analysis of the results has revealed that as the tiredness levels increased the driver was more likely to yawn and close his eyes simultaneously. This indicates that there is a strong connection between closed eyes, yawning and drowsiness [18] [21].

1.3.6 Head Pose Estimation

The muscles of a sleep-deprived driver will slowly start relaxing, the driver's head begins to lean forward due to its own weight, as a result, it is harder for the driver to breath, causing his head to return to its normal position. This moving of the head position will keep occurring, causing a nodding motion that can easily be identified by head pose estimation based on facial landmarks, suggesting that the driver is drowsy [5].

If nodding persists and the driver continues driving, it is very likely that he will fall into a micro-sleep and if not woken in time, it can turn into a regular sleep. A driver who is experiencing the stage of micro-sleep will be completely unaware that it is even occurring. Micro-sleep will often occur when the driver's eyes are open, it will only last a few seconds and unless the driver starts to drift out of his lane, he most likely won't notice that he should stop driving, because the next time micro-sleep occurs, it could result in a road collision [22] [23].

Head pose estimation can be achieved by using facial landmark detection, as shown in Figure 5, various landmark points are then used to estimate the pose of the head, this can be used to determine in which direction the driver is looking.

Figure 5: The output from Dlib facial landmark detection, on an image from HELEN dataset [24].

2 Technical Foundations

This section will identify the underlying technologies and theories that will be used in this project to accomplish an accurate real-time sleep and drowsiness detection system for drivers.

2.1 Hardware

An ideal implementation of this project would run on a Raspberry Pi 3, making it portable and easy to mount inside a car. However, in this project I want to achieve high accuracy, that will require a large amount of processing power that a Raspberry Pi 3 won't be able to supply. It is possible that in the future this implementation will be optimised to run on a Raspberry Pi 3.

Initially, this solution will run on a MacBook Pro running macOS Mojave, this computer is equipped with 16GB RAM and a powerful 2.2GHz i7 processor. The video will be captured using a plug-and-play compatible 720p webcam, with an attached infrared illumination ring that will switch on, upon detecting low lighting. The infrared illumination ring will consist of 48 IR LEDs, a light sensor and it will be powered by a 12V 300mA power supply.

2.2 Libraries

This project will be implemented using Python 3 because it offers extensive support for libraries and allows for quick development due to its simplicity. The simplicity of Python will have a positive impact on my productivity and it will allow me to quickly develop prototypes.

I will be using Dlib library for facial landmark detection and tracking. Dlib is a library with an extensive collection of fast and accurate algorithms for; Machine Learning, Computer Vision, Image Processing and Linear Algebra.

The wide range of functionality, good documentation combined with the simplicity of use, make this library an optimal choice for this project. Dlib is primarily a C++ library that provides access to a number of its tools for Python applications through the Python API.

To achieve the best possible performance I will make use of the NumPy library. This library handles large, multi-dimensional arrays and matrices, in addition it contains an extensive collection of high-level mathematical functions that can be used to operate on such arrays and matrices.

Some of the benefits that this library will provide are, vectorized and matrix operations that allow for elementwise multiplication and addition, boolean selection as well as contiguous allocation in memory. NumPy will reduce memory usage as well as increase read and write times which are crucial in a real-time based system.

In addition to using the NumPy library, I will also make use of the Savitzky–Golay filter implementation in the SciPy library which offers a good compromise between performance, accuracy and computational cost. The use of this library will allow for faster project development by removing the need to implement and test my own implementation of the Savitzky–Golay filter.

2.3 Savitzky–Golay Filter

The Savitzky–Golay smoothing and differentiation filter is a particular type of a low-pass filter. Such filter can be used to smooth a set of data points in order to increase the accuracy of the data without distorting the signal tendency, as shown in Figure 6. The Savitzky–Golay filter is commonly used for the purpose of smoothing high-frequency signals with a large amount of noise [25] [26] [27]. As it can be observed from the Figure 6(a) the electroencephalogram (EEG) signal has a lot of low amplitude noise which is effectively filtered by the Savitzky–Golay filter as shown in Figure 6(b).

The facial landmark detector will be continuously tracking the drivers face, generating hundreds of varying landmark coordinates that correspond to the position of the driver's facial features, the size and variation in the data will make it difficult to analyse when a blink has occurred. Therefore, I will use the Savitzky–Golay filter to smooth the set of EAR values collected during sleep and drowsiness detection, in order to improve the accuracy of the system.

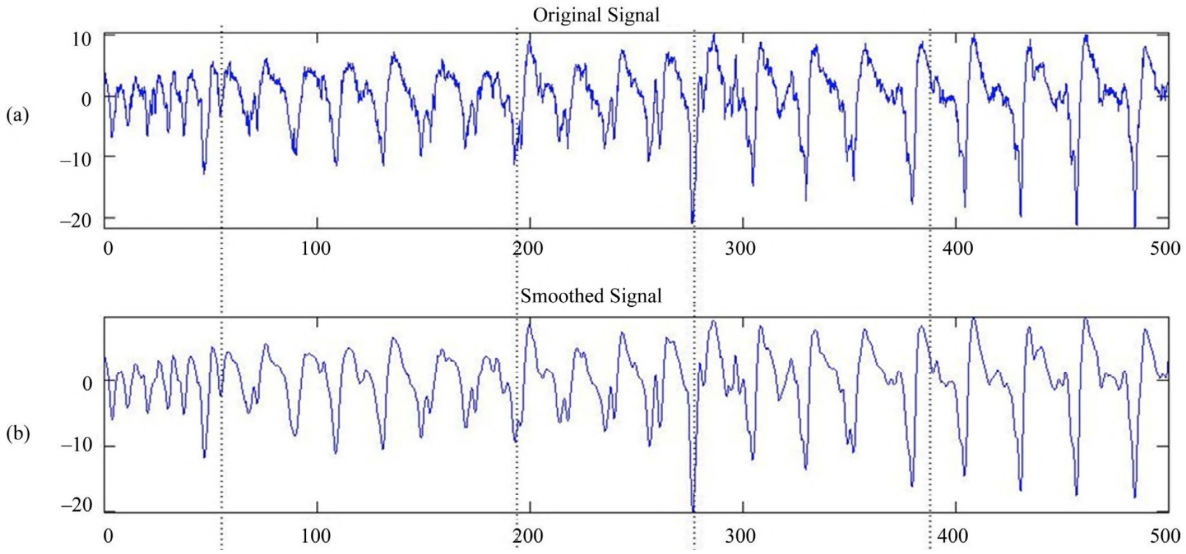

Figure 6: Signal segmentation in real EEG data: (a) Original signal; (b) Filtered signal by Savitzky-Golay filter [28].

2.4 Dataset

In this section, I will discuss the choice of the dataset for facial landmark detection. There are various datasets publicly available and I have decided to implement this project using iBUG 300-W face landmark dataset. The reason for this choice is that Dlib comes with a landmark predictor that was trained on the iBUG 300-W face landmark dataset Figure 7.

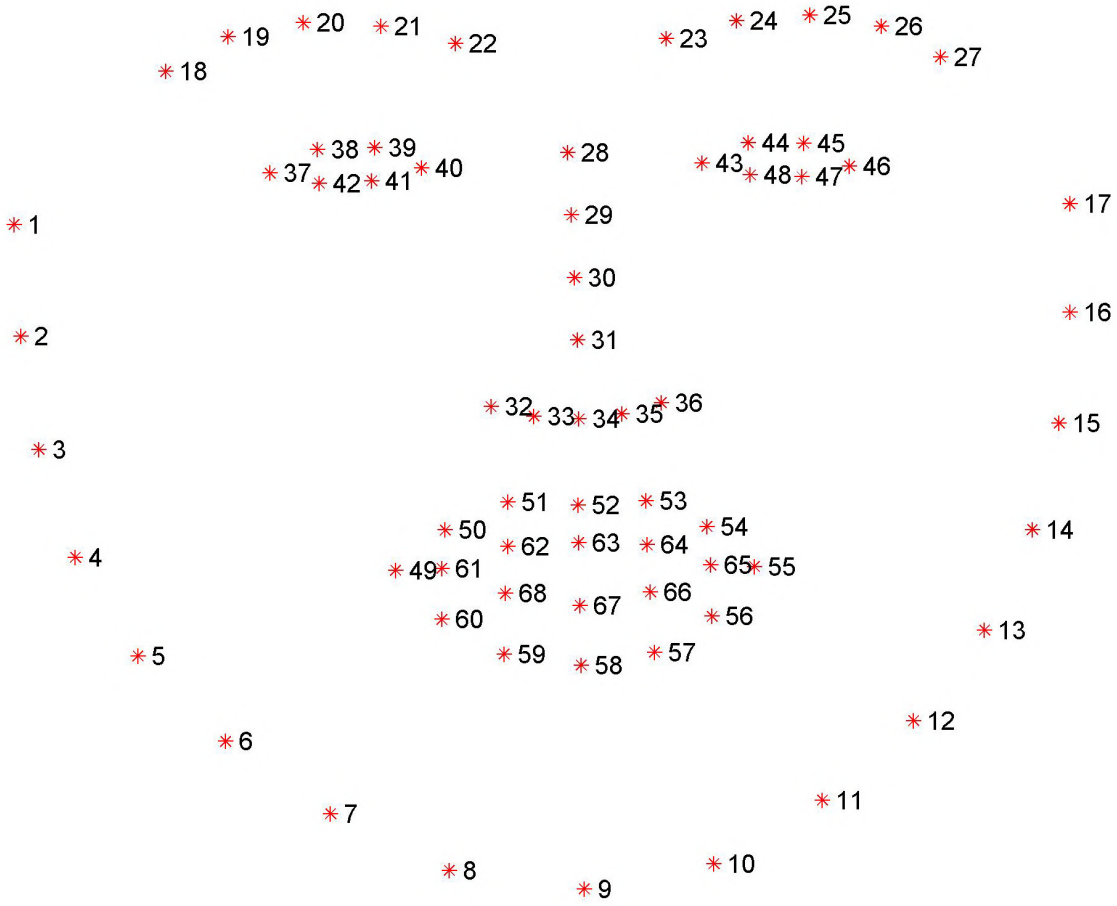

Figure 7: Visualization of the 68-facial landmark coordinates from the iBUG 300-W dataset [29].

Existing facial landmark datasets cover a large variety of different faces, poses, expressions and illumination levels. However, the issue with most of these datasets is that they cover a relatively small subset of the overall images. Also, the accuracy of annotations in some datasets is not very accurate, possibly caused by human fatigue of the person manually annotating the images.

The 68-point iBUG 300-W dataset that the Dlib facial landmark predictor was trained on, is suitable for this solution because, it used a semi-automatic annotation methodology that should be more accurate than datasets annotated manually [30].

There also exists a dataset with a higher number of annotation points, such as the 195-point model that can be trained on the HELEN dataset, as shown by diagram (f) in Figure 8. I believe that a higher number of landmark points would significantly reduce the performance of my solution since, the system would have to track more coordinates in real-time.

Therefore, after an in-depth analysis of the different landmark configurations, as shown in Figure 8, it's apparent that the only suitable choices are either iBUG or HELEN, because these datasets have significantly more coordinates around eyes and my solution will depend on the accuracy of detecting the eyes. Datasets such as LFPW and XM2VTS could not be taken into consideration as they don't have enough landmarks around the eyes, which would seriously affect the accuracy of my solution which must be avoided at all cost.

In conclusion, I have decided to use the iBUG dataset with 68 facial landmark coordinates, as they mark all the facial coordinates that are relevant to my solution. A dataset with a lower number of coordinates would provide better performance, but unfortunately at the cost of accuracy, which is more important in this project.

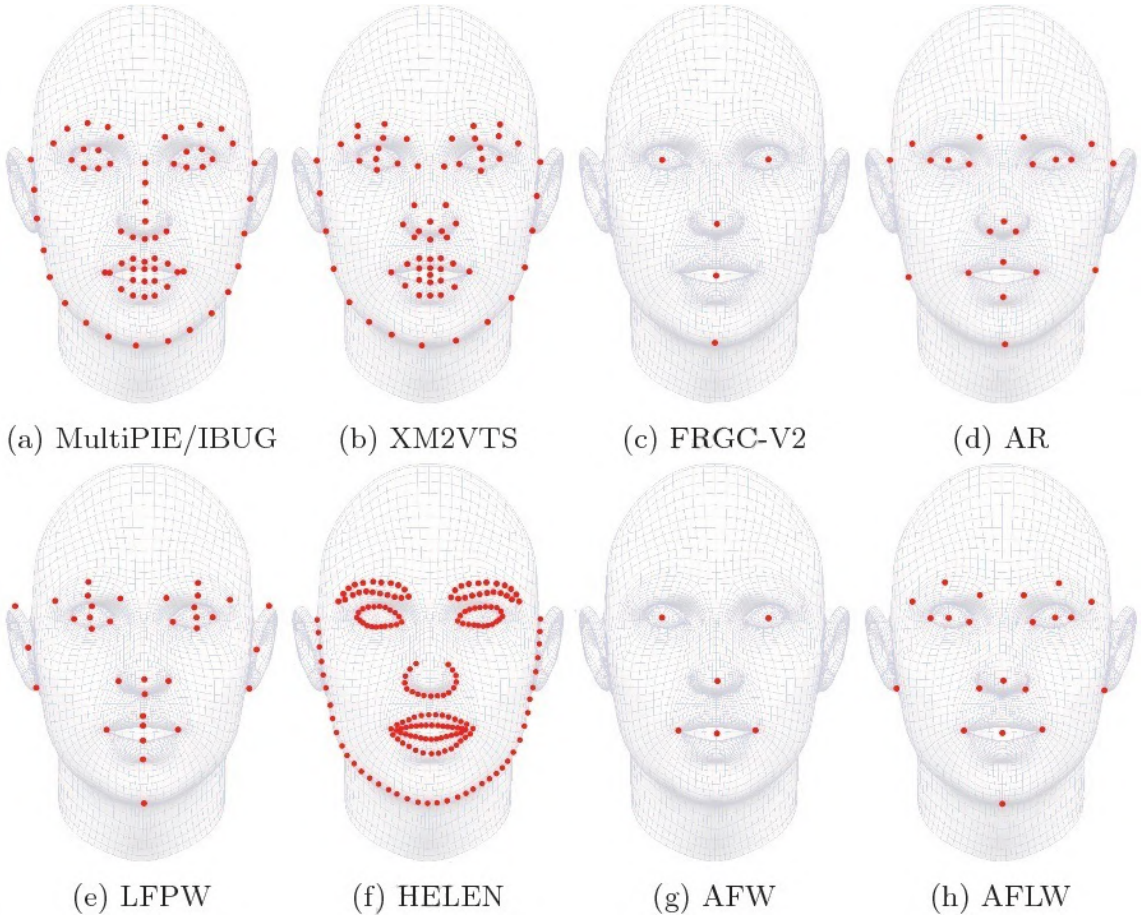

Figure 8: Datasets with different landmark configurations as well as number of coordinates [30].

2.5 Facial Landmark Detection

Facial landmark detection is a process of identifying key features of the face such as eyes, eyebrows, mouth, face outline and nose, as shown by Figure 7. Key features of the face are labelled using coordinates, the number and exact position of these coordinates, depends on the dataset being used, as shown by Figure 8.

In this project, I will perform facial landmark detection using Dlib facial landmark predictor that was trained on the iBUG dataset which uses 68 facial landmark coordinates. The landmarks identified by this system will be used to detect driver drowsiness, landmarks will provide real-time information about the driver's eyes, mouth and head pose.

It's crucial that identified landmarks are accurate since, the entire solution is based on the accuracy of these landmarks. Incorrectly identified landmarks will cause the system to misidentify the driver's state, making this system unreliable and ineffective.

3 Project Management

3.1 Project Planning and Schedule

During the initial phase of this project, I have created a project schedule which outlines the most important stages of this project as shown in Figures 9 and 10.

This project schedule follows an Agile methodology where each major phase of the project is grouped into an iteration. This project has twelve iterations, each iteration has a specific start and a completion date. Within each iteration, there are smaller tasks that are completed in the order they are listed and the end of an iteration is marked by a milestone. A milestone is an orange diamond which is used to mark the completion of an iteration.

The initial iteration is the research phase, as shown in Figure 9 where it has been marked by light blue colour. Completion of this phase will ensure that I have enough background knowledge to start working on the project development, as well as the initial documentation.

This project consists of five iterations dedicated to the development of the system; face detection, facial landmark detection; eye aspect ratio; head pose estimation; sleep and drowsiness detection. Each development iteration consists of seven subtasks that need to be completed in the order they are listed. Subtasks within each development iteration are colour grouped: development tasks are pink; unit testing tasks are yellow; debugging tasks are dark blue; integration tasks are dark green; integration testing tasks are grey.

Each iteration begins with the development of a module which is followed by unit testing, integration and then integration testing. The unit and integration tests are followed by debugging which has the purpose of ensuring that all issues are identified and fixed before continuing onto the next task or an iteration.

The iterations dedicated to research or documentation, overlap with other iterations in order to ensure that the project deadline is met. As shown in Figure 9, the initial documentation overlaps with the development iterations.

Whilst still working on the head pose estimation iteration, I have allocated sufficient amount of time for the Gregynog presentation iteration as shown in Figure 10, which includes the creation of a project demo as well as a presentation on the progress of the project.

Halfway through the completion of the last development iteration, I will start working on the final dissertation iteration. As, I will be capable of creating an outline of the final dissertation, whilst I continue working on the project which will be in its final stages.

Upon completion of the final development iteration, I will start working on the release testing and verification iteration. During this iteration, the final system release will be tested and verified. This iteration will ensure that the final system release successfully passes system, as well as acceptance testing and that any identified issues are fixed before the final evaluation of the system.

Once the final project has been submitted the project fair iteration will start, during this iteration I will be working on my project fair stall. The project fair day is after the final project submission deadline, hence why the completion is at 80% for this iteration.

I have taken into account that the project schedule has to accommodate for any unforeseen challenges, that are encountered before the final project submission. Therefore, I have allocated an entire iteration to handle any unforeseen challenges between the final project deadline and the project fair.

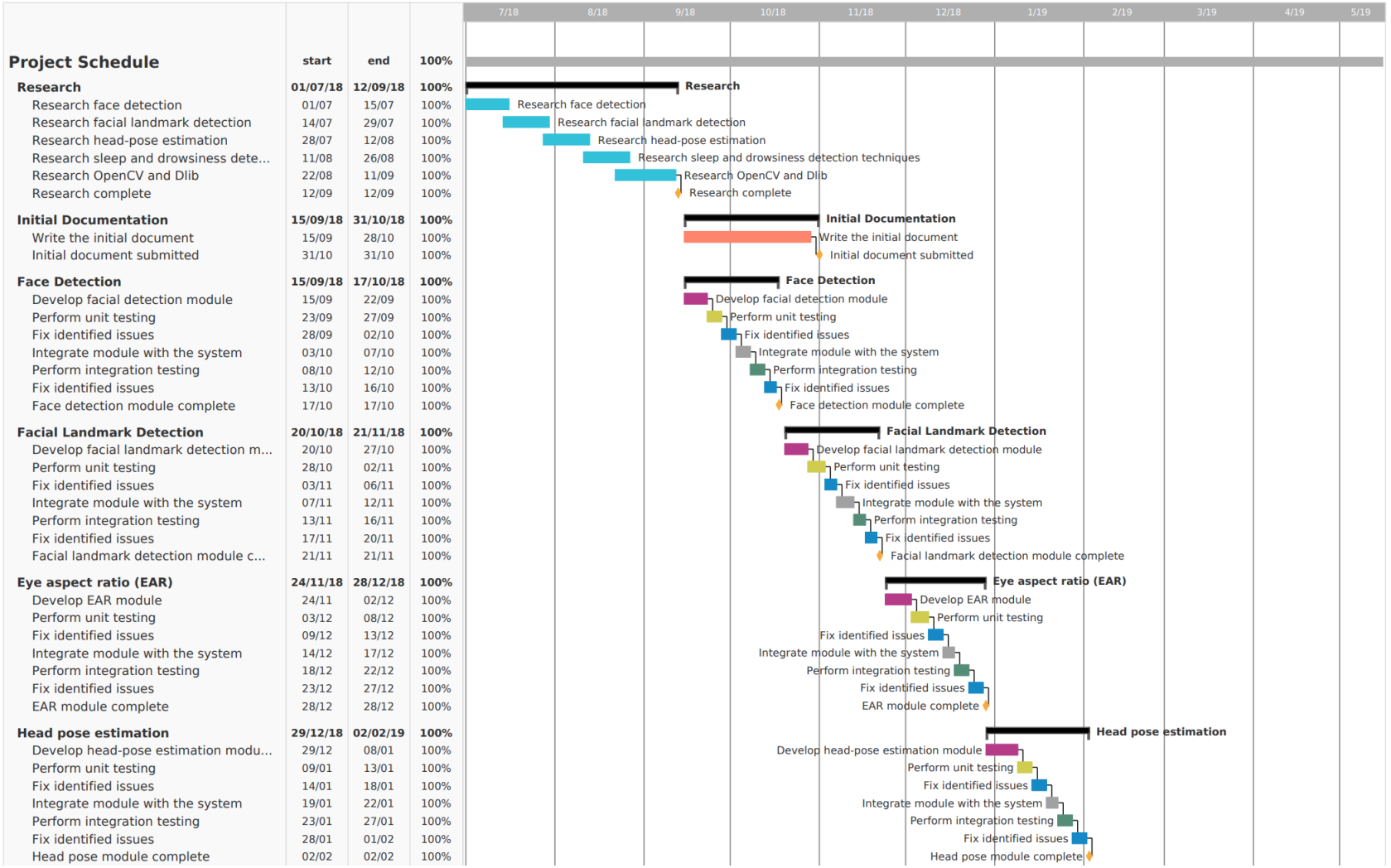

Figure 9: The project schedule part 1 of 2, with a detailed explanation of the project work for the period from 01/07/18 to 03/02/19.

