Future University in Egypt

Faculty of Computers and Information Technology

DRIVER DISTRACTION DETECTION USING FACIAL LANDMARKS

STUDENT NAME	STUDENT ID
Salma Yasser	20183154
Marwan Mohamed	20184133
Mohamed Ashraf	20183706
Mohamed Magdy	20182573
Ahmed Hussein	20184017

Supervised by : Dr. Heba Hamdy

Table Of Figures

Table Of Contents

Abstract

Driver inattention and distraction are the main causes of road accidents, many of which result in fatalities. To reduce road accidents, the development of information systems to detect driver inattention and distraction is essential. Currently, distraction detection systems for road vehicles are not yet widely available or are limited to specific causes of driver inattention such as driver fatigue. Despite the increasing automation of driving due to the availability of increasingly sophisticated assistance systems, the human driver will continue to play a longer role as supervisor of vehicle automation. With a goal to reduce traffic accidents and improve transportation safety, this study proposes a driver distraction detection system which identifies various types of distractions through a camera observing the driver. A driver distraction detection system is developed for the purpose of making driving safer. With this in mind, we review the published scientific literature on driver distraction detection methods and integrate the identified approaches into a holistic framework that is the main contribution of the paper. Based on published scientific work, our driver distraction detection framework contains a structured summary of reviewed approaches for detecting three main distraction detection approaches: drowsiness , distraction by billboards and distraction by picking up something form backward or down during driving. Our framework visualizes the whole detection information chain from measured data, computed data, computed events. Besides providing a sound summary for researchers interested in distracted driving, we discuss several practical implications for the development of driver distraction detection systems that can also combine different approaches for higher detection quality.

CHAPTER 1

INTRODUCTION

Outlines:

INTRODUCTION

This chapter presents an introduction about the research in general, some medical background, the problem that we tried to solve, the motivation and objective of this research, and finally this thesis outline.

1.1. Motivation

Now-a-days, there is huge increase in private transportation day by day in this modernize world. It will be tedious and bored for driving when it is for long time distance. One of the main causes behind the driver’s lack of alertness is due to long time traveling without sleep and rest. Tired driver can get drowsy while driving. Every fraction of seconds drowsiness can turn into dangerous and life-threatening accidents may lead to death also. To prevent this type of incidents, it is required to monitor driver’s alertness continuously and when it detects drowsiness, the driver should be alerted. Through this we can reduce significant number of accidents and can save lives of people.

Drowsiness is always been part of the main causes of severe traffic accidents occurring in our daily life. According to the National Sleep Foundation’s 2005 Sleep in America poll, 60% of adult drivers – about 168 million people – say they have driven a vehicle while feeling drowsy in the past year, and more than one-third, (37% or 103 million people), have actually fallen asleep at the wheel! In fact, of those who have nodded off, 13% say they have done so at least once a month. Four percent – approximately eleven million drivers – admit they have had an accident or near accident because they dozed off or were too tired to drive. [1] According to the US National Highway Traffic Safety Administration, approximately 100,000 crashes occur in US each year due to drivers’ drowsiness resulting in an estimated 1,550 deaths, 71,000 injuries and $12.5 billion losses [2]. Another report [3] states that the US government and businesses spend an estimated $60.4 billion per year on accidents related to drowsy driving and it also costs the consumer about $16.4 billion in terms of property damages, health claims, lost time and productivity due to drowsy driving. In 2010, the National Sleep Foundation (NSF) reported that 54% adult drivers had driven a vehicle while feeling drowsy and 28% had actually fallen asleep [4]. The German Road Safety Council (DVR) claims that one in four highway traffic fatalities are a result of momentary driver drowsiness [5].

Motor companies like Toyota, Ford, Mercedes-Benz and others are also currently employing car safety technologies to prevent accidents from happening when the driver is getting drowsy. This trend is expected to make the cars smarter and significantly reduce the accidents caused by drowsiness of drivers. Adhering to these endeavors, our research is motivated by the statistical significance of accidents due to drowsiness and provides an improved and systematic approach to drowsiness detection.

1.2. Problem Definition

Existence of camera that detect drowsiness level of the driver does not exist in more than 95% percent of cars and ones who have are luxurious ones and are not affordable to most of people.

A driver who falls asleep at the wheel loses control of the vehicle, an action which often results in a crash with either another vehicle or stationary objects. In order to prevent these devastating accidents, the state of drowsiness of the driver should be monitored. The following measures have been used widely for monitoring drowsiness

The National Highway Traffic Safety Administration estimates that every year about 100,000 police-reported, drowsy-driving crashes result in nearly 800 fatalities and about 50,000 injuries. The real number may be much higher, however, as it is difficult to determine whether a driver was drowsy at the time of a crash.

A study by the AAA Foundation for Traffic Safety estimated that 328,000 drowsy driving crashes occur annually. That's more than three times the police-reported number. The same study found that 109,000 of those drowsy driving crashes resulted in an injury and about 6,400 were fatal. The researchers suggest the prevalence of drowsy driving fatalities is more than 350% greater than reported.

Beyond the human toll is the economic one. NHTSA estimates fatigue-related crashes resulting in injury or death cost society $109 billion annually, not including property damage.

1.3. Solution approach

The Driver distraction detection aims to ensure a safe transport system for all road users. The goal is to develop an eye tracking algorithm to enable us to bypass the limitations of eye tracking technologies.

In this work, an approach is proposed to solve the problem of eye tracking, a framework for eye movement recognition is built after detecting the contours of the eye, the distance or gap between the upper and lower points is calculated based on these actions, the system will be able to determine whether the eye is open or closed So the system can alert the driver if the eyes are closed.

A real-time algorithm to detect eye blinks, yawn and lack of focusing in a video sequence from a standard camera is proposed. Recent landmark detectors, trained on in-the-wild datasets exhibit excellent robustness against a head orientation with respect to a camera, varying illumination and facial expressions. The proposed algorithm therefore estimates the landmark positions, extracts a single scalar quantity – eye aspect ratio (EAR) and mouth aspect ratio (MAR) – characterizing the eye and the mouth opening in each frame.

1.4. Scope of the work

For the scope of this project, we will focus on using a highly efficient deep learning model to classify different driver distractions at runtime using facial landmarks. This proposed project detects eye blinks, yawn, lack of focusing and sudden face disappearance. The Driver distraction detection framework presented in this work is Digital camera. This camera used in cars while driving. It is suitable for all types of cars. Any Driver can used it, it's translate the reactions of the driver’s face to increase the driver’s attention and to reduce the possibility of an accident.

CHAPTER 2
LITERATURE REVIEW

Outlines:

2.1. Introduction

Every year the lives of approximately 1.3 million people are cut short as a result of a road traffic crash. Between 20 and 50 million more people suffer non-fatal injuries, with many incurring a disability as a result of their injury. A heartbreaking statistic in this report is that injuries caused by road accidents lead to the death of young people between 5 and 29 years of age, the common distractions are eating and drinking while driving, communicating with fellow passengers, using in-vehicle electronic devices, observing roadside digital billboards/advertisings/logo signs, and using a mobile phone to call or text.

Road traffic injuries cause considerable economic losses to individuals, their families, and to nations as a whole. These losses arise from the cost of treatment as well as lost productivity for those killed or disabled by their injuries, and for family members who need to take time off work or school to care for the injured. Road traffic crashes cost most countries 3% of their gross domestic product. [6]

Currently, distracted driving is among the most important causes of traffic accidents. Consequently, intelligent vehicle driving systems have become increasingly important. Recently, interest in driver-assistance systems that detect driver actions and help them drive safely has increased.

2.2. Driver Distraction Detection System

An effective way to address distracted driving is by developing a system that can monitor the distractions. Some techniques were introduced that adapt the in-vehicle information-system functionalities according to the driver’s state. Correctly identifying the driver’s state is very important.

The authors of this paper developed a support vector machine-based **(SVM-based)** model to detect the use of a mobile phone while driving, the authors of another study did some extraordinary work by sitting on a chair and mimicking a certain type of distraction (e.g., talking on a mobile phone). In this work, the AdaBoost classifier was applied along with hidden Markov models to classify Kinect RGB-D data, a faster Region Based Convolutional Neural Networks (R-CNN) model was designed, the authors created a dataset for hand detection in an automotive environment and achieved an average precision of 70.09% using the aggregate channel features object detector, a **SVM** classier was applied to determine the driver's actions; 90% and 94% accuracy were achieved for hand actions and face and hand-related information, respectively, The dataset consists of labels for each image for these four driver actions].

Amongst the various reasons, the major one is the driver's fatigue and drowsy state. This reduces driver's decision-making ability to control the car. Symptoms of being drowsy or sleepy include-difficulty in focusing, frequent eye blinking, day-dreaming, missing traffic signs, yawning repeatedly etc. the drivers who are deprived of more than 4 hours of sleep are 10.2 times more likely to indulge in accidents. According to the statistics it has been estimated that driver's drowsiness kills 1, 500 people and leaves 71, 000 people injured in US road accidents every year. Considering the Australian survey, about 20% of severe road accidents and 30% of fatal crashes involve driver's mistake. Further, a survey in Norway found that 3.9% of the accidents were sleep related and almost 20% of night-time accidents involved driver's drowsiness. Therefore, it is necessary that the next set of cars coming out in the market should have an additional safety feature to alert sleepy drivers and handling auxiliary task using hand gesture.

As of now there has been an extensive amount of work done on drowsiness detection. But here they specify only a few important and relevant literature works. Chellappa et al. [7] uses physiological and physical signs for drowsiness detection. System takes input from both the factors and uses a combination of these as a parameter. Physiological inputs include body temperature and pulse rate, physical inputs comprise of yawning and blinking. Eventually they result in annoyance to the driver thus not reliable as compared to non-intrusive approaches. One of the non-intrusive approaches include the HOG descriptor algorithm which was introduced by Dalal et al. [8] for face detection. Also, Comparative analysis of performance of HOG descriptors against generalized haar wavelets, PCA-SIFT descriptors and shape context descriptors commenced on different dataset. Linear SVM is then trained on HOG features to classify the facial expression. Ngxande et al. [9] give a meta-analysis on three machine learning techniques i.e. Support Vector Machines (SVM), Convolutional Neural Networks (CNN) and Hidden Markov Models (HMM) to figure out the behaviour aspects. It then concludes that out of the three techniques the SVM technique is most commonly used but the convolutional neural networks give finer results than the other two. Wang et al. [10] proposed framework for driver drowsiness detection where the method PATECP (Percentage and Time that Eyelids Cover the Pupils) and PATMIO (Percentage and Time that Mouth Is Open) is used to decide whether the driver is drowsy by setting a particular threshold. Zhao et al. [11] proposed a schema for recognizing drowsy expressions which is based on facial dynamic fusion information and a deep belief network (DBN). To make the system more robust in different light conditions in [12] uses various visual cues along with remotely located camera with infrared illuminators. Assari et al. [13] also proposes a system that used infrared camera, to capture the video stream of the driver, in order to resolve the issues imposed by lightning conditions, presence of glasses, beard etc. Further face region is detected and facial components are extracted from the stream and then compared with that of a drowsy person using template matching. Many other theories were also implemented for the same.

In this paper, the researchers use a literature review to identify existing driving distraction detection methods for driver monitoring in the vehicle cabin, analyze them, and extract an information flow for distraction detection. Thereby, the literature review process follows common and established guidelines for scientific rigor.

The approach researchers used to identify the relevant literature included a broad sampling frame that covers a wide range of academic disciplines including computer science, mobility and transport, mathematics, information systems, human factors, and psychology that are interested in the interdisciplinary topic driving distraction focusing on both manual and automated driving. Researchers are especially interested in published scientific works that have published driving distraction approaches and related methods (e.g., for detection of certain human poses). Researchers started their literature review by conducting a keyword search, mainly using Google Scholar using keywords such as ‘‘driver distraction’’, ‘‘driver monitoring’’, or ‘‘distraction detection’’. However, they also searched the references of the identified papers for cited related work and added additional papers to their review sample by applying forward and backward reference searches.

2.3. Drowsiness Driving Detection

There are more serious injuries and deaths than minor injuries, and the damage due to major accidents is increasing. In particular, heavy cargo trucks and high-speed bus accidents that occur during driving in the middle of the night have emerged as serious social problems. Therefore, in this part, a drowsiness prevention system was developed to prevent large-scale disasters caused by traffic accidents.

Eye-Blink Detection Face recognition should be first performed in order to detect eye blinking. Therefore, the system recognizes the pupils of the driver’s eyes after recognizing the face and examines the blink speed of their eyelids to detect drowsiness. In Figure [2.1], the Haar Cascade technique, which uses patterns of light and shade in OpenCV, is applied to recognize the faces of human beings. On drivers' faces, the eyes are dark and the nose is bright. Therefore, the technique extracts face information by analyzing the pattern in the black and white image. Further, extracted face information is recognized by using the Haar Cascade in OpenCV [14]. The study actually applied this technique to recognize the eyes of a human face. However, the Raspberry Pi environment used in this study was poor in terms of its performance. Thus, this study referred to Korean standard face data to determine the eye position of drivers. Figure [2.2] shows that the eyes of drivers can be continuously tracked by applying the mean-shift method to continue to track them even when they move. The mean-shift is a method used to find the peak or center of gravity of data distribution, which indicates the algorithm is moving to a data-dense area and the center of the distribution. When the data are distributed on a 2D plane, the process of finding the densest peak point of the data is constructed by the following methods [15]:

(1) Obtain data originating from the radius, r, from the current position.
(2) Move the current position to the coordinates of the center of gravity.
(3) Repeat step 1 and 2 until the position converges.

Determining the position of the eyes and the open eyes. A module was created that store the image of the user's eye area in the category.number.jpg format, a vector with 1024 features was created. This value was determined through a hyperparameter tuning experiment. Softmax was used as the activation function. Stockastic gradient descent was used as the learning model Carbon Dioxide Detection as a result of a questionnaire survey, it was found that the occurrence of many drowsy driving operations depends on the air quality in vehicles. Therefore, this study tried to prevent drowsy driving by detecting the concentration of carbon dioxide in vehicles. Figure [2.3] represents a sensor for measuring the concentration of carbon dioxide of the NDIR (Non-Dispersive Infrared) system. If the concentration of carbon dioxide was over 1500 ppm, it was expected that drowsiness would appear. Further, when the concentration of carbon dioxide was high, this not only caused drowsiness and stiffness but also caused dizziness, headache and health problems. This sensor measures the concentration of carbon dioxide to the extent that refraction is caused by gas concentration using a non-distributed infrared emitting unit. The sensor has high durability and high accuracy, thereby detecting drowsy driving quickly. From the possible semiconductor resistance, electrochemical, and NDIR sensors, this study used the NDIR method to measure driving conditions, due to the considerations of cost-effectiveness and efficiency.

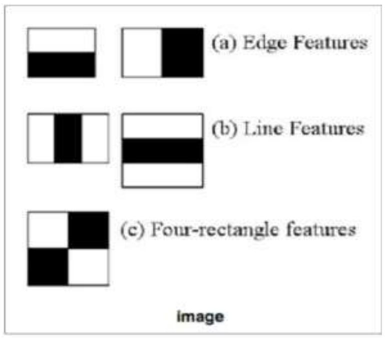


Figure 2.1 : Haar Cascade

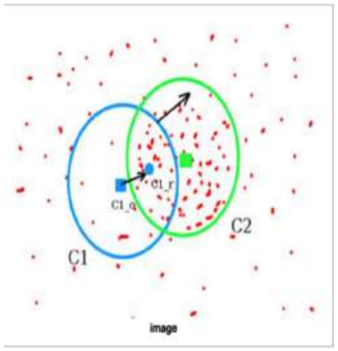


Figure 2.2 : Mean-Shift Method.

Figure 2.3 : Carbon Dioxide concentration sensor.

The author of this paper proposed a real time system to monitor driver’s state such as sleepiness, yawning and if drivers fall sleep or yawn more than 4sec, this systems alert the driver to be in normal driving state. A RGB camera is mounted at front windows and constantly looking at drivers face. He also set a timer for each driver such that if driver is driving constantly till 12 hours, this proposed system triggers an alarm to switch off from driving. First step of his proposed approach is to detect the face from each frame and recognize the face to check whether it is same driver or different driver. If it is the same driver he constantly monitoring eye closeness and yawning and simultaneously timer is also increases. If it is different driver, a separate timer is initialized and start monitoring the driver state. To find the eye closeness and yawning, they first find the facial landmarks and compute the aspect ratio of mouth and eye. If aspect ratio is beyond certain threshold this proposed system triggers an alarm to warn the driver. The working module of this flowchart is given in Figure [2.4].

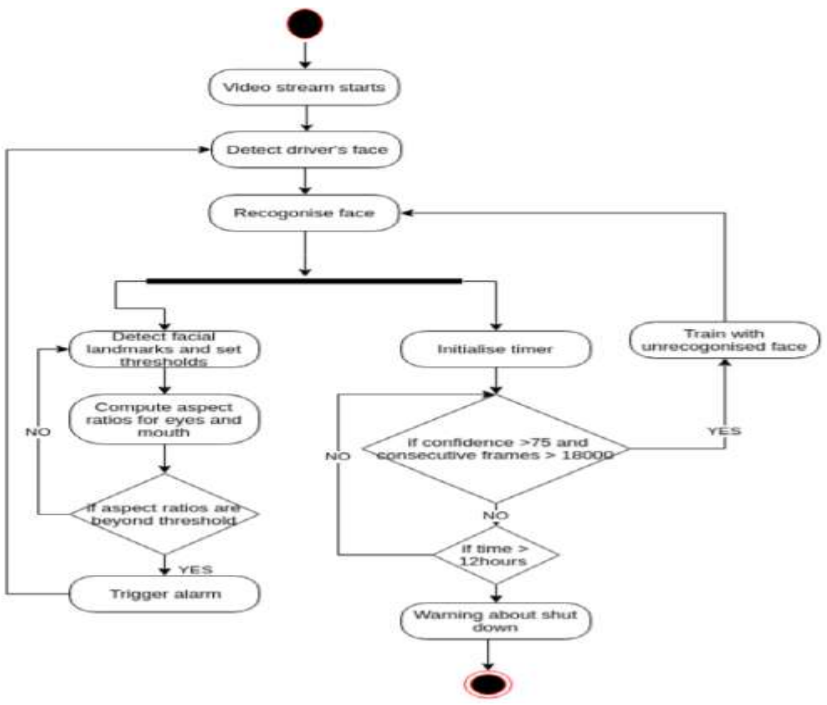

Video stream starts
Detect driver's face
Recogonise face
Detect facial landmarks and set thresholds
Initialise timer
Train with unrecogonised face
Compute aspect ratios for eyes and mouth
NO
if confidence >75 and consecutive frames > 18000
YES
NO
if aspect ratios are beyond threshold
NO
if time > 12hours
YES
Trigger alarm
Warning about shut down

Figure 2.4 : Module (Flowchart)

Input Data: an RGB camera was used to capture the video stream.

Face Detection: The system begins with the face detection process using Histogram of Oriented Gradients (HOG) which is a feature descriptor used for object detection. This technique relies on distribution of intensity gradients or the edge directions for detection features.

A detection window of specified pixels is passed over the image such that gradients are computed using Equation in figure [2.5] for every pixel within the cell. Gradients include pixel orientation and magnitude.

$$\textbf{gradient magnitude:} \quad gm = \sqrt{gm_x^2 + gm_y^2}$$

$$\textbf{gradient direction:} \quad \theta = arctan\frac{gm_y}{gm_x}$$

Figure 2.5 : Gradient Equation

Dlib facial landmarks: The next step is to acquire the facial landmarks. The basic idea of this technique is to locate 68 specific points on face such as corners of the mouth, along the eyebrows, on the eyes, and so forth. It is a pre-trained detector in the dlib 1 library that is able to find these 68 co-ordinates on any face. Sample facial landmark are shown in figure [2.6] and experiments on each frame are shown in experimental section. EAR and MAR calculation: Eye aspect ratio can be using equation in figure [2.7].

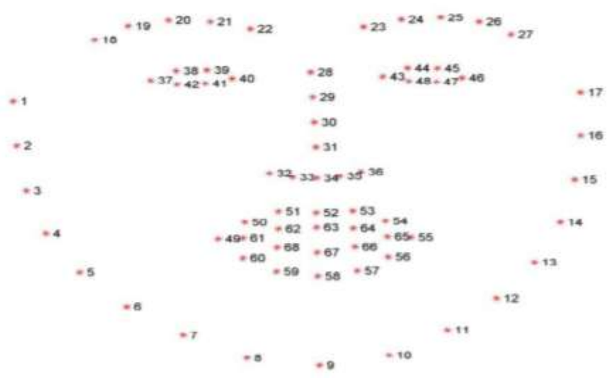


Figure 2.6 : Facial Landmarks

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

Figure 2.7 : Eye Aspect Ratio Equation

In this part of the research, the author developed software that could receive color frames from the camera placed in front of the driver and calculate coordination of facial details. Finally, by comparing and fissuring, the information obtained from interpreting assessment criterion about the alertness or sleepiness together with information resulted from image processing, the software for detection of the levels of drowsiness was developed and promoted. This method is conducted in several steps as follows: first, the image of driver's facial features taken by the camera is transferred to a processor to be processed

Next, the facial features and location of the eyes are determined by Viola-Jones algorithm. For convenience, categorizers are utilized in the cascade sequence figure [2.8]. In this method, driver's face was detected with regard to the oval shape of the head, hue and the eye sockets. Then, the zone of the eyes was recognized by considering the various changes in derivatives (between pupil and white part of eyes) of the visual information

Finally, to recognize whether eyes are closed or open, images are converted into gray scale format. Then, by calculating the mean of component V, illumination of images is normalized. Afterward, image of eyes is converted into binary by determining threshold via OTSU method. This conversion reduces the volume of data. After that, the image is divided into upper and lower part. When eyes are open, due to the color of pupils and eyelash, the ratio of dark pixels in upper part of eye is greater than the state of closed eyes because in the case of closed eyes, eyelids with light color cover pupils. In addition to this, in this state eyelash is in the lower part. To improve images, combining and extending wear were utilized to remove black spots. Then, the ration of black pixels to the whole pixels was calculated in both upper and lower parts. Finally, the ration of these amounts was used to recognize whether eyes are open or closed figure [2.9] the procedure mentioned above: This figure is two sections for differences between open and closed eye.