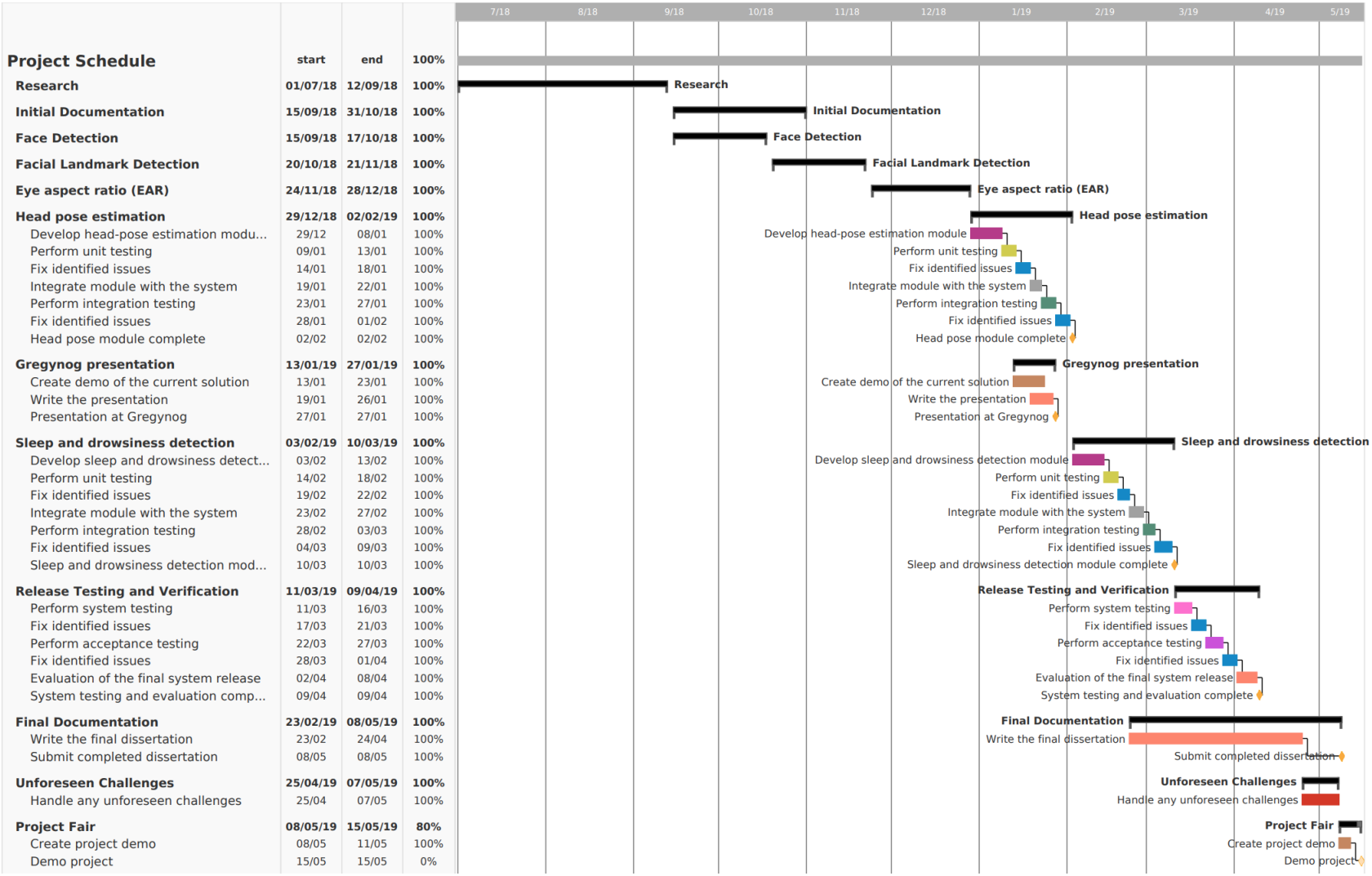

Figure 10: The project schedule part 2 of 2, with a detailed explanation of the project work for the period from 13/01/19 to 15/05/19.

3.2 Risk Assessment

This section outlines potential risks that could occur during this project and the strategies that will be used to avoid them. The risk assessment has been conducted during the writing of the initial document, to ensure that they are avoided, and their occurrence does not have a significant impact on the project development. In this risk assessment I have identified three types of risks; personal risks, section 3.2.1; technical risks, section 3.2.2; project risks, section 3.2.3. For each identified risk, a likelihood of occurrence and severity level has been stated, in addition to an appropriate avoidance strategy. The higher the severity and likelihood levels of a risk, the higher the negative impact that the risk will have on the project, if not avoided.

3.2.1 Personal Risks

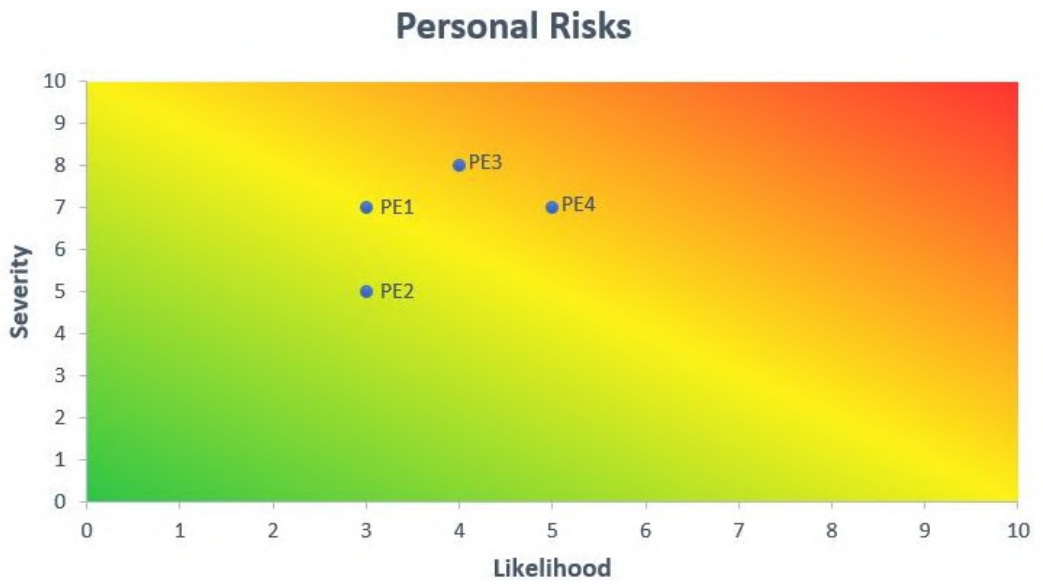

Figure 11: Personal Risks.

Type	Risk 1	Avoidance Strategy
PE1	Illness	Leave enough time at the end of the project to handle any unexpected issues.
PE2	Poor time management	Create a detailed schedule that will be strictly followed and allocate time for non-working days such as holidays.
PE3	Difficulty understanding new technologies such as Python, Dlib or OpenCV	I allocated sufficient time during the summer for research and learning new technologies.
PE4	Lack of background knowledge	An adequate amount of time has been allocated to research drowsiness detection during summer, before starting work on the initial document.

Table 1: Personal Risks.

3.2.2 Technical Risks

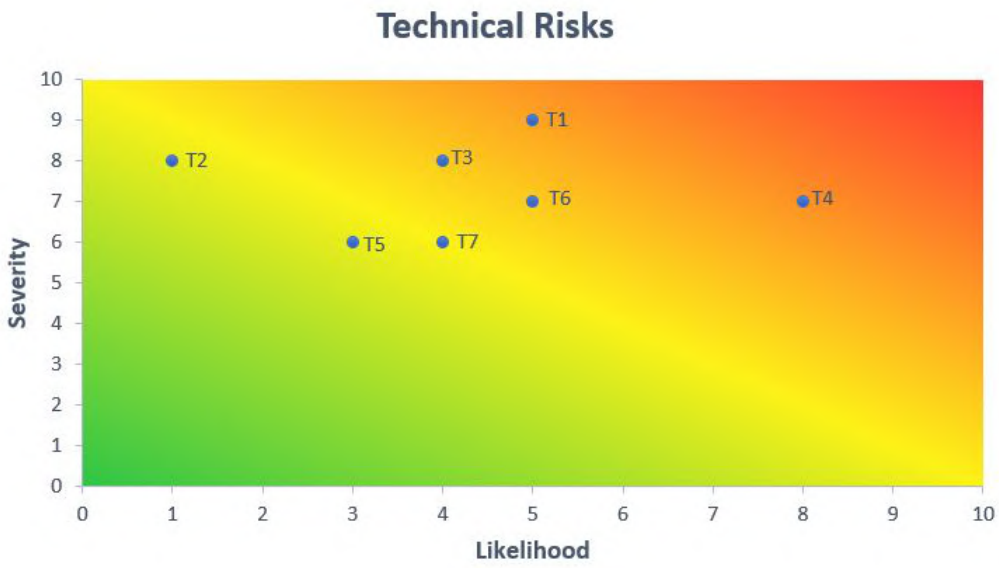

Figure 12: Technical Risks.

Type	Risk	Avoidance Strategy
T1	Inaccurate facial landmark detection	Research and compare the accuracy of annotations in various datasets by developing prototypes.
T2	Loss of access to Dlib Python API	Explore alternate implementation approaches.
T3	Insufficient system performance for detecting driver drowsiness in real-time	Explore alternate drowsiness measures that require less processing power and develop prototypes to test which measures perform better.
T4	Inability to detect facial landmarks in poor lighting conditions	Conduct testing in various lighting conditions to tune and develop prototypes that handle poor lighting conditions better.
T5	Mistakes in used libraries such as Dlib and OpenCV	Perform tests and compatibility analysis as well as explore alternate libraries that offer similar functionality.
T6	Inaccurate yawn detection which misinterprets talking or laughing as yawning	Explore alternate yawn detection measures in addition to developing and testing prototypes to improve accuracy.
T7	Inaccurately calculated PERCLOS value	Test the parts of the system responsible for calculating the PERCLOS value plus blink detection. Analyse test result and implement improvements.

Table 2: Technical Risks.

3.2.3 Project Risks

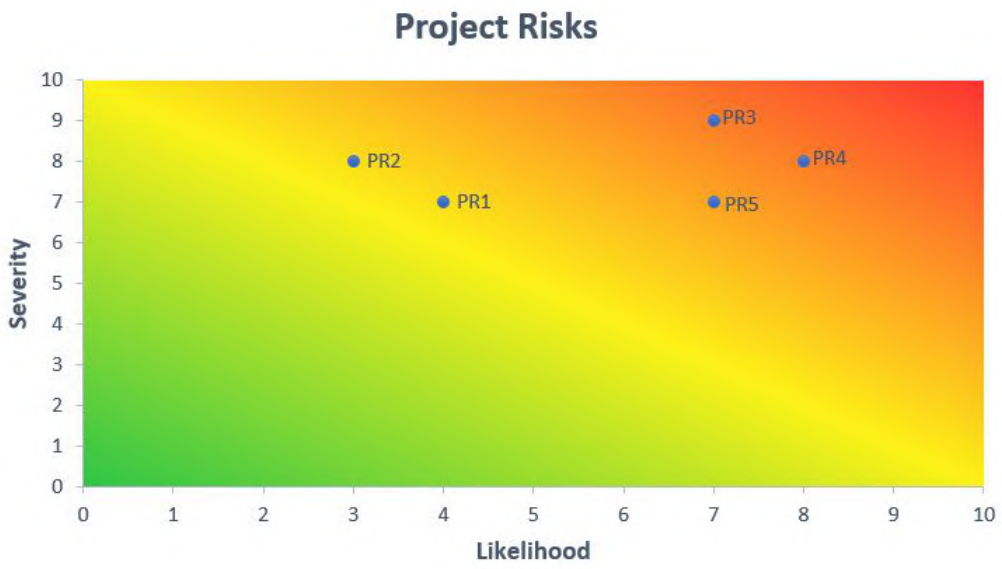

Figure 13: Project Risks.

Type	Risk	Avoidance Strategy
PR1	Overly optimistic schedule	Allocate sufficient time in the project schedule for; planning, development, testing, bug fixing, re-testing, changes, documentation and non-working time such as holidays.
PR2	Insufficient development time	Adequate time has been allocated to each stage of development in the schedule, plus extra time at the end of the project to handle any unexpected issues.
PR3	Loss of data	The project will be backed up regularly onto external storage and cloud. If a major milestone is reached, the project will be backed up at that time otherwise, it will be done daily.
PR4	Hardware failure	The project will be developed on a personal computer and all data is going to be regularly backed up onto external storage and cloud. A second set of hardware will be prepared prior to the project start date, to reduce potential delays as much as possible.
PR5	Poorly defined requirements	Develop; clear, complete, detailed, realistic, and testable requirements that have been discussed with the supervisor. Also, develop prototypes to investigate the possible lack of detail in the defined requirements.

Table 3: Project Risks.

3.3 Software Life Cycle

Agile software development life cycle is based on iterative and incremental development, it is ideal for projects where the requirements are constantly evolving, and solutions are being improved throughout the development process to ensure that the delivered product is fully functional and complies with all of the project requirements.

The requirements of this project have been well defined. However, the problem of accurate drowsiness detection will require small modifications to the requirements throughout the project. The development of this project is going to be iterative and incremental. Therefore, as new drowsiness detection measures are implemented it might be necessary to replace a certain measure or maybe an additional measure will be required, to improve the system performance or accuracy. Agile will allow for such unexpected changes at any time during the implementation phase, without causing any issues with the existing part of the system that is effectively functioning.

Agile software life cycle is an approach that produces a new release with slight modifications from the previous release. At each iteration, the release is thoroughly tested to ensure that it is ready for the next improvement. Testing performed after each release ensures that issues won't evolve into larger problems further into the development process. This methodology ensures that there is a working solution at each iteration and it provides the developer with the ability to introduce unexpected changes at any stage during the development.

The ability to perform tests with each release is crucial in this project since, drowsiness detection will require constant improvements to ensure that it's accurate and reliable. Testing, performed at each phase, will allow the developer to eliminate any issues that some of the drowsiness detection measures are causing, and it will optimize the development process and productivity.

In this project, not all the aims and requirements are clear, but they will become clearer with each iteration. Therefore, it will be possible to develop an early solution to this problem. Improvements to the solution at each iteration will tackle the problem of drowsiness detection more effectively.

The ability to return to a previous solution or a specific release is vital for a developer. It is not always possible, or feasible, to continue with the implementation of a certain measure or technology, if it's not proving to be as accurate as expected. Therefore, a developer will have the ability to abandon an unsuccessful release and continue with an earlier release that has been fully tested, without exposing this project to a failure.

Regular meetings with my supervisor, at the end of each major development phase, as well as strictly following the defined project schedule, will ensure that this project is being developed to the highest standard. This will avoid any complications arising from poor project management or miscommunication with the client, in this case, it's my supervisor.

3.4 Testing Strategy

During the testing phase of this project, I will utilise numerous testing methodologies to ensure that the final solution is accurately detecting a drowsy driver or sleeping driver in real-time. The system will be developed using iterative and incremental development, following the project schedule. At the end of each iteration, the release will have to meet the aims specified in the project schedule. This will be confirmed by thoroughly testing the release using a combination of functional and non-functional testing.

At the end of each iteration and before the submission of the final solution, I will perform grey-box testing to ensure that the implementation doesn't contain any hidden errors, a maximum code coverage is achieved, and the code has been optimised. Grey-box testing will also allow me to verify the implementation based on the requirements of the design. This will ensure that all requirements have been achieved and identify any areas for future improvement.

I will begin testing the facial landmark detection by running the system with a large database of face images, taken in various lighting conditions, starting with good lighting and concluding with poor lighting.

The purpose of this test is to let the system annotate detected faces with facial landmarks and then save them. As soon as the system has finished processing all the images, I will review them manually, checking for facial landmarks that are missing or are inaccurate.

Once the facial landmarks have been accurately detected in various lighting conditions in still images, the same testing technique will be used. However, still images will be replaced with various publicly available video files, mainly of people driving and with the camera focused on their face. These video files will again be processed and annotated by the system with recognised facial landmarks. On completion of the video file processing, I will manually review them searching for any issues with performance, as well as the accuracy of facial landmark detection.

These tests will allow me to measure the accuracy of the system in identifying facial landmarks, as well as detecting any patterns in the results. Identified patterns could aid the search for poor performing code, that may be responsible for incorrect facial landmark detection and possible improvements that could be made to improve the accuracy of facial landmark detection in different lighting conditions.

Once it is certain that the facial landmark detection is accurate, the system will be tested for detecting drowsiness and sleep, using head pose estimation and PERCLOS. These two drowsiness measures will be combined, and also used separately during testing to ensure that both are performing efficiently. Drowsiness detection testing will start with the system processing a large collection of video files, in which people are drowsy or falling asleep. The system will annotate videos with values from the drowsiness detection measures and the results will then be analysed to check if the drowsiness and sleep detection is accurate. This testing technique will be repeated until the required level of accuracy is achieved, the technique may vary slightly with each iteration to target a specific part of the system.

In conclusion, the above testing techniques should allow me to implement a system, that is capable of accurately detecting a sleeping or drowsy driver in real-time.

4 Project Specification

This section outlines the real-time sleep and drowsiness detection system from the hardware as well as software engineering perspective. Section 4.1 explains the chosen methodology when implementing this system. Whereas Sections 4.2 and 4.3 outline the requirements the implemented system must meet, in order for the project to be considered a success

4.1 Sleep and Drowsiness Detection

This system will perform several steps before it is able to confirm that the driver being monitored is drowsy or sleeping. The process of real-time drowsiness detection will start by analysing the input video feed of the driver's face, frame by frame.

Facial landmark detection will be performed on each frame and the result generated from each frame will be collected and then applied to the appropriate algorithms. To overcome the problem of finding the driver's face and performing unnecessary adjustments to the cameras view angle and position; I will assume for this project that the camera is fixed on top of the car's dashboard, directly facing the driver but without obstructing his view.

The process of drowsiness detection will make use of two measures, PERCLOS and head pose estimation. The use of both measures will increase accuracy, particularly when the drivers head is in a position where the camera is unable to see the whole face. Upon detecting that the drivers head is not in an ideal position, the system will discard any data recorded at that time to avoid any false positive result.

One of the biggest challenges in this project is facial landmark detection because a human face can vary significantly in size, shape, skin tone, as well as in the size and shape of facial features. To overcome this problem, I am planning on performing facial detection using Dlib facial landmark predictor that was trained on the iBUG dataset which uses 68 facial landmark coordinates. Dlib facial landmark predictor will allow me to extract data about the driver's facial features in various lighting conditions.

Eye coordinates extracted by facial landmark predictor will be used to calculate the PERCLOS value, which will be used as a measure of drowsiness. If the driver's eyes have been closed at least 40% of the time, then the system will identify the driver's state as drowsy. However, if the driver's eyes have been closed at least 70% of the time, then the system will identify the driver's state as sleeping. Upon detecting that the driver is either drowsy or sleeping, an audible warning will be played.

The system will collect data about the behaviour of the driver's eyes using 12 coordinates identified by the facial landmark detection, labelled 37 to 48, as shown in Figure 7. Each eye is represented by 6 coordinates.

For every video frame, the system will calculate the EAR values of the left and right eye, as described in Section 1.3.3. Upon successfully calculating the EAR value of every eye, an average EAR value will be calculated for both eyes.

The system will collect average EAR values for 6 seconds before it will attempt at calculating the EAR threshold that will be used for blink detection. The EAR threshold is simply the lower quartile value of the collected EAR values over the time period of 6 seconds.

Once the system has calculated the EAR threshold, it will monitor for eye blinks. Upon detecting an eye blink it will start a sleep and drowsiness detection iteration that has a duration of 3 seconds.

During the sleep and drowsiness detection iteration, the system will count the number of sampled EAR values and check how many have been detected as blinks. At the end of the sleep and drowsiness detection iteration, the system will calculate the PERCLOS value for the collected EAR values and check for signs of sleep and drowsiness.

In conclusion, combining PERCLOS and EAR measure with head pose estimation will allow me to implement a reliable and accurate drowsiness detection system.

4.2 System Requirements

The system implemented for this project must meet the following requirements:

R.1) The system should use a Dlib HoG Face Detector, in order to detect drivers face in various lighting conditions, without being affected by abnormal facial expressions or head poses.

R.2) The system hardware should consist of an infrared illumination module, in order to make it possible to detect the drivers face in extremely poor lighting. The module must be automatically switched on when the amount of light is insufficient to detect the drivers face.

R.3) The system should consist of: the eye aspect ratio module; the head pose estimation module; the sleep and drowsiness detection module.

R.4) The system should be capable of an accurate 68-point facial landmark detection in real-time, without being affected by varying lighting conditions. The detection of facial landmarks must be performed using Dlibs facial landmark predictor.

R.5) The system should be capable of accurately calculating the EAR values for both eyes, including an average of the computed EAR values, using the 12 coordinates identified by the facial landmark detection, labelled 37 to 48 as shown in Figure 7.

R.6) The system should be capable of calculating the lower quartile value (EAR Threshold) of the set of EAR values that have been filtered by the Savitzky–Golay filter.

R.7) The system should be capable of recalculating the EAR threshold every 6 seconds, to ensure that blink detection is accurate.

R.8) The system should be capable of real-time blink detection using the computed average EAR value and EAR threshold.

R.9) The system should be capable of an accurate head pose estimation based on the position of facial landmarks. This includes the ability to accurately estimate whether yaw, pitch or roll is negative, positive or neutral.

R.10) The system should discard any EAR values collected whilst, head yaw is negative or positive.

R.11) The system should be capable of measuring the percentage of eye closure for a time period of 3 seconds based on the recorded set of EAR values.

R.12) The system must be able to detect when the driver is feeling drowsy or is sleeping, by using the calculated PERCLOS value.

R.13) When the system identifies that the driver is drowsy or sleeping, the system must proceed with audible warnings in an attempt to waken the sleeping driver.

R.14) The system must be able to run flawlessly on a machine with a macOS Sierra operating system, at least 8GB of RAM, 5GB of free storage and an equivalent to a 2.8 GHz Intel Core i5 processor (4308U).

5 Implementation

In this section I will explain the process of implementing a real-time sleep and drowsiness detection system. This section will also give an in depth overview of the implemented algorithms, used hardware and any aspects of the project that have not been implemented. The complete source code for this system is the Appendix A.

5.1 Hardware

This section will look at the hardware used to capture the video footage of a driver in real-time, whilst he is driving. The video footage is captured by a Logitech C270 HD USB webcam as shown in Figure 14(b), this webcam has a max resolution of 720p/30fps, 60 degree field of view and a standard lens.

In order to make it possible for the webcam in Figure 14(b) to record in poor light, it had to be modified. The modifications included attaching an infrared illuminator module shown in Figure 14(a), on the front of the camera as shown in Figure 15(a) and Figure 15(b).

The infrared illuminator module Figure 14(a), is powered by a 12V 300mA power supply, consists of 48 IR LEDs and a light sensor. The IR LEDs are automatically switched on when the light sensor detects an insufficient amount of light for the webcam to record in.

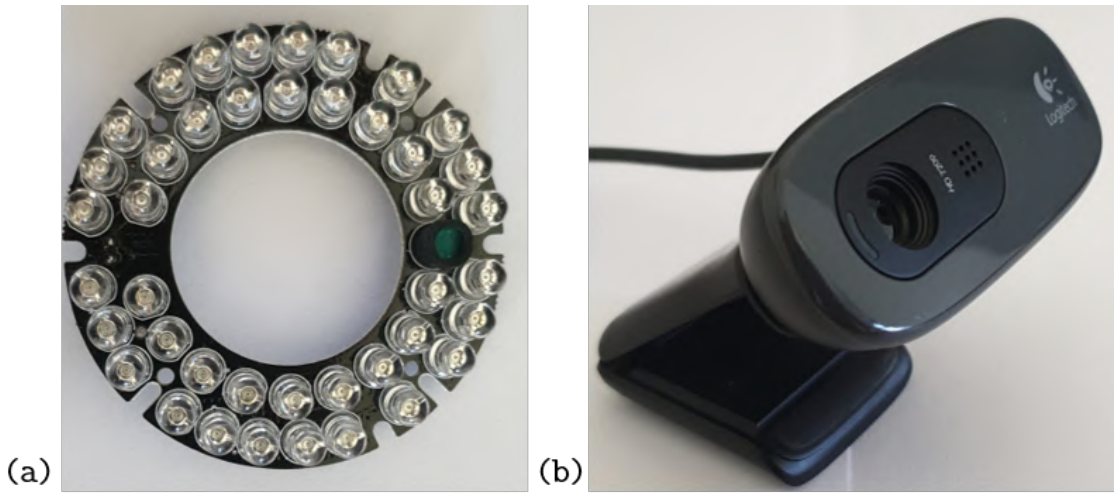

Figure 14: Hardware components used for video capture before the integration: (a) a 48 LED infrared illuminator ring; (b) Logitech 720p HD USB webcam.