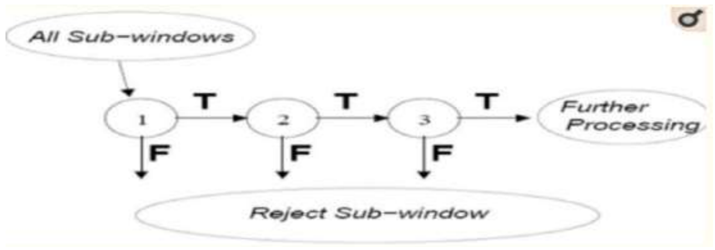


Figure 2.8 : Facial Features by Viola-Jones

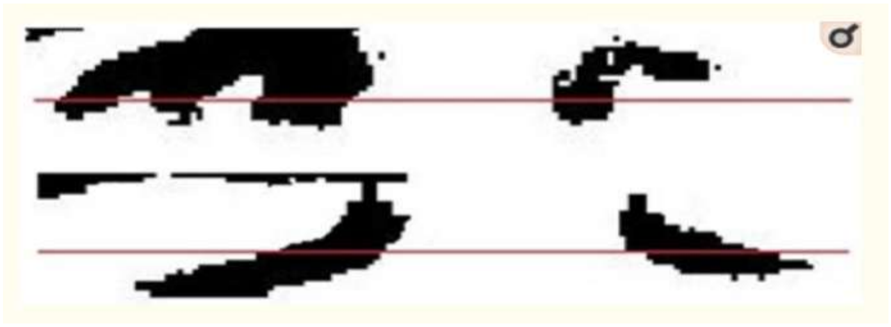


Figure 2.9 : differences between open and closed eye

The system is basically developed to detect drivers dozing at the wheel at night time driving. The system uses an infra-red night vision camera that points directly towards the driver's face and monitors the driver's eyes in order to detect fatigue. In such a case when fatigue is detected, a warning signal is issued to alert the driver. This paper discusses the algorithms that have been used to detect drowsiness. The decision whether the driver is dozing or not is taken depending on whether the eyes are open for a specific number of frames. If the eyes are found to be closed for a certain number of consecutive frames, then the driver is alerted with an alarm.

2.4. Speech Recognition Driver Assistance

The automotive industry is integrating more and more technologies into modern cars. They are equipped with numerous interactive technologies including high quality audio/video systems, satellite navigation, hands-free mobile communication, control over climates and car behavior. With increasing use of such on-board electronics and in-vehicle information systems, the evaluation of driver task demand is becoming an area of increasing importance to both government agencies and industry .With potentially more complex devices for the driver to control, risk of distracting driver's attention increases. Current research and attention theory have suggested that speech-based interactions are less distracting than interaction with visual display. Therefore, to improve both comfort and safety in the car, the driver assistance technologies need effective speech interface between the driver and the infotainment system.

This paper refers to a distant-talking interaction system in the car environment being developed at the labs. The system shares many aspects with the European project VICO (Virtual Intelligent CO-driver), to which ITC- first and Bosch GmbH contributes. Target of the project is to allow a real natural dialogue with a user while driving a car, enabling him/her to get driving assistance, hotel and restaurant reservation and tourist information. The system is based on a HMM speech recognizer for the Italian language, a Dialog Manager module that handles the interaction with the user and a module that retrieves data from databases, called Car Wide Web. In this paper they focus on the structure of the Car Wide Web module, the databases containing the information being used in the project, the specific API developed to access the databases and the corpus of spontaneous speech interactions collected using the Wizard-of-Oz method.

The Car Wide Web is the module responsible for the re tried of the data that is needed by the interaction. Al- most all the dialogue between the system and the user is based on information resident in databases, which can be static or dynamic. A static database may contain historical information regarding a city being visited. a dynamic database (through an Internet connection) may contain weather news or could allow a hotel repsenation. CWW primary task is to understand the queries coming from the Dialog Manager and to retrieve the data from the databases. The actual connection to the data is delegated to a specific API so CWW maintains a certain independence to the physical structure of the databases.

Exchange Protocol in order to have a flexible exchange of data between the DM and the CWW, all the requests and responses are wrapped in an XML format. It has been decided to define two XSDs (XML Schema Definition), one for the data going to the CWW and another for the data coming from the CWW. XSDs can be seen as grammars defining the format of XML files by putting some restrictions on the structure of the files and on the values, contained. With an XSD associated to the XML requests, it comes easy to discover errors in both the structure and' the content of the queries. This leads to an efficient error handling that allows to check the requests before the real connection with the databases.

Internal Structure the CWW module can be subdivided in the following sub modules: XML Parser: The requests coming from the DM are in XML format and need to be unwrapped and understood. This is done with an XML parser using the DOM (Document Object Model) strategy. The parser creates a tree in memory containing all the elements of the request. The parser checks also the XSD file associated to the query and reports possible didation erron. Request Interpretation: The tree is then navigated and interpreted. The first elements of the tree describe the kind of request to be fulfilled, so the related actions are taken. The data needed to create a query to the databases are extracted and manipulated and, through the specific API developed, the results are retrieved. Response Processing: Each data chunk retrieved from the databases is already wrapped in an XML format, but all the respokes coming from the API are wm- posed to create a single string to return to the Dialog Manager. XML Parser: Before being returned, the string is parsed in order to check its validity and to assure a consistent. Response. To do the validation, the parser uses the XSD file associated with the response file.

Functionalities the VICO project status is close to its first prototype delivery. This implies that CWW is not in a find state and not all the foreseen functionalities have been developed.

In order to analyze the structure of effects of dialling a number in comparison to having a conversation on the phone a review was conducted including 19 studies.

Different aspects of driving behavior were examined in these studies. The review summarizes these results with regard to lane keeping, speed and distance regulation and reaction to special stimuli. Ten studies were done in a driving simulator, five in real traffic and two on test tracks. In two additional studies tests were conducted in a laboratory to examine certain tasks similar to driving. The studies used different speech tasks to induce heavy cognitive load of the driver. Figure [2.10] gives the results. Dialling substantially impairs the lateral control of the vehicle. 43% of the studies found an impaired lane keeping. In 50% of the studies, leaving the lane occurred more often. Sometimes the drivers drove slower (20% of the studies). The distance towards preceding cars did not change. If responses to external stimuli were required dialling did not impair the reaction time but more errors occurred (33% of the studies). When having a conversation on the phone, lane keeping was impaired in 21% of the studies. However, other 21% found an improvement of lane keeping. Leaving the lane occurred less often during conversation than while dialling (25% of the studies). In 56% of the studies driver went more slowly while talking. However, in 60% of the studies smaller distances towards preceding cars were found. Finally, 63% of the studies showed prolonged reaction times and more errors (40% of the studies).

Better lane keeping Worse lane keeping Leave lane Slower speed Smaller distance Slow reaction Errors % of studies Conversation Dialling Figure [2.10]: Effects of dialling and having a conversation on the phone Overall dialling clearly impairs lane keeping while the longitudinal control changes only insignificantly. This may be explained by the short duration of this activity. Thus, significant effects on longitudinal control might be expected if these tasks took longer. Similarly, reaction time to sudden events is not changed but the number of errors increases. In summary, 4 in accordance with the multiple resources model dialling disturbs the driving task since same resources are used. When having a conversation on the phone the effects in the area of lateral control are smaller. The effects in the area of longitudinal control show that the drivers reduce speed. This may be a compensatory response to make the driving task easier. However, the smaller distances towards preceding cars combined with slowed down reactions and more frequent errors may lead to dangerous situations during car follow. Comparable effects are found in newer studies not included in this review to mention just a few of the many studies). This interpretation is supported by accident studies demonstrating an increased accident risk when telephoning. This negative effect of phoning is hard to explain with the multiple resources model. It cannot be explained by drivers holding the phone as in 10 of the 19 studies a hands-free phone was used. In a study where the effects of handsfree and handheld phones were compared, both modes lead to an increased workload. Additionally, while drivers compensated talking on a handheld phone by reducing speed, this was not found in handsfree phone conversations. On the one hand the results of this short review support the hypothesis that perceiving additional visual information and handling the phone while dialling substantially impairs driving. According to the multiple resources model it should be possible to avoid these interferences by using auditory information and speech. Unfortunately, this hypothesis is not supported by the review as having a conversation was shown to impair driving even when handsfree phones are used.

This conclusion is supported by other studies showing that conversations themselves decrease performance while driving. Other factors not included in the multiple resources model have to be considered to explain this. The following section examines under which circumstances negative effects of verbal interactions arise. The influence of complexity and emotional quality of verbal interactions

Studies examining the influence of passengers on driving show that driving with passengers decreases accident risk as compared to driving alone . This result points to the interesting fact that under certain circumstances verbal interaction during driving can even increase safety. In a set of experiments in their laboratory the effect of different kinds of interactions were examined in a simple driving simulation (for the complete description. A simplified representation of a car had to be kept on a curvy road by means of a video game steering wheel, gas pedal and brake. Additionally, at some points of barriers suddenly appeared in front of the car. In order to avoid a ‘crash’ drivers had to break fast and hard. These parts of the driving task were termed ‘freeway driving’. Within certain visually defined areas these barriers appeared very frequently. A crash could only be avoided if the speed had been reduced when entering this area. These parts were termed ‘city driving’. In a control condition the subjects drove with a quiet passenger on the right front seat. Three different interaction conditions were induced as role-plays. In the condition ‘small talk’ subjects talked about inconsequential, simple topics without large emotional or cognitive involvement. The condition ‘argument’ induced a negative emotional atmosphere where both participants tried to win the upper hand in a controversy. Finally, in the ‘complex’ condition subjects invented a story together by alternatively producing a sentence. In this condition the analysis was also carried out separately for periods of speaking and listening. For the analysis, 5 the number of crashes was counted when the drivers were not able to avoid the barrier. From this, a relative accident risk was calculated where the number of accidents with a quiet front seat passenger served as reference, i.e. the relative risk in this condition was set to "1". Figure [2.11] shows the relative risks in the different conditions. 012345 Quiet passenger Small talk Argument Complex Listening Speaking Relative Risk City Freeway Figure [2.11]: Relative risks for different conversations on a simulated freeway and city during city driving both small talk and argument clearly increase accident risk. The strongest effect is found for the complex interaction. However, the increased accident risk in this condition is limited to speaking and not found while listening. At first glance a completely different picture emerges for freeway driving. Here only the argument condition leads to an increased accident risk. In the complex interaction the accident risk is even decreased and particularly when speaking. The explanation for these differences is found when speed is considered during freeway driving Figure [2.12]. Quiet passenger Small talk Argument Complex Listening Speaking Speed [%] Speed on the freeway Average speed in the different conversation situations the speed while driving with a quiet passenger was set to 100%. The speed in the other conditions is given as a percentage in relationship to that speed. Speed is reduced during small talk and the complex interaction and particularly when speaking during the complex interaction. In contrary, speed is hardly reduced in the argument condition. The speed reduction in the other conditions can be understood as efforts of the driver to compensate the 6 increased cognitive load caused by the interaction. Drivers know that talking reduces their attention for the driving task and that they cannot react as effectively and drive as safely as without talking. To counteract this, drivers reduce their speed which represents an effective means to provide more time for any necessary reactions. During city driving the speed is already reduced so much that a further reduction of the speed is no longer possible (and thus not shown in the Figure). Thus, in city driving the negative effects of the interactions become visible as an increase of the accident risk. The compensatory reaction of speed reduction on the freeway is not done in the argument condition resulting in an increased accident risk in this situation. This condition was described by the drivers as the most emotional and involving situation. Additionally, the aggressions resulting from the conversation work against a relaxed driving.

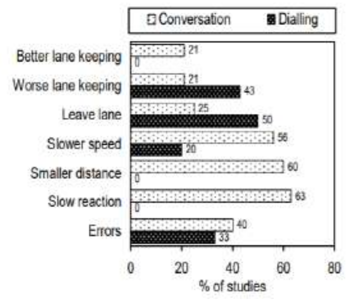


Figure 2.10 : results of different speech tasks to induce heavy cognitive load of the driver.

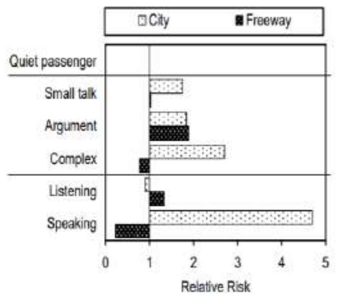


Figure 2.11 : relative risks in the different conditions.

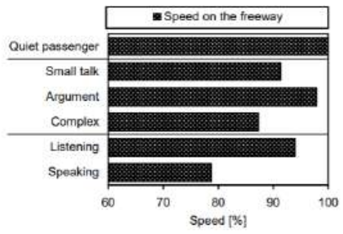


Figure 2.12 : when speed is considered during freeway driving.

2.5. Driver Drowsiness Detection by Applying Deep Learning Techniques

This work presents the development of an ADAS (advanced driving assistance system) focused on driver drowsiness detection, whose objective is to alert drivers of their drowsy state to avoid road traffic accidents. In a driving environment, it is necessary that fatigue detection is performed in a non-intrusive way, and that the driver is not bothered with alarms when he or she is not drowsy. Our approach to this open problem uses sequences of images that are 60 s long and are recorded in such a way that the subject's face is visible. To detect whether the driver shows symptoms of drowsiness or not, two alternative solutions are developed, focusing on the minimization of false positives. The first alternative uses a recurrent and convolutional neural network, while the second one uses deep learning techniques to extract numeric features from images, which are introduced into a fuzzy logic-based system afterwards. The accuracy obtained by both systems is similar: around 65% accuracy over training data, and 60% accuracy on test data. However, the fuzzy logic-based system stands out because it avoids raising false alarms and reaches a specificity (proportion of videos in which the driver is not drowsy that are correctly classified) of 93%. Although the obtained results do not achieve very satisfactory rates, the proposals presented in this work are promising and can be considered a solid baseline for future works.

The aim of this work is to develop a system that is able to estimate the fatigue of a driver by using sequences of images that are recorded in such way that the face of the subject is visible.

The drowsiness detection system developed in this work is part of a driver-based ADAS system[16,17], with two important restrictions: early detection and minimization of the number of false positives. The idea is that the system will warn the driver only in real cases of fatigue, to avoid false positives, which would cause boredom in the driver, causing them to turn off the ADAS, without executing the rest of the functionalities.

When it comes to the recording of the driver, it is important to determine the frame rate that the camera has to communicate to the system. A high frame rate will overload the system because of the high number of frames per second (FPS) that have to be evaluated, but a low amount of FPS can affect negatively the system performance. In this domain, it is necessary that there are enough FPS to appreciate details of the image sequence that have a very short duration, such as blinks.

Since the average blink duration ranges from 100 to 400 ms [18], in this work, a frame rate of 10 FPS is used, which is enough to detect blinks and avoid overloading the system. This way, 600 frames are evaluated every time a new frame is captured by the camera. To do this, the system stores the previous 599 frames, so that a full sequence of 60 s is analyzed at each instant.

This work proposes two alternative solutions to estimate drivers' drowsiness. The first alternative uses a recurrent and convolutional neural network, while the second one uses AI and deep learning techniques to extract numeric features from images, and then introduces them into a fuzzy logic-based system. However, both solutions follow the same process structure, which consists of three phases: preprocessing, analysis, and alarm activation. Figure 2.13 shows the three different phases.

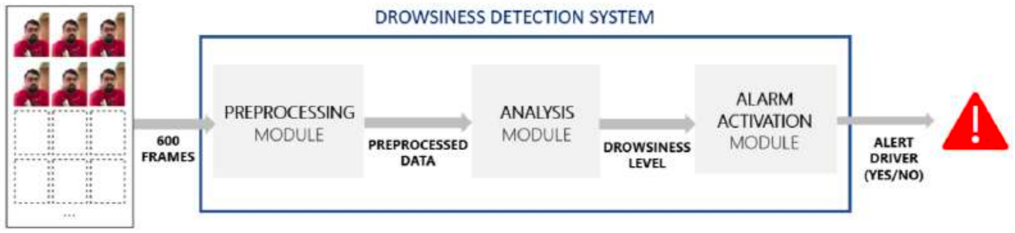


Figure 2.13 : System overview.

As shown in Figure 2.13, the system receives 600 frontal images of the driver, correspond- ing to the last 60 s recorded by a camera placed on the vehicle (for example, on the driver's dash) at 10 frames per second (FPS).

These images are received by the *preprocessing module*, whose objective is to trans- form the received image into data that can be used by the drowsiness detection model. The preprocessed data are then sent to the *analysis module*, which performs the fatigue detection tasks and assesses the level of drowsiness of the driver at that moment, based on the information from the last 60 s. Lastly, the calculated drowsiness level is transmitted to the *alarm activation module*, which uses the last levels of drowsiness to determine whether it is necessary to alert the driver or not.

As mentioned above, one of the main objectives of the alarm activation module is to minimize the number of false positives of the system (drowsiness alerts when the driver is actually awake), since a high number of false positives disturbs the driver and increases the possibility of turning off the system. This is one of the reasons that motivate the experimentation over videos instead of frames and make the tests more exacting, since the activation of a single alarm in a 10 min video means that, regardless of what is detected before or after that moment, the classification of the video will be considered "drowsy".

Once the system has made its decision on whether to alert the driver or not (a yes/no possible outcome), it will communicate its decision to the human–computer interaction system responsible for warning the driver by using visual and/or sound stimuli.

The three modules (preprocessing, analysis and alarm activation) of each of the two alternative solutions are described below.

3.1. Alternative I: Recurrent and Convolutional Neural Network

The first alternative, although it is focused on using deep learning techniques, uses one artificial intelligence technique (linear SVM combined with HOG) to preprocess the driver's image and extract the face. This new image is sent to the *analysis module*, that applies deep learning techniques to analyze the fatigue of the driver at that moment.

In this case, the *analysis module* is composed of a recurrent and convolutional neural network (which we will call "recurrent CNN"). This recurrent CNN is responsible for detecting the fatigue of the driver at the current moment by calculating a numerical output that represents the estimated drowsiness level of the driver. This value is sent to the *alarm activation module*, where it is decided whether or not to activate the corresponding alarm.

Figure 2.14 shows a diagram representing the process followed by the system, divided in 3 stages that are detailed in this section.

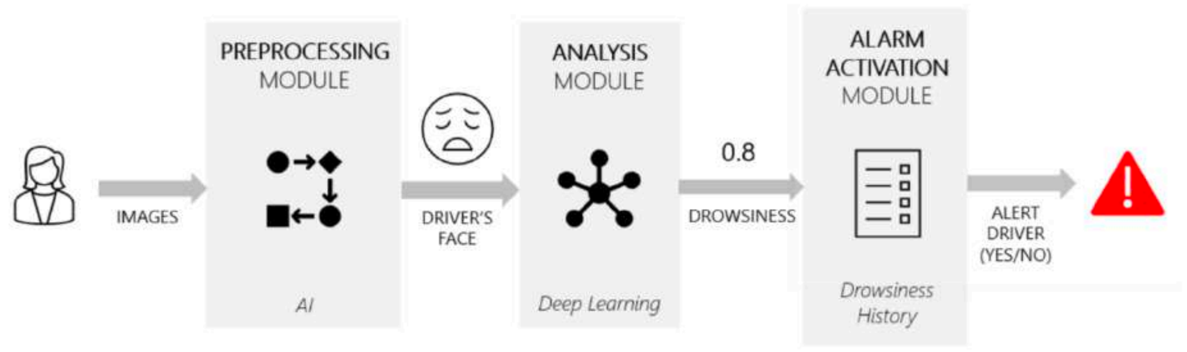


Figure 2.14 : Alternative I overview.

3.1.1. Preprocessing

As previously mentioned, before sending the image to the recurrent CNN, we crop and preprocess the original frame at the *preprocessing module*. To avoid that the model is affected by possible noise, the first step is to apply a Gaussian blur to the original image. Blurring images is a common technique used to smooth edges and remove noise from an image, while leaving most of the image intact.

From the blurred image, we extract the image region that contains the face, for which DLIB's library [18] is used. In particular, we use its face detector, which calculates the coordinates of the face location by using histograms of oriented gradients (HOG) and a linear supporting vector machine (SVM) [19]. This way, this AI technique is used to crop the original image and leave only the face of the driver.

After this, we scale the face to 64 × 64 px, a size that allows us to preserve the details of the face and considerably reduces the computation time required to process each image. Next, we perform a histogram equalization to adjust the contrast of the image and avoid unnecessary details. We apply ImageNet mean subtraction, a technique that reduces the probability that the classification is affected by the illumination of the image, and that allows to use transfer learning with models that have been trained over the ImageNet domain. Figure 3 shows an example of the preprocessing that is performed.

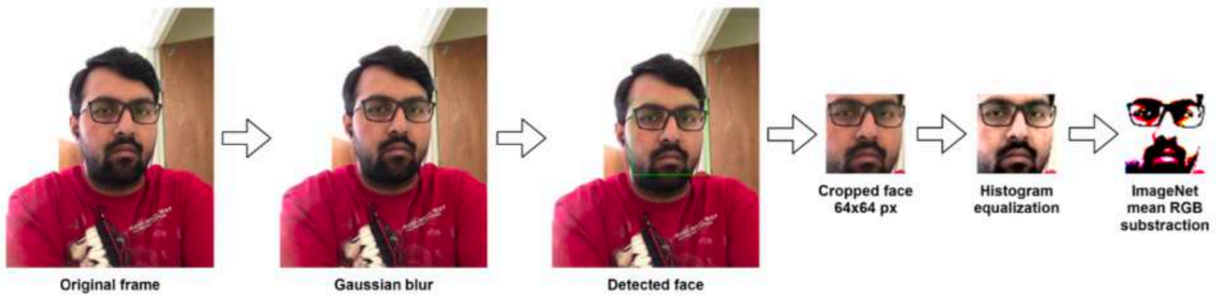


Figure 2.15 : Preprocessing of the driver's face.

3.1.2. Analysis

The analysis module uses a recurrent and convolutional neural network to estimate the drowsiness level of the driver. The CNN is based on the EfficientNetB0 architecture [20], which presents a lightweight model that is highly precise. Smaller models are processed faster, and EfficientNetB0 presents the smallest model of the EfficientNet series. Since the differences in accuracy are not significant in our domain when upgrading to a superior model, we consider EfficientNetB0 to be the most adequate model for this case, where the model needs to quickly obtain a prediction. This way, we perform transfer learning on this model by using previously trained weights that have great performance in recognizing objects on images from the ImageNet dataset [21].

The EfficientNetB0 architecture is divided into 9 blocks, each of them formed by multiple layers. Figure 2.16 shows an overview of this architecture, where the 9 main blocks are represented.