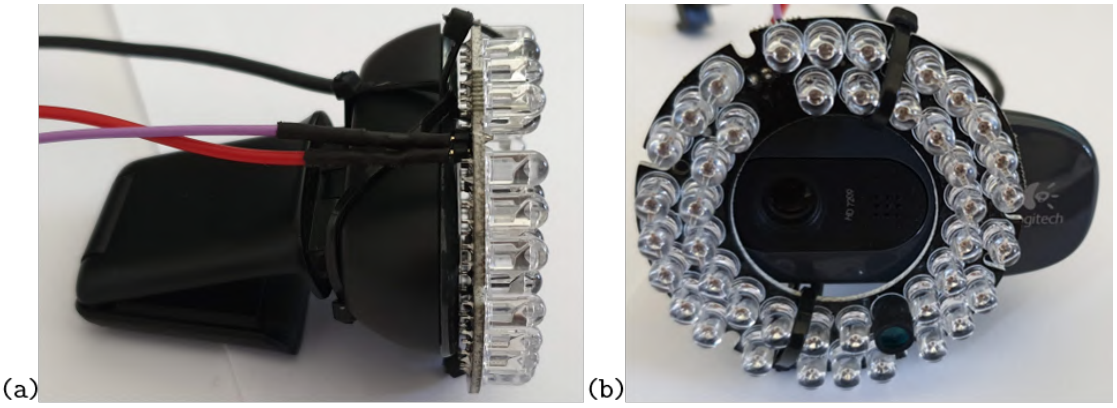

Figure 15: Infrared illuminator ring attached to the webcam: (a) webcam side view; (b) webcam front view.

5.2 Face Detection

Facial detection can be performed using four common methods: Haar Cascade Face Detector in OpenCV; DNN Face Detector in OpenCV; HoG Face Detector in Dlib; CNN Face Detector in Dlib.

Extensive research of the facial detection methods has revealed that the two methods which offer the best compromise between performance and accuracy are, HoG Face Detector in Dlib as shown in Figure 16(b), and Haar Cascade Face Detector in OpenCV as shown in Figure 16(a).

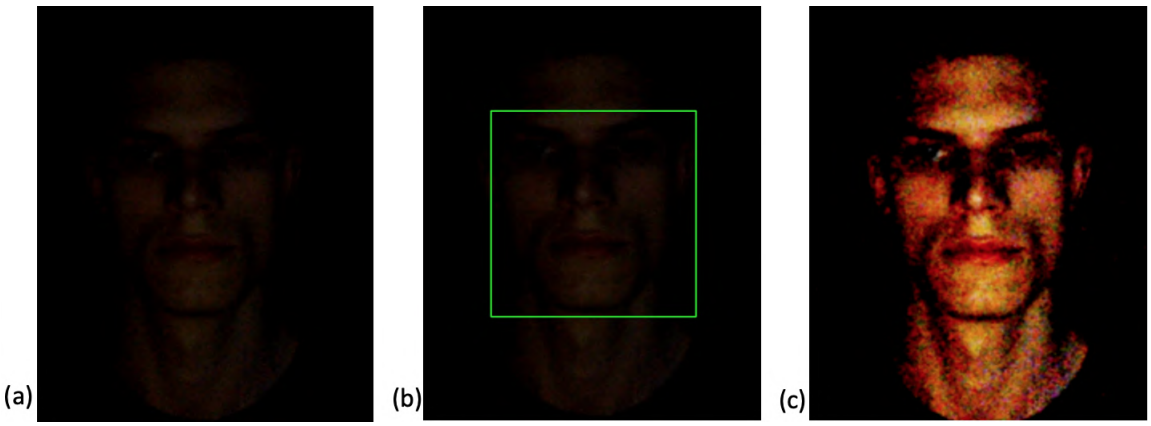

Figure 16: Result of face detection in poor light with a bright comparison image: (a) OpenCV-Haar face detector; (b) Dlib-HoG face detector; (c) a bright pre-processed image for comparison.

The DNN Face Detector in OpenCV offers better accuracy at the cost of speed. Whereas, the CNN Face Detector in Dlib has been rejected due to being unsuitable for facial detection in real-time, caused by its slow algorithm.

Therefore, I have decided to implement facial detection using Haar Cascade Face Detector in OpenCV as shown by Listing 2 and HoG Face Detector in Dlib as shown by Listing 1.

```
import cv2
import dlib

IMAGE_PATH_INPUT = 'faces/face_pic.png'
IMAGE_PATH_OUTPUT = 'faces/out.jpg'

img = cv2.imread(IMAGE_PATH_INPUT)
detector = dlib.get_frontal_face_detector()
detector_result = detector(img, 1)
face = detector_result[0]
cv2.rectangle(img, (face.left(), face.top()), (face.right(),
    face.bottom()), (0, 255, 0), 2)
cv2.imwrite(IMAGE_PATH_OUTPUT, img)
```

Listing 1: Implementation of the Dlib-HoG face detector for a single image.

```
import cv2

HAAR_CASCADE_FRONT = 'haarcascade_frontalface_alt.xml'
IMAGE_PATH_INPUT = 'faces/face_pic.png'
IMAGE_PATH_OUTPUT = 'faces/out.jpg'

img = cv2.imread(IMAGE_PATH_INPUT)
face_cascade = cv2.CascadeClassifier(HAAR_CASCADE_FRONT)
faces = face_cascade.detectMultiScale(img, 1.3, 5)

for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

cv2.imwrite(IMAGE_PATH_OUTPUT, img)
```

Listing 2: Implementation of the OpenCV-Haar face detector for a single video frame.

It is a requirement that the system is capable of detecting a drivers face in poor lighting. Testing of Haar and HoG has demonstrated that Haar is unable to detect faces in poor lighting, as shown in Figure 16(a), while HoG succeeds in accurately detecting a face in the same lighting conditions.

5.3 Facial Landmark Detection

The facial landmark detection is a two step process. The first step is to detect a face as described in section 5.2. Upon successful detection of a face, the area where a face was detected is converted into a grayscale image and then analysed for facial structures.

The facial landmark detection implemented in this system is based on the pre-trained facial landmark predictor inside the Dlib library. This pre-trained predictor is used to estimate the location of the 68 (x, y)-coordinates that correspond to the facial features of the detected face.

The Dlib's facial landmark detector implements a paper, that can detect facial landmarks in just one millisecond [31].

However, in this system it is impossible to perform facial landmark detection at 1000 fps, due to the delay caused by face detection. Hence why the performance of the facial landmark detection is highly limited by the computational capacity of the machine it is running on and it doesn't exceed 60fps.

An example implementation for a single video frame is shown in Listing 3. This code takes as input a single video frame, which is then resized to a smaller image and converted to a grayscale image in order to speed up face detection.

Upon successful detection of a face, the face is analysed for facial landmarks, if the facial landmarks are successfully detected the program draws the coordinates that correspond to the facial features onto the image.

However, when a real-time video is being processed, instead of drawing facial landmarks onto the image, the location of those landmarks is stored in a NumPy array for further processing. Storing the location of facial landmarks in a NumPy array allowed me to reduce memory usage as well as speed up the processing of the array.

```
1  import numpy as np
2  import imutils
3  import dlib
4  import cv2
5
6  PREDICTOR_PATH = 'shape_predictor_68_face_landmarks.dat'
7  IMAGE_PATH_INPUT = 'faces/face.jpg'
8  IMAGE_PATH_OUTPUT = 'faces/out.jpg'
9  FRAME_WIDTH = 800
10 FRAME_HEIGHT = 450
11
12 detector = dlib.get_frontal_face_detector()
13 predictor = dlib.shape_predictor(PREDICTOR_PATH)
14
15 def get_landmarks(gray_frame, face):
16     shape = predictor(gray_frame, face)
17     coords = np.zeros((shape.num_parts, 2), dtype=int)
18
19     for i in range(0, shape.num_parts):
20         coords[i] = (shape.part(i).x, shape.part(i).y)
21
22     return coords
23
24 def get_face(gray_frame):
```

```
25      faces_detected = detector(gray_frame, 0)
26
27      if len(faces_detected) > 0:
28          return faces_detected[0]
29
30      return None
31
32  def main():
33      image = cv2.imread(IMAGE_PATH_INPUT)
34      image = imutils.resize(image, width=FRAME_WIDTH,
        height=FRAME_HEIGHT)
35      gray_frame = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
36      face = get_face(gray_frame)
37
38      if face is not None:
39          landmarks = get_landmarks(gray_frame, face)
40
41          for (x, y) in landmarks:
42              cv2.circle(image, (x, y), 1, (0, 0, 255), -1)
43
44          cv2.imwrite(IMAGE_PATH_OUTPUT, image)
45
46
47  if __name__ == '__main__':
48      main()
```

Listing 3: Implementation of Dlib's pre-trained facial landmark predictor for a single video frame.

5.4 Eye Aspect Ratio

Eye aspect ratio (EAR) allows for an accurate analysis of the driver's eyes at a low computational cost. The eye aspect ratio calculation is based on the position of the twelve eye landmarks. The EAR is calculated for each eye separately, and then an average of the two is computed and returned since a person blinks both eyes at the same time.

The initial step of this function is to extract twelve landmarks that correspond to the position of the left and then the right eye, six landmarks per eye as shown by Figure 17. Upon successful retrieval of the twelve eye landmarks, I can calculate the eye aspect ratio for each eye, using the euclidean distance function from the Scipy library as shown in Listing 4, then the Euclidean distance is used to calculate the average of the two calculated EAR values.

Figure 17: Annotated left and right eye landmarks using twelve red dots.

```
1  from scipy.spatial import distance as dist
2  from imutils import face_utils
3
4  (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
5  (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
6
7  # Calculate eye aspect ratio (ear) for a single eye
8  def calc_ear(eye):
9      # Vertical distance between two first eye co-ordinates
10     v_a = dist.euclidean(eye[1], eye[5])
11     # Vertical distance between the second two eye co-ordinates
12     v_b = dist.euclidean(eye[2], eye[4])
13     # Horizontal distance between the corners of an eye
14     h_a = dist.euclidean(eye[0], eye[3])
15
16     # Computing eye aspect ratio
17     computed_ratio = (v_a + v_b) / (2.0 * h_a)
18     return computed_ratio
19
20
21 # Calculates avg ear for both eyes
22 def get_avg_ear(landmarks):
23     left_eye = landmarks[lStart:lEnd]
24     right_eye = landmarks[rStart:rEnd]
25
26     # Compute eye aspect ratio for both eyes
27     left_eye_ar = calc_ear(left_eye)
28     right_eye_ar = calc_ear(right_eye)
29
30     # Average the eye aspect ratio for both eyes
31     avg_eye_ar = (left_eye_ar + right_eye_ar) / 2.0
32
33     return avg_eye_ar
```

Listing 4: Implementation of the two eye aspect ratio functions. The first function line 8 calculates the eye aspect ratio. The second function line 22, calculates the average eye aspect ratio for both eyes.

5.5 Head Pose Estimation

The accuracy of facial landmark detection is dependant on the accurate estimation of the head pose. Accurate head pose estimation avoids false positive result due to inaccurate landmark detection that is caused by unusual head poses.

The classification of the different head poses estimated by the system, is based on the visualisation shown in Figure 18. To estimate the head pose of a driver, the system first calculates the roll, pitch and yaw angles of the drivers head. Once, the system has calculated the angle values for the three head poses, it moves onto classifying whether each of the head poses is neutral, positive or negative, as shown in Figure 18 and Figure 19.

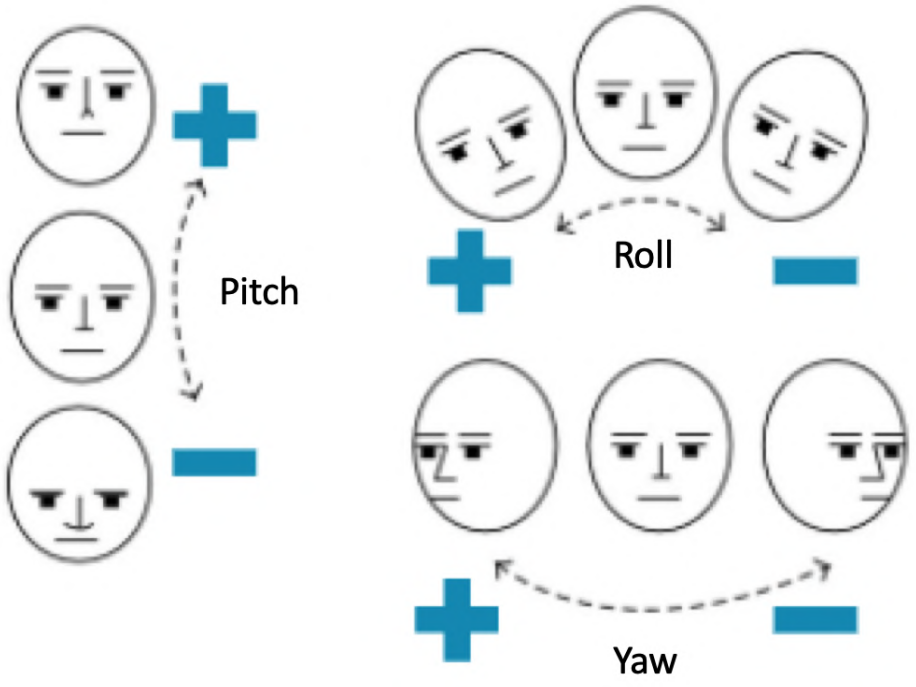

Figure 18: Visualisation of the pitch, roll and yaw head poses that are recognised by the system.

Figure 19: Result of the head pose estimation in poor lighting, on a single video frame, with annotated (x, y, z)-coordinate values, as well as simplified description of those values.

However, if the system detects that the yaw value is either, positive or negative, it will ignore the value of the roll angle. Since the head pose estimation algorithm will get confused by positive yaw, combined with a positive roll which could potentially produce a false positive result that will make the sleep and drowsiness detection inaccurate.

The implementation of the head pose estimation algorithm is strongly dependant on six facial landmarks as shown in Figure 20: tip of the nose; chin; left corner of the left eye; right corner of the right eye; left corner of the mouth; right corner of the mouth.

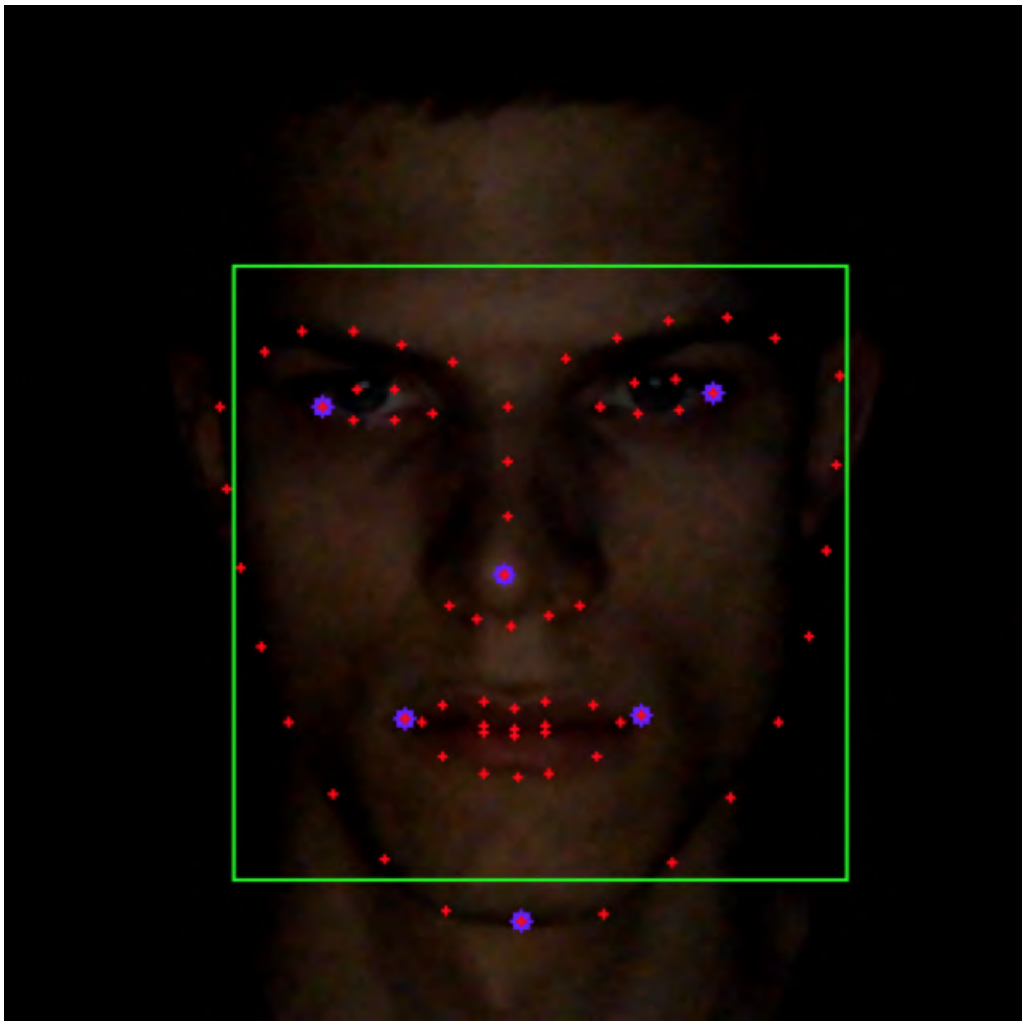

Figure 20: Visual representation of the six facial landmarks used for head pose estimation, marked with pink circles. The red circles annotate 68 facial landmarks.

The head pose estimation is implemented using the OpenCV function solvePnP, which implements several algorithms for pose estimation. In this system, I have decided to use the solvePnP-iterative method which is based on the Levenberg-Marquardt optimisation, the result can be seen in Figure 19.

The distinction between neutral, positive and negative head poses is determined by the value of the angle for each head pose. I have tested the head pose estimation algorithm with a hundred example images of various head poses in order to find the range of values that gives the most accurate result. The following are the angle values that have proven to give the best result.

The head pitch:

- Positive if the angle value is between 0 and 155.
- Negative if the angle value is less than -160 or greater than 175.
- Neutral if the angle value is between 155 and 175.

The head yaw:

- Positive if the angle value is greater than 19.
- Negative if the angle value is less than -19.
- Neutral if the angle value is between -19 and 19.

The head roll:

- Positive if the angle value is greater than 15.
- Negative if the angle value is less than -15.
- Neutral if the angle value is between -15 and 15.

If the algorithm encounters a computational error, then the head pose is marked as unknown, and will not be taken into consideration when detecting sleep and drowsiness to ensure that the result produced is accurate. For complete source code of the head pose estimator, please refer to Appendix A, Section A.2.

5.6 Eye Blink Detection

Eye blink detection is performed by analysing the value of the eye aspect ratio (EAR), in real-time for every video frame. If the EAR value drops below a set threshold, then the system registers an eye blink and further analysis is performed to detect sleep and drowsiness as explained in Section 5.7.

The threshold of the eye aspect ratio is calculated using the function shown in Listing 5. This function takes as input a list of EAR values, and using the average as well as a minimum of the EAR values, it calculates the lower quartile, which is returned as the EAR threshold.

At first, it may seem reasonable to use the minimum EAR value as the threshold. However, not all eye blinks will result in the same minimum EAR value, as a matter of fact, a lot of the EAR values that represent eye blinks, are significantly smaller than the minimum EAR value as it can be seen in Figure 21. Therefore, it was necessary to use the lower quartile value instead of the minimum or average which would have resulted in an inaccurate eye blink detection.

```
1 # Calculates ear threshold
2 def calc_ear_thrs(ear_values):
3     avg_ear = sum(ear_values) / len(ear_values)
4     min_ear = min(ear_values)
5     ear_threshold = min_ear + ((avg_ear - min_ear) / 2)
6     return ear_threshold
```

Listing 5: Implementation of the function that is used to calculate the threshold value for a set of eye aspect ratio (EAR) values.

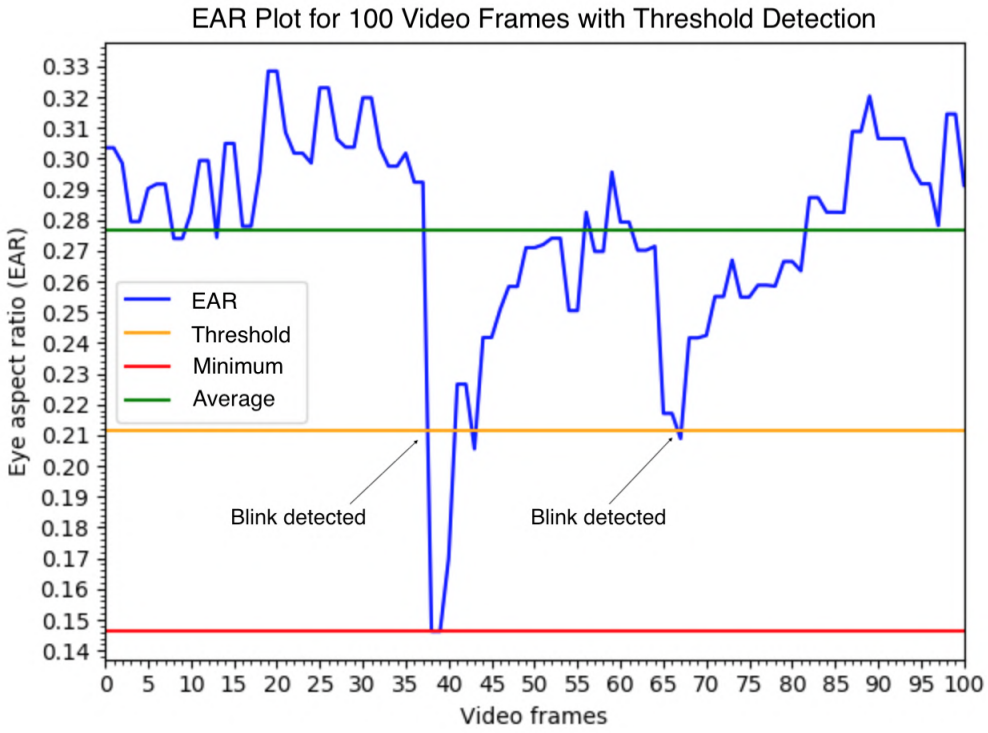

Figure 21: An EAR plot showing two eye blinks, the first at the 37th and the second at the 64th video frame.