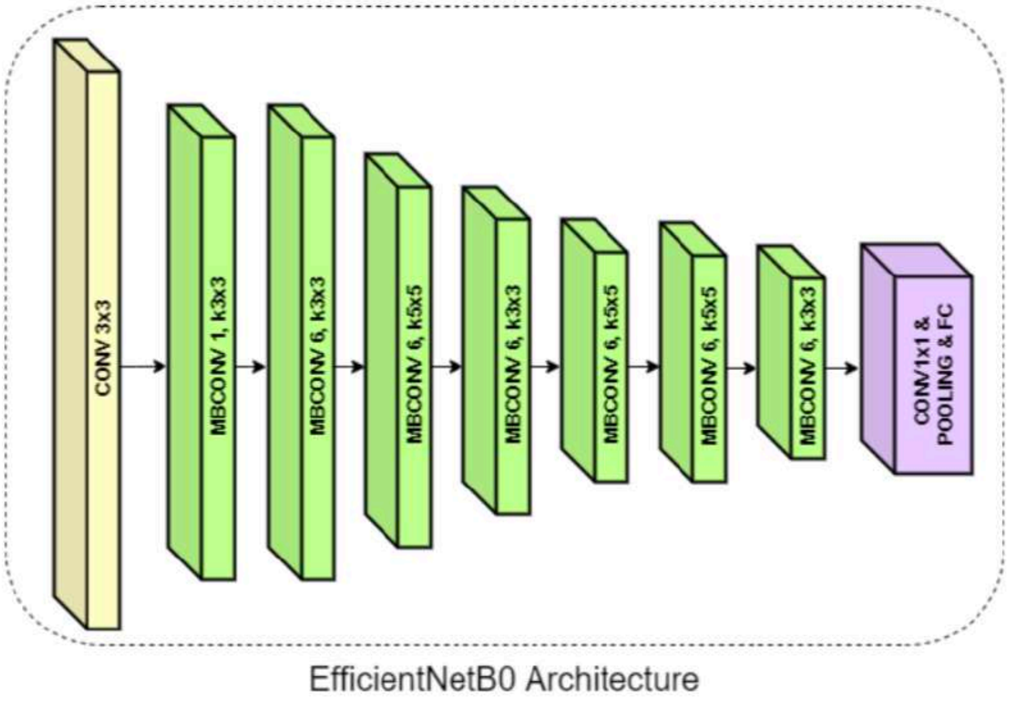


Figure 2.16 : EfficientNetB0's architecture overview.

In this work, we focused on tuning the number of frozen layers of the transferred model to obtain the highest possible accuracy. For this, we trained our system using four different configurations, named after the amount of layers that are trained within the EfficientNetB0 model:

- *No training:* All layers are frozen, no layer from EfficientNetB0 is trained.
- *Top training:* All layers are frozen except for the layers of Block 9 (pooling, flatten, dense and dropout layers).
- *Partial training:* In addition to layers of Block 9, layers from Block 8 (the last MBConv block) are also trained.
- *Full training:* All layers remain unfrozen.

To test these configurations, a preliminary evaluation was performed, where each configuration was trained over 25 epochs. After analyzing the training accuracy obtained from this experimentation, we concluded that the best performing configuration was *top training*. So, the weights of the model are frozen at every layer, except for the last block of the layers (which consists of pooling, flatten, dense and dropout layers), preventing the information loss of the early layers while training the new model. In this case, the Efficient- NetB0 architecture is used as the base of a GRU (gated recurrent unit) recurrent neural network, combining, in this way, a CNN with a RNN.

The GRU network receives 600 images with the face of the driver, and it has to identify the characteristics that reveal if the driver was drowsy during that last minute. The output of the GRU is received by the final classifier, where some dense and dropout layers are used to estimate the drowsiness level of the driver, which is a number between 0 and 1.

Figure 2.17 shows the final architecture of the network. As mentioned, transfer learning is applied in the module labeled GRU in Figure 2.17, where

EfficientNetB0 is implemented. The rest of the additional layers are those that are trained to transform the knowledge provided by EfficientNetB0 into a drowsiness level estimation.

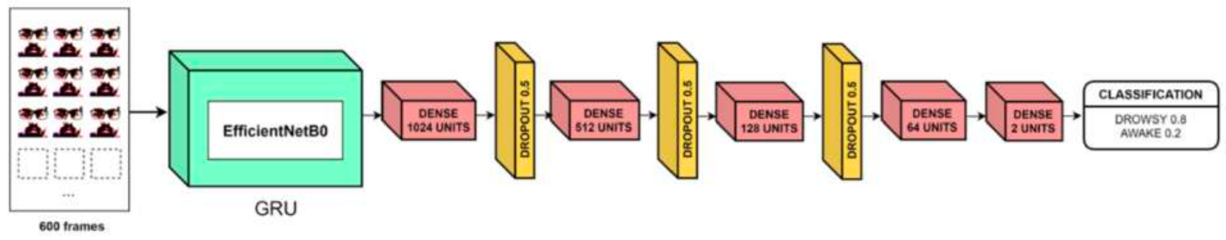


Figure 2.17 : Architecture of the model used for drowsiness estimation.

3.1.3. Alarm Activation

The following conditions must be met to alert the driver:

- The driver is considered to be drowsy at a specific moment when the output of the analysis module is greater than a threshold, called *drowsiness_threshold*. This value ranges between 0 and 1.

- The driver has to be considered drowsy for multiple instants of the last 60 s to raise an alarm. This is determined by the variable *min_time*, which represents how many seconds the driver has to be drowsy before alerting him or her. This value ranges between 0 and 60.

This way, when the driver is considered drowsy for at least the established minimum time, an alarm is raised. If the condition to raise an alarm continues after alerting the driver, a second alarm is not raised. Instead, another alarm is raised only if the conditions for alerting the driver are no longer met, and after that, drowsiness is detected again.
To calculate the optimal values of variables *drowsiness_threshold* and *min_time*, multi- ple tests were performed.

3.2. Alternative II: Deep Learning Combined with Fuzzy Logic

In this case, the images are preprocessed by using artificial intelligence (linear SVM combined with HOG and ensemble of regression trees) and deep learning techniques (pre-trained CNN), which extract numerical characteristics that can be introduced on a fuzzy inference system (FIS). After this, the FIS returns a numerical output that represents the estimated drowsiness level of the driver, and this value allows the system to raise an alarm if needed.

Figure 2.18 shows a diagram representing the process followed by the system, divided into 3 stages, which are explained in this section.

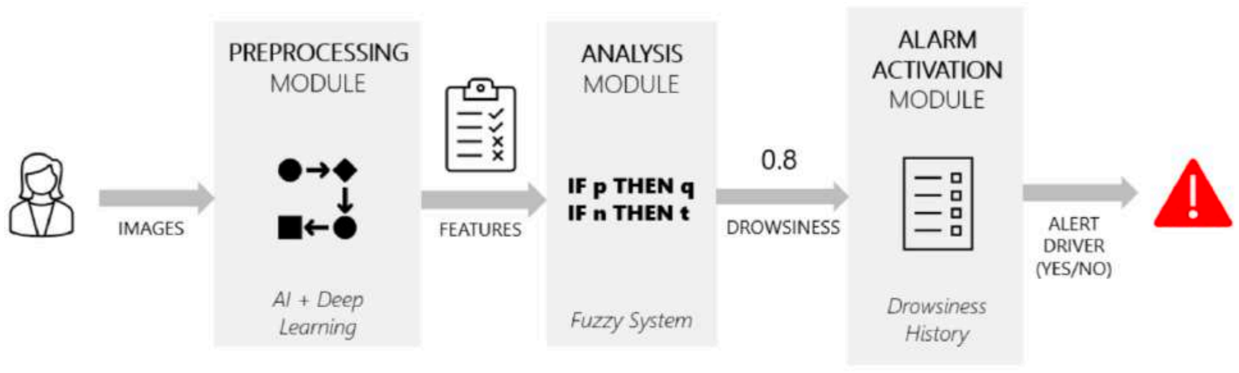


Figure 2.18 : Alternative II overview.

3.2.1. Preprocessing

The parameters that we are interested in discovering from the driver's image are the following:

- Number of blinks (*blinks*).
- Average blinking duration (*avg_blink*).
- Number of microsleeps, i.e., blinks with a duration of over 500 ms (*microsleeps*).
- Number of yawns (*yawns*).
- Time spent yawning (*avg_yawn*).

The first step to calculate these parameters is locating the driver's face. To do this, DLIB's face detector is used once again. However, in this case, we also use the landmark detector that DLIB provides, which is an implementation of [22], where the shape predictor uses an ensemble of regression trees. DLIB's landmark detector represents the facial features with a group of 68 points, 12 of them representing the eyes (6 points per eye), which allows us to know the position of the driver's eyes.

2.6. Driver distraction using deep learning approach

The authors of developed a support vector machine-based (SVM-based) model to detect the use of a mobile phone while driving. The dataset used in this work consists of frontal images of the driver's face. The pictures used to develop the model showed both hands and face with a pre-made assumption and were focused on two driver actions, ia driver with a phone and one without a phone. A SVM classifier was applied to detect the driver actions and frontal images of the drivers were used. SVM classification was used to detect the use of a mobile phone while driving. The dataset used in this study was collected using transportation imaging cameras placed on highways and traffic lights. The authors of another study did some extraordinary work by sitting on a chair and mimicking a certain type of distraction (e.g., talking on a mobile phone). In this work, the AdaBoost classifier was applied along with hidden Markov models to classify Kinect RGB-D data. However, this study had two limitations, i.e., the effect of lighting and the distance between the Kinect device and the driver. In real-time applications, the effect of light is very important, and this should be taken into consideration for efficient results. The authors of suggest using a hidden conditional random-fields model to detect mobile-phone usage. The database was created using a camera mounted above the dashboard and the hidden conditional random-fields model used to detect cell-phone usage. The model considers the face, mouth, and hand features of images obtained from a camera mounted above the dashboard. In another study related to phone usage ,a faster Region Based Convolutional Neural Networks (R-CNN) model was designed. In that study, both the use of a phone by the driver and hands on the wheel were detected. Here, the main contribution was the segmentation of hand and face images. The dataset used to train the faster R-CNN was the same as that used in, in which a novel methodology called multi- scale faster R-CNN was proposed. With this methodology, higher accuracy with low processing time was obtained. In, the authors created a dataset for hand detection in an automotive environment and achieved an average precision of 70.09% using the aggregate channel features object detector. Another study discusses the detection of cell-phone usage by a driver. In that work, the authors applied a histogram of gradients (HOGs) and AdaBoost classifiers. Initially, the supervised descent method is applied to locate the landmarks on the face, followed by extraction of bounding boxes from the left-hand side of the face to the right-hand side. A classifier was applied to train these two regions to detect the face, followed by left- and right-hand detection. Finally, after applying segmentation and training, 93.9% accuracy at 7.5 frames per second was obtained. Motivated by the same, several researchers worked on cell-phone usage detection while driving. In , the authors designed a more inclusive distracted-driving dataset with a side view of the driver considering four activities: Safe driving, operating the shift lever, eating, and talking on a cell phone. The authors achieved 90.5% accuracy using the contourlet transform and random forest. The authors also proposed a system using a pyramid of histogram of gradients (PHOG) and multilayer perceptron that yields an accuracy of 94.75% and Faster R-CNN . Also investigated in was action detection using three actions, hands on the wheel, interaction with the gears, and interaction with the radio; an attempt was then made to classify these actions. Separate cameras were fixed to capture face and hand actions. A SVM classier was applied to determine the driver's actions; 90% and 94% accuracy were achieved for hand actions and face and hand-related information, respectively. In another study, the Southeast University dataset was used. In this study, the authors focused on hands on the wheel, usage of a phone, eating, and operating the shift lever. The dataset consists of labels for each image for these four driver actions. The authors applied various classifiers and obtained 99% accuracy for the CNN classifier. Another study focused on the following seven actions: Checking the left-hand mirror, checking the right-hand mirror, checking the rear mirror, handling in-hand radio devices, using the phone (receiving calls and texting), and normal driving. In a pre-processing step, the author cropped the driver's body from the image, removed the background information, and applied the Gaussian mixture model; after pre-processing the data, CNN classification was applied to classify these actions. Using AlexNet, 91% accuracy was achieved for predicting driver distraction. In other studies, segmentation was performed to extract the drivers' actions by applying Speeded Up Robust Features (SURF) key points. After pre-processing, HOG features were extracted and K-NN classification applied. A recognition rate of approximately 70% was achieved. In, a similar methodology was applied. However, in that study, two different CNN models were applied to classify the actions. In the first model, triplet loss was applied to increase the overall accuracy of the classification model. By doing this, the authors achieved a 98% accuracy rate. They used 10 actions and the StateFarm dataset, which comprises labeled images, for distraction detection. Here, the detected actions included safe driving, texting using the right hand, talking on the phone using the right hand, texting using the left hand, talking on the phone using the left hand, changing the radio station, drinking, talking to the passenger sitting behind the driver, checking hair and makeup, and talking to the passenger sitting beside the driver. In, the American University in Cairo (AUC) dataset was used to detect distraction. This dataset is similar to the StateFarm dataset. In training was done using five different types of CNNs. After applying the CNN, the experimental results were weighted by implementing a genetic algorithm to further classify the actions. In another work conducted by the authors of, a modified CNN was applied, followed by numerous regularization techniques. These techniques helped reduce the overfitting problem. The accuracy achieved by that work was 96% for driver distraction. A deep-learning technique was applied in using the AUC and StateFarm datasets in which the video version of the dataset was worked on. The authors proposed a deep neural network approach called multi-stream long short-term memory (LSTM) (M-LSTM). The features used in this work were from, with contextual information, which was retrieved from another pertaining CNN; 52.22% and 91.25% accuracy were achieved on the AUC and the StateFarm distracted-driver datasets, respectively. In [43], a deep-convolution- network technique was applied for the recognition of large-scale images. Similar work for the recognition of face was implemented in [44] in three dimensions by applying feature-extraction techniques and classification methodologies. Another recognition technique has been applied for palmprints by applying a robust two-dimensional Cochlear transformation technique.

CHAPTER 3

SYSTEM OVERVIEW

Outlines:

3.1. System Architecture

The system is developed using python language. When camera opens, it starts a video stream automatically and starts in detecting driver's face. Then, each captured image (frame) is transferred to the shape predictor model. The model detects facial landmarks to find the coordinates of eyes and mouth to detect drowsiness and lack of focusing. The EAR and MAR thresholds are set to 0.3 and 0.6. When EAR and MAR reach thresholds, drowsiness is detected and an alarm is triggered. At the same time, when the camera is not able to detect the driver's full face, an unfocus alarm is triggered. Another process in the same time is when the driver suddenly disappears from the frame, the same alarm is triggered.

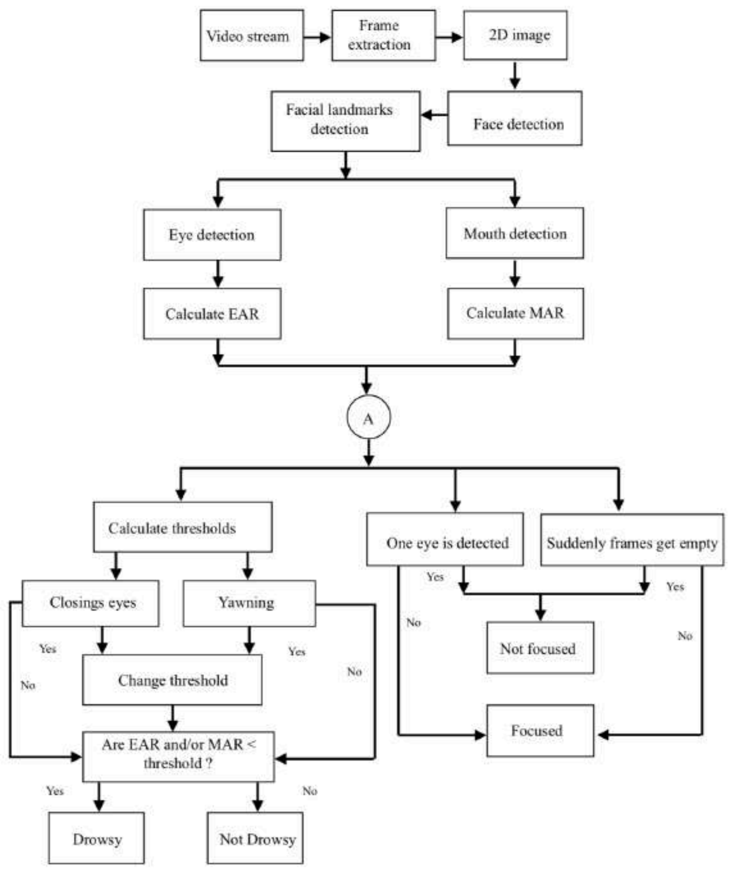


Figure 3.1 : Activity Diagram.

3.2. Use Case Diagram

This is a representation of the way the system works using use case diagram.

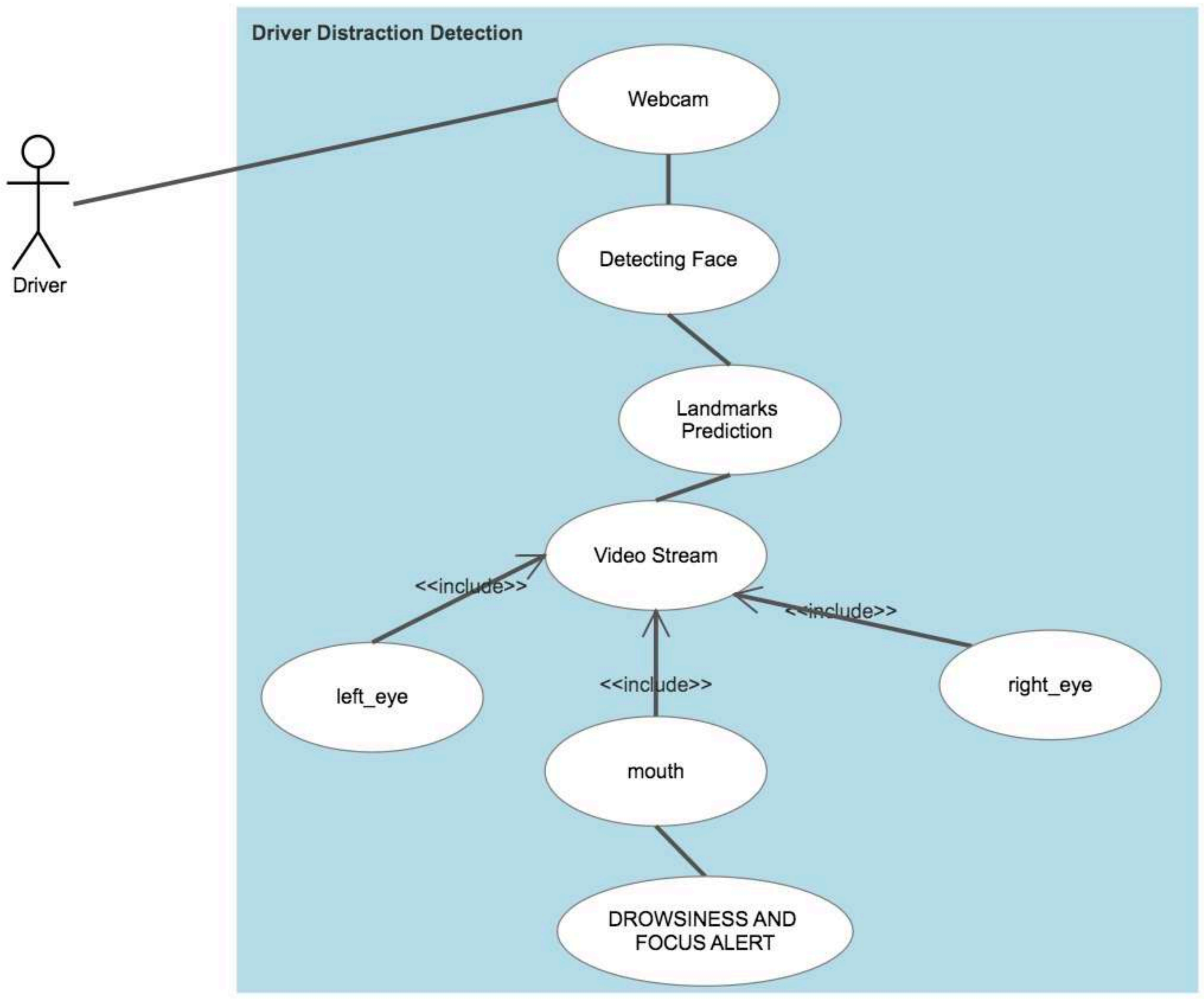


Figure 3.2 : Use Case Diagram.

3.3. Activity Diagram

Another representation provides a view of the behavior of a system by describing the sequence of actions in a process using activity and class diagrams.

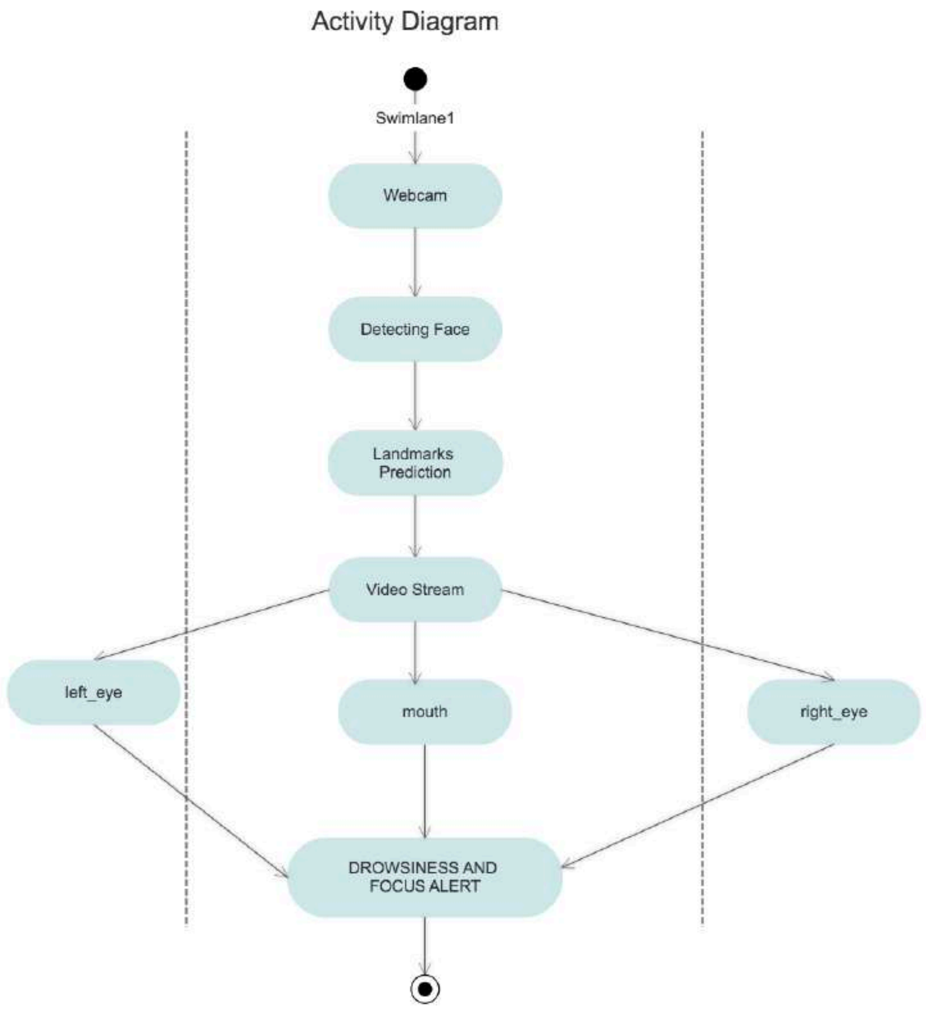


Figure 3.3 : Activity Diagram.