Figure 21 is used to demonstrate why the calculated minimum and average of the EAR values have been discarded as the threshold values. This graph shows two eye blinks, the first eye blink occurs at the 37th video frame and lasts for approximately 23 video frames, this eye blink sets the minimum EAR value at less than 0.15. The second eye blink occurs at the 64th video frame and lasts approximately just as long as the first eye blink, and it has a minimum EAR value of 0.21. The extremely low EAR value of the first eye blink results in an extreme minimum which cannot be met by the second eye blink. In addition, the calculated average of the EAR values doesn't cover peaks and hence it would detect almost every EAR value as an eye blink.

Therefore, the lower quartile value is the most optimal value out of the three possible threshold values. Since it allows for accurate detection of various types of eye blinks and it is not limited to an extremely small EAR value which would rarely ever be met.

In order to further improve the detection of the EAR threshold, I have applied a Savitzky–Golay filter from the SciPy library, to the set of EAR data points in order to smooth the data, which in result increases the precision of the data without distorting the signal tendency as shown in Figure 22. The implementation of this filter is demonstrated in Listing 6 as well as Appendix A, Section A.1, Listing 8 code lines 120 and 121.

```
1                       sg_ear_vals = sc.savgol_filter(ear_vals, 17, 7)
2                       ear_thrs = ear.calc_ear_thrs(sg_ear_vals)
```

Listing 6: Demonstration of how the Savitzky–Golay filter from the SciPy library is used to smooth a set of EAR values.

The calculation of the EAR threshold based on raw EAR data has proven to be strongly affected by the noise introduced by the detection of facial landmarks. I have used the Savitzky–Golay filter built into the SciPy library with a window size of 17 and a polyorder of 7, in order to smooth the EAR values before calculating the new EAR threshold as shown in Listing 6.

The smoothing of the EAR values using the Savitzky–Golay filter, has allowed me to compute a threshold that can be used to detect a rapid eye blink, as shown below in the Figure 22. Rapid blinking often results in a slight change of the EAR value, often caused by the slow performance of the facial landmark detection or outside factors such as lighting.

Frequently the EAR measure is affected by noise from the facial landmark detection. Such noise can reduce the EAR value of a rapid eye blink as it can be seen in Figure 22, here a rapid eye blink occurs at the 25th video frame followed by a regular eye blink at 63rd video frame. The first eye blink would not be detected by the threshold calculated from the raw EAR values, as it's too small. However, the application of the Savitzky–Golay filter has smoothed the EAR values reducing unnecessary noise, and in result creating a slightly bigger EAR threshold which is capable of detecting the first eye blink, from the raw EAR values.

In addition, the bigger EAR threshold allows for an earlier eye blink detection as it can be seen in Figure 22, where the second eye blink would be detected at the 61st instead of the 62nd video frame. This small change might not make a huge difference to eye blinks that been recorded accurately. However, the smoothing and bigger threshold will improve the detection of eye blinks that have been mixed with a lot of noise.

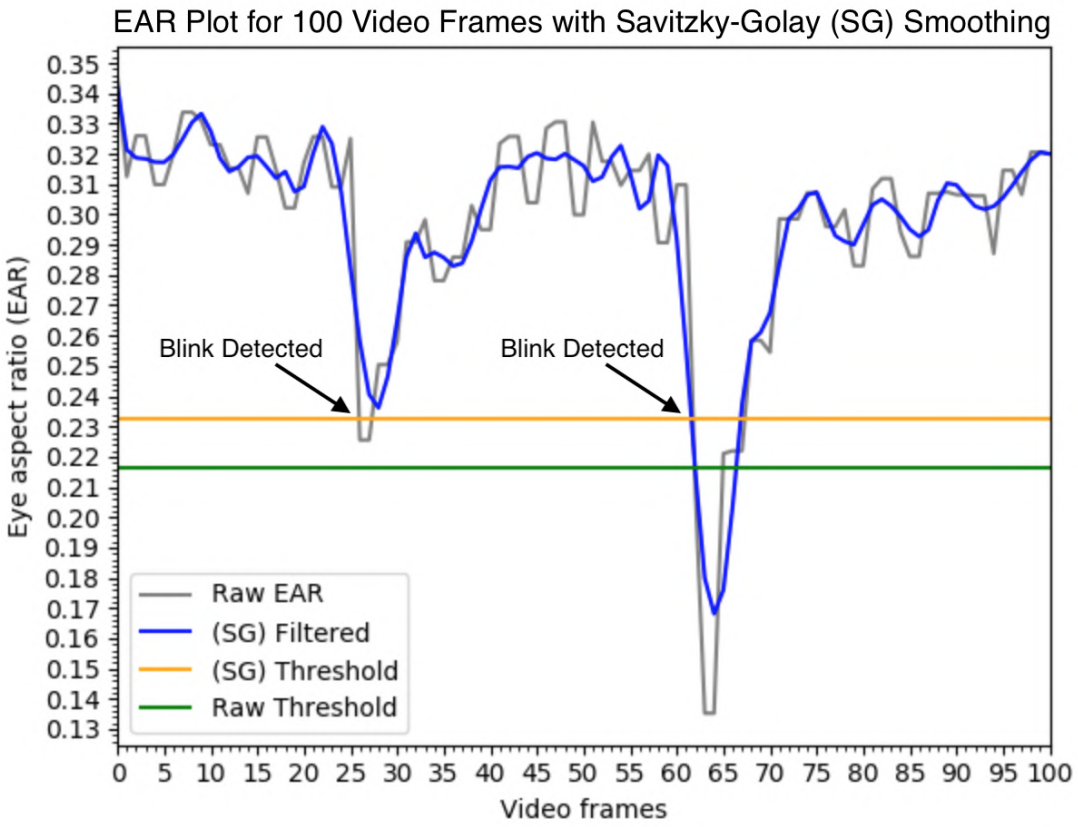

Figure 22: Raw and Savitzky–Golayand smoothed EAR measure, plotted over an observation period of 100 video frames, with two eye blinks, and a plotted threshold values for both graph plots.

To ensure that the eye blink detection remains accurate throughout the whole time, the EAR threshold is recalculated approximately every 6 seconds. This recalculation accommodates for any rapid changes to the drivers head position, as well as lighting conditions that could potentially decrease or increase the value of the EAR that is calculated for every video frame.

5.7 Sleep and Drowsiness Detection

The sleep and drowsiness detection is based on the analysis of drivers blink frequency and the resulting PERCLOS value. By analysing the blink frequency over a set period of time, the system can detect when the driver is beginning to feel drowsy or is sleeping, by calculating the PERCLOS value for the set of recorded EAR values, as shown in Appendix A, Section A.1, Listing 8 code lines 57-61.

The sleep and drowsiness detection is performed over 45 video frames, this process is referred to as the sleep and drowsiness detection iteration. Such iteration is only started upon a successful detection of a blink, as shown in the Appendix A, Section A.1, Listing 8 code lines 131-132.

An iteration that has been started, will start collecting the EAR values over the next 45 video frames as shown in the Appendix A, Section A.1, Listing 8 code lines 135-136. Once an iteration has finished, a PERCLOS value is calculated for the set of collected EAR values.

The calculated PERCLOS value is then used to check if the driver is drowsy, or sleeping, as shown in the Appendix A, Section A.1, Listing 8 code lines 138-144 which end an iteration, and the code lines 63-78 which perform sleep and drowsiness detection by analysing the value of PERCLOS.

The sleep and drowsiness detection function, as shown in the Appendix A, Section A.1, Listing 8 code lines 53-80, checks for a successive detection of drowsiness as it can be seen in code line 63. If the system detects that the driver is drowsy three times or more consecutively, it will mark the driver's state as sleeping.

A graphical representation of the PERCLOS boundaries used for the detection of sleep and drowsiness is shown in Figure 23. If the PERCLOS value is below 40% the system estimates that the driver is awake and no actions are taken. However, if the PERCLOS value is above 40% but less than 70%, the driver is considered to be drowsy, in result a short beep is played to make the driver aware of his current fatigue level.

If the PERCLOS value exceeds 70%, the system predicts that the driver is sleeping and tries to wake up the driver by issuing a 3-second audible warning. If a sleeping driver is not woken up by the 3-second audible warning, then the warning is played again until the PERCLOS value is below 70%.

This method is considered to be accurate because, as explained in Section 5.6 on average an adult will blink 20 times in a minute, with each blink lasting from 0.2 to 0.3 seconds. Therefore, it is possible to accurately determine whether a driver is drowsy, by collecting the EAR values over a duration of 45 video frames, which is approximately 3 seconds.

An awake driver will blink approximately once every 45 video frames (3 seconds), with each blink lasting up to 4.5 frames (0.3 seconds), such values will result in a PERCLOS value of approximately 10%. As mentioned in Section 5.6, a drowsy person will have a higher blink frequency and a longer blink duration, typically a blink of a drowsy person will exceed 1 second. Therefore, for a drowsy driver, the value of PERCLOS will be above 40%.

The duration of 45 video frames is approximately 3 seconds, only if the system is running on the hardware mentioned in Section 5.1. But, the number of frames over which the EAR values are sampled is dependant on the computational capacities of the hardware used. Since significantly more powerful hardware will be capable of capturing and analysing the video footage at a higher rate of fps. In result, video capture of 45 video frames might be approximately 1.5 seconds long, and hence it would be necessary to increase the duration of a sleep and drowsiness detection iteration to 90 video frames.

The use of frame count to depict the duration of a sleep and drowsiness detection iteration has allowed me to improve the performance of the system. Since the system is not overloaded with the processing of various time counters and instead it checks the size of the set of EAR values, which simply depicts the number of frames that have passed, as shown in the Appendix A, Section A.1, Listing 8 code lines 139.

In conclusion, I have implemented an accurate sleep and drowsiness detection method that is not computationally expensive and can be easily improved by the addition of other sleep and drowsiness detection measures such as yawn detection.

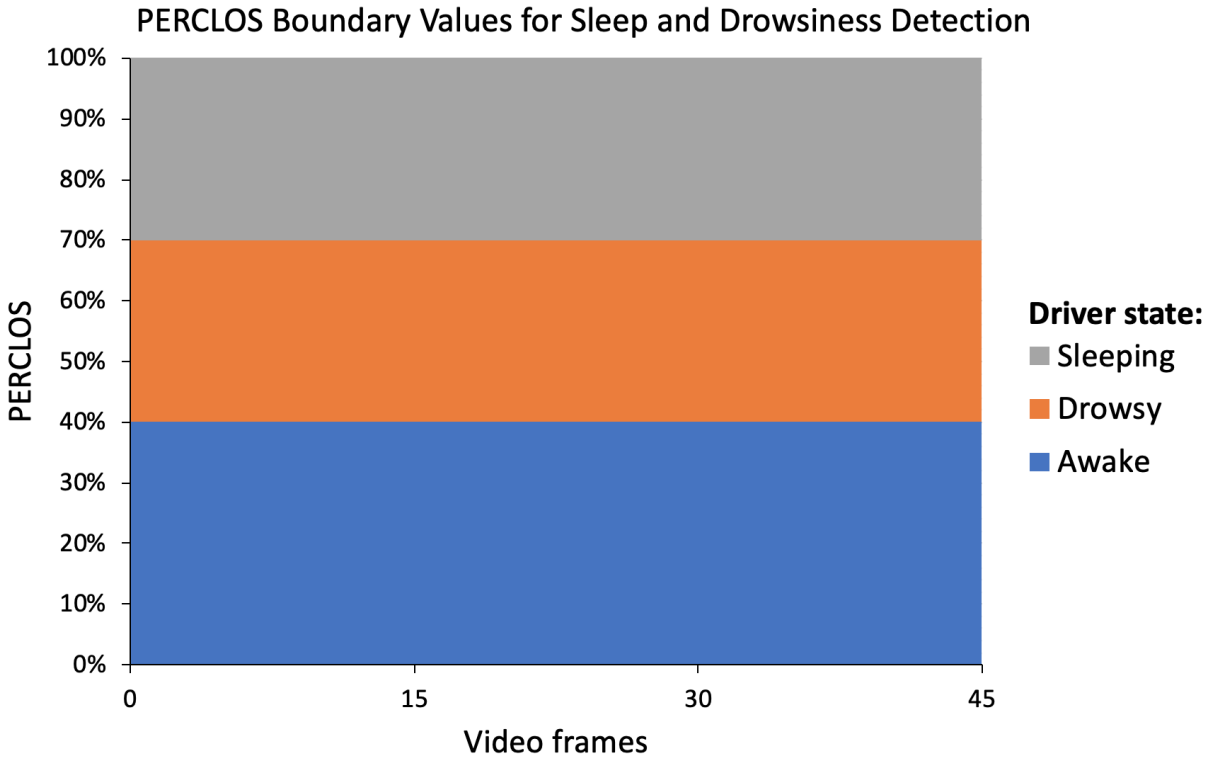

Figure 23: A graphical representation of the three PERCLOS boundary values for an awake, drowsy and sleeping driver. This representation is based on a single sleep and drowsiness detection iteration that has a duration of 45 video frames.

5.7.1 Warning System

The implementation of the warning system used to alert the driver of his current state, and to prevent him from falling asleep, is rather simple yet very effective.

When this system detects that the driver is asleep, the system will play a siren sound for 3 seconds, this is effectively going to make the driver aware of what is happening to him. However, if the driver is drowsy the system will play a short beep sound.

Sleep and drowsiness detection is performed every 3 seconds once an eye blink has been registered. Therefore, if a driver is asleep the system will be able to continue playing the same 3 seconds long sound until the driver's state changes to awake or drowsy.

The warning sound is played using a separate thread as shown in the Appendix A, Section A.1, Listing 8 code lines 66-68 and 73-75, this ensures that whilst the warning system is trying to alert the driver, the sleep and drowsiness detection algorithms continue analysing drivers facial features. This allows for continuous analysis of the driver's drowsiness level and ensures that if the first warning sound doesn't alert the driver, then the warning sound is played until the driver becomes aware of his current state.

This system makes use of the playsound library to play the sound on the hardware mentioned in Section 5.1, the usage of this library can be seen in Listing 7.

```
1 # Play sound
2 def play_alarm(path):
3     playsound.playsound(path)
```

Listing 7: Implementation of the warning function that is used to play a sound from the input file path upon detecting a drowsy or a sleeping driver.

6 Testing

In this section, I will discuss the testing methods that have been used to test this system in order to ensure that the achieved accuracy and performance satisfies the system requirements and the objectives of the project aim.

This system has been developed using iterative and incremental development, where each unit has been individually tested and then integrated with the project. Such development practice allowed me to focus on the development of modules and only perform integration testing once the module has passed unit testing.

For the purpose of this document, I will focus on showing the most challenging test cases as it is natural that if a challenging test case passes, then the equivalent basic test case will also pass. However, some basic test cases will also be shown to demonstrate that the system requirements have been met.

6.1 Testing Environment

For the purposes of testing this system, I have set-up a workstation and mounted the webcam on the top of one of the monitors as shown by Figure 24. The nature of this system makes it unethical and dangerous to test it on public roads.

The workstation I have set-up acted as a driving simulator and allowed me to create an ideal testing environment where I could instantly vary the lighting levels as well as change positions that would otherwise be impossible to perform inside of a vehicle, especially when driving.

Figure 24: Webcam mounted onto a PC monitor: (a) photo of the monitor from a distance; (b) magnified view of the webcam mount.

In addition to testing this system in an ideal testing environment, I have also conducted tests inside a vehicle that was used to model a real driving scenario as shown by Figure 25. However, to ensure that the testing performed was ethical and safe, the vehicle has been stationary on private property, at all times.

The system set up inside the car was primitive and temporary. The webcam has been glued onto the dashboard just above the steering wheel at drivers eye level as shown in Figures 25 and 26, and the video footage from the webcam was being transferred via a USB, to a laptop placed on the passenger's seat as shown in Figure 25. The laptop was used to run the sleep and drowsiness detection program, whilst recording results for future analysis, a view of a processed and annotated video frame can be seen in Figure 27.

Figure 25: System setup inside a vehicle with the sleep and drowsiness system running on a laptop, that is placed on the passengers seat and webcam mounted on the dashboard above the steering wheel at drivers eye level.

Figure 26: A closer view of the webcam mounted inside a car, on top of the dashboard at drivers eye level.

Figure 27: View from the webcam mounted inside the car, that has been processed and annotated by the sleep and drowsiness detection system.

6.2 Face Detection Testing

When testing face detection it was important that I tested every possible scenario. Face detection in this system is the first module that processes the real-time raw footage of the driver's face and then passes the processed result to the remaining modules. A faulty face detection would render the entire system useless. Therefore, to ensure that this module was thoroughly tested I performed tests under three lighting conditions; good lighting; poor lighting; infrared lighting.

The testing processes involved reading a real-time video capture of my face, and for each frame, the system was expected to detect and draw a rectangle where the face has been found and finally save the annotated image to a results folder, that was later analysed.

In addition to using real-time video capture, I have also tested facial detection with the iBUG 300-W dataset and obtained satisfactory accuracy. Since the system is only required to detect a face when the driver is looking directly ahead of him, I was able to ignore other head poses that could yield a high number of false positive result during the detection of facial landmarks.

The facial detection has proven to be very accurate when detecting faces in good lighting as shown by Figure 28(a), and an infrared lighting as shown by Figure 28(c) and Figure 29(d). However, even though I have managed to achieve facial detection in poor light as shown by Figure 28(b). The facial detection in poor lighting is not reliable and often yields no result. Therefore, the use of infrared lighting increases the accuracy of the facial detection module in poor lighting almost to the same level of accuracy that has been achieved in good lighting.

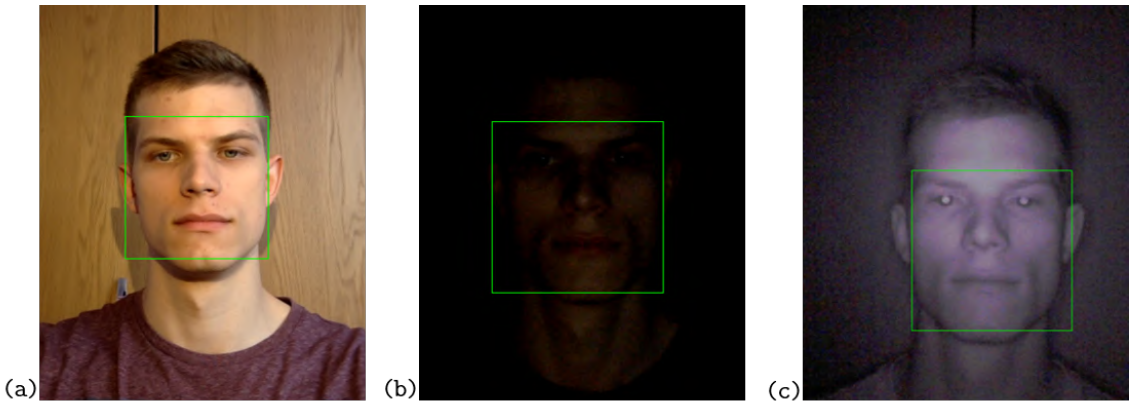

Figure 28: Successful facial detection in various lighting conditions, with the person looking directly ahead: (a) good lighting; (b) poor lighting; (c) infrared lighting.

In addition to testing the face detection under varying lighting conditions, I have also introduced different head poses. I have achieved accurate facial detection in infrared light with the head turned sideways, as shown in Figure 29(a) as well as in cases where the head is not only turned sideways but it also has a negative roll, as shown in Figure 29(b).

Furthermore, the facial detection has also proven to be capable of detecting partially covered faces, as demonstrated in Figure 29(c), which was an attempt at mimicking a yawn.

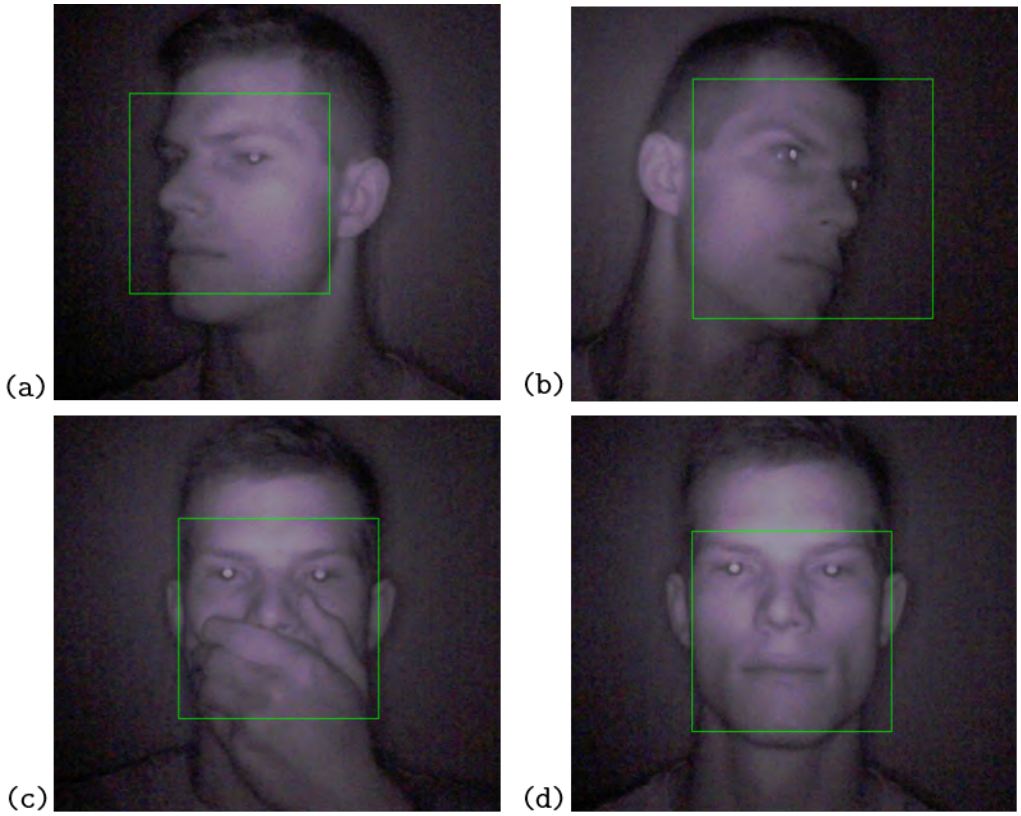

Figure 29: Succesful facial detection in infrared lighting: (a) head turned to the right; (b) head turned to the left with a slight negative roll; (c) face partially covered by a hand, mimicking a yawn; (d) face looking directly ahead.

Upon successfully testing the face detection module, I was able to integrate it with the main project without encountering any issues during integration testing.

In conclusion, the face detection module has proven to be almost 99% accurate when detecting faces in good and infrared lighting even with various head poses. However, in poor light if the face was too far away from the camera or the lighting conditions were severely bad, the system was unable to detect a face because it was unable to find the face outline in a mostly black image.

6.3 Facial Landmark Detection Testing

Testing of the facial landmark detection algorithm was conducted under three lighting conditions: good as shown in Figures 30(a) and 31; poor as shown in Figures 30(b) and 32; infrared as shown in Figures 30(c) and 33.