3.4. Class Diagram

Here is a blueprints of the system to model the objects that make up the system using class diagram.

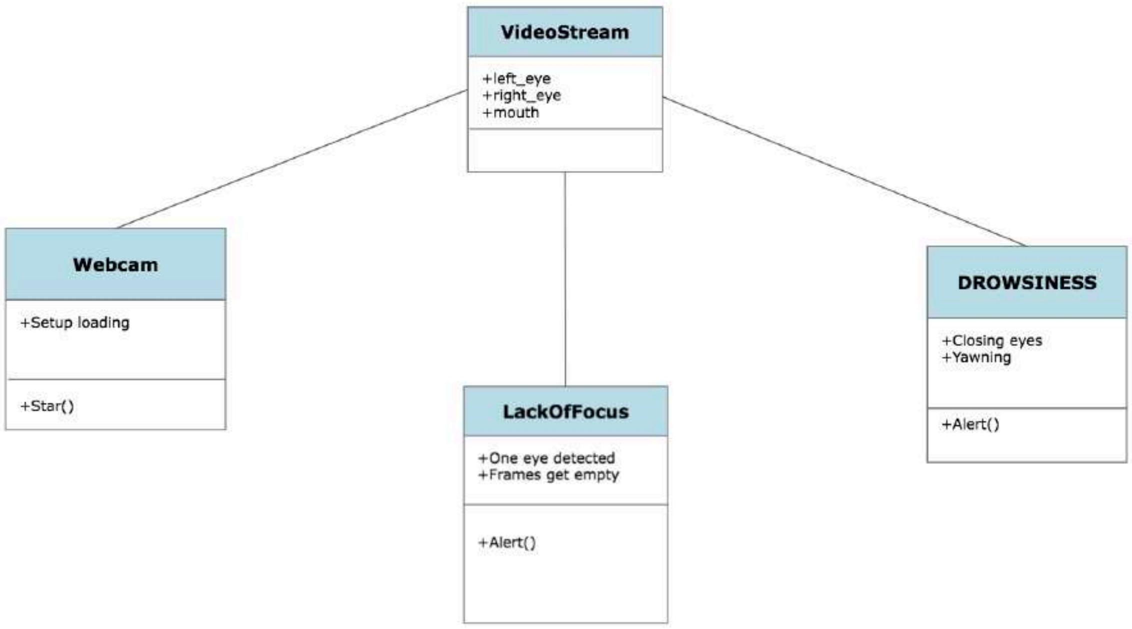


Figure 3.4 : Class Diagram.

3.5. State Chart Diagram

This is a representation by state chart diagram to describe the states of different objects in its life cycle. These states of objects are important to analyze and implement them accurately.

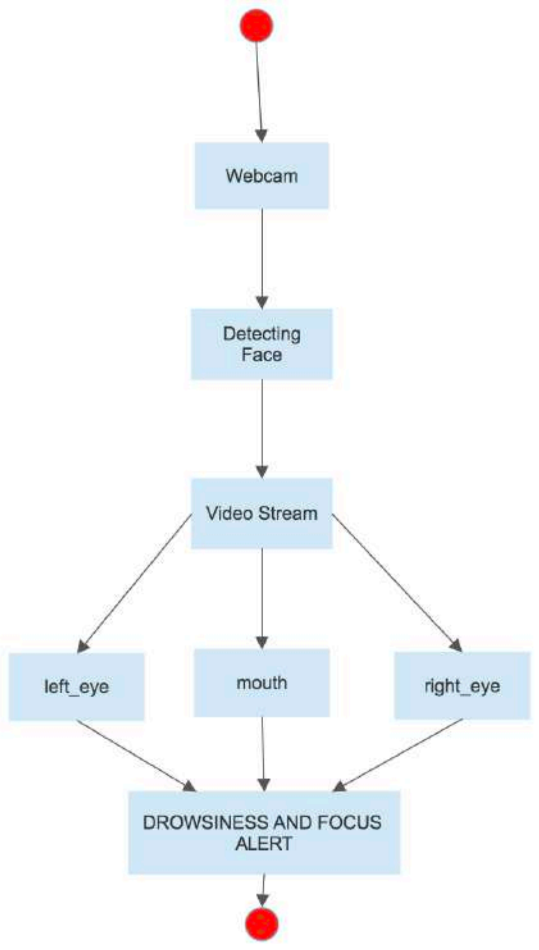


Figure 3.5 : State Chart Diagram.

3.6. Deployment Diagram

Here is a representation using deployment diagram. Deployment diagram is used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagram is used to describe the static deployment view of a system.

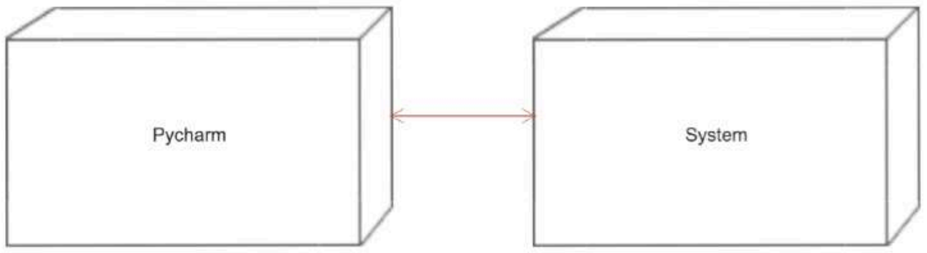


Figure 3.6 : Deployment Diagram.

CHAPTER 4

DRIVER DISTRACTION DETECTION

Outlines:

4.1. Driver distraction detection

This is the block diagram that shows the steps of the system. The system starts with video stream, then it starts to detect the driver's face. The next step is detecting facial landmarks and setting EAR and MAR thresholds, if they are beyond threshold an alarm is triggered. In parallel, another process is working to detect when only half the face of the driver is detected to trigger an unfocus alarm. Another parallel process to see if suddenly frames got empty after a face was detected to trigger an alarm.

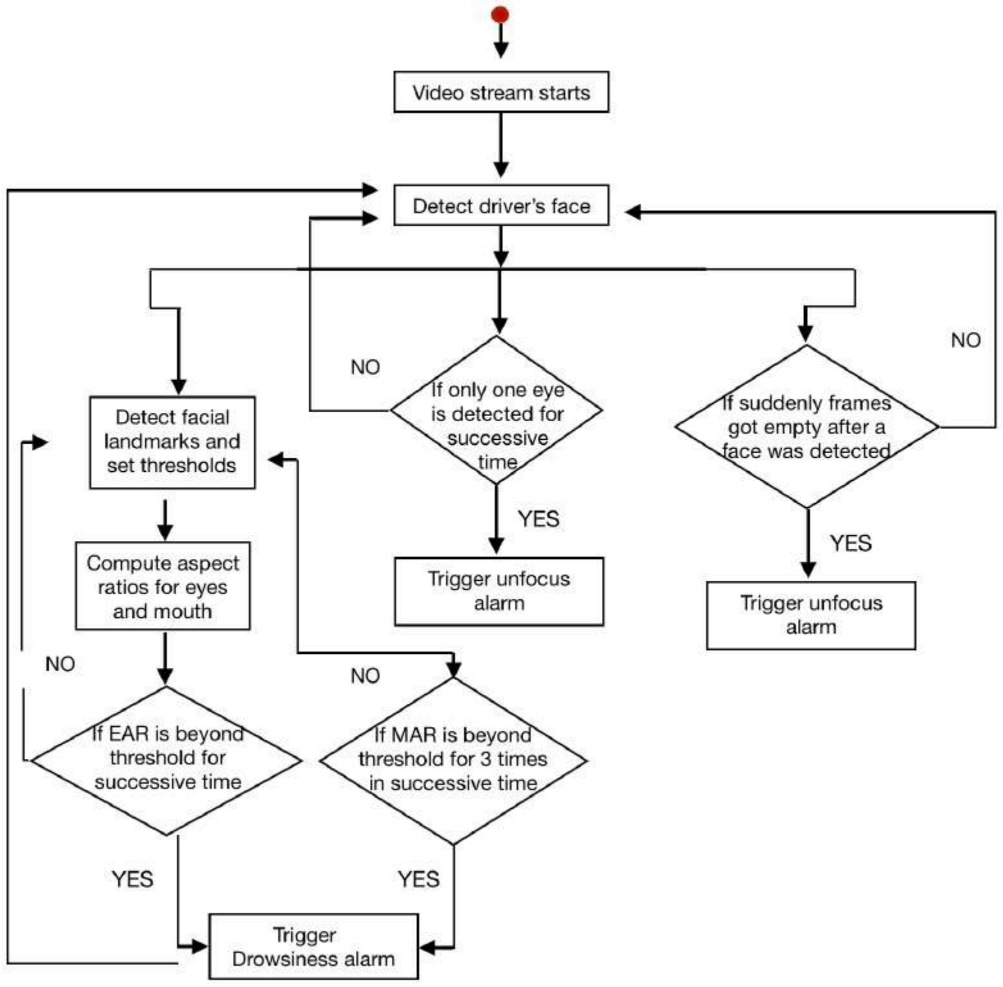

Video stream starts
Detect driver's face
NO
If only one eye is detected for successive time
NO
If suddenly frames got empty after a face was detected
Detect facial landmarks and set thresholds
YES
YES
Compute aspect ratios for eyes and mouth
Trigger unfocus alarm
Trigger unfocus alarm
NO
NO
If EAR is beyond threshold for successive time
If MAR is beyond threshold for 3 times in successive time
YES
YES
Trigger Drowsiness alarm

Figure 4.1 : An overview of our proposed work.

4.1.1. Video stream

- The car was rigged with a drowsiness detector.
- The camera was used for this project was a Webcam Night Vision 300 GoldShip Mod.3526 - Leadership
- This is the preferable camera as it:

⇒ Is relatively affordable.

⇒ It has 6 LEDs for use in dark environments.

⇒ Maximum resolution: 5M Pixels

⇒ Is plug-and-play compatible with nearly every device was tried it with (including the Raspberry Pi).

- The camera was taken and mounted it to the top of the dash using some double sided tape to keep it from moving around during the drive.
- The camera was then connected to the raspberry Pi on the seat next to the driver.
- The installed position of the camera module should avoid blocking the sight of the driver to ensure it could capture the driver’s face fully.
- The Raspberry Pi module is powered-up by the car adapter.
- When the system starts, it will collect the frontal scene as the HD stream. Later, this video stream will be resized to 640×480 pixels with 30 frames per seconds encoded in H264 format, in grayscale.

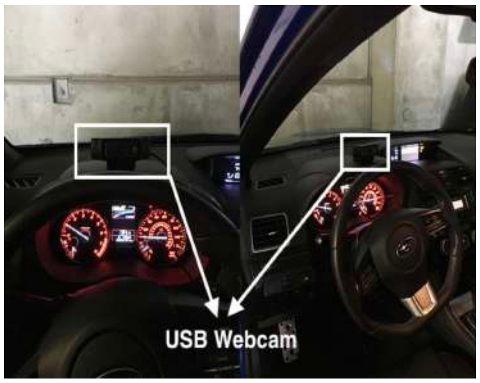


Figure 4.2 : Connecting Camera

4.1.2. Face Detection With Dlib (HOG + Linear SVM)

4.1.2.1. Steps of Histogram of Oriented Gradients (HOG) in Dlib

One of the most popular implementation for face detection is offered by Dlib and uses a concept called Histogram of Oriented Gradients (HOG). This is an implementation of the original paper by Dalal and Triggs [18]. The dataset used for training, consists of 2825 images which are obtained from LFW dataset and manually annotated by Davis King, the author of Dlib [23]. The idea behind HOG is to **extract features into a vector**, and feed it into a classification algorithm like a Support Vector Machine for example that will assess whether a face (or any object you train it to recognize actually) is present in a region or not.

The features extracted are the distribution (histograms) of directions of gradients (oriented gradients) of the image. Gradients are typically large around edges and corners and allow us to detect those regions.

Dlib was originally introduced as a C++ library for machine learning by Davis King. But later, a Python API was also introduced which we can easily install using the `pip` package manager.

Dlib's face detection APIs became quite popular among machine learning and computer vision practitioners due to their ease of use. Specifically, Dlib provides two different methods for face detection:

- Face detection using HOG and Linear SVM. (which we will use in our project).
- Face detection using CNN.

We used HOG + Linear SVM face detector as it is accurate and computationally efficient.

Dlib's HOG + Linear SVM face detector is fast and efficient. By nature of how the Histogram of Oriented Gradients (HOG) descriptor works, it is *not* invariant to changes in rotation and viewing angle.

Figure 4.3 displays the results of applying dlib's HOG + Linear SVM face detector to an input image containing multiple faces.

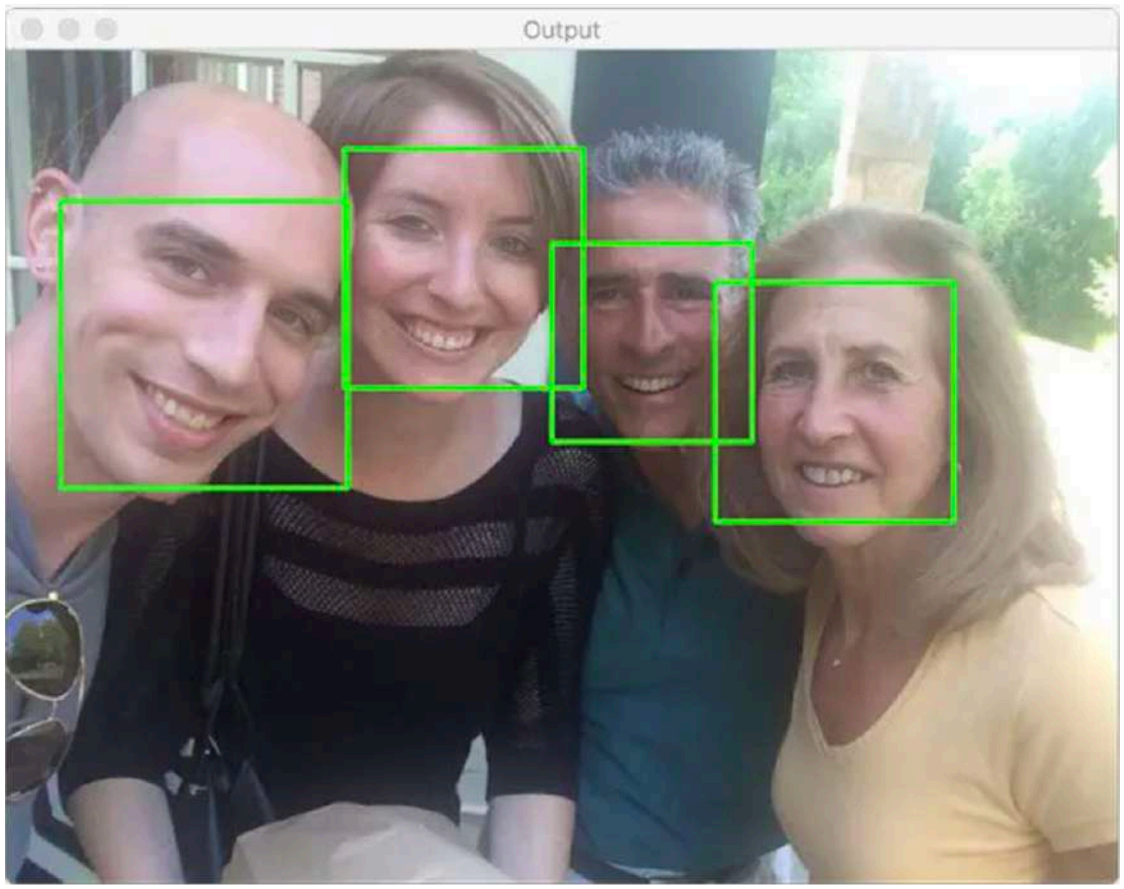


Figure 4.3: Successfully applying dlib's HOG + Linear SVM face detector

In the original paper, the process was implemented for human body detection, and the detection chain was the following :

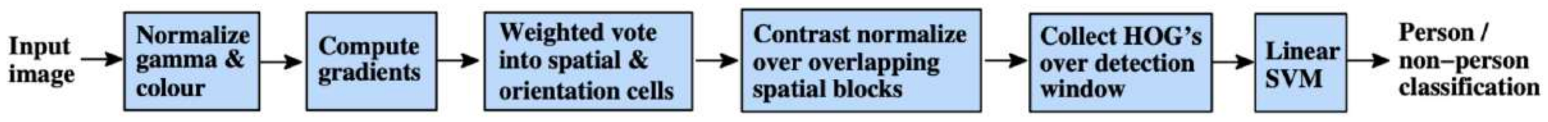

Input image
Normalize gamma & colour
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window
Linear SVM
Person / non−person classification

Figure 4.4 : HOG Detection Chain

1) Computing the Gradient Images

The first step is to compute the horizontal and vertical gradients of the image, by applying the following kernels :

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Figure 4.5 : Kernels to compute the Gradients

The gradient of an image typically removes non-essential information.

2) Compute tyhe HOG

The image is then divided into 8x8 cells to offer a compact representation and make our HOG more robust to noise. Then, we compute a HOG for each of those cells. To estimate the direction of a gradient inside a region, we simply build a histogram among the 64 values of the gradient directions (8x8) and their magnitude (another 64 values) inside each region. The categories of the histogram correspond to angles of the gradient, from 0 to 180°. Ther are 9 categories overall : 0°, 20°, 40°… 160°.

We then calculate 2 information :

1. Direction of the gradient
2. Magnitude of the gradient

When we build the HOG, there are 3 subcases :

1. The angle is smaller than 160° and not halfway between 2 classes. In such case, the angle will be added in the right category of the HOG
2. The angle is smaller than 160° and exactly between 2 classes. In such case, we consider an equal contribution to the 2 nearest classes and split the magnitude in 2

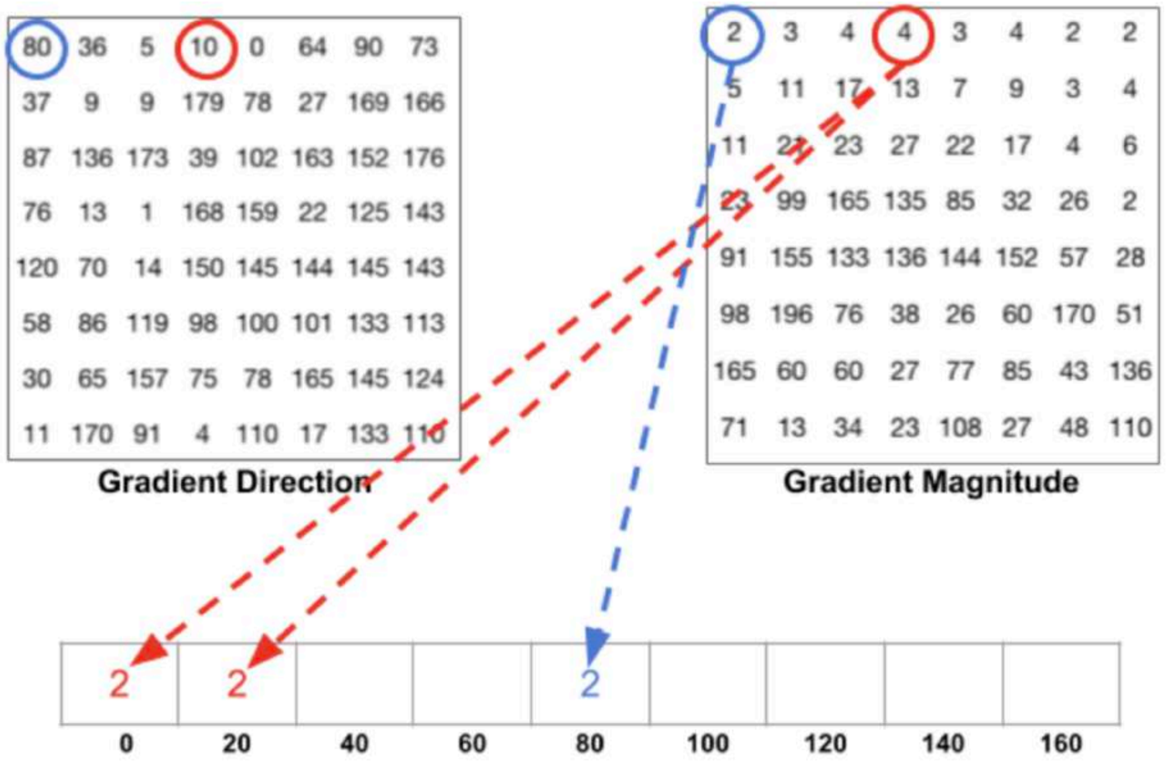
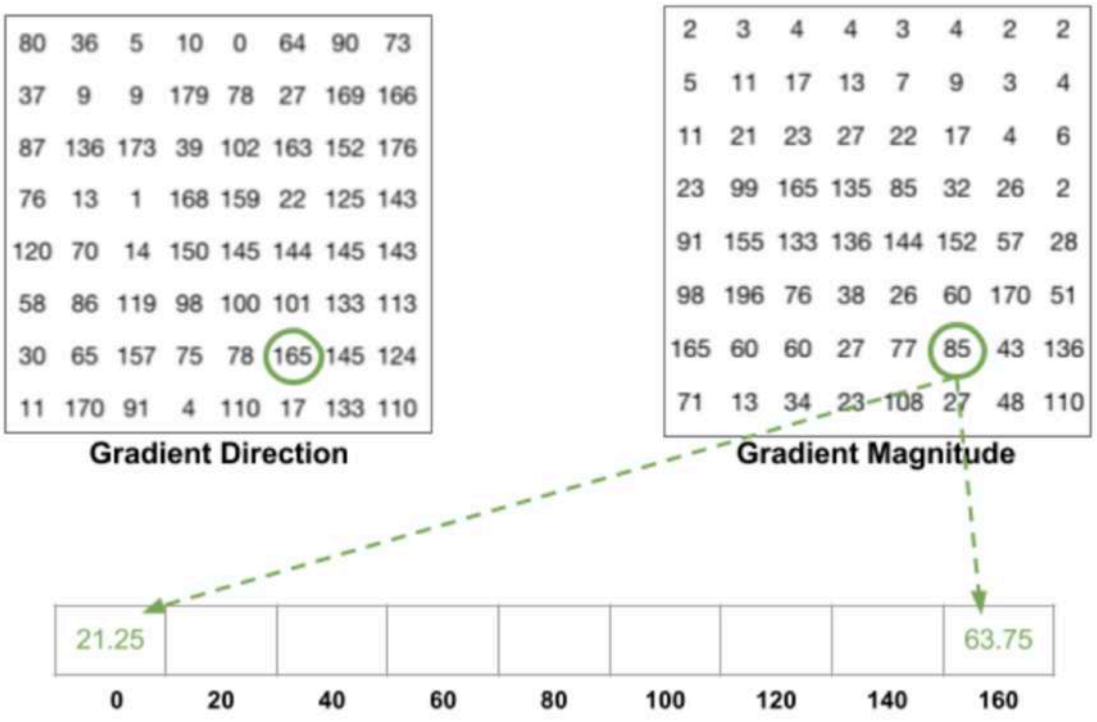