In order to prove that the 68 facial landmark points are annotated correctly onto each frame of the video, I ran the facial landmark detection algorithm for all the images in the iBUG-300W dataset and checked each annotation for errors. The algorithm has proven to be very accurate as it can be seen in Figure 30, where the annotations are accurate, but also the lighting conditions have not impacted the accuracy of the algorithm.

Tests performed under good lighting conditions, for a resting facial expression and a straight head pose as shown in Figures 31 and 30(a), have been a success and I was able to achieve 100% accuracy.

Upon completion of the testing under good lighting conditions, I have moved onto testing the solution in poor and infrared lighting on real-time video footage, as shown in Figures 30(c), 32 and 33, where I have extracted a single video frame for demonstration purposes.

The accuracy of the facial landmark predictor was not affected by the change of lighting conditions. However, in extreme circumstances, I have noticed that it was unable to detect the facial landmarks for videos recorded in extremely poor lighting. But, once the infrared module has been switched on, the landmarks were accurately detected.

Hence, the inability of detecting facial landmarks in poor lighting is only an issue if the sleep and drowsiness detection system is not equipped with an infrared illumination ring.

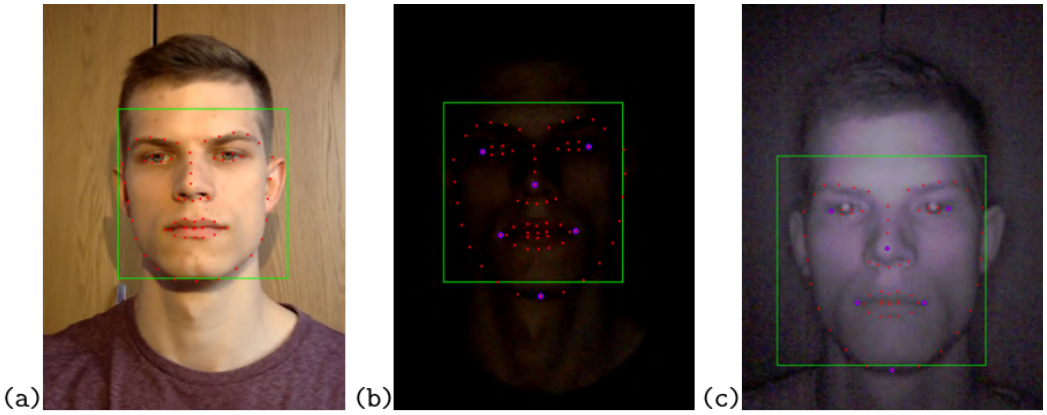

Figure 30: Side by side comparison of facial landmark detection for a single video frame, under different lighting conditions; (a) good lighting; (b) poor lighting; (c) infrared lighting.

In addition to varying the lighting conditions, I have also tested the facial landmark detection algorithm with various facial expressions. The testing has shown that the system accurately detects facial landmarks for facial expressions imitating a yawn as it can be seen in Figures 31(a) and 33(b).

The detection of abnormal facial expressions, in various lighting conditions has also proven to be very accurate as it can be seen in Figures 31(c), 33(a), 33(c) and 33(e).

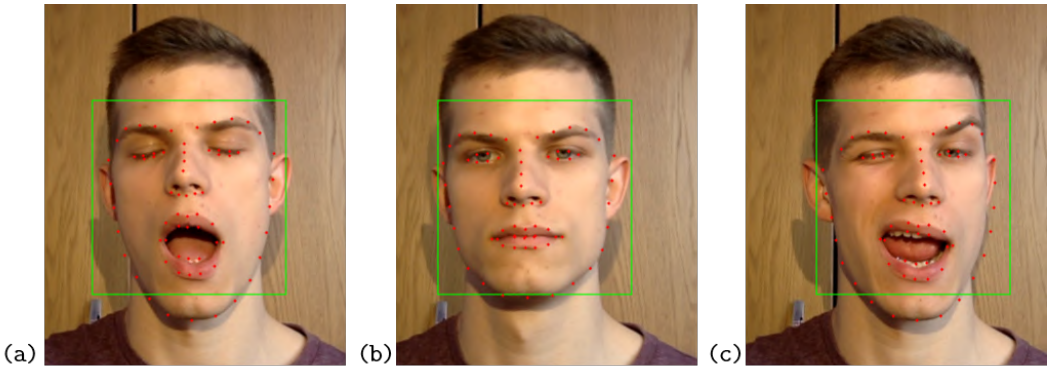

Figure 31: The result of facial landmark detection for a single video frame in good lighting with various facial expressions: (a) Detection of facial landmarks for a face imitating a yawn; (b) Detection of facial landmarks for a resting face; (c) Detection of facial landmarks for a face with an abnormal facial expression.

The most challenging aspect of facial landmark detection was the ability to accurately estimate the position of the facial landmarks if the head wasn't in a resting position. The head roll had no impact on the accuracy, however, if the head was turned or titled the accuracy varied significantly, especially when the head position was exposing less than 3/4 of the face to the camera.

I have managed to obtain accurate facial landmark detection on turned and tilted head in extremely poor lighting as shown in Figure 32. However, I was unable to obtain accurate facial landmark detection in poor lighting with a maximum head turn, but it was possible in infrared lighting as shown in Figure 33(d) and 33(e).

This shows that the use of an infrared illumination ring is necessary in order to maintain high accuracy of facial landmark detection in poor lighting. It's almost impossible to accurately detect facial landmarks for a head that has been turned almost by 90-degrees since most of the face is invisible to the camera.

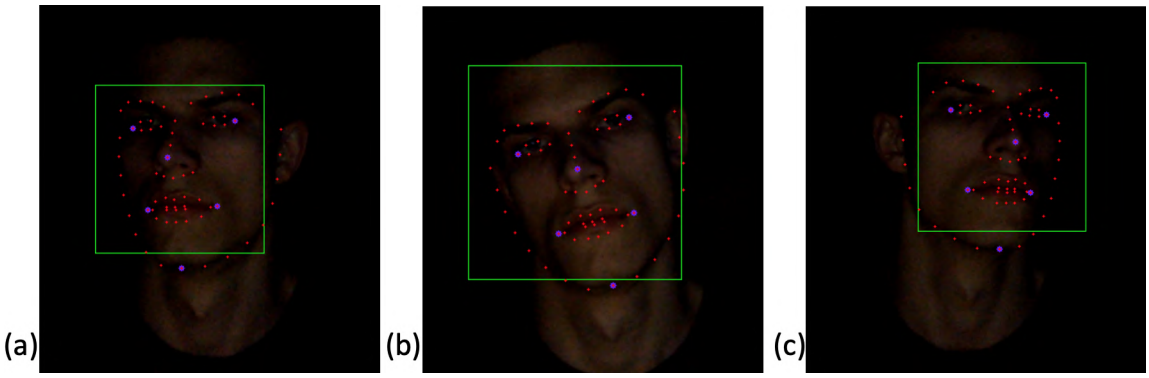

Figure 32: The result of facial landmark detection for a single video frame in poor lighting with various head poses: (a) Head turned to the right; (b) Head tilted to the right; (c) Head turned to the left.

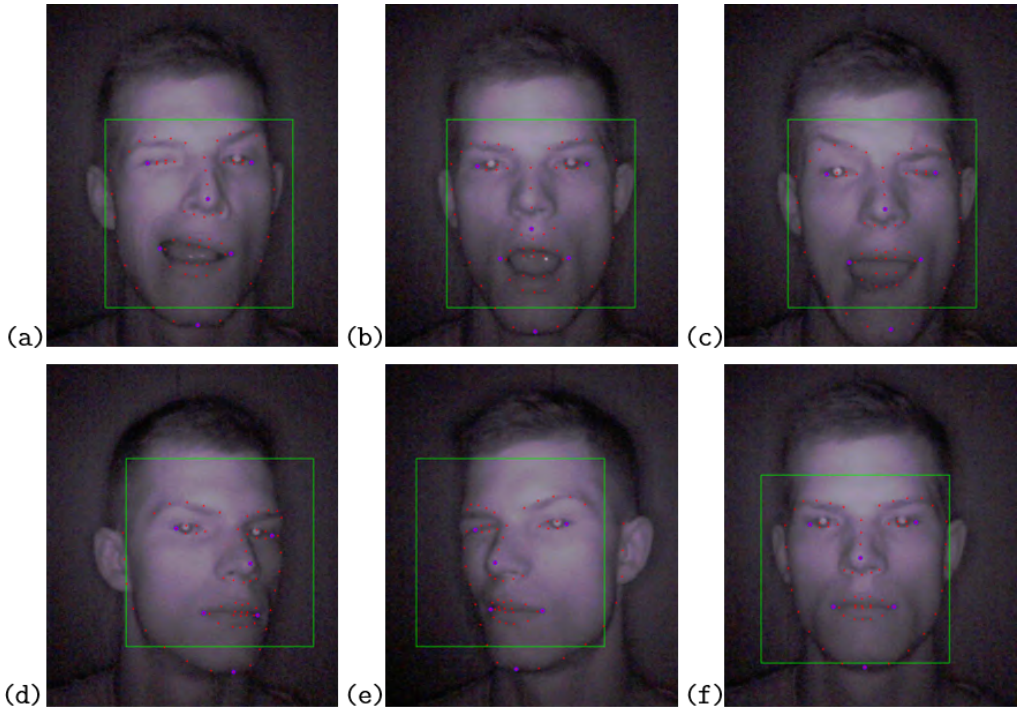

Figure 33: The result of facial landmark detection for a single video frame in infrared lighting with various facial expressions and head poses: (a) Neutral head pose with an abnormal facial expression; (b) Neutral head pose with a face imitating a yawn; (c) Neutral head pose with an abnormal facial expression; (d) Resting face with the persons head turned left; (e) Abnormal facial expression with the head turned right; (f) Neutral head pose with a resting facial expression.

During the integration of the facial landmark detection module with the rest of the system, there have been no unexpected issues. As the facial landmark detection doesn't collide with any of the face detection logic. In addition, both the facial landmark predictor and the facial detection modules used, are from the Dlib library. Therefore, the integration of those modules has been performed by the author of this library to ensure that the two work together and that there are no performance issues caused by running both at the same time.

6.4 Eye Aspect Ratio Testing

The testing of the eye aspect ratio has been performed under three lighting conditions: good lighting as shown in Figures 34(a) and 34(d); poor lighting as shown in Figures 34(b) and 34(e); infrared lighting as shown in Figures 34(c) and 34(f).

During the integration of the EAR module, there have been no issues. Since this module doesn't interfere with the facial detection nor the facial landmark detection modules and it only makes use of the data supplied by the facial landmark predictor.

The purpose of the initial tests was to estimate the accuracy of the calculated average eye aspect ratio for fully open and fully closed eyes, in various lighting conditions with a neutral head pose as well as resting facial expression as it can be seen in Figure 34.

As it can be seen in Figures 34(a) and 34(b) the average EAR values are approximately the same and a difference of 0.01 is normal due to slight variation in landmark detection caused by different lighting conditions. The average EAR value significantly decreases to 0.24 during tests performed in infrared lighting as it can be seen in Figure 34(c), this is a major difference and further testing has identified that the infrared light illuminates the skin around the eye region reducing the shadows which make the eye region seem smaller.

Whilst testing the average EAR values for closed eyes as it can be seen in Figures 34(d), 34(e) and 34(f), the average EAR value started off at 0.9 and increased up to 0.11, this is a minor difference considering the major changes in the lighting conditions between tests. Overall, the average EAR values have been calculated accurately, and the system will have to adjust to the slight variations in average EAR values as the drivers face changes over time with the lighting conditions.

Figure 34: The result of calculating the average eye aspect ratio for a resting facial expression in various lighting conditions with closed and fully open eyes: (a) good lighting with fully open eyes and an average EAR value of 0.28; (b) poor lighting with fully open eyes and an average EAR value of 0.29; (c) good lighting with fully open eyes and an average EAR value of 0.24; (d) good lighting with fully closed eyes and an average EAR value of 0.09; (e) poor lighting with fully closed eyes and an average EAR value of 0.10; (f) infrared lighting with fully closed eyes and an average EAR value of 0.11.

Upon proving that the average EAR values are calculated accurately for a resting facial expression and a neutral head pose, I have moved onto testing with a tilted head.

A heavily tilted head as shown in Figures 35(a) and 35(b) makes the eyes of an individual look smaller due to the angle at which the camera is placed. Therefore, from the Figure 35(a) we can see that the biggest change of 0.07 in respect to the average EAR value from Figure 34(c) is introduced when an individual tilts the head backwards. So, it will be necessary to accurately estimate the head pose in order to avoid false positive results. But, the value of 0.17 from Figure 35(a) is still not close enough to the value from Figure 34(d) that would indicate that the eyes are fully closed.

There's a minor difference of 0.02 in the average EAR value for a head tilted forwards as shown in Figure 35(b) in respect to a straight head as shown in Figure 34(c). Such a difference doesn't cause any issues as the system would be expecting values in the region of 0.28 as shown in Figure 34(a).

Therefore, overall the average EAR values have been calculated accurately despite the variation in lighting conditions and various head poses.

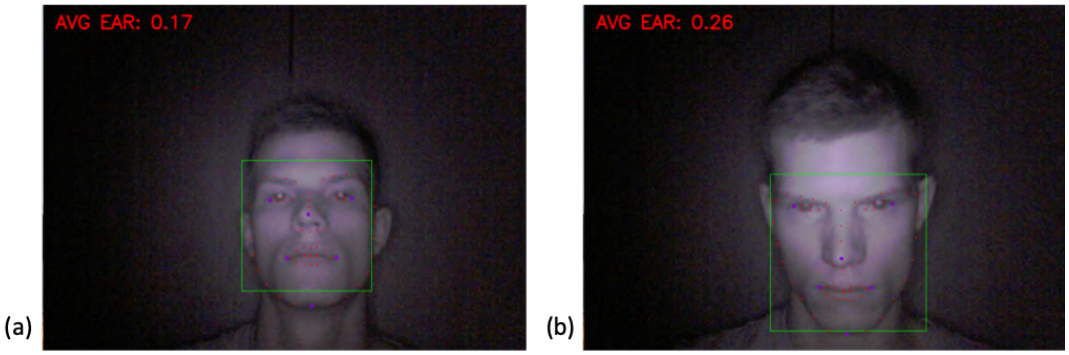

Figure 35: The result of calculating the average eye aspect ratio for a resting facial expression in infrared lighting for a tilted head: (a) head tilted backwards with an average EAR value of 0.17; (b) head tilted forwards with an average EAR value of 0.26.

6.5 Head Pose Estimation Testing

The head pose estimation testing had been conducted in good, poor and infrared lighting conditions as well as for various angles of head roll, yaw and pitch.

The detection of a negative and positive head roll, despite the big changes in lighting conditions has been accurate, as it can be seen in Figure 36. The estimation of the head roll has been significantly more accurate than the estimation of a head pitch as shown in Figure 38 and head yaw as shown in Figure 37. The head roll has been easier to estimate since this head pose doesn't hide any of the landmarks and the distance between landmark points is constant as the face is still positioned perpendicularly to the camera.

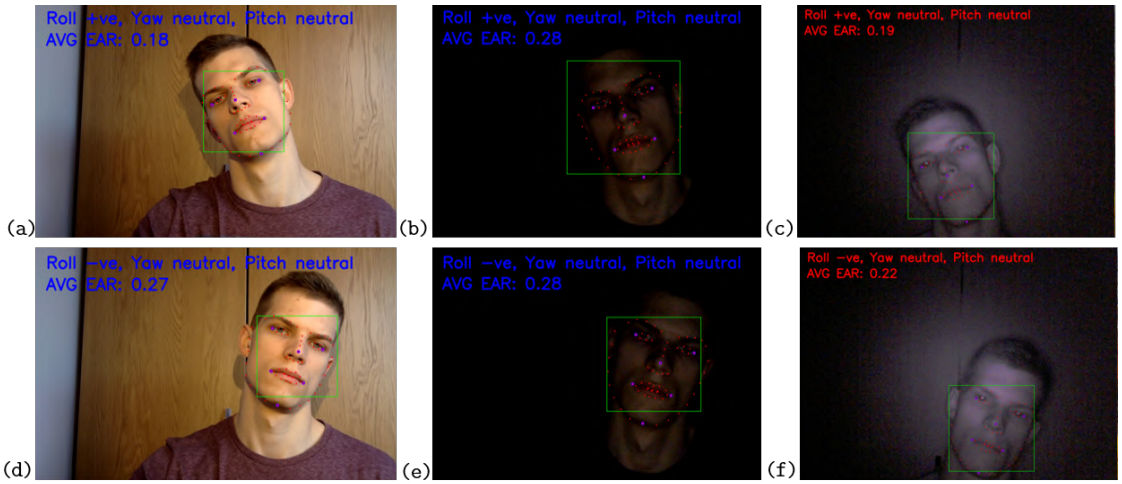

Figure 36: The result of estimating positive and negative head roll in various lighting conditions: (a) positive head roll in good lighting; (b) positive head roll in poor lighting; (c) positive head roll in infrared lighting;(d) negative head roll in good lighting; (e) negative head roll in poor lighting; (f) negative head roll in infrared lighting.

The estimation of a negative, positive and neutral head yaw has been very accurate in various lighting conditions, as it can be seen in Figure 37. During the testing of the head yaw, a neural head roll, as well as a neutral head pitch, has been tested, as shown in Figures 37(b), 37(e) and 37(h).

The system will ignore any changes to head roll whilst the system is detecting a positive or a negative head yaw, as it can be seen in Figures 37(a), 37(d), 37(g), 37(c), 37(f) and 37(i). Hence why the neutral head roll has been tested whilst testing neutral head yaw. In addition, a neutral head yaw means that the head pitch is also neutral as it can be seen in Figures 37(b), 37(e) and 37(h).

Testing of a negative and positive head yaw has been challenging. As when the head is being turned away from the camera, the facial landmarks on one side of the face are being hidden by the side of the face that is being fully exposed to the camera. In order to avoid false positive result, I have estimated the maximum angle of the head yaw that can be accepted and anything past this angle range is ignored as explained in section 5.5.

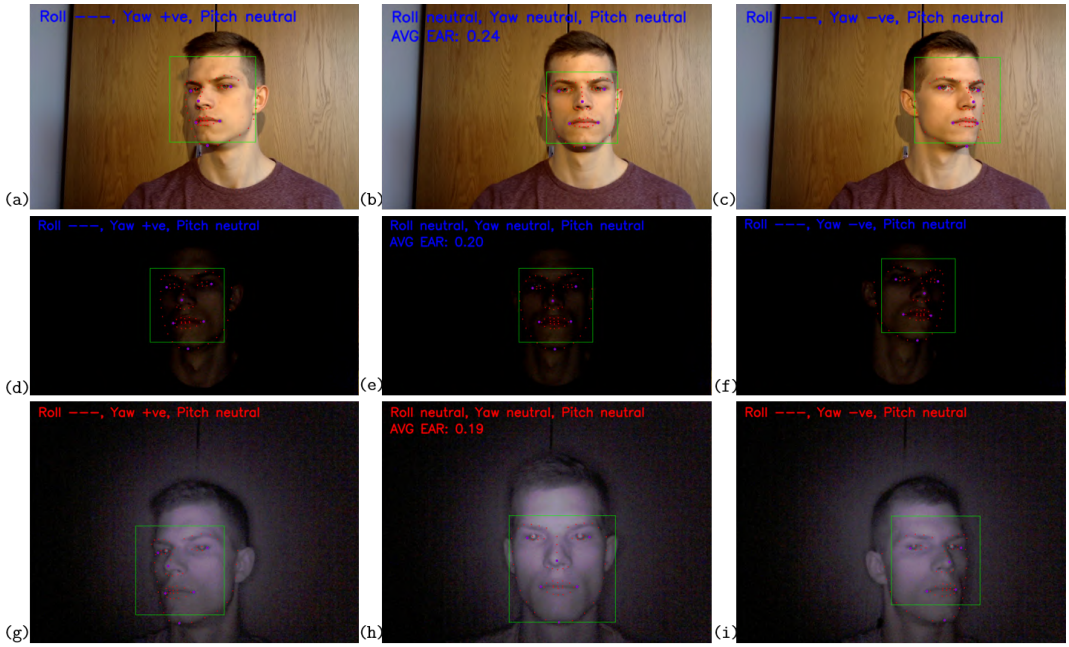

Figure 37: The result of estimating positive, negative and neutral head yaw in various lighting conditions: (a) positive head yaw in good lighting; (b) neutral head yaw in good lighting; (c) negative head yaw in good lighting; (d) positive head yaw in poor lighting; (e) neutral head yaw in poor lighting; (f) negative head yaw in poor lighting; (g) positive head yaw in infrared lighting; (h) neutral head yaw in infrared lighting; (i) negative head yaw in infrared lighting.

The head pose estimation of a positive and a negative head pitch is very accurate, even in extreme lighting conditions as shown by Figures 38(c) and 38(d). Whilst testing the estimation of a head pitch in good, poor and infrared lighting as shown in Figure 38, I have experienced difficulties in detecting positive head pitch in extremely poor lighting. Since, as the head is tilted backwards in extremely poor lighting the eye region becomes invisible as it moves away from the camera.

However, whilst testing in infrared lighting I have proven that the estimation of extreme head pitch angles is accurate as shown in Figures 38(e) and 38(f). Throughout testing, the estimation of a negative head pitch has been very accurate and there have been no issues. The lack of issues in detecting negative head pitch was because as the head is tilted forwards, the facial landmarks become clearer to the camera as the top of the head moves towards the camera but the rest of the face remains in the same position.

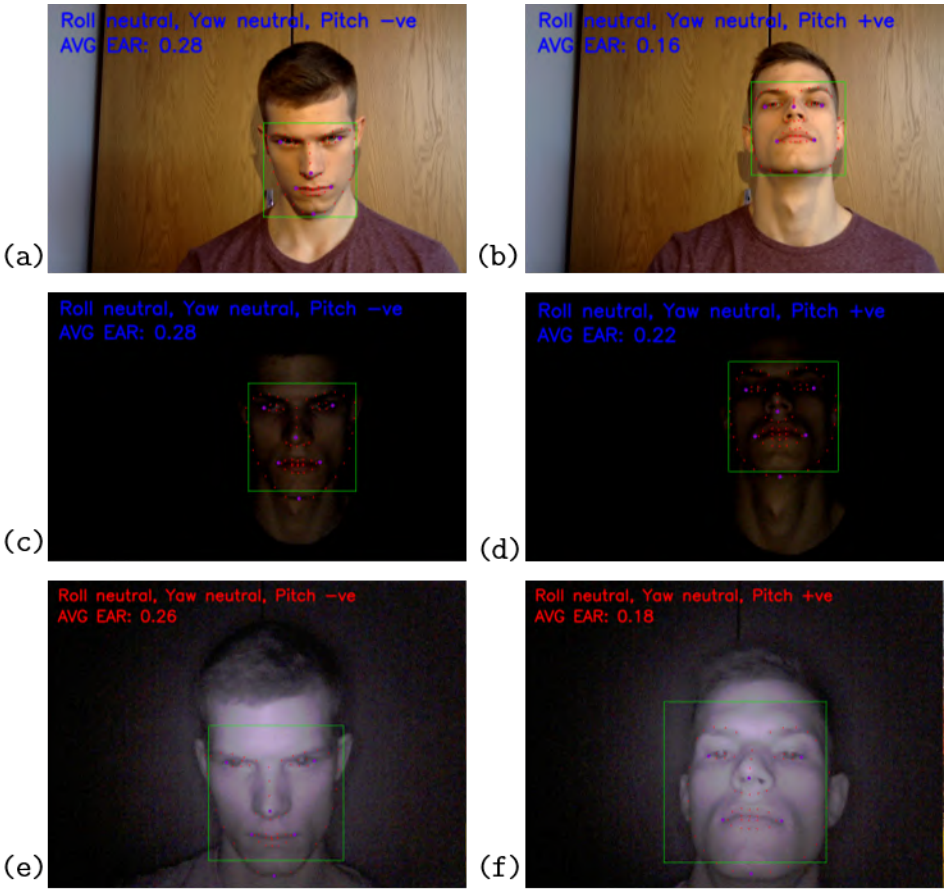

Figure 38: The result of estimating positive, negative and neutral head pitch in various lighting conditions: (a) negative head pitch in good lighting; (b) positive head pitch in good lighting; (c) negative head pitch in poor lighting; (d) positive head pitch in poor lighting; (e) negative head pitch in infrared lighting; (f) negative head pitch in infrared lighting.