Figure 4.6 : Histogram of Gradients

3. The angle is larger than 160°. In such case, we consider that the pixel contributed proportionally to 160° and to 0°.



80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

0	20	40	60	80	100	120	140	160
21.25								63.75

Figure 4.7 : Histogram of Gradients

The HOG looks like this for each 8x8 cell :

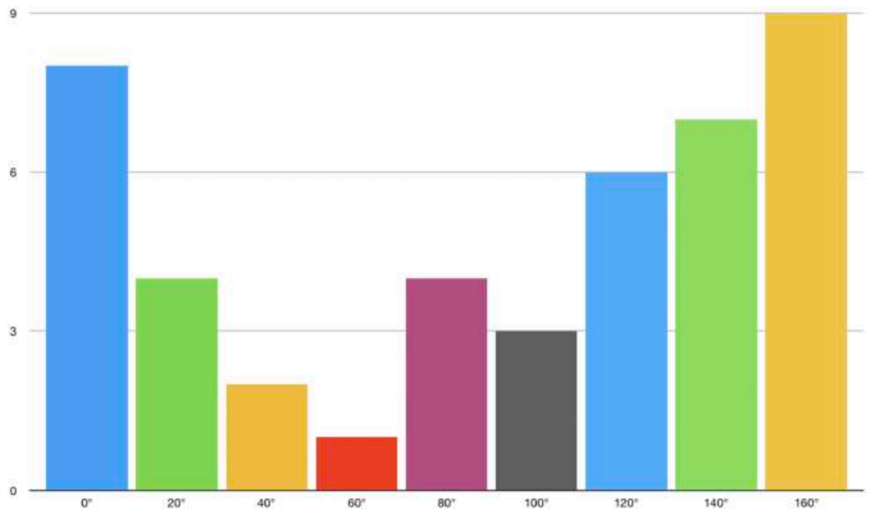


Figure 4.8 : HOG for each 8x8 cell

3) Block Normalization

Finally, a 16x16 block can be applied in order to normalize the image and make it invariant to lighting for example. This is simply achieved by dividing each value of the HOG of size 8x8 by the L2-norm of the HOG of the 16x16 block that contains it, which is in fact a simple vector of length 9*4 = 36.

Finally, all the 36x1 vectors are concatenated into a large vector. And we are done ! We have our feature vector, on which we can train a soft SVM classifier.

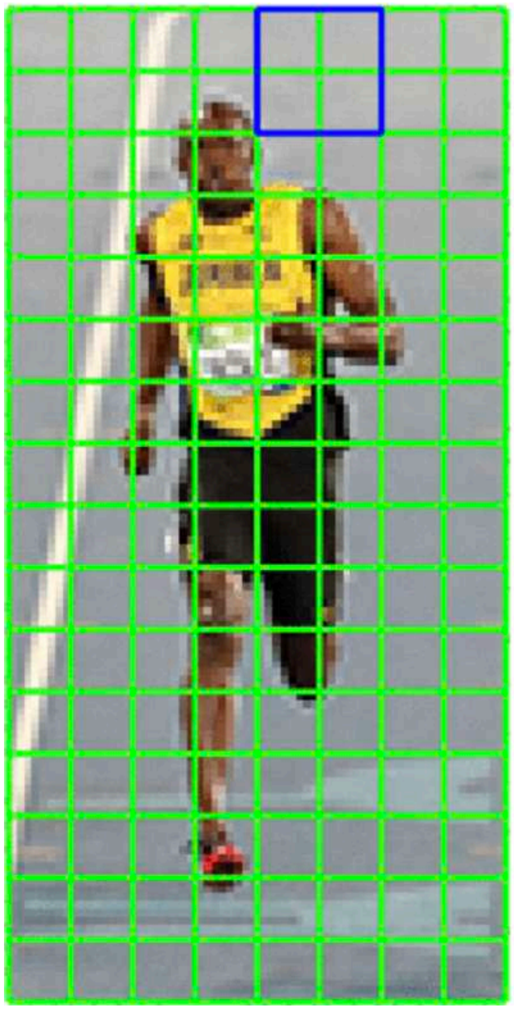


Figure 4.9 : Block Normalization

4.1.2.2. Pros

1. Works very well for frontal and slightly non-frontal faces.
2. Light-weight model as compared to CNN.
3. Works under small occlusion.
4. Basically, this method works under most cases except a few.

4.1.2.3. Cons

1. The major drawback is that it does not detect small faces as it is trained for minimum face size of 80×80. Thus, you need to make sure that the face size should be more than that in your application. You can however, train your own face detector for smaller sized faces.
2. The bounding box often excludes part of forehead and even part of chin sometimes.
3. Does not work very well under substantial occlusion.
4. Does not work for side face and extreme non-frontal faces, like looking down or up.
5. Really slow for real time detections.

4.1.3. Facial landmarks detection

The pre-trained facial landmark detector inside the dlib library in OpenCV is used to estimate the location of *68 (x, y)-coordinates* that map to facial structures on the face.

The indexes of the 68 coordinates can be visualized on the figure [4.10].

These annotations are part of the 68 point **iBUG 300-W dataset** which the dlib facial landmark predictor was trained on. [23]. Other flavors of facial landmark detectors exist, including the 194 point model that can be trained on the **HELEN dataset**.[24].

Regardless of which dataset is used, the same dlib framework can be leveraged to train a shape predictor on the input training data — this is useful if you would like to train facial landmark detectors or custom shape predictors of your own.

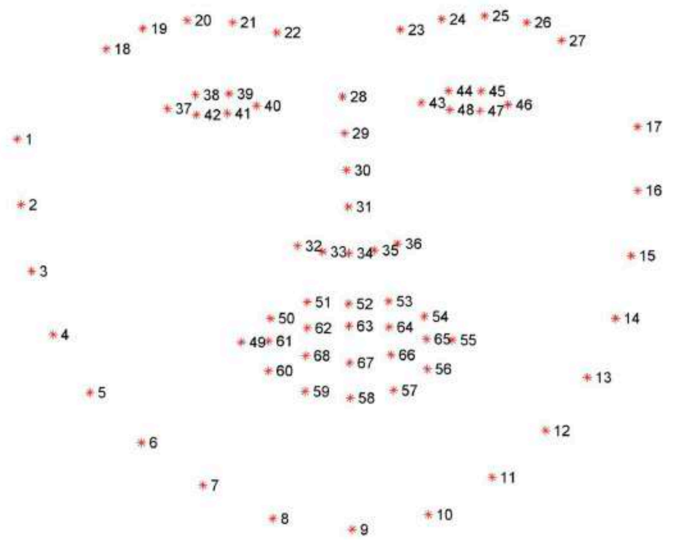


Figure 4.10: The pre-defined 68 co-ordinates of face defined in dlib library.

4.1.4. Eye aspect ratio (EAR) and mouth aspect ratio (MAR)

⇒ Eye aspect ratio (EAR)

From the landmarks detected in the image, we derive the eye aspect ratio (EAR) that is used as an estimate of the eye opening state. Since the per-frame EAR may not necessarily recognize the eye blinks correctly, a classifier that takes a larger temporal window of a frame into account is trained. For every video frame, the eye landmarks are detected. The eye aspect ratio (EAR) between height and width of the eye is computed.

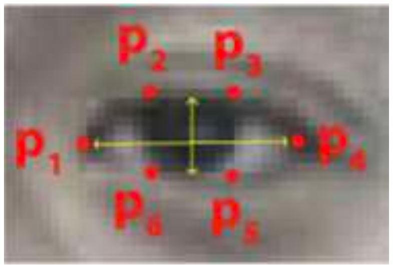


Figure 4.11 : Each eye is represented by 6(x, y)-coordinates facial landmark of eye region starting from left corner in clock wise direction.

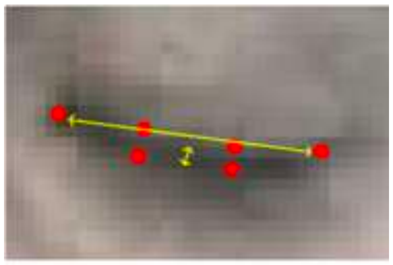


Figure 4.12 : Mouth is represented by 8(x, y)-coordinates facial landmark of mouth region starting from inner lip left corner.in clockwise direction.

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

Figure 4.13 : Eye Aspect Ratio (EAR) Equation

⇒ **Mouth aspect ratio (MAR)**

The horizontal and vertical distance of the mouth is calculated as :

MAR=(50-60)+(51-59)+(52-58)+(53-57)+(54-56)/(2*(49-55)).

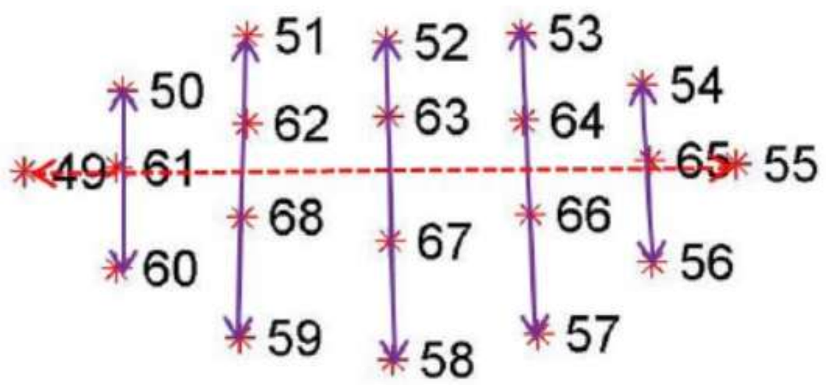
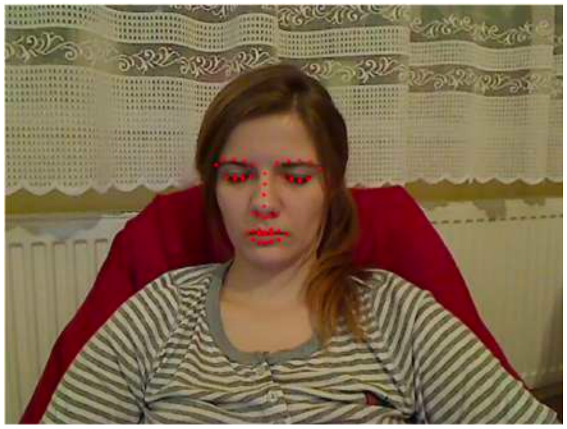


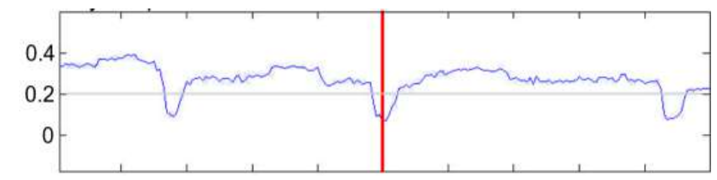
Figure 4.14 : Mouth Aspect Ratio (MAR)

4.1.5. EAR is beyond threshold

Example of detected blinks. The plots of the eye aspect ratio EAR results of the EAR thresholding (threshold set to 0.2). Input image with detected landmarks (depicted frame is marked by a red line).



EYE ASPECT RATION



EAR THRESHOLDING (T=0.2)

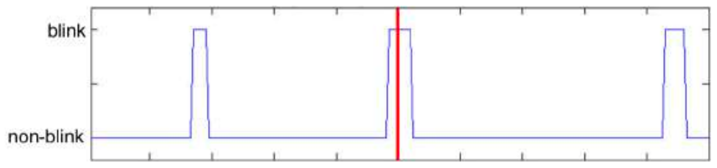
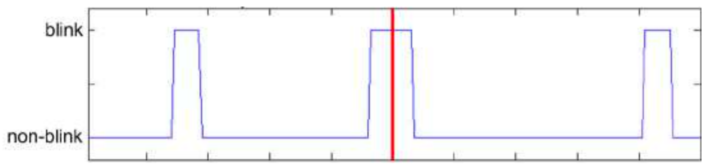
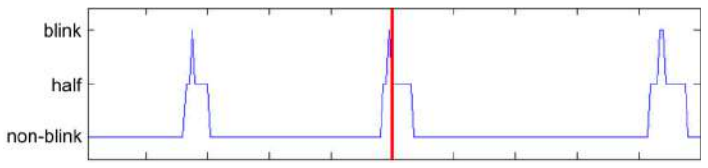


Figure 4.15 : EAR is beyond threshold example

EAR SVM OUTPUT



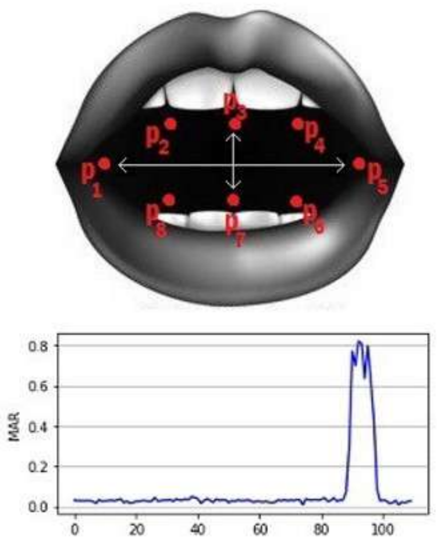
EAR SVM OUTPUT



4.1.6. MAR is beyond threshold

This is a similar mathematical approach as that of EAR, as you would expect, measures the ratio of the length of the mouth to the width of the mouth. We can calculate the length of the mouth by considering average distance between points p2, p8; p3, p7; p4, p6 and width by considering distance between points p1, p5. Our hypothesis was that as an individual becomes drowsy, they are likely to yawn and lose control over their mouth, making their MAR to be higher than usual in this state.

When the driver yawns three times in a short period of time(1000 frames, 41 seconds), the system will notice drowsiness and then trigger a drowsiness alarm.



$$\text{MAR} = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2\,\|p_1 - p_5\|}$$

Figure 4.16 : Mouth Aspect Ratio (MAR) Equation

4.1.7. One eye is detected for successive time

If the process of face detection failed to detect the two eyes and the complete mouth, this means that the driver is looking right, left or down. In this case, a different alarm will be triggered to make him focus while driving.

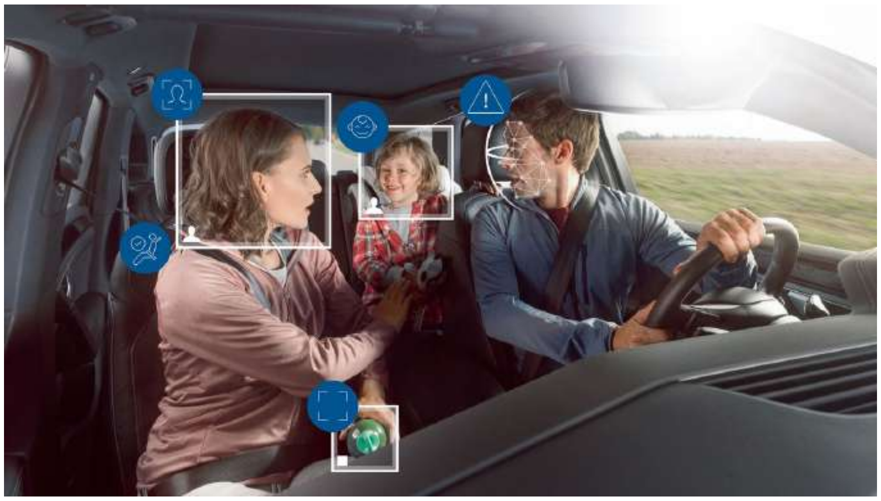


Figure 4.17 : One eye is detected for successive time

4.1.8. Suddenly frames got empty after a face was detected

If suddenly frames got empty after a face was detected, this means that the driver may have been distracted that he got off to pick up something from down or might be doing something behind. In this case, unfocus alarm is triggered to make the driver focus while driving.

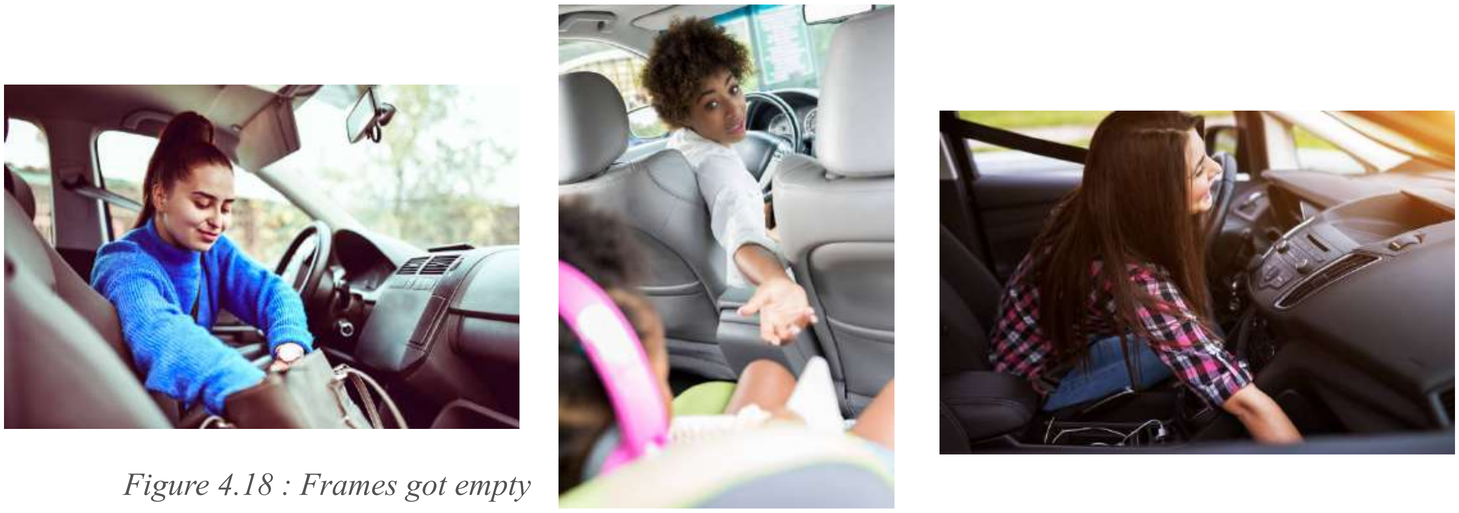


Figure 4.18 : Frames got empty

4.2. Hardware

4.2.1. Raspberry Pi

- Single-board computer; RAM: 2GB; Flash: 8GB; ARM A53 Quad-Core
- The Raspberry Pi is mounted with a Webcam Night Vision with a Buzzer, to capture the driver's condition, and mounting the system in the car, to alert the driver by providing a medium loud beeping sound.
- credit-card sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse.
- It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.
- Rasperri pi support programming languages as python, C++ and scratch.
- Rasperri pi suppor micro HDMI ports and USB ports.

Figure 4.19 : Raspberry Pi

4.2.2. Webcam Night Vision 300 GoldShip Mod.3526 - Leadership

- It has 6 LEDs for use in dark environments.
- Automatic light sensor that lights up the LEDs if the ambient lighting is not adequate.
- Electronic exposure control.

▸ Focus:

- 10 cm to infinity.
- Maximum Resolution: 5M Pixels.

▸ Frame Rate:

- Up to 30fps(CIF Mode).
- Up to 12fps (VGA Mode).

Figure 4.20 : Webcam Night Vision

4.2.3. HDMI Cable

Figure 4.21 : HDMI Cable

4.2.4. Buzzer

A buzzer, which is enclosed inside the Raspberry Pi casing, is connected to the GPIO of Raspberry Pi, to provide a medium loud beeping sound. This sound is used to alert the driver if he/she is in a drowsy condition or unfocused condition.

Figure 4.22 : Buzzer

4.2.5. USB Mouse

Input for Raspberry pi.

4.2.6. USB Keyboard

Input for Raspberry pi.

The whole unit is then mounted on the top of the dash facing the driver. Then, the device is being powered-up by plugging the micro USB power supply adapter into the cigarette lighter socket in the car, connecting the Raspberry Pi to the power supply.

CHAPTER 5

Implementation

Outlines:

5.1. User Interface

When opening the system by plugging the micro USB power supply adapter into the cigarette lighter socket in the car, connecting the Raspberry Pi to the power supply, the camera starts video streaming with continuously monitoring and writing the state of the eyes if they are either opened or closed. Also monitoring the current eye aspect ratio (EAR) and printing it on the screen.