Overall the estimation of a head pose has been very accurate in various lighting conditions, and even extreme angles have been successfully detected. The integration of this module with the rest of the system has been simple and no issues were encountered. This module only uses the predicted facial landmarks and doesn't collide with any of the remaining logic in this system and hence it could be easily used in other projects where head pose estimation is necessary.

6.6 Eye Blink Detection Testing

Testing of the eye blink detection, was based on checking if the recorded EAR values meet the dynamically calculated EAR threshold during eye blinks. The Figure 39 demonstrates dynamic threshold adjustment over 1000 video frames, each threshold adjustment takes place every 101 video frames which is approximately 6 seconds in my testing environment.

In Figure 39 there are 10 threshold adjustment iterations, overall the EAR values remain steady and there are no outliers, on average there are 3 eye blinks per iteration. The eye blink detection module has achieved an accuracy of a 100% by detecting all of the 36 eye blinks.

If the system doesn't detect eye blinks in an iteration, the application of the Savitzky–Golay filter to the EAR values, will cause the EAR threshold to reduce in the next iteration. In result, the reduced EAR threshold will allow for the detection of eye blinks in the coming iterations.

In conclusion, the eye blink detection based on the analysis of EAR and its threshold has proven to be very accurate. On average I have been able to detect 98% of the eye blinks using this method. The 2% remains undetectable due to being heavily distorted by noise from the facial landmark detection as well as outside factors such as lighting conditions.

The 2% of eye blinks that remain undetectable by this method are mostly rapid eye blinks, such as the 3rd eye blink of the 7th iteration in Figure 39. However, in this case the rapid eye blink has been registered accurately. A rapid eye blink is prone to incorrect detection caused by the small change in the EAR value which sometimes due to noise won't be able to meet the set EAR threshold.

However, an incorrect eye blink detection is not very common and often only affects a single eye blink once every 60 seconds. Such accuracy is acceptable and won't have a strong impact on the accuracy of the sleep and drowsiness detection, as the system will be measuring the properties of several consecutive eye blinks, rather than a single eye blink.

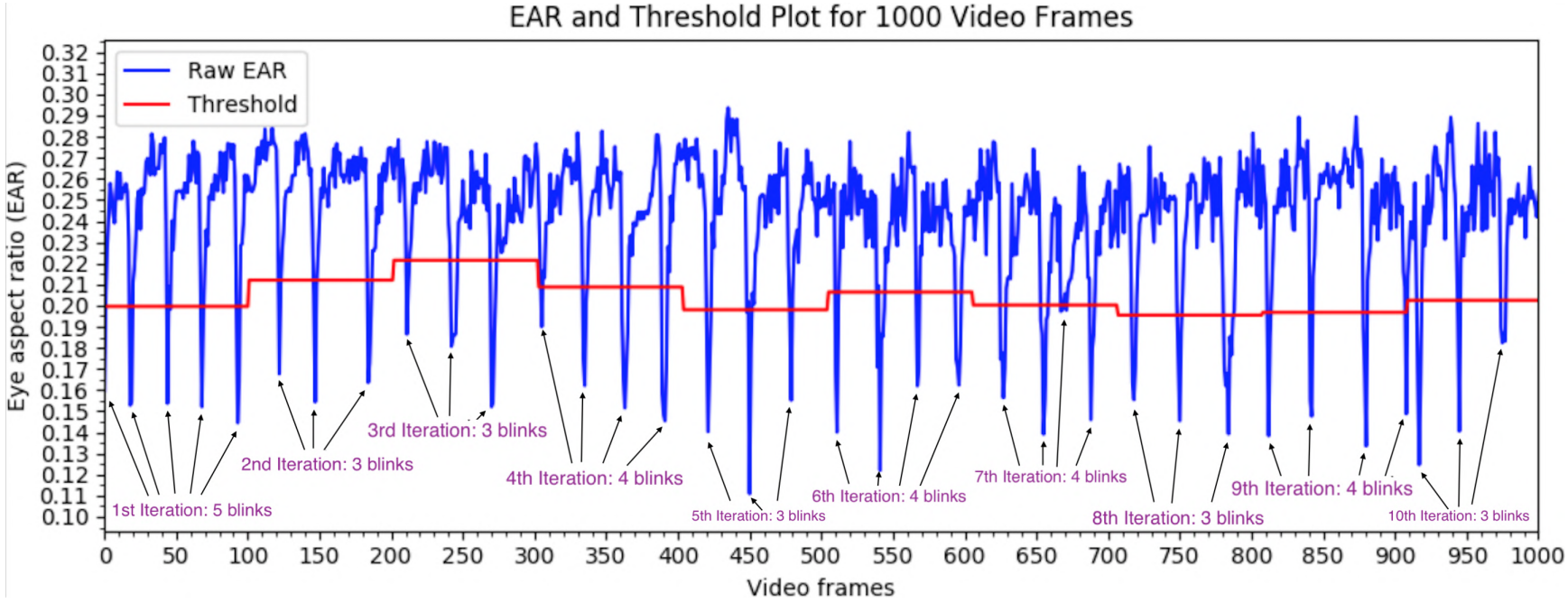

Figure 39: The EAR measure and its threshold plotted over an observation period of 1000 video frames, with 10 threshold iterations and 36 eye blinks, for a driver in an awake state.

6.7 Sleep and Drowsiness Detection Testing

The testing of sleep and drowsiness has been conducted for a driver in an awake, drowsy and a sleeping state. For each state, the EAR values have been recorded over a sleep and drowsiness detection iteration which lasts 45 video frames, approximately 3 seconds in this testing environment. The accuracy of the sleep and drowsiness detection is dependant on the accuracy of blink detection. However, the PERCLOS calculation based on the recorded EAR values and their threshold is 100% accurate as this calculation is based on pre-processed data.

As mentioned in Section 5.7, on average an awake person will have an eye blink duration of approximately 0.2 to 0.3 seconds, this is approximately 4.5 video frames in this testing environment. I have started testing by analysing the drowsiness level of an awake driver. For the purpose of this test I have extracted one of the sleep and drowsiness detection iterations, which is used to demonstrate the eye blink behaviour as well as the accuracy of PERCLOS calculations.

The Figure 40 shows a single sleep and drowsiness detection iteration for an awake driver. The plotted EAR values in this iteration show two eye blinks, one at the beginning of the iteration, and a second eye blink at the end of the iteration, for both eye blinks the duration cannot be estimated.

However, the analysis of this graph shows that, during this iteration there have been no rapid eye blinks and no long eye blinks, such as the ones detected for a drowsy driver in Figure 41. Overall, the system calculated that for this iteration, the awake driver has a PERCLOS value of 30%, this is 10% less than the drowsiness boundary. Therefore the prediction is 100% accurate since, the driver was fully awake for this test.

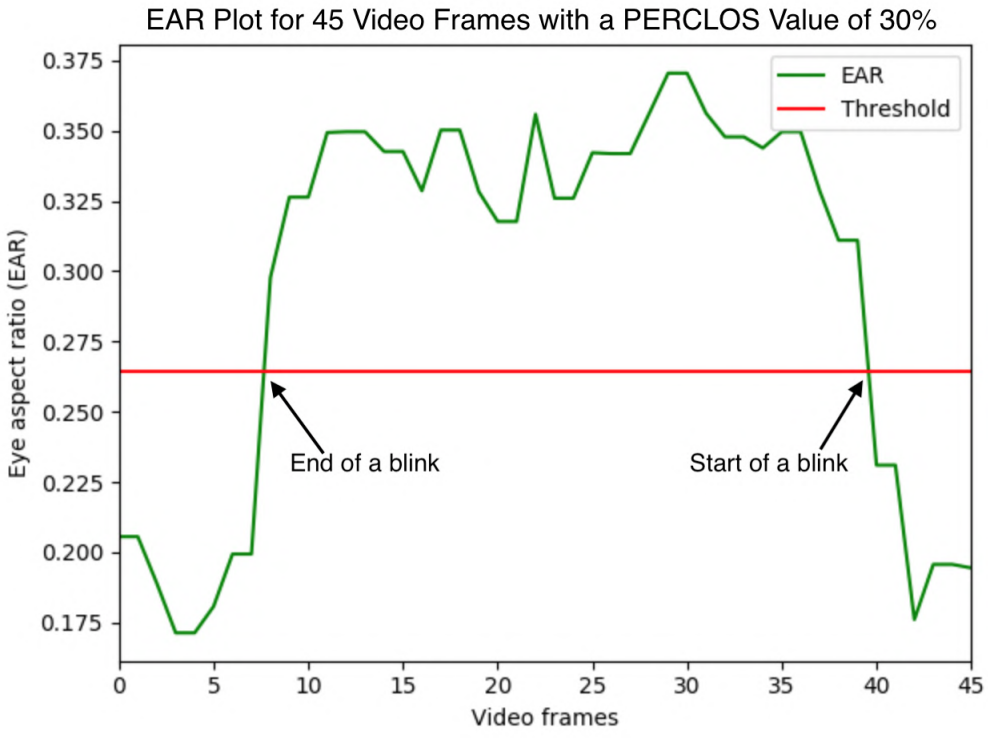

Figure 40: The EAR measure plotted over an observation period of a single sleep and drowsiness detection iteration with a PERCLOS value of 4%, for a driver in an awake state.

The Figure 41, shows a single sleep and drowsiness detection iteration for a drowsy driver. In this iteration the plotted EAR values show two rapid eye blinks with a longer that average duration, this is an indication of driver drowsiness. The PERCLOS value for this iteration is 43%, this indicates that this driver is drowsy. The prediction that the driver is drowsy is accurate and the PERCLOS value has been calculated correctly.

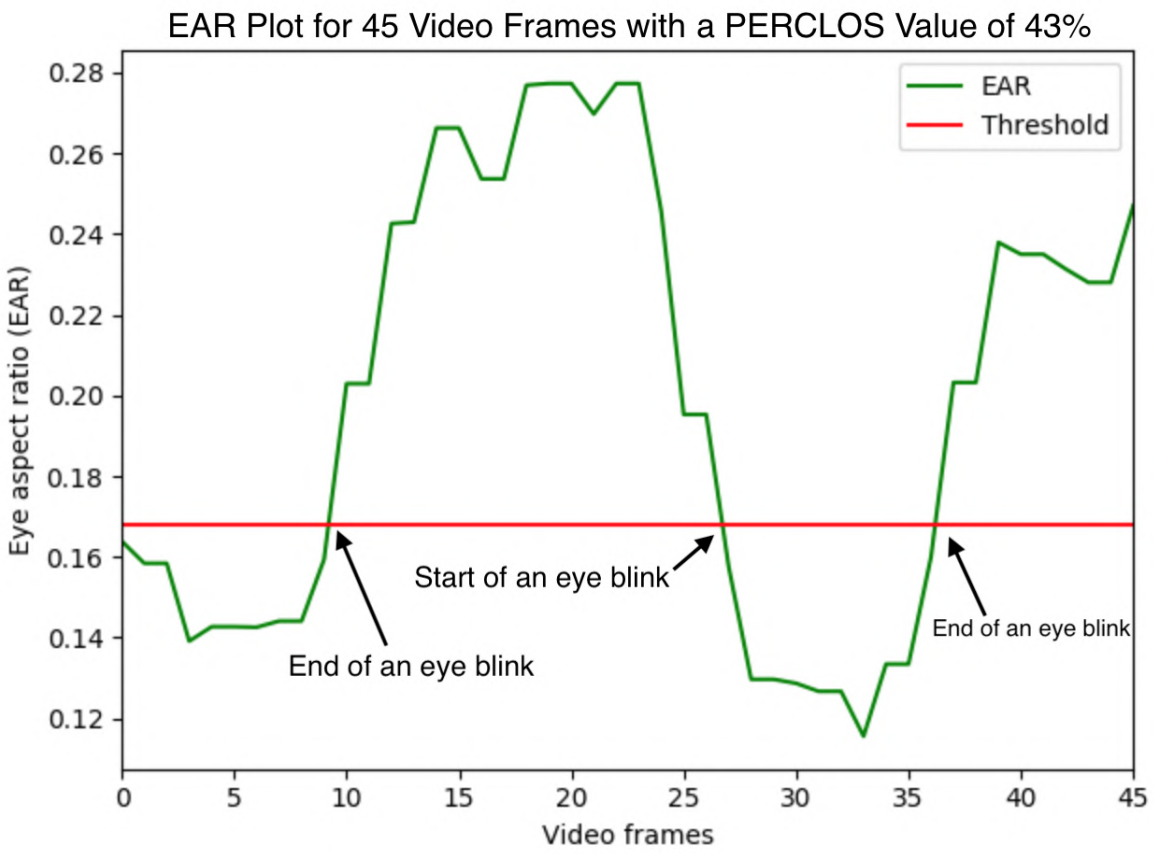

Figure 41: The EAR measure plotted over an observation period of a single sleep and drowsiness detection iteration with a PERCLOS value of 43%, for a driver in a drowsy state.

Upon proving that the sleep and drowsiness detection, is capable of accurately distinguishing between a drowsy and an awake driver, I have moved onto testing the system with a sleeping driver. The Figure 42 shows an EAR plot for a driver who is completely asleep. In this case the analysis of the graph is simple, as the EAR values are clearly below the threshold with no blinks present.

Therefore, the calculated PERCLOS value of a 100% is accurate, proving that the system is capable of detecting a driver who is completely asleep. In addition to proving that the driver was completely asleep during this test, I was also able to demonstrate the noise introduced by facial landmark detection and how it causes the EAR values to fluctuate, even when the eyes are completely shut.

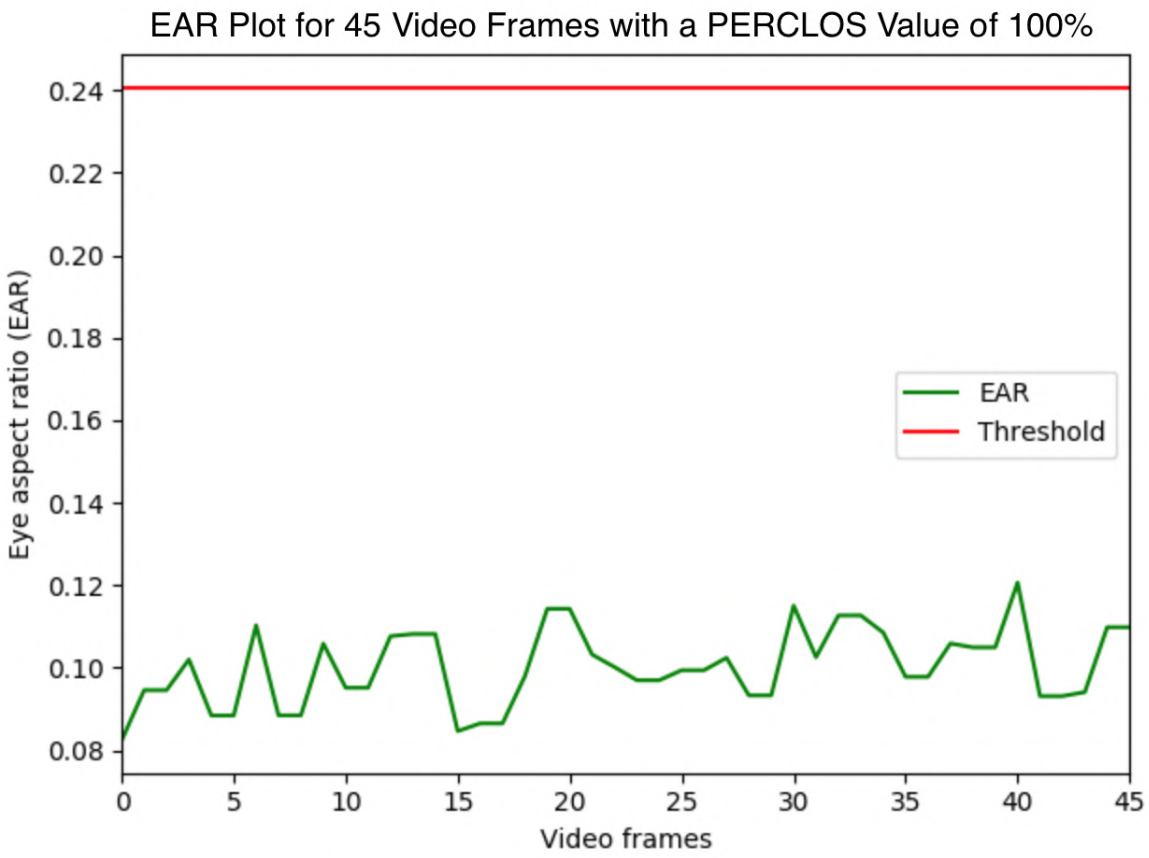

Figure 42: The EAR measure plotted over an observation period of a single sleep and drowsiness detection iteration with a PERCLOS value of 100%, for a driver in a sleeping state.

For the last test, I have tested the system with data of a driver who is severely drowsy and is about to fall asleep. The Figure 43, demonstrates a single very long eye blink which is also a sign of micro-sleep. In this instance the value of PERCLOS was 91% which is significantly higher than the threshold of 70%. Therefore, the system was capable of accurately detecting a sleeping as well as a partially sleeping driver.

Overall, sleep and drowsiness detection based on the evaluation of the PERCLOS value for an iteration of 45 video frames, has proven to be very effective and accurate at distinguishing between an awake, drowsy and a sleeping driver.

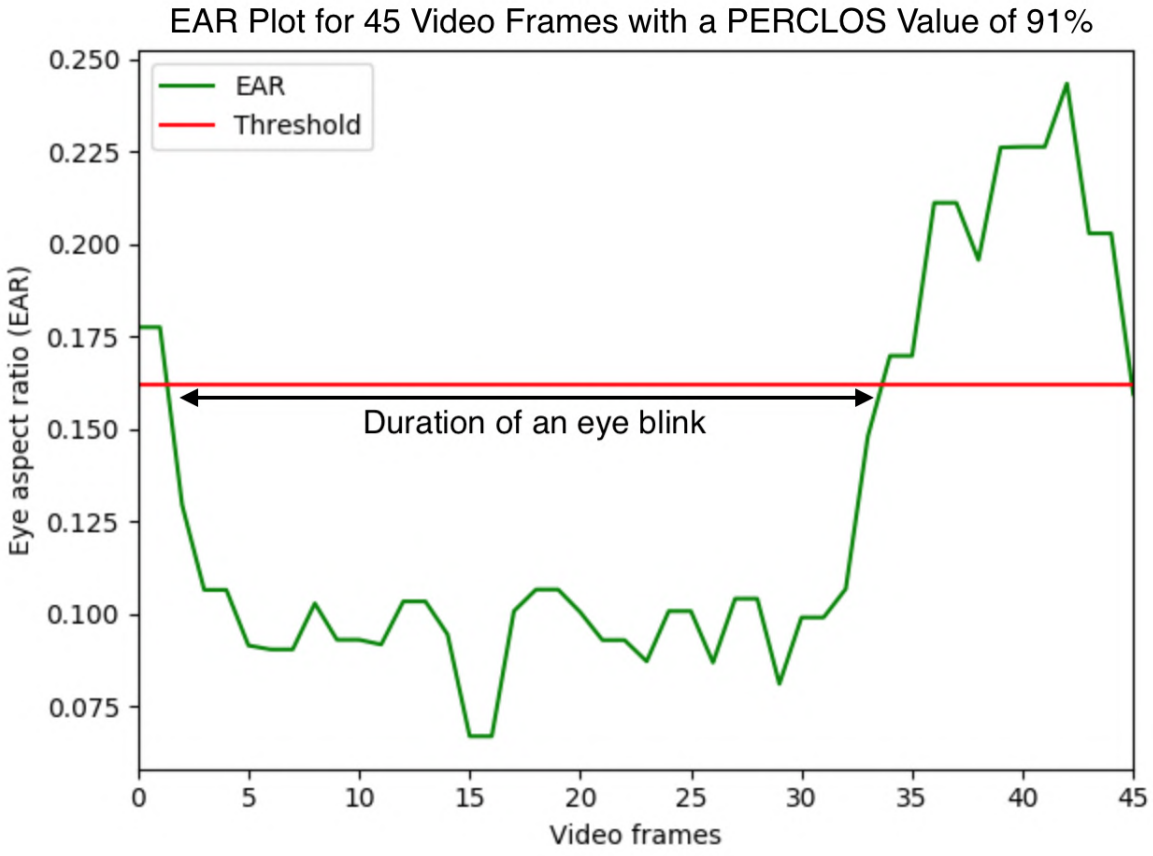

Figure 43: The EAR measure plotted over an observation period of a single sleep and drowsiness detection iteration with a PERCLOS value of 91%, for a driver in a sleeping state.

6.8 Testing Evaluation

I believe that testing of the sleep and drowsiness detection system and its subcomponents was sufficient enough to prove that the system works and meets all of the system requirements. The testing performed proves that the system is capable of an accurate sleep and drowsiness detection from the analysis of drivers facial landmarks, in various lighting conditions.

It has been extremely challenging and time-consuming to accurately test sleep and drowsiness detection. Since, the conditions for testing had to be optimal in order to ensure that there are no outliers in the recorded data, purely caused by factors such as constantly varying lighting conditions.

In addition, when testing sleep and drowsiness detection I have spent two days playing driving simulator games, to ensure that the system is accurate. During this testing, I have kept a log of how I feel, to ensure that when I am analysing the produced results by the sleep and drowsiness detection system, it is accurate and that it matches with what I have stated in my log.

Overall the system has been thoroughly tested, but there are aspects of testing that I would have changed if I was to do this project again. The budget and time constraints have certainly limited testing of this project. If I had more time, and a bigger budget, the main change that I think would be necessary, is the use of a driving simulator in a room with variable lighting conditions, where the outside light cannot affect the recorded data. This could significantly reduce any noise from the facial landmark detection and varying lighting conditions.