The system can differentiate between the normal eye blinks and drowsiness by counting the time the eyes remain closed.

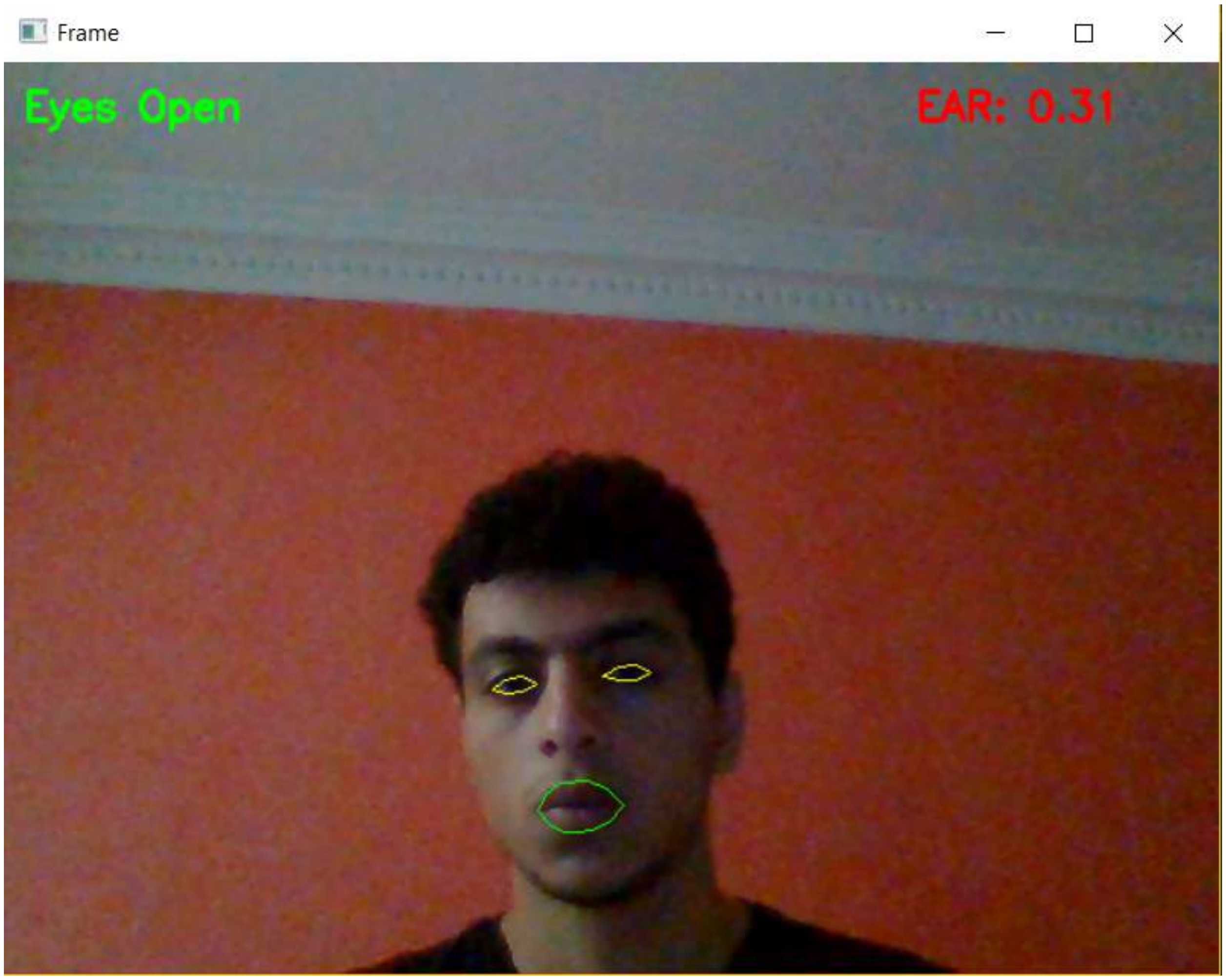


Figure 5.1 : Focusing and No Drowsiness

When the system detects that it's not a normal eye blink and the driver is drowsy and is going to fall asleep, by monitoring the eye aspect ratio (EAR) that will reach threshold in this case, while it's monitored and printed on the screen, the system will beep an alarm to alert the driver and will print on the screen "DROWSINESS ALERT!".

The alarm will not be turned off unless the driver opens his eyes, so that the eye aspect ratio returns bigger than the threshold.

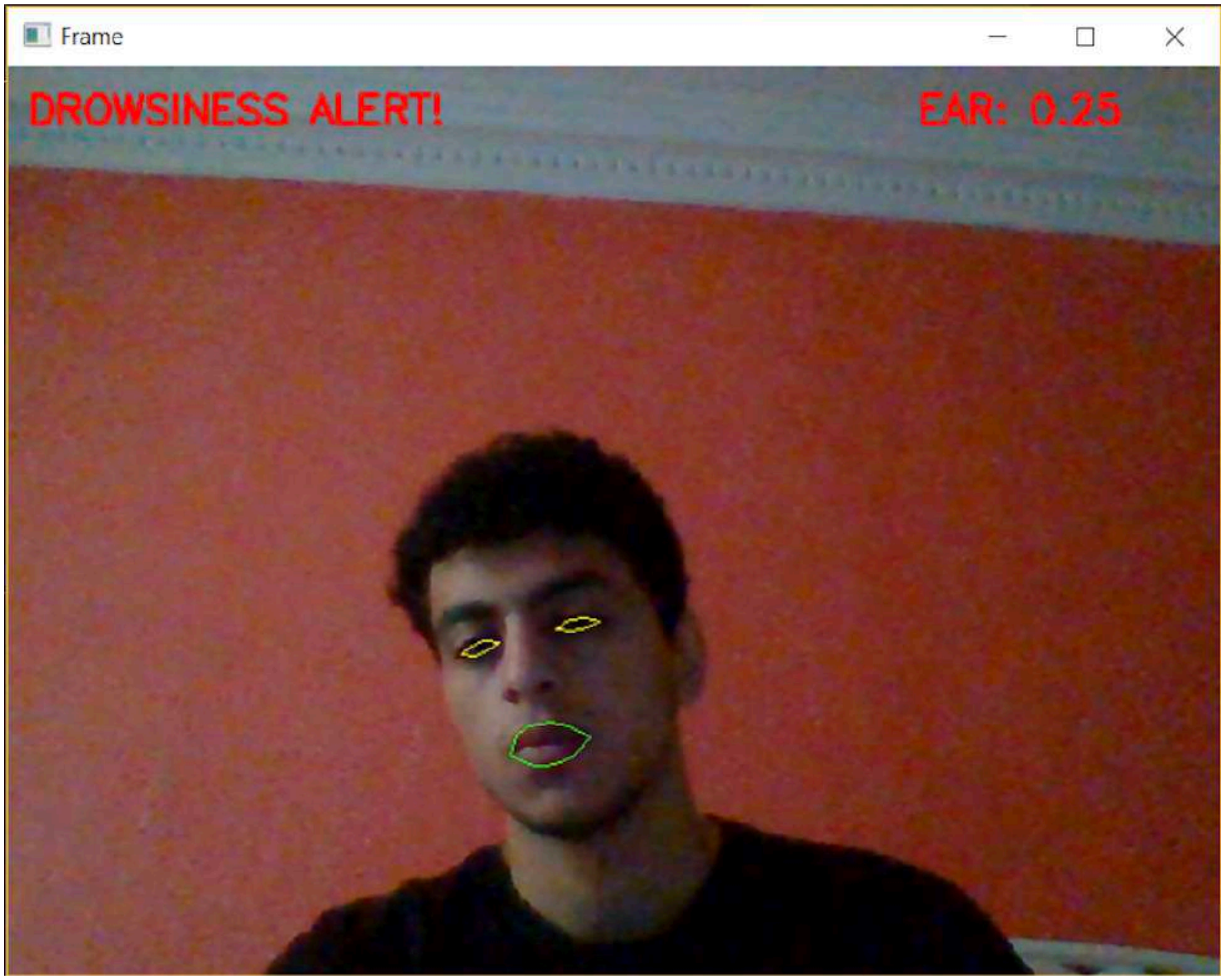


Figure 5.2 : Drowsiness detected (closing eyes)

Another way to detect drowsiness is when the driver starts yawning. Like the eye aspect ratio (EAR), the mouth aspect ratio (MAR) is calculated and MAR threshold is set monitoring the driver's state if he's either yawning or not and how many times he yawned. In case of yawning, the state of yawning is printed on the screen with the count of yawns. The state of eyes and EAR also will remain printed on the screen in all cases.

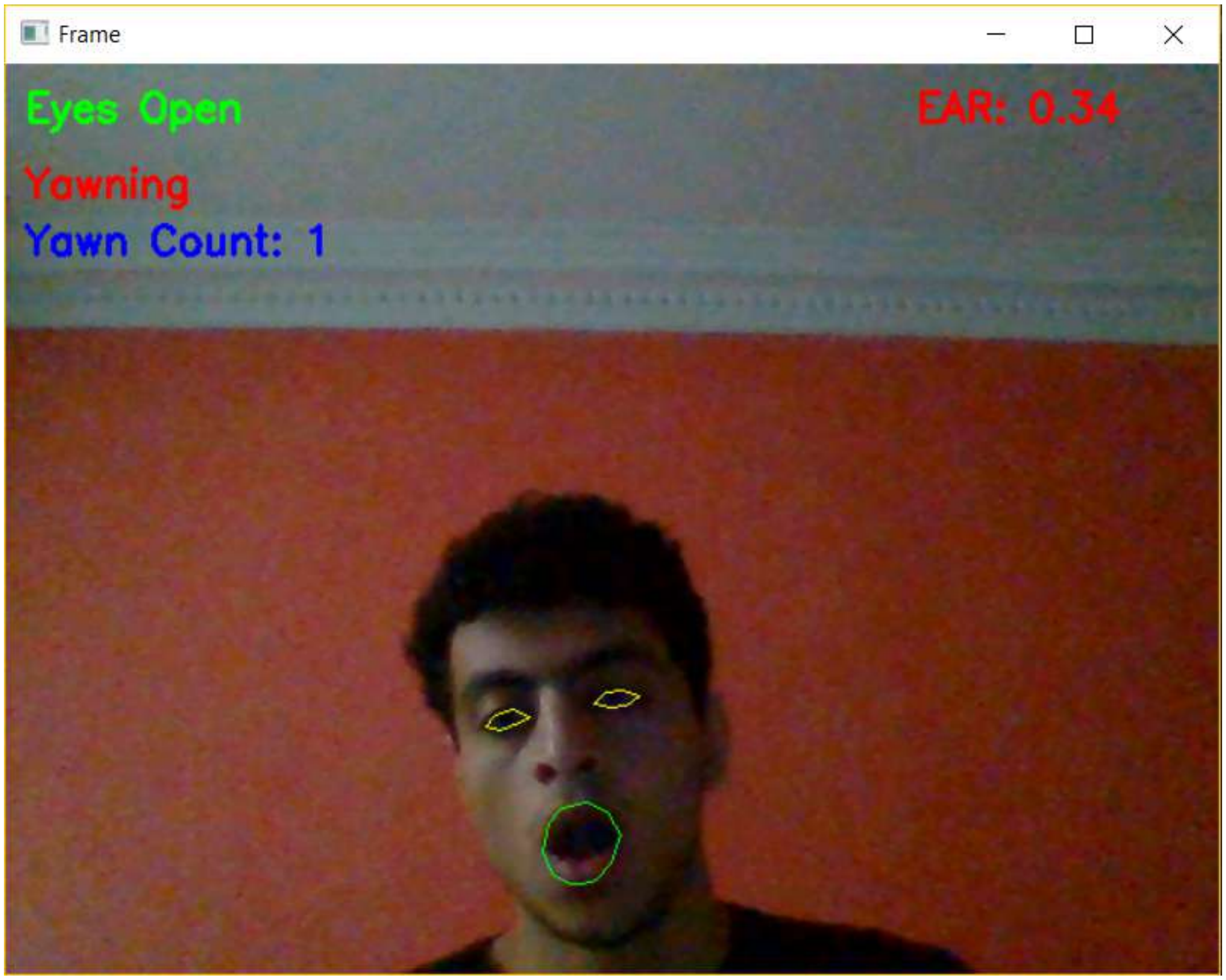


Figure 5.3 : First Yawn

The system will monitor the driver yawning in a specific period of time (1000 frames, 41 seconds). If the driver yawned for three successive times in 41 seconds, the system will consider this as a sign of drowsiness and starts to trigger a drowsiness alarm to alert the driver.

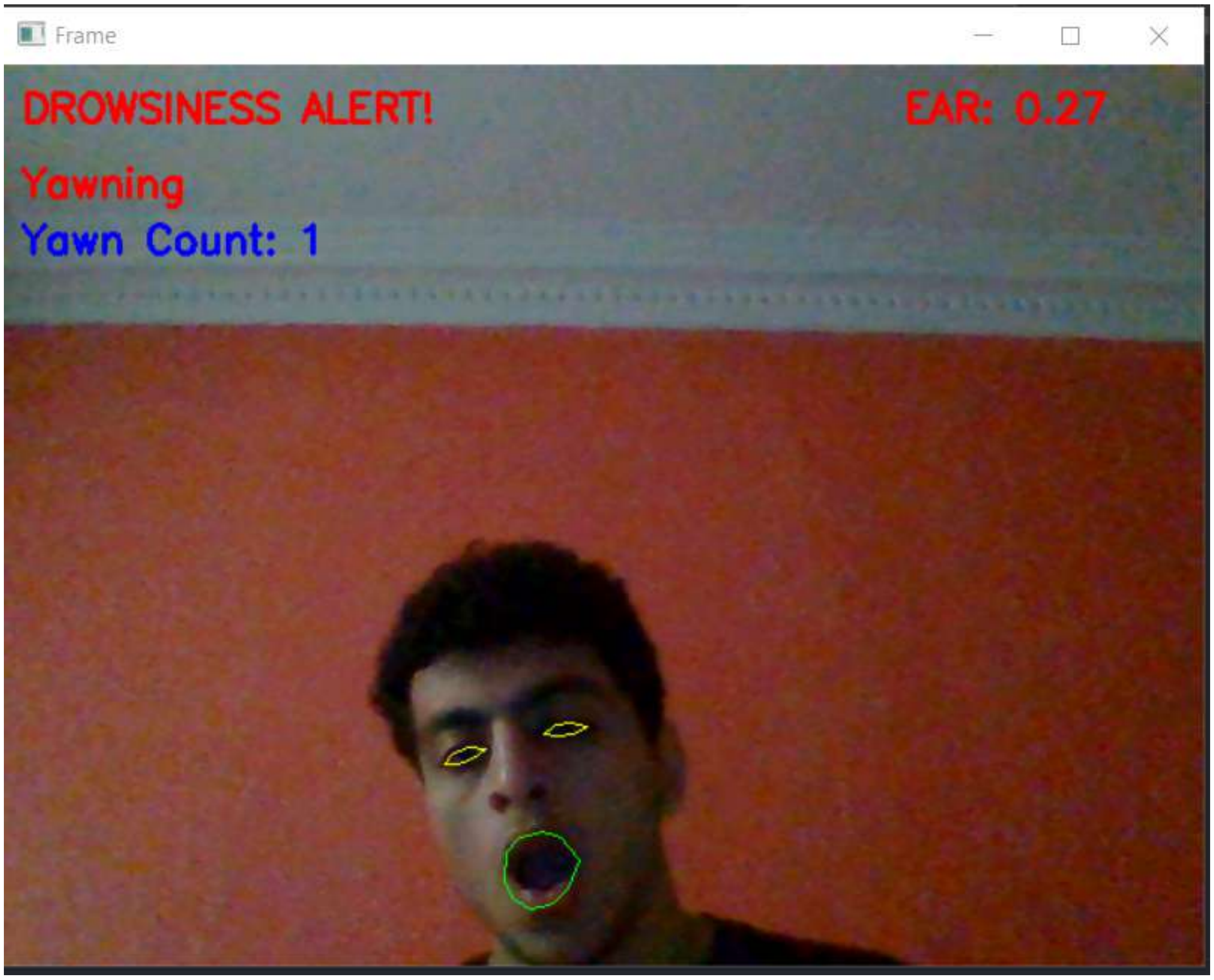


Figure 5.4 : Drowsiness detected (Yawning)

The yawns count is printed on the screen.

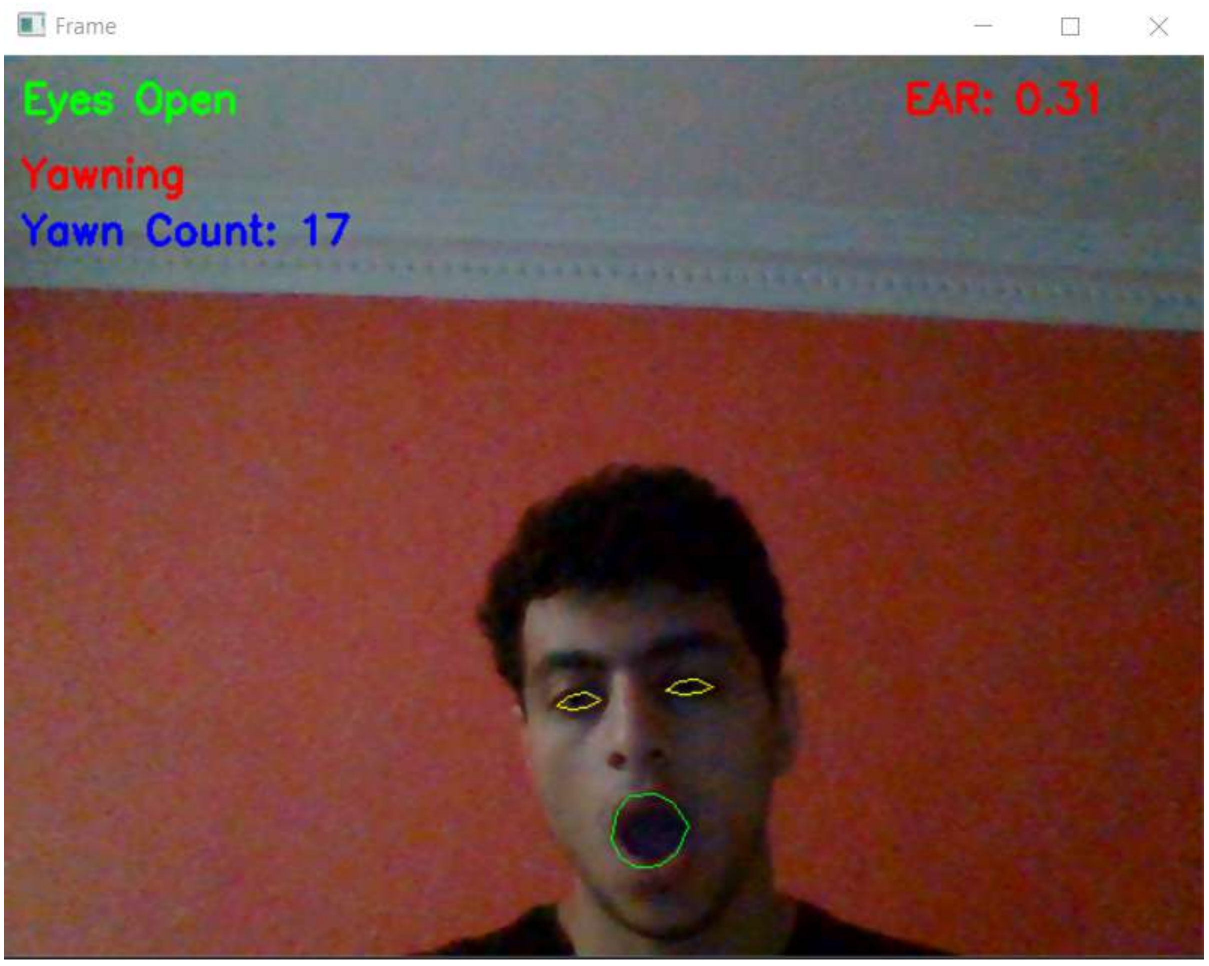


Figure 5.5 : Yawns Count

When the driver gets distracted by billboards, talking to someone next to him or picking up something form backward or down during driving and only the half of his face is detected, the system triggers an alarm to alert him.

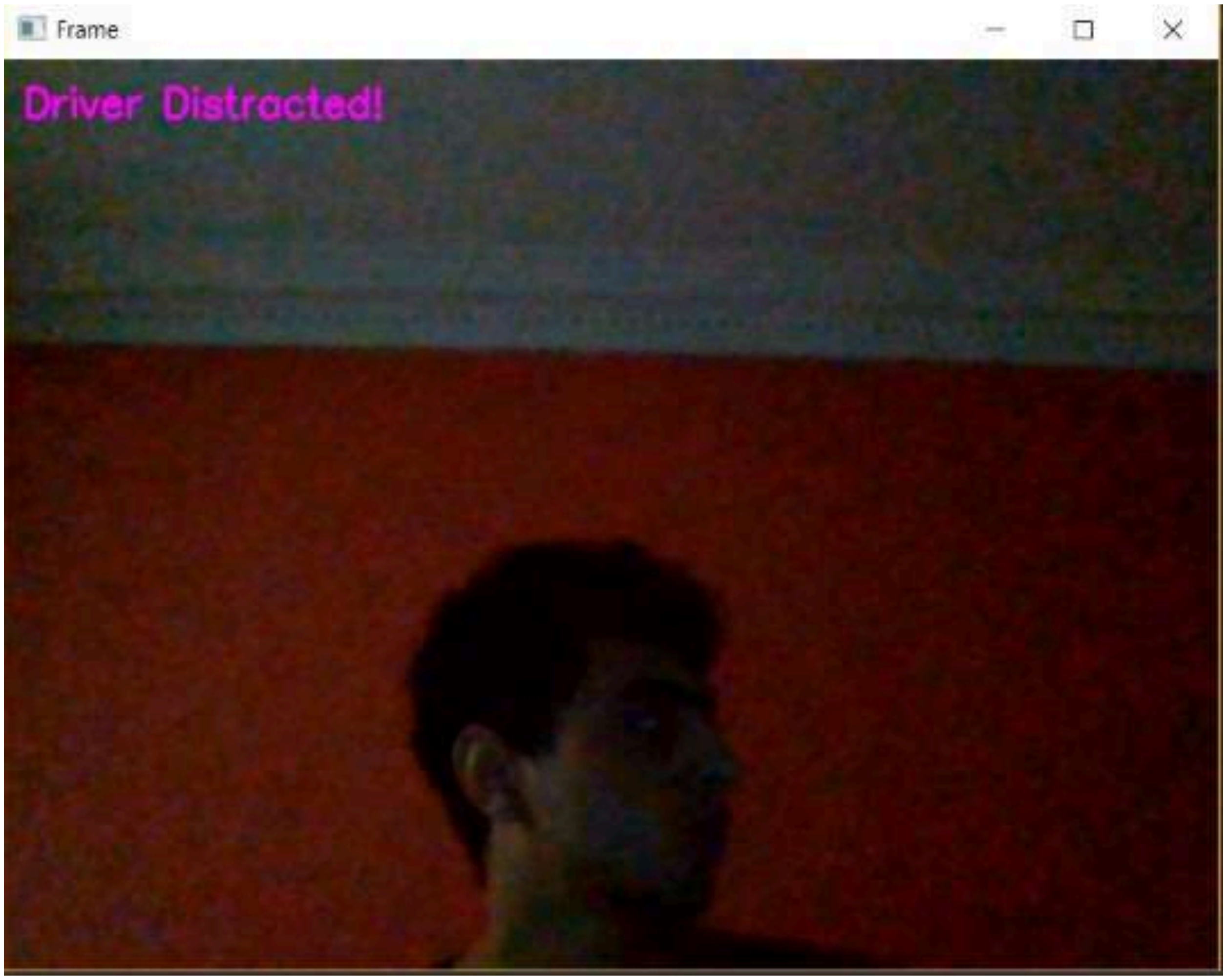


Figure 5.6 : Distraction detected (Half the face only is detected)

When the driver suddenly moves and his face couldn't be detected for some reason like talking to someone in behind or picking up something from down or behind, the system alerts him by triggering an alarm.

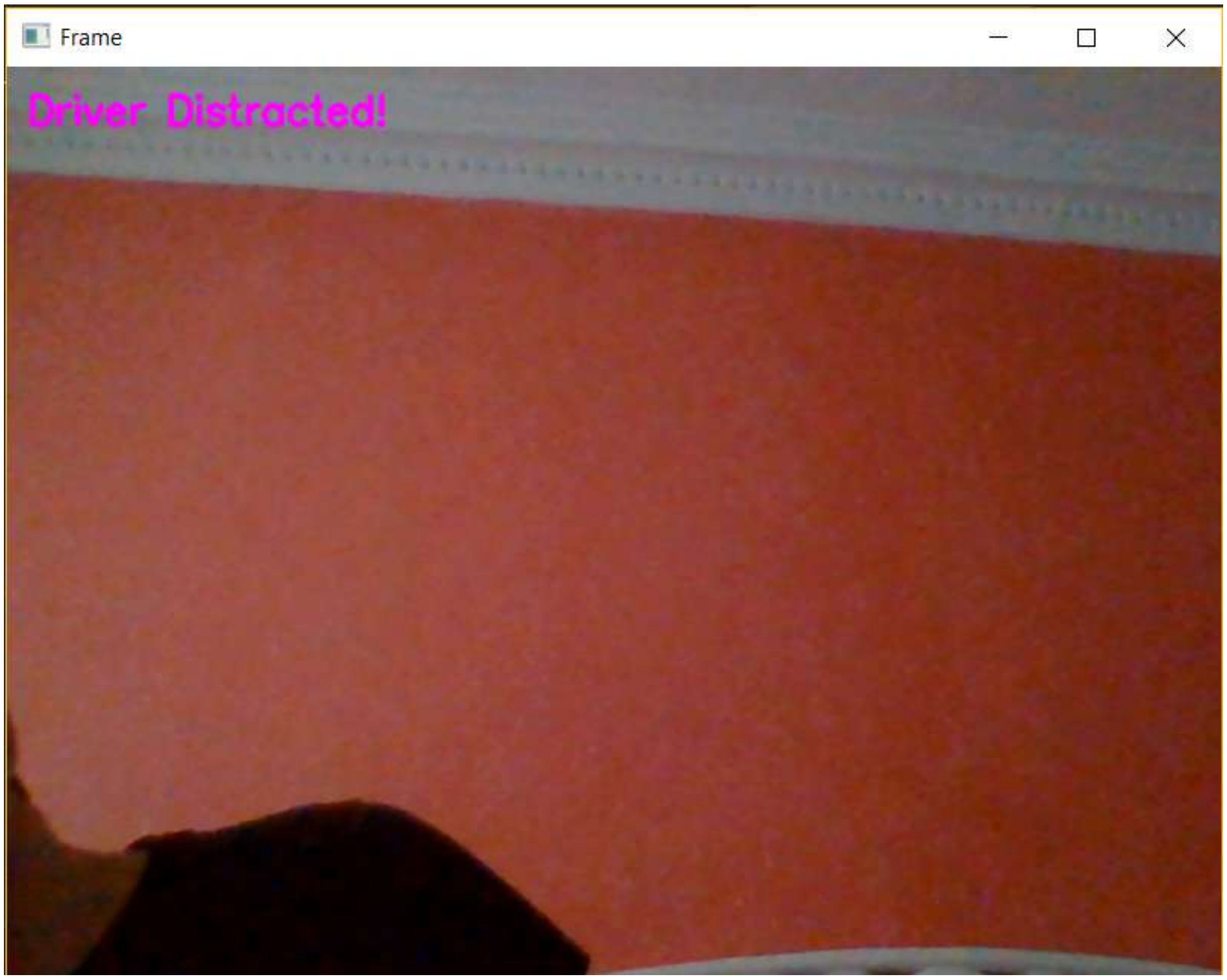
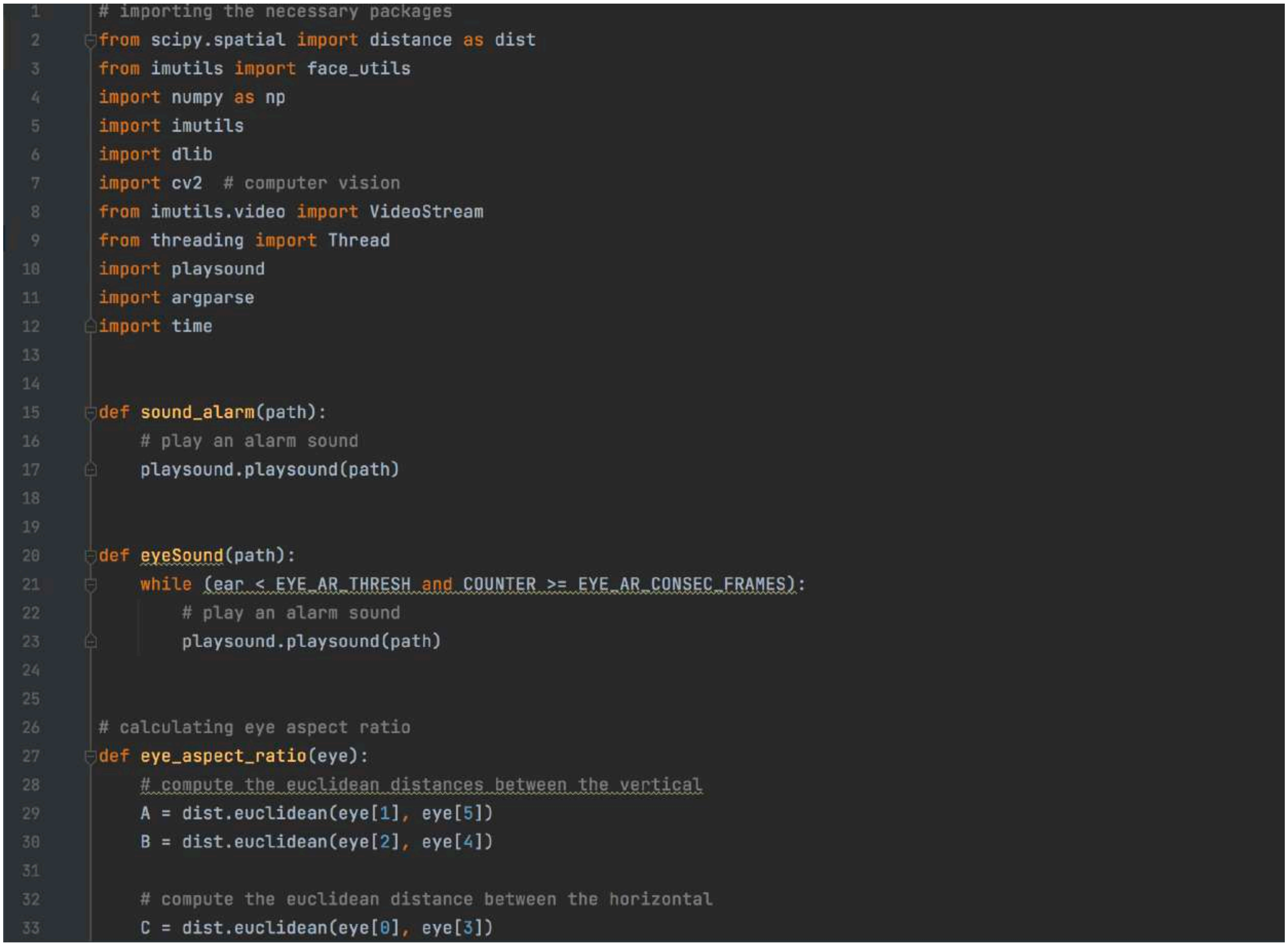
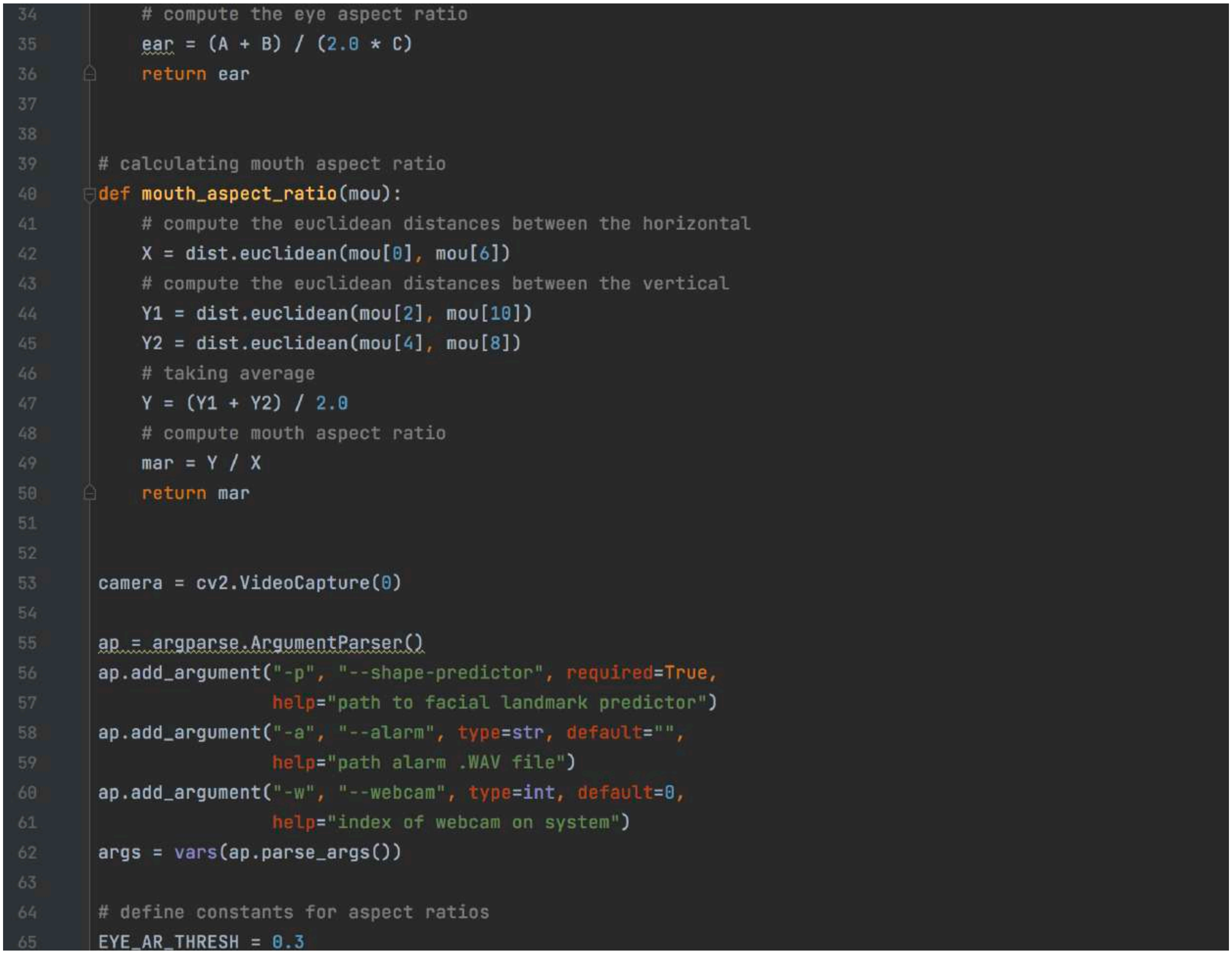


Figure 5.7 : Distraction detected (suddenly frames got empty after a face was detected)

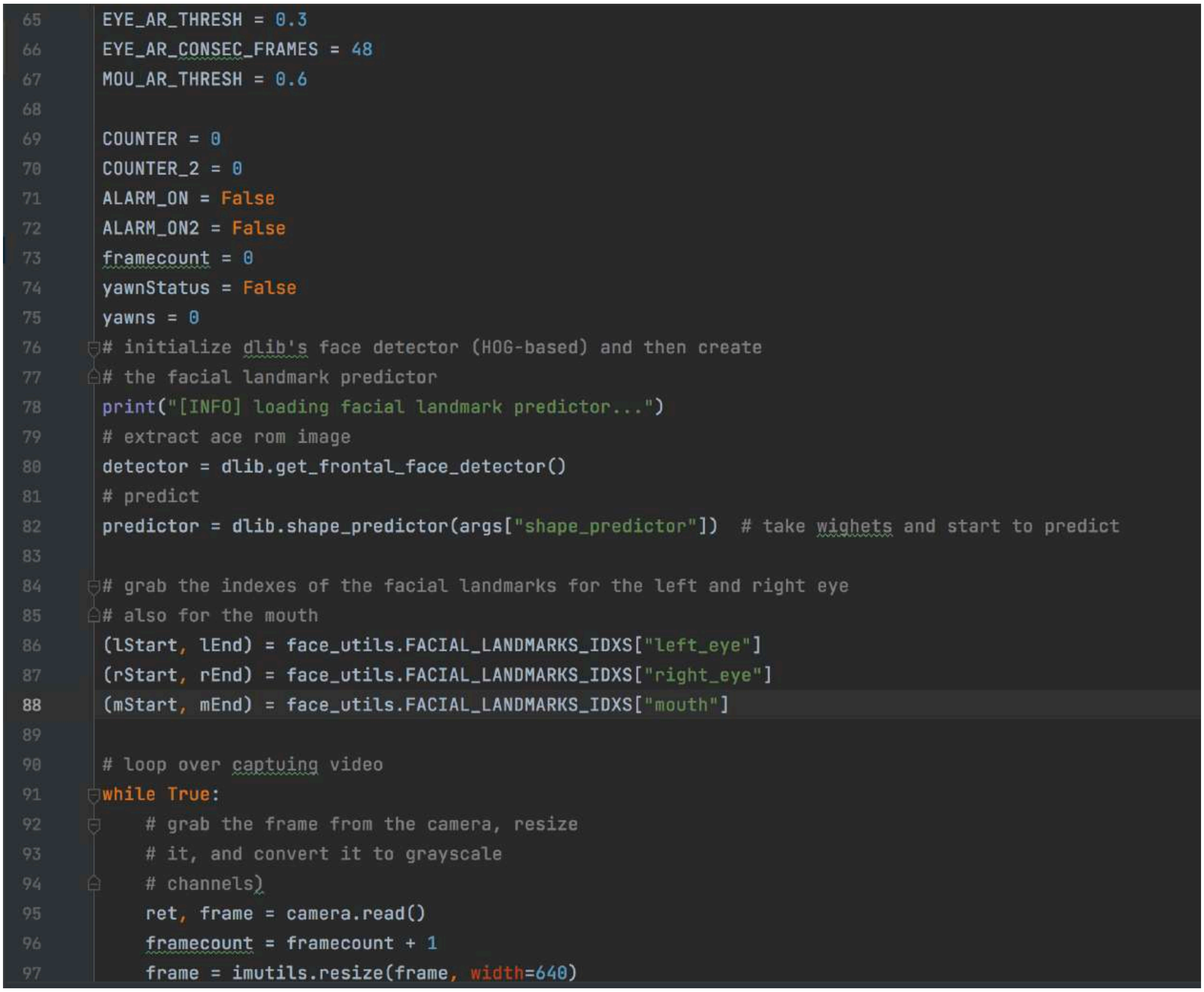
5.2. Screenshots of the code



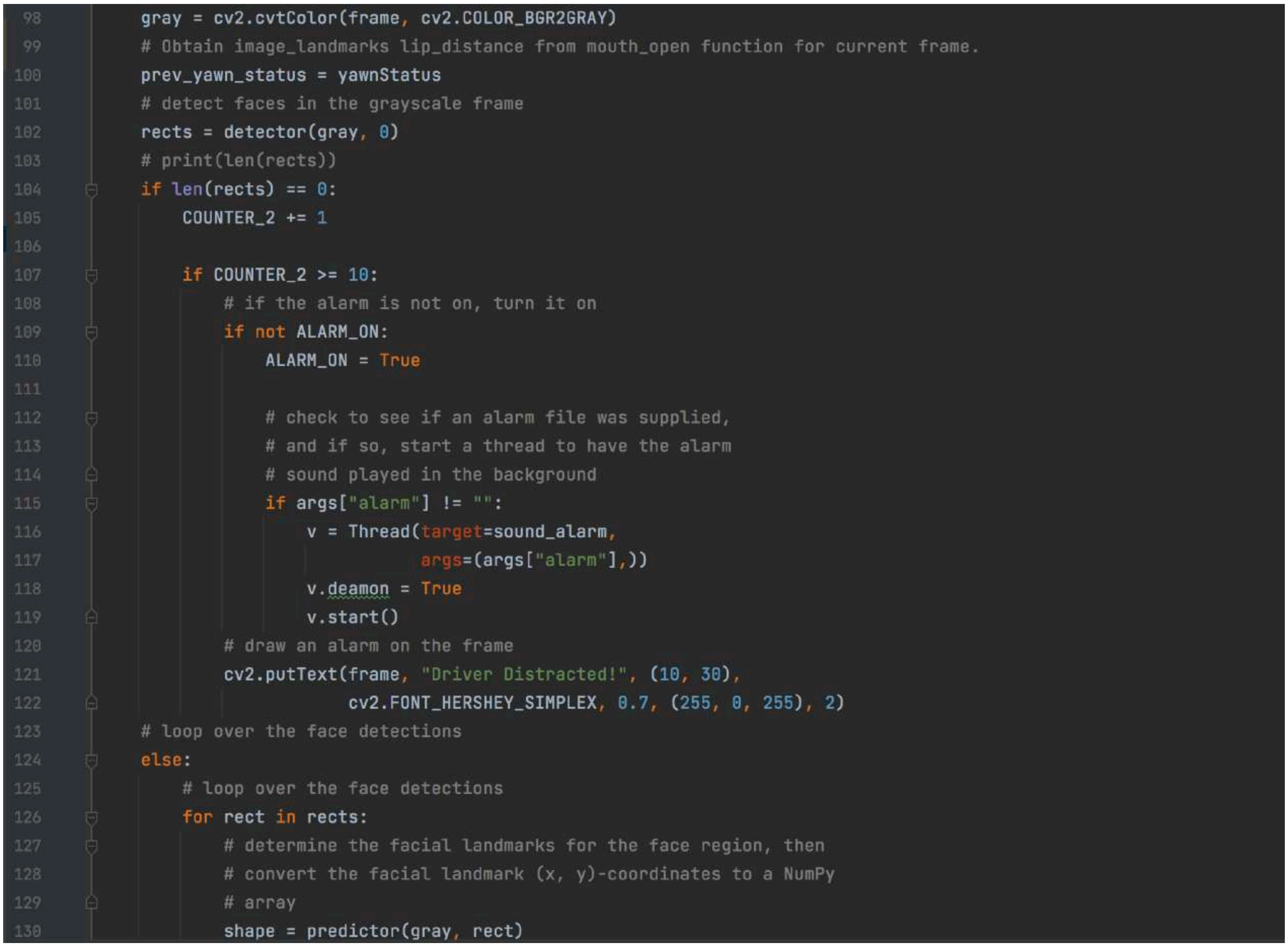
```python
# importing the necessary packages
from scipy.spatial import distance as dist
from imutils import face_utils
import numpy as np
import imutils
import dlib
import cv2  # computer vision
from imutils.video import VideoStream
from threading import Thread
import playsound
import argparse
import time


def sound_alarm(path):
    # play an alarm sound
    playsound.playsound(path)


def eyeSound(path):
    while (ear < EYE_AR_THRESH and COUNTER >= EYE_AR_CONSEC_FRAMES):
        # play an alarm sound
        playsound.playsound(path)


# calculating eye aspect ratio
def eye_aspect_ratio(eye):
    # compute the euclidean distances between the vertical
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])

    # compute the euclidean distance between the horizontal
    C = dist.euclidean(eye[0], eye[3])
```