In addition, the use of a driving simulator would allow me to test the system in various lighting conditions at any time of the day, without being restricted to natural lighting. The use of a driving simulator could potentially allow for a small study group, that could be used to collect the necessary data to make this system accurate for any kind of face, facial expression as well as head pose.

In conclusion, the system had been thoroughly tested and is capable of accurate sleep and drowsiness detection. However, given the time and budget constraint, it was impossible to introduce more robust testing techniques and therefore, some aspects of the testing could significantly be improved in the future, given a bigger budget as well as time frame.

7 Results and Evaluation

In this section, I will present and explain the achieved results of the implemented real-time sleep and drowsiness detection system. I will also analyse and make observations on the accuracy of the said system.

The accuracy of the sleep and drowsiness detection system is measured by the ability to accurately estimate the PERCLOS value for a driver in an awake, drowsy and a sleeping state.

The implemented system is capable of accurately distinguishing between an awake, drowsy and a sleeping state, as shown in Figures 44 and 45. The results obtained are based on the analysis of the PERCLOS values for a person playing a driving simulator game.

The first result is three PERCLOS plots over an observation time of 100 minutes, for a driver in an awake, drowsy and a sleeping state as shown in Figure 44. This result demonstrates the systems ability to accurately distinguish between an awake, drowsy and a sleeping state.

Analysis of the result in Figure 44 shows that for an awake driver, the system recorded a stable PERCLOS plot between 10% and 30% for the first 48 minutes. The PERCLOS values begin to increase from the 48th minute onwards, and start to fluctuate between 18% and 35%. An increase in the PERCLOS value indicates that the monotonous activity of playing a driving simulator has made the driver tired. If the observation period was more than 100 minutes, it is very likely that the PERCLOS values of this driver would have met the drowsiness threshold value of 40%.

Furthermore, the results in Figure 44 demonstrate clear boundaries between the PERCLOS values of a driver who is awake, drowsy or sleeping. It can be observed that the fluctuating PERCLOS values for a drowsy driver are closer to the sleeping threshold of 70% rather than the drowsy threshold of 40%. Overall, all three plots of the PERCLOS values gradually increase in the 10 minute period, between the 40th and 50th minute of the observation.

In Figure 44, the plot of the PERCLOS values for a sleeping driver doesn't reach the mark of 100%, and overall it fluctuates a lot. The fluctuating PERCLOS values are a result of the noise introduced by facial landmark detection. During the observation, the driver was extremely tired from the 50th minute onwards but was still capable of playing the driving simulator.

Overall, the ability of the system to distinguish whether the driver is awake, drowsy or sleeping is satisfactory. In order to show that the system is capable of detecting all three states during one observation, I have conducted a second observation with a duration of 200 minutes.

Figure 45 shows a PERCLOS plot for a 200-minute observation, of a driver playing a driving simulator game. The analysis of this PERCLOS plot has shown that the system is capable of accurately detecting the gradual changes between the three states.

Figure 45 shows that the driver is starting to feel drowsy as the PERCLOS values start to gradually increase from the 70th minute. From the 130th and 165th minute, the system accurately estimated that the driver is feeling drowsy and that from the 165th minute onwards the driver is starting to fall asleep.

Overall, the implemented system has proven to be very accurate at detecting drowsiness levels over a long period of time. The results have shown no false positive predictions as well as the lack of outliers. This is very satisfying, as it proves that the system has been tested well and that any noise introduced by facial landmark detection was effectively removed from the recorded EAR data.

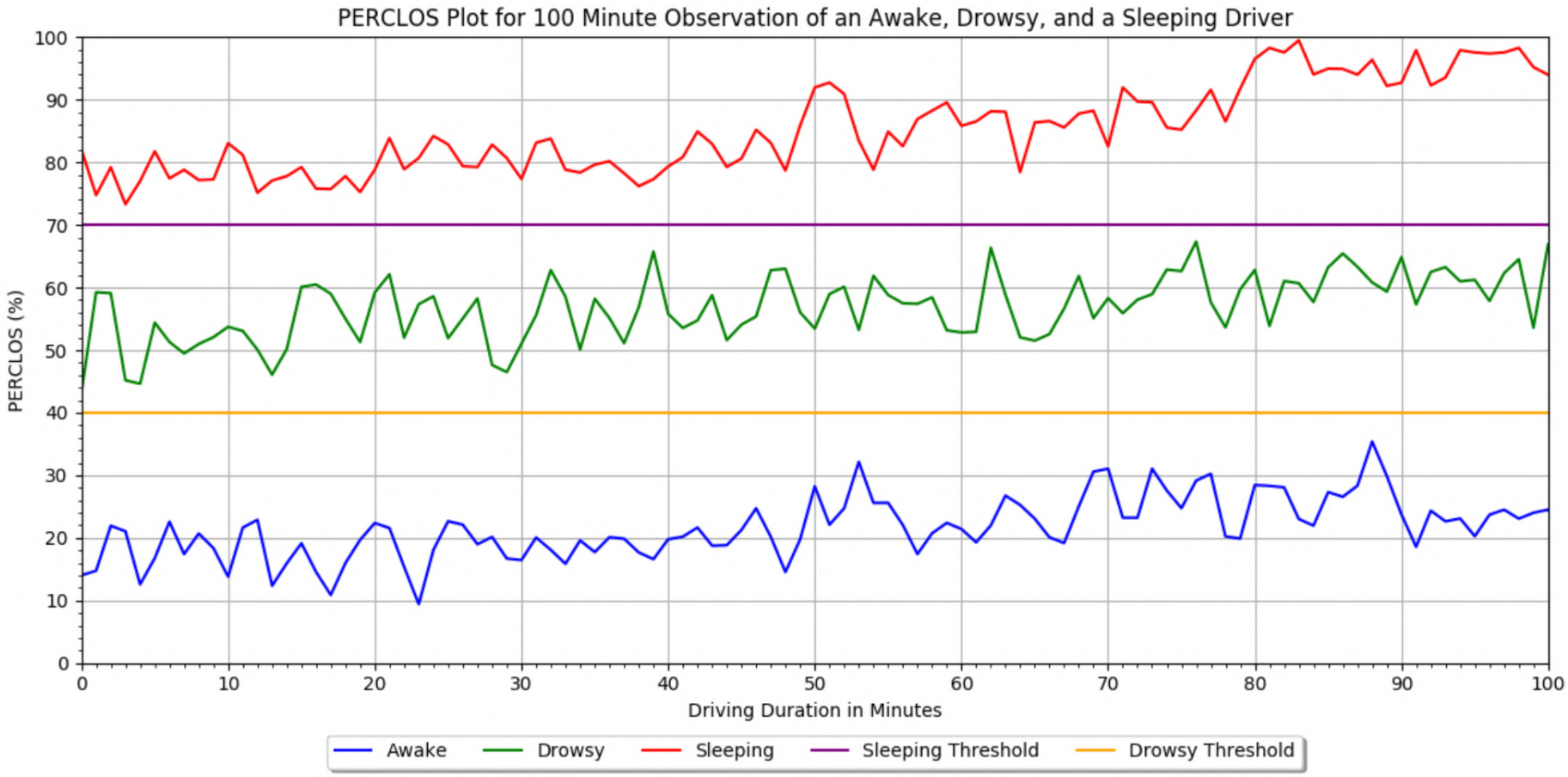

Figure 44: The PERCLOS measure plotted over an observation period of 100 minutes, for a driver in an awake, drowsy and a sleeping state.

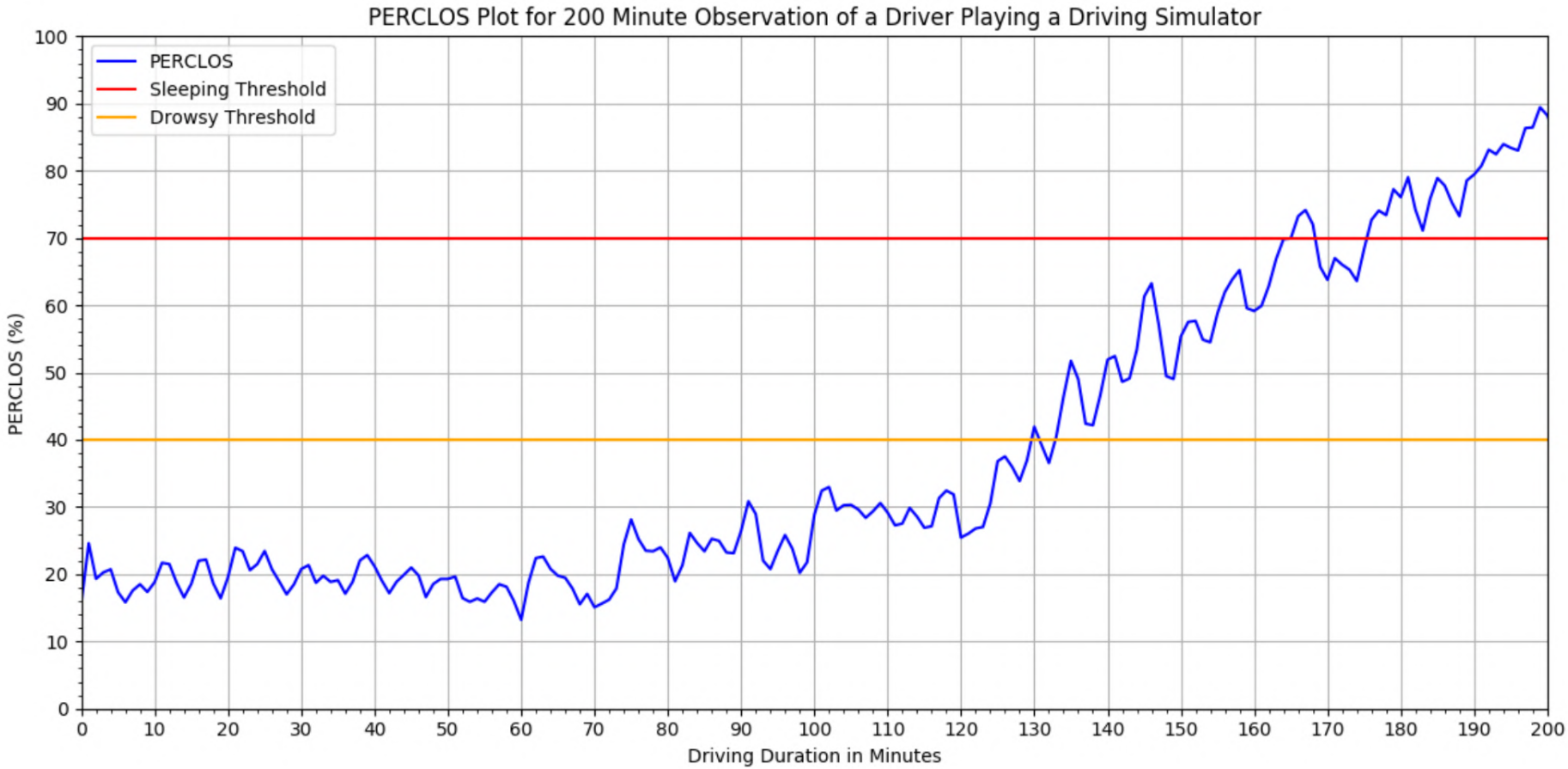

Figure 45: The PERCLOS measure plotted over an observation period of 200 minutes, for a driver who is playing a driving simulator. This graph shows a gradual increase in the PERCLOS value over time, for a driver who is slowly falling asleep.

7.1 Evaluation of Project Management

In this section I will evaluate the effectiveness of my project schedule which has been described in depth in Section 3.1, and shown in Figures 9 and 10.

Throughout the development of this project, I have managed to strictly follow the set time frames and meet every single major milestone on time. The division of the project schedule into 12 sprints, has allowed me to accurately plan my week and focus on completing modules, rather than the whole implementation at once.

I have found that the allocated time for each task was sufficient, this allowed me to move onto the next task without having any overdue tasks affect my schedule or the quality of my work. However, during the final documentation sprint, I have gotten overly ambitious and spent too much time on some sections of the document. In addition, I have underestimated the amount of time it would take to summarise the testing results. Therefore, I have had to use the time frame allocated to unforeseen challenges.

In conclusion, I have strictly followed the project schedule and was able to finish this project on time before the deadline, without having any overdue tasks or neglecting any aspects of this project.

7.2 Evaluation of Project Aim and Objectives

In this section I will evaluate the results obtained against the project aim and the six objectives.

O.1) Objective 1 has been met. Since, the system is capable of detecting facial landmarks in good and poor light conditions.

O.2) Objective 2 has been met. Since, the system is capable of detecting when a driver is drowsy, which results in an audible warning.

O.3) Objective 3 has been met. Since, the system is capable of detecting when a driver is sleeping, which results in an audible warning.

O.4) Objective 4 has been met. Since, the system is capable of an accurate blink detection in real-time which is used for calculating the PERCLOS value.

O.5) Objective 5 has been met. Since, the system is capable of accurate head pose estimation in real-time.

O.6) Objective 6 has been met. Since, the system is capable of analysing drivers facial features to detect sleep and drowsiness for a time period of at least 100 minutes, with an accuracy of 98% as shown in Figures 44 and 45.

The aim of this project is:

> *To develop a system that is capable of real-time driver drowsiness and sleep detection by analysing facial features.*

This aim has been broken down into six objectives, that when met, will determine whether the implemented system has met the project aim. The system implemented meets all six objectives and in result it achieves the aim of this project.

7.3 Evaluation of System Requirements

In this section I will identify and evaluate which system requirements have been met and if any have been neglected.

R.1) This requirement has been met since the system uses Dlib HoG Face Detector, that is capable of accurately detecting drivers face despite the variations in lighting, facial expression and head pose.

R.2) This requirement has been met since, the system hardware consists of an infrared illumination module, which consists of 48 LEDs and a light sensor, that automatically switches the LEDs on when the amount of light is insufficient to detect the drivers face.

R.3) This requirement has been met since the system consists of the eye aspect ratio module, the head pose estimation module and the sleep and drowsiness detection module.

R.4) This requirement has been met since, the system uses Dlibs facial landmark predictor to achieve accurate 68-point facial landmark detection in real-time, without being affected by varying lighting conditions.

R.5) This requirement has been met since, the system is capable of calculating the EAR value for the left and right eye, as well as the average EAR value, using the 12 coordinates identified by the facial landmark detection, labelled 37 to 48 as shown in Figure 7.

R.6) This requirement has been met since, the system is capable of filtering the collected EAR values using the Savitzky–Golay filter and then calculating the lower quartile value (EAR Threshold).

R.7) This requirement has been met since, every 6 seconds the system recalculates the EAR threshold to ensure blink detection is accurate, despite the variations in lighting, facial expression or head pose.

R.8) This requirement has been met since, the system is capable of accurate blink detection in real-time, based on the average EAR and EAR threshold.

R.9) This requirement has been met since the system is capable of accurate head pose estimation based on the position of facial landmarks. This includes the ability to accurately estimate whether yaw, pitch or roll is negative, positive or neutral.

R.10) This requirement has been met since, the system discards any EAR values collected whilst, head yaw is negative or positive.

R.11) This requirement has been met since, the system is capable of measuring the percentage of eye closure, for a time period of 3 seconds based on the recorded EAR values.

R.12) This requirement has been met since, the system is able to detect when the driver is feeling drowsy or is sleeping, by using the calculated PERCLOS value.

R.13) This requirement has been met since, when the system identifies that the driver is drowsy or sleeping, the system issues an audible warning in an attempt to waken the sleeping driver.

R.14) This requirement has been met since, the system runs flawlessly on a machine with a macOS Sierra operating system, at least 8GB of RAM, 5GB of free storage and an equivalent to a 2.8 GHz Intel Core i5 processor (4308U).

In conclusion, the implemented system has met all of the 14 system requirements. Therefore, the system has been developed to a high standard and the desired sleep and drowsiness detection methods have been successfully implemented.

7.4 Results and Evaluation Conclusion

The ability of this implemented system, to accurately estimate the PERCLOS value for a driver in an awake, drowsy and a sleeping state, is very high, despite the various issues I have faced with the removal of the noise introduced by facial landmark detection.

The obtained results are shown in Figures 44 and 45, and prove the accuracy of the system in detecting sleep and drowsiness over a long period of time, despite the constant variations in lighting, drivers head pose or facial expressions.

Overall, the project has been well planned and managed as discussed in Section 7.1 and the achieved results satisfy the project aim and requirements as discussed in Sections 7.2 and 7.3. However, I believe that further work is required on this system, before it could be tested on public roads, due to the unpredictability of the facial landmark detection.

8 Challenges

In this section, I will explain the challenges I faced over the entire course of this project, and how they have been overcome.

The most challenging aspect of this project was the ability to detect and track facial landmarks of a driver at night, when the light levels inside a car are very low and a standard camera is unable to register the shape of the drivers head, resulting in a pitch black video input.

I have been able to overcome this issue by attaching an infrared illumination module onto the camera. The infrared module is equipped with a light sensor and when the light level drops below a certain value, it automatically turns on the infrared LEDs, that illuminate the drivers face allowing the system to analyse the facial features.

Maintaining a good performance without losing the accuracy of the sleep and drowsiness detection system has been challenging. This system is written using a high-level language Python. Code written in this language is interpreted rather than compiled, which makes it slower than code written in, for example, the c++.

Since Python has no primitives, everything is stored as objects which require more memory. In addition, a Python list can store objects of a different type, so each individual entry has to contain additional data about its type. The lack of primitive types and the excessive amount of memory consumed by lists, severely impact the runtime as well as the memory consumption, which wouldn't take place if the code had been written in c++.

I have been able to maintain good performance by using the NumPy library instead of pure Python. The use of this library has allowed me to reduce memory usage as well as increase the read and write times, improving the overall performance of real-time sleep and drowsiness detection.

The accuracy of this system is highly dependent on the ability to detect the drivers face. However, whilst driving often the surrounding environment as well as other road users, distract the driver, causing him to look away from the camera, which introduces false positive result due to inaccurate facial landmark detection.

It has been a challenge to maintain accurate facial landmark detection whilst the driver is looking away from the camera. Therefore, in order to overcome this issue, I have developed an algorithm for estimating the drivers head pose. The head pose estimation algorithm ignores any data that is captured whilst the driver is not looking directly at the camera, which ensures that the detected facial landmarks are detected accurately.

9 Reflection

I have found the research presented in this paper, intellectually stimulating and challenging. This project has allowed me to broaden my knowledge in the facial recognition and sleep detection area, as well as understand why it is incredibly difficult to develop a real-time sleep, and drowsiness detection system for drivers. I have also been able to fully grasp the understanding of the underlying performance issues, that are encountered during real-time processing of the driver's facial features.

During development, I have encountered numerous challenges that I have had to overcome in order to keep on top of the project schedule. Most of the challenges have been caused by the inability to accurately detect facial landmarks or they were purely performance issues. However, good hardware and software mitigation strategies have helped me overcome and avoid challenges.

The testing stage of this project was tricky and time-consuming since, it's incredibly difficult to test a system where the input values such as the facial landmarks are constantly varying due to changing light conditions, the distance of the face from the webcam, facial expressions or head pose. But, overall I have managed to successfully test every module of this implementation by strictly following the project schedule and by performing unit as well as integration testing for each development iteration.

Overall this project was a success, as I was able to achieve the project aim and its objectives, as well as all system requirements without neglecting any aspects of this project. The implemented system is capable of detecting sleep and drowsiness based on the analysis of facial features and application of the three drowsiness detection measures: the eye aspect ratio; head pose estimation; the percentage of eye closure over time.

This is a promising project with a big potential that will introduce a lot of interesting findings into the research area of real-time sleep and drowsiness detection. I believe that the results I have achieved show that it is possible to detect sleep based on the analysis of facial features.

10 Future Work

During the course of this project, I was able to completely achieve a real-time driver sleep and drowsiness detection system which is based on the analysis of facial features.

However, even though the results obtained are satisfactory, various parts of this project have shown the need for further investigation for which I did not have the time, in the near future I plan to continue work on this project.

While the implemented system has exhausted the basic idea of sleep and drowsiness detection, I would like to spend another 6-months developing convolutional neural networks (CNNs) that could be trained to distinguish between the different levels of tiredness.

The use of convolutional neural networks would significantly improve the accuracy of this system as it would remove the high number of false positive result that the current system suffers from.

However, if I could spend an additional 12-months instead of 6, I could combine the CNNs with recurrent neural networks (RNNs) to potentially yield more powerful and generic models. During this time I would conduct research to review which type of neural network is more suitable and if combining them has proven to have a major improvement on the real-time driver sleep and drowsiness detection system, in terms of accuracy and performance achieved.

Furthermore, during the work on CNNs and RNNs, I would like to introduce additional measures based on biometrics such as heart rate, as well as yawn recognition to improve the accuracy of the drowsiness detection system.

Nevertheless, the main obstacle during this project was the lack of time to conduct a thorough study based on real driving scenarios, performed in a driving simulator. Prior to introducing CNNs and RNNs into this system, it will be necessary to spend at least 3-months gathering and processing this data in order to create a large realistic dataset for drowsiness detection. Such a dataset would contain a lot of variety in terms of different subjects, poses, drowsiness states, illumination levels, as well as occlusions.

11 Conclusion

In this paper, I have addressed the problem of detecting a drowsy or a sleeping driver in real-time using the analysis of facial features. An in-depth discussion on the variety of different sleep and drowsiness detection measures has been provided. The implemented system has been based on three measures, the analysis of eye aspect ratio, head pose estimation and the percentage of eye closure over time. The chosen measures have offered the best compromise between performance, accuracy and computational cost.

The focus of this paper was on the real-time analysis of the facial landmarks in various lighting conditions, to accurately detect the different levels of drowsiness and sleep. An innovative solution has been developed that is capable of sleep and drowsiness detection purely using, head pose estimation and the analysis of the eye aspect ratio that allows for the calculation of the percentage of eye closure over time.

The implemented system has achieved accurate sleep and drowsiness detection based on the measurement of the percentage of eye closure over time. The system is capable of accurately distinguishing between a driver who is drowsy and has a percentage of eye closure above 40% and a driver who is sleeping and has a percentage of eye closure above 70%. Despite the strong variations in lighting, facial expressions and head pose.

The system is capable of accurately estimating the driver's head pose which aids the process of discarding any data calculated for a head in an extreme pose that could result in false positive predictions. In addition, the system automatically adjusts to poor lighting conditions and can maintain good accuracy by automatically illuminating the driver's face using an infrared illumination module.

The developed system and research conducted during this project will contribute to the future sleep and drowsiness detection systems, that could be developed not only for cars but also for all kinds of other vehicles such as trains and aircraft.

Overall, I am satisfied with the quality of this project. Since the implemented system has achieved an outstanding accuracy of sleep and drowsiness detection using analysis of facial features. Not to mention that the system has met all system requirements and achieved the project aim and its objectives.

Furthermore, I will continue working on this project, and the results achieved as well as source files of this system will be released publicly to benefit the community. In addition, in the near future, I plan to accomplish a more robust sleep and drowsiness detection system that could benefit the automotive industry.

Appendices

A Appendix: Software Source Code

A.1 main.py

```
import dlib
import imutils
import playsound
import cv2
import time
import numpy as np
import constants as c
import head_pose_estimator as hp
import eye_aspect_ratio as ear
from imutils.video import WebcamVideoStream
from imutils import face_utils
from imutils.video import FPS
from threading import Thread
from scipy import signal as sc

# Initialize dlib face detector (HOG-based)
detector = dlib.get_frontal_face_detector()

# Create facial landmark predictor
predictor = dlib.shape_predictor(c.PREDICTOR_PATH)

FONT = cv2.FONT_HERSHEY_SIMPLEX


# Play sound
def play_alarm(path):
    playsound.playsound(path)


# Detect a single face
def get_face(gray_frame):
    faces_detected = detector(gray_frame, 0)

    if len(faces_detected) > 0:
        return faces_detected[0]

    return None


# Predict facial landmarks
def get_landmarks(gray_frame, face):
    shape = predictor(gray_frame, face)

```

```
    # Convert to NumPy array
    coords = np.zeros((shape.num_parts, 2), dtype=int)
    for i in range(0, shape.num_parts):
        coords[i] = (shape.part(i).x, shape.part(i).y)

    return coords


# Analyse ear values for sleep
def detect_sleep(ear_values, ear_thrs, drowsy_count):
    status_msg = ''
    shut_eye_count = 0

    for ear_val in ear_values:
        if ear_val < ear_thrs:
            shut_eye_count += 1

    perclos = (shut_eye_count / len(ear_values)) * 100

    if perclos >= 70 or drowsy_count > 2:
        status_msg = "SLEEPING"
        drowsy_count = 0
        t = Thread(target=play_alarm, args=("sounds/alarm.mp3",))
        t.daemon = True
        t.start()

    elif 70 > perclos >= 40:
        status_msg = "DROWSY"
        drowsy_count += 1
        t = Thread(target=play_alarm, args=("sounds/blink.mp3",))
        t.daemon = True
        t.start()

    else:
        drowsy_count = 0

    return status_msg, drowsy_count


def main():
    print("--- Starting live video feed ---")
    vs = WebcamVideoStream(src=c.CAMERA_MODE).start()

    time.sleep(1.0)
    fps = FPS().start()
    frame_count = 0
    ear_vals = []
    ear_thrs = 0
    avg_ear = 0
    it_ear_vals = []
    collect_blinks = False
```

```
    status_msg = ''
    drowsy_cnt = 0
    # Visual info flag, set it to False to improve performance
    display_visuals = True

    while True:
        frame = vs.read()
        frame = imutils.resize(frame, c.FRAME_WIDTH,c.FRAME_HEIGHT)

        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        frame_count += 1

        face = get_face(gray_frame)

        if face is not None:
            blink_msg = ''
            landmarks = get_landmarks(gray_frame, face)
            head_pose, img_points = hp.get_head_pose(landmarks)

            if head_pose[c.YAW] == c.POSE_NEUTRAL:
                avg_ear = ear.get_avg_ear(landmarks)
                ear_vals.append(avg_ear)

                # Update ear threshold approximately every 6s
                if frame_count % 101 == 0 and len(ear_vals) > 17:
                    sg_ear_vals = sc.savgol_filter(ear_vals, 17, 7)
                    ear_thrs = ear.calc_ear_thrs(sg_ear_vals)

                    frame_count = 0
                    ear_vals.clear()

                # Check for a blink
                if avg_ear < ear_thrs:
                    blink_msg = 'BLINK DETECTED'

                # Start new sleep detection iteration
                if avg_ear < ear_thrs and collect_blinks == False:
                    collect_blinks = True

                # Collect ear values for sleep detection iteration
                if collect_blinks == True:
                    it_ear_vals.append(avg_ear)

                # Has sleep detection iteration finished
                if len(it_ear_vals) >= 45:
                    collect_blinks = False
                    status_msg, drowsy_cnt = \
                        detect_sleep(it_ear_vals, ear_thrs,
                                     drowsy_cnt)
                    it_ear_vals.clear()

```

```
146             # Flag for the visual info overlay
147             if display_visuals == True:
148                 cv2.putText(frame, blink_msg, (20, 110), FONT, 0.8,
149                             (0, 255, 255), 2)
150                 cv2.putText(frame, status_msg, (20, 140), FONT,
151                             0.8, (0, 255, 255), 2)
152                 cv2.putText(frame, (hp.read_head_pose(head_pose)),
153                             (20, 20), FONT, 0.8, (255, 0, 0), 2)
154                 cv2.putText(frame, "THRS EAR: {:.2f}".format(
155                     ear_thrs), (20, 50), FONT, 0.8,
156                             (255, 0, 0), 2)
157                 cv2.putText(frame,
158                             "AVG EAR: {:.2f}".format(avg_ear),
159                             (20, 80), FONT, 0.8, (255, 0, 0), 2)
160
161                 for p in img_points:
162                     cv2.circle(frame, (int(p[0]), int(p[1])), 3,
163                                (255, 0, 100), -1)
164
165                 (x, y, w, h) = face_utils.rect_to_bb(face)
166                 cv2.rectangle(frame, (x, y), (x + w, y + h),
167                               (0, 255, 0), 1)
168
169                 for (x, y) in landmarks:
170                     cv2.circle(frame, (x, y), 1, (0, 0, 255), -1)
171
172         cv2.imshow("Sleep Detection", frame)
173
174         if cv2.waitKey(1) & 0xFF == ord(c.EXIT_KEY):
175             break
176
177         fps.update()  # update the FPS counter
178
179     # stop the timer and display FPS information
180     fps.stop()
181     print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
182     print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
183     print("--- Ending live video feed ---")
184
185     # Cleanup upon exit
186     cv2.destroyAllWindows()
187     vs.stop()
188
189
190 if __name__ == '__main__':
191     main()
```

Listing 8: Implementation of the sleep and drowsiness detection module.

A.2 head_pose_estimator.py

```
import constants as c
import numpy as np
import cv2


def get_head_pose_vectors(landmarks):
    # 2D image points
    img_points = np.array([landmarks[c.NOSE_TIP],
                           landmarks[c.CHIN],
                           landmarks[c.LEFT_EYE_LEFT_CORNER],
                           landmarks[c.RIGHT_EYE_RIGHT_CORNER],
                           landmarks[c.LEFT_MOUTH_CORNER],
                           landmarks[c.RIGHT_MOUTH_CORNER]
                           ], dtype="double")
    # 3D model points.
    model_points = np.array([
        c.NOSE_TIP_3D_POINT,
        c.CHIN_3D_POINT,
        c.LEFT_EYE_LEFT_CORNER_3D_POINT,
        c.RIGHT_EYE_RIGHT_CORNER_3D_POINT,
        c.LEFT_MOUTH_CORNER_3D_POINT,
        c.RIGHT_MOUTH_CORNER_3D_POINT
    ])

    # Camera internals
    focal_length = c.FRAME_WIDTH
    frame_center = (c.FRAME_WIDTH / 2, c.FRAME_HEIGHT / 2)

    camera_matrix = np.array([[focal_length, 0, frame_center[0]],
                              [0, focal_length, frame_center[1]],
                              [0, 0, 1]], dtype="double")

    dist_coeffs = np.zeros((4, 1))  # Assuming no lens distortion

    success, rotation_vector, \
    translation_vector = cv2.solvePnP(model_points,
                                      img_points,
                                      camera_matrix,
                                      dist_coeffs,
                                      cv2.cv2.SOLVEPNP_ITERATIVE)

    return img_points, rotation_vector, translation_vector


def get_euler_angles(rotation_vec, translation_vec):
    rotation_mat, _ = cv2.Rodrigues(rotation_vec)
    pose_mat = cv2.hconcat((rotation_mat, translation_vec))
    return cv2.decomposeProjectionMatrix(pose_mat)[6]

```

```
50
51 def get_head_pose(landmarks):
52     img_points, rotation_vector, translation_vector = \
53         get_head_pose_vectors(landmarks)
54
55     euler_angles = get_euler_angles(rotation_vector,
56                                     translation_vector)
57
58     head_pose = np.full(3, c.POSE_UNKNOWN)
59
60     head_pose = calc_head_pose(head_pose, euler_angles, c.PITCH)
61     head_pose = calc_head_pose(head_pose, euler_angles, c.YAW)
62     head_pose = calc_head_pose(head_pose, euler_angles, c.ROLL)
63
64     return head_pose, img_points
65
66
67 def calc_head_pose(head_pose, euler_angles, angle_type):
68     angle = euler_angles[angle_type, 0]
69
70     negative_pose_angles = c.ANGLE_VALUES[angle_type][
71         c.POSE_NEGATIVE]
72     neutral_pose_angles = c.ANGLE_VALUES[angle_type][
73         c.POSE_NEUTRAL]
74     positive_pose_angles = c.ANGLE_VALUES[angle_type][
75         c.POSE_POSITIVE]
76
77     if angle_type == c.PITCH:
78         head_pose[angle_type] = calc_pitch(angle,
79                                            negative_pose_angles,
80                                            neutral_pose_angles,
81                                            positive_pose_angles)
82     elif angle_type == c.YAW:
83         head_pose[angle_type] = calc_yaw(angle,
84                                          negative_pose_angles,
85                                          neutral_pose_angles,
86                                          positive_pose_angles)
87     elif angle_type == c.ROLL and head_pose[
88         c.YAW] == c.POSE_NEUTRAL:
89         head_pose[angle_type] = calc_roll(angle,
90                                           negative_pose_angles,
91                                           neutral_pose_angles,
92                                           positive_pose_angles)
93     else:
94         head_pose[angle_type] = c.POSE_UNKNOWN
95
96     return head_pose
97
98
99 # Calculates head pitch angle (nodding)
100 def calc_pitch(angle, negative_pose_angles,
```

```
101                 neutral_pose_angles,
102                 positive_pose_angles):
103     if positive_pose_angles[0] < angle < positive_pose_angles[1]:
104         return c.POSE_POSITIVE
105     elif neutral_pose_angles[0] <= angle <= neutral_pose_angles[1]:
106         return c.POSE_NEUTRAL
107     elif negative_pose_angles[0] > angle or angle > \
108             negative_pose_angles[1]:
109         return c.POSE_NEGATIVE
110     else:
111         return c.POSE_UNKNOWN
112
113
114 # Calculates head roll angle
115 def calc_roll(angle, negative_pose_angles,
116                 neutral_pose_angles,
117                 positive_pose_angles):
118     if angle < negative_pose_angles:
119         return c.POSE_POSITIVE
120     elif angle > positive_pose_angles:
121         return c.POSE_NEGATIVE
122     elif neutral_pose_angles[0] <= angle <= neutral_pose_angles[1]:
123         return c.POSE_NEUTRAL
124     else:
125         return c.POSE_UNKNOWN
126
127
128 # Calculates head yaw angle
129 def calc_yaw(angle, negative_pose_angles, neutral_pose_angles,
130                 positive_pose_angles):
131     if angle < negative_pose_angles:
132         return c.POSE_NEGATIVE
133     elif angle > positive_pose_angles:
134         return c.POSE_POSITIVE
135     elif neutral_pose_angles[0] <= angle <= neutral_pose_angles[1]:
136         return c.POSE_NEUTRAL
137     else:
138         return c.POSE_UNKNOWN
139
140
141 # Converts head yaw angle value to a descriptive string
142 def read_yaw(head_pose):
143     result = ''
144
145     if head_pose[c.YAW] == c.POSE_NEUTRAL:
146         result += 'Yaw neutral, '
147     elif head_pose[c.YAW] == c.POSE_NEGATIVE:
148         result += 'Yaw -ve, '
149     elif head_pose[c.YAW] == c.POSE_POSITIVE:
150         result += 'Yaw +ve, '
151     else:
```

```
152         result += '--- '
153
154     return result
155
156
157 # Converts head pitch angle value to a descriptive string
158 def read_pitch(head_pose):
159     result = ''
160
161     if head_pose[c.PITCH] == c.POSE_NEUTRAL:
162         result += 'Pitch neutral '
163     elif head_pose[c.PITCH] == c.POSE_NEGATIVE:
164         result += 'Pitch -ve '
165     elif head_pose[c.PITCH] == c.POSE_POSITIVE:
166         result += 'Pitch +ve '
167     else:
168         result += '--- '
169
170     return result
171
172
173 # Converts head roll angle value to a descriptive string
174 def read_roll(head_pose):
175     result = ''
176
177     if head_pose[c.ROLL] == c.POSE_NEUTRAL:
178         result += 'Roll neutral, '
179     elif head_pose[c.ROLL] == c.POSE_NEGATIVE:
180         result += 'Roll -ve, '
181     elif head_pose[c.ROLL] == c.POSE_POSITIVE:
182         result += 'Roll +ve, '
183     else:
184         result += 'Roll ---, '
185
186     return result
187
188
189 # Head pose summary
190 def read_head_pose(head_pose):
191     result = ''
192     result += read_roll(head_pose)
193     result += read_yaw(head_pose)
194     result += read_pitch(head_pose)
195     return result
```

Listing 9: Implementation of the head pose estimation module.

A.3 eye_aspect_ratio.py

```python
from scipy.spatial import distance as dist
from imutils import face_utils

(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

# Calculate eye aspect ratio (ear) for a single eye
def calc_ear(eye):
    # Vertical distance between two first eye co-ordinates
    v_a = dist.euclidean(eye[1], eye[5])
    # Vertical distance between the second two eye co-ordinates
    v_b = dist.euclidean(eye[2], eye[4])
    # Horizontal distance between the corners of an eye
    h_a = dist.euclidean(eye[0], eye[3])

    # Computing eye aspect ratio
    computed_ratio = (v_a + v_b) / (2.0 * h_a)
    return computed_ratio


# Calculates avg ear for both eyes
def get_avg_ear(landmarks):
    left_eye = landmarks[lStart:lEnd]
    right_eye = landmarks[rStart:rEnd]

    # Compute eye aspect ratio for both eyes
    left_eye_ar = calc_ear(left_eye)
    right_eye_ar = calc_ear(right_eye)

    # Average the eye aspect ratio for both eyes
    avg_eye_ar = (left_eye_ar + right_eye_ar) / 2.0

    return avg_eye_ar


# Calculates ear threshold
def calc_ear_thrs(ear_values):
    avg_ear = sum(ear_values) / len(ear_values)
    min_ear = min(ear_values)
    ear_threshold = min_ear + ((avg_ear - min_ear) / 2)
    return ear_threshold
```

Listing 10: Implementation of the eye aspect ratio module.

A.4 constants.py

```
# OPTIONS
PREDICTOR_PATH = 'shape_predictor_68_face_landmarks.dat'
CAMERA_MODE = 0  # 0 - Built in cam (laptops), 1 - External cam
FRAME_WIDTH = 800
FRAME_HEIGHT = 450
INIT_TIME = 60
EXIT_KEY = 'q'

# HEAD POSE - YAW ANGLE VALUES
YAW = 1

YAW_NEUTRAL = -19, 19
YAW_NEGATIVE = -19
YAW_POSITIVE = 19

# HEAD POSE - ROLL ANGLE VALUES
ROLL = 2

ROLL_NEUTRAL = -15, 15
ROLL_NEGATIVE = -15
ROLL_POSITIVE = 15

# HEAD POSE - PITCH ANGLE VALUES
PITCH = 0

PITCH_NEUTRAL = 155, 175
PITCH_NEGATIVE = -160, 175
PITCH_POSITIVE = 0, 155

POSE_UNKNOWN = -1      # Head position unknown
POSE_NEGATIVE = 0      # Head yaw left, pitch down and roll left
POSE_NEUTRAL = 1       # Head in neutral (straight position)
POSE_POSITIVE = 2      # Head yaw right, pitch up and roll right


ANGLE_VALUES = [PITCH_NEGATIVE, PITCH_NEUTRAL, PITCH_POSITIVE], \
               [YAW_NEGATIVE, YAW_NEUTRAL, YAW_POSITIVE], \
               [ROLL_NEGATIVE, ROLL_NEUTRAL, ROLL_POSITIVE]

# Facial Landmark 2D Points
NOSE_TIP = 30
CHIN = 8
LEFT_EYE_LEFT_CORNER = 36
RIGHT_EYE_RIGHT_CORNER = 45
LEFT_MOUTH_CORNER = 48
RIGHT_MOUTH_CORNER = 54

# Facial Landmark 3D Points
NOSE_TIP_3D_POINT = (0.0, 0.0, 0.0)
```

```
50 CHIN_3D_POINT = (0.0, -330.0, -65.0)
51 LEFT_EYE_LEFT_CORNER_3D_POINT = (-225.0, 170.0, -135.0)
52 RIGHT_EYE_RIGHT_CORNER_3D_POINT = (225.0, 170.0, -135.0)
53 LEFT_MOUTH_CORNER_3D_POINT = (-150.0, -150.0, -125.0)
54 RIGHT_MOUTH_CORNER_3D_POINT = (150.0, -150.0, -125.0)
55
56 # AVG eye aspect ratio indexes
57 EAR_NEGATIVE = 0
58 EAR_NEUTRAL = 1
59 EAR_POSITIVE = 2
```

Listing 11: The file used to defined constants used throughout the whole system.

References

[1] J A Horne and L A Reyner. Sleep related vehicle accidents. *BMJ*, 310(6979):565–567, 1995.

[2] Drew Dawson and Kathryn Reid. Fatigue, alcohol and performance impairment. *Nature*, 388:235, 08 1997.

[3] Delphine Robineau. Reported road casualties in great britain: 2017 annual report. Report, Department for Transport, 09 2018. Last accessed: 27/10/2018.

[4] T. Igasaki, K. Nagasawa, N. Murayama, and Z. Hu. Drowsiness estimation under driving environment by heart rate variability and/or breathing rate variability with logistic regression analysis. In *2015 8th International Conference on Biomedical Engineering and Informatics (BMEI)*, pages 189–193, Oct 2015.

[5] In-Ho Choi and Yong-Guk Kim. Head pose and gaze direction tracking for detecting a drowsy driver. In *2014 International Conference on Big Data and Smart Computing (BIGCOMP)*, pages 241–244, Jan 2014.

[6] Volkswagen. 2019. driver alert system. https://www.volkswagen.co.uk/technology/car-saftey/driver-alert-system. Last accessed: 5/05/2019.

[7] P. Chen, W. Yeh, P. Li, and K. Tu. Study on using the heart rate peak sensor to detect the first moment of doze. In *2013 International Symposium on Next-Generation Electronics*, pages 531–533, Feb 2013.

[8] K. Naito, T. Isioka, H. Takano, and K. Nakamura. Real time doze detection method using closed eye time during blink burst and isolated blinks. In *2012 Proceedings of SICE Annual Conference (SICE)*, pages 1837–1840, Aug 2012.

[9] W. W. Wierwille, L. A. Ellsworth, S. S. Wreggit, R. J. Fairbanks, and C. L. Kirn. Research on vehicle-based driver status/performance monitoring; development, validation, and refinement of algorithms for detection of driver drowsiness. 12 1994. (Report No. DOT HS 808 247).

[10] A. G. Mavely, J. E. Judith, P. A. Sahal, and S. A. Kuruvilla. Eye gaze tracking based driver monitoring system. In *2017 IEEE International Conference on Circuits and Systems (ICCS)*, pages 364–367, Dec 2017.

[11] Klauer S. G. Dingus, T. A. Neale, V. L. Sudweeks, and D. J. J. D. Ramsey. The impact of driver inattention on near-crash/crash risk: An analysis using the 100-car naturalistic driving study data. 04 2006. (Report No. DOT-HS-810594).

[12] J. Yan, H. Kuo, Y. Lin, and T. Liao. Real-time driver drowsiness detection system based on perclos and grayscale image processing. In *2016 International Symposium on Computer, Consumer and Control (IS3C)*, pages 243–246, July 2016.

[13] Tereza Soukupová and Jan Cech. Real-time eye blink detection using facial landmarks. 2016. Center for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University in Prague.

[14] Lunbo Xu, Shunyang Li, Kaigui Bian, Tong Zhao, and Wei Yan. Sober-drive: A smartphone-assisted drowsy driving detection system. In *2014 International Conference on Computing, Networking and Communications (ICNC)*, pages 398–402, Feb 2014.

[15] LIZHEN WEI CUIQING ZHANG and PEI ZHENG. Research on driving fatigue detection based on perclos. 2017. Technical College of Mechanics and Electrics. University of Technology. Inner Mongolia, Hohhot, China.

[16] Xiaopeng Ni Jiawen Wang Hao Zhang Xiao Li Lizong Lin, Chao Huang and Zhiqin Qian. Driver fatigue detection based on eye state. *Technology and Health Care*, 23(s2):453–463, 2015.

[17] Z. Jie, M. Mahmoud, Q. Stafford-Fraser, P. Robinson, E. Dias, and L. Skrypchuk. Analysis of yawning behaviour in spontaneous expressions of drowsy drivers. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 571–576, May 2018.

[18] M. Omidyeganeh, A. Javadtalab, and S. Shirmohammadi. Intelligent driver drowsiness detection through fusion of yawning and eye closure. In *2011 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems Proceedings*, pages 1–6, Sept 2011.

[19] B. Akrout and W. Mahdi. Yawning detection by the analysis of variational descriptor for monitoring driver drowsiness. In *2016 International Image Processing, Applications and Systems (IPAS)*, pages 1–5, Nov 2016.

[20] S. Abtahi, S. Shirmohammadi, B. Hariri, D. Laroche, and L. Martel. A yawning measurement method using embedded smart cameras. In *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1605–1608, May 2013.

[21] B. N. Manu. Facial features monitoring for real time drowsiness detection. In *2016 12th International Conference on Innovations in Information Technology (IIT)*, pages 1–4, Nov 2016.

[22] H. Kang. Various approaches for driver and driving behavior monitoring: A review. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 616–623, Dec 2013.

[23] P. Hrubes, J. Faber, and M. Novak. Analysis of eeg signals during micro-sleeps. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 4, pages 3775–3780 vol.4, Oct 2004.

[24] The output of dlib face landmark detection, on one of the images from the helen dataset. http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html. Last accessed: 27/10/2018.

[25] S. V. Menon and C. S. Seelamantula. Robust savitzky-golay filters. In *2014 19th International Conference on Digital Signal Processing*, pages 688–693, Aug 2014.

[26] Jianwen Luo, Kui Ying, and Lijing Bai. Savitzky–golay smoothing and differentiation filter for even number data. *Signal Processing*, 85:1429–1434, 07 2005.

[27] Ronald W Schafer et al. What is a savitzky-golay filter. *IEEE Signal processing magazine*, 28(4):111–117, 2011.

[28] Hamed Azami, Karim Mohammadi, and Behzad Bozorgtabar. An improved signal segmentation using moving average and savitzky-golay filter. *Journal of Signal and Information Processing*, 3(01):39, 2012.

[29] The ibug68 facial landmark annotation template. https://ibug.doc.ic.ac.uk/resources/facial-point-annotations. Last accessed: 27/10/2018.

[30] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: database and results. *Image and Vision Computing*, 47:3 – 18, 2016. 300-W, the First Automatic Facial Landmark Detection in-the-Wild Challenge.

[31] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.