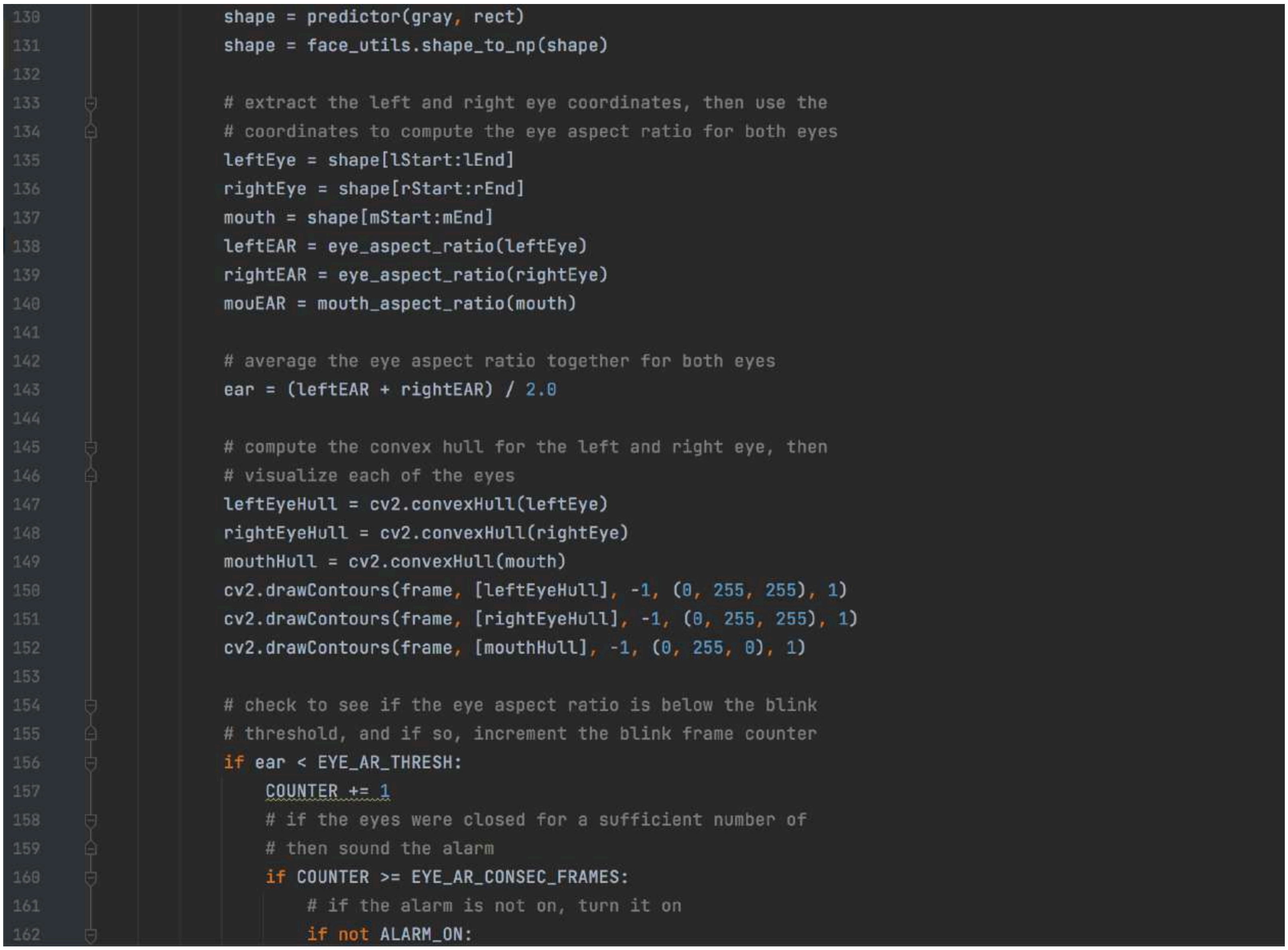
```python
    # compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)
    return ear


# calculating mouth aspect ratio
def mouth_aspect_ratio(mou):
    # compute the euclidean distances between the horizontal
    X = dist.euclidean(mou[0], mou[6])
    # compute the euclidean distances between the vertical
    Y1 = dist.euclidean(mou[2], mou[10])
    Y2 = dist.euclidean(mou[4], mou[8])
    # taking average
    Y = (Y1 + Y2) / 2.0
    # compute mouth aspect ratio
    mar = Y / X
    return mar


camera = cv2.VideoCapture(0)

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
                help="path to facial landmark predictor")
ap.add_argument("-a", "--alarm", type=str, default="",
                help="path alarm .WAV file")
ap.add_argument("-w", "--webcam", type=int, default=0,
                help="index of webcam on system")
args = vars(ap.parse_args())

# define constants for aspect ratios
EYE_AR_THRESH = 0.3
```



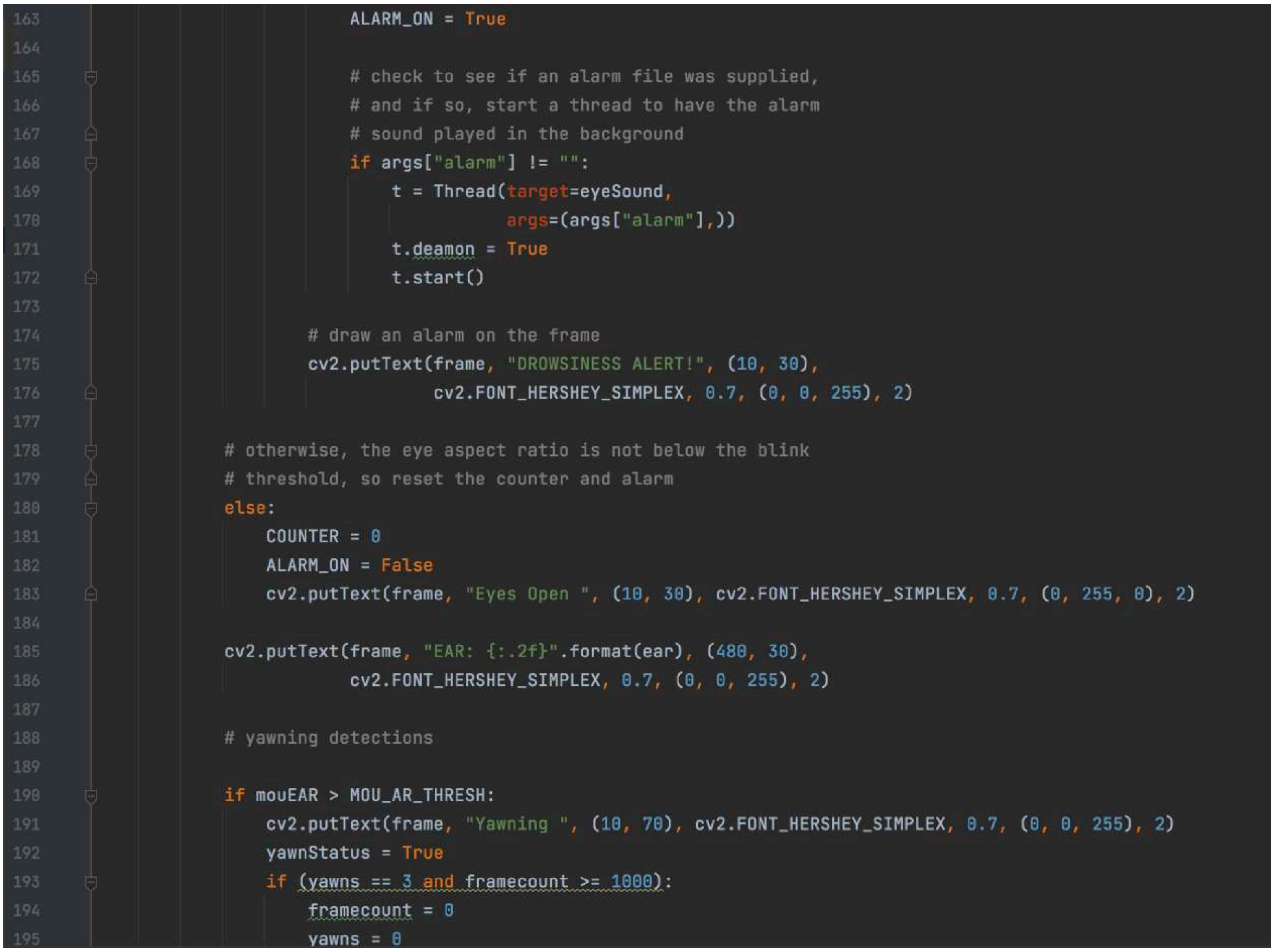
```
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 48
MOU_AR_THRESH = 0.6

COUNTER = 0
COUNTER_2 = 0
ALARM_ON = False
ALARM_ON2 = False
framecount = 0
yawnStatus = False
yawns = 0
# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")
# extract ace rom image
detector = dlib.get_frontal_face_detector()
# predict
predictor = dlib.shape_predictor(args["shape_predictor"])  # take wighets and start to predict

# grab the indexes of the facial landmarks for the left and right eye
# also for the mouth
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]

# loop over captuing video
while True:
    # grab the frame from the camera, resize
    # it, and convert it to grayscale
    # channels)
    ret, frame = camera.read()
    framecount = framecount + 1
    frame = imutils.resize(frame, width=640)
```



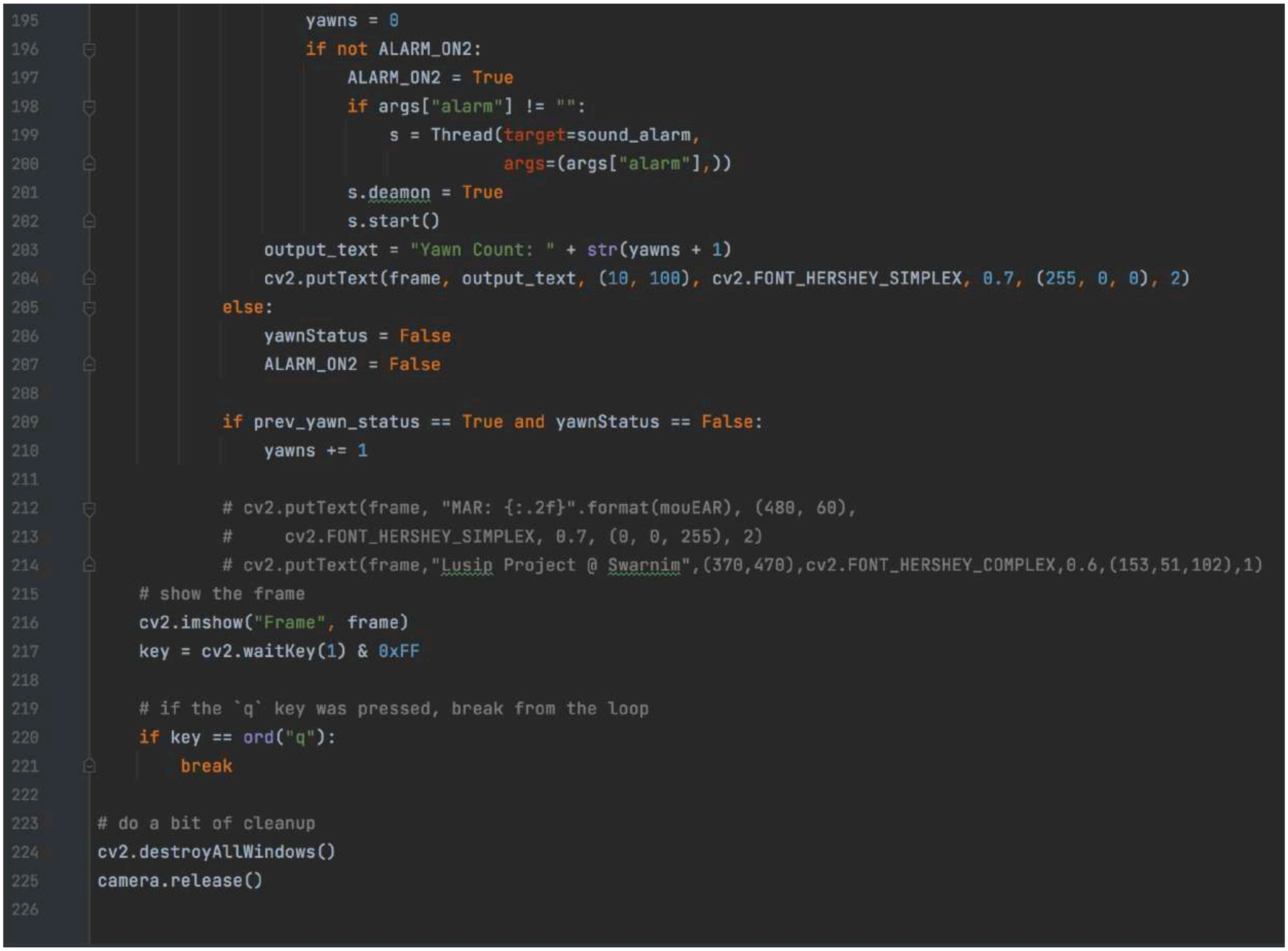
```
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Obtain image_landmarks lip_distance from mouth_open function for current frame.
        prev_yawn_status = yawnStatus
        # detect faces in the grayscale frame
        rects = detector(gray, 0)
        # print(len(rects))
        if len(rects) == 0:
            COUNTER_2 += 1

            if COUNTER_2 >= 10:
                # if the alarm is not on, turn it on
                if not ALARM_ON:
                    ALARM_ON = True

                    # check to see if an alarm file was supplied,
                    # and if so, start a thread to have the alarm
                    # sound played in the background
                    if args["alarm"] != "":
                        v = Thread(target=sound_alarm,
                                   args=(args["alarm"],))
                        v.deamon = True
                        v.start()
                # draw an alarm on the frame
                cv2.putText(frame, "Driver Distracted!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 255), 2)
        # loop over the face detections
        else:
            # loop over the face detections
            for rect in rects:
                # determine the facial landmarks for the face region, then
                # convert the facial landmark (x, y)-coordinates to a NumPy
                # array
                shape = predictor(gray, rect)
```



```
            shape = predictor(gray, rect)
            shape = face_utils.shape_to_np(shape)

            # extract the left and right eye coordinates, then use the
            # coordinates to compute the eye aspect ratio for both eyes
            leftEye = shape[lStart:lEnd]
            rightEye = shape[rStart:rEnd]
            mouth = shape[mStart:mEnd]
            leftEAR = eye_aspect_ratio(leftEye)
            rightEAR = eye_aspect_ratio(rightEye)
            mouEAR = mouth_aspect_ratio(mouth)

            # average the eye aspect ratio together for both eyes
            ear = (leftEAR + rightEAR) / 2.0

            # compute the convex hull for the left and right eye, then
            # visualize each of the eyes
            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull = cv2.convexHull(rightEye)
            mouthHull = cv2.convexHull(mouth)
            cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 255), 1)
            cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 255), 1)
            cv2.drawContours(frame, [mouthHull], -1, (0, 255, 0), 1)

            # check to see if the eye aspect ratio is below the blink
            # threshold, and if so, increment the blink frame counter
            if ear < EYE_AR_THRESH:
                COUNTER += 1
                # if the eyes were closed for a sufficient number of
                # then sound the alarm
                if COUNTER >= EYE_AR_CONSEC_FRAMES:
                    # if the alarm is not on, turn it on
                    if not ALARM_ON:
```



```
                    ALARM_ON = True

                    # check to see if an alarm file was supplied,
                    # and if so, start a thread to have the alarm
                    # sound played in the background
                    if args["alarm"] != "":
                        t = Thread(target=eyeSound,
                                   args=(args["alarm"],))
                        t.deamon = True
                        t.start()

                # draw an alarm on the frame
                cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        # otherwise, the eye aspect ratio is not below the blink
        # threshold, so reset the counter and alarm
        else:
            COUNTER = 0
            ALARM_ON = False
            cv2.putText(frame, "Eyes Open ", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)

        cv2.putText(frame, "EAR: {:.2f}".format(ear), (480, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        # yawning detections

        if mouEAR > MOU_AR_THRESH:
            cv2.putText(frame, "Yawning ", (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            yawnStatus = True
            if (yawns == 3 and framecount >= 1000):
                framecount = 0
                yawns = 0
```



```python
                    yawns = 0
                    if not ALARM_ON2:
                        ALARM_ON2 = True
                        if args["alarm"] != "":
                            s = Thread(target=sound_alarm,
                                       args=(args["alarm"],))
                        s.deamon = True
                        s.start()
                output_text = "Yawn Count: " + str(yawns + 1)
                cv2.putText(frame, output_text, (10, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)
            else:
                yawnStatus = False
                ALARM_ON2 = False

            if prev_yawn_status == True and yawnStatus == False:
                yawns += 1

            # cv2.putText(frame, "MAR: {:.2f}".format(mouEAR), (480, 60),
            #     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            # cv2.putText(frame,"Lusip Project @ Swarnim",(370,470),cv2.FONT_HERSHEY_COMPLEX,0.6,(153,51,102),1)
    # show the frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

# do a bit of cleanup
cv2.destroyAllWindows()
camera.release()
```

CHAPTER 6

Conclusion and Future Work

Outlines:

6.1. Conclusion

The driver distraction detection system developed is capable of detecting **drowsiness** and **distraction** in a rapid manner. The system which can differentiate normal eye blink and drowsiness which can prevent the driver from entering the state of sleepiness while driving. During the monitoring, the system is able to decide if the eyes are opened or closed, the driver is yawning many times in a short period of time and the lake of focusing, such as when the driver gets distracted by billboards or picking up something form backward or down during driving. When the eyes have been closed for about two seconds, and the driver is yawning for more than 3 times in 41 seconds, and when he's not completely focused in driving, alarm beeps to alert the driver. By doing this, many accidents will be reduced and provides safe life to the driver and vehicle safety. A system for driver safety is presented only in the luxurious costly cars. Using drowsiness detection system, driver safety can be implemented in normal cars also.

6.2. Future work

we can enhance the distraction detection system by using accelerometer to detect whether the car engine is running and some sensors to know if the driver started to drive or not to start monitoring his behavior.

6.2.1. Utilization of outer factors

The future works may focus on the utilization of outer factors such as vehicle states, sleeping hours, weather conditions, mechanical data, etc, for fatigue measurement. Driver drowsiness pose a major threat to highway safety, and the problem is particularly severe for commercial motor vehicle operators. Twenty-four hour operations, high annual mileage, exposure to challenging environmental conditions, and demanding work schedules all contribute to this serious safety issue. Monitoring the driver’s state of drowsiness and vigilance and providing feedback on their condition so that they can take appropriate action is one crucial step in a series of preventive measures necessary to address this problem. Currently there is not adjustment in zoom or direction of the camera during operation. Future work may be to automatically zoom in on the eyes once they are localized.

6.2.2. Car insurance mobile application

In the future, this project can be used as a car insurance mobile application, used to monitor the driver’s behavior before any accident. This is the same idea of **black box**. A flight recorder is an electronic recording device placed in an aircraft for the purpose of facilitating the investigation of aviation accidents and incidents. This can be useful for car insurance companies. This idea may be obligatory in all car insurance companies soon. This project can be implemented in the form of mobile application to reduce the cost of hardware.

6.2.3. Automatic speed control

The drowsiness detector is the initiator of the process. It will sense the condition of the driver and send the feedback to the micro-controller. If the eyelids of the driver are remained closed for a prolonged time than the usual the drowsiness detector will sense this and will pass on the information to the micro-controller. The micro-controller will initiate the working of the other components. The emergency lights will be switched as a signal to the other drivers who are driving; the servomotor which is fitted at the brake pedal will increase its angle gradually at regular intervals, this action will promise the gradual deceleration of the vehicle at a slower pace so that the driver will not be awaken immediately which will disrupt the normal working of the system. The vibrator motor is turned on to awaken the driver form his sleep after the vehicle has reached a complete halt position. The driver is awaken comfortably and the end of the process and he can resume his driving. The process is stopped once the sleep detector senses the eyelids of the driver is opened.

6.2.4. IoT-Based smart alert system for drowsy driver detection

Here, a Pi camera is used to regularly record the total movement of an eye through which the threshold value of an EAR is calculated. A counter is also included in it for counting occurrence of frames. Suppose that it exceeded above a range of 30. In that case, a voice is activated by a speaker and a mail is automatically sent to an authorized person of the vehicle which is generally processed at the time of drowsiness detection. The mail is received by an authorized one, who can alert the sleepy driver by ringing to him, if that drive is still not awake after turning on the voice alert message in the speaker. Thus, this process is successfully performed to detect a driver's drowsiness and detect the collision impacts due to braking of the vehicle through crash sensor and Force Sensitive Sensor (FSS). Due to the occurrence of the collision, the data collected from sensors and the location data are sent to an authorized person (owner) or any near hospitals with the help of a GPS module (GoogleMaps link).

6.2.5. Emotional recognition using facial expression

In the future, the project be updated and read the facial expression of the driver and monitor his emotionst then analyze it and take action. If the facial expression is sad the car play a motivating song. If tired a caution message appear to the driver that he needs a break and a cup of coffee. If the driver is in a happy mood the car plays happy songs. If he is angry the car play relaxing sounds if the nature

Facial expression recognition (FER) algorithm will be used in the project and excessive amount of memory would be required

References

[1] Clumbs Sleep Consultants, https://columbussleep.com/sleep-driving#:~:text=The%20National%20Highway%20Traffic%20Safety,%2412.5%20billion%20in%20monetary%20losses.

[2] Hartman, K. and J. Strasser, Saving Lives Through Advanced Vehicle Safety Technology:Intelligent Vehicle Initiative Final Report. 2005, Department of Transportation: Washington, DC. p. 12.

[3] A., E., M. A., and S. R., Drowsy and Fatigued Driving Problem Significance and Detection Based on Driver Control Functions. Handbook of Intelligent Vehicles: SpringerReference, ed. A. Eskandarian. 2012, Berlin, Germany: Springer-Verlag Berlin Heifelberg.

[4] Drivers Beware: Getting Enough Sleep Can Save Your Life this Memorial Day. 2010, National Sleep Foundation: Arlington, VA.

[5] Husar, P., Eyetracker warns against momentary driver drowsiness. 2010, Fraunhofer-Gesellschaft: Munich, Germany.

[6] World Health Organization Global status report on road safety 2018

[7] Y. Chellappa, N. N. Joshi, V. Bharadwaj, Driver fatigue detection system, in: Signal and Image Processing (ICSIP), IEEE International Conference on, IEEE, 2016, pp. 655–660.

[8] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005.IEEE Computer Society Conference on, Vol. 1, IEEE, 2005, pp. 886–893.

[9] M. Ngxande, J.-R. Tapamo, M. Burke, Driver drowsiness detection using behavioral measures and machine learning techniques: A review of stateof-art techniques, in: Pattern Recognition Association of South Africa and

Robotics and Mechatronics (PRASA-RobMech), 2017, IEEE, 2017,pp. 156–161.

[10] P. Wang, L. Shen, A method of detecting driver drowsiness state basedon multi-features of face, in: Image and Signal Processing (CISP), 2012 5th International Congress on, IEEE, 2012, pp. 1171–1175.

[11] L. Zhao, Z. Wang, X. Wang, Q. Liu, Driver drowsiness detection using facial dynamic fusion information and a dbn, IET Intelligent Transport Systems.

[12] Q. Ji, Z. Zhu, P. Lan, Real-time nonintrusive monitoring and predictionof driver fatigue, IEEE transactions on vehicular technology 53 (4)(2004) 1052–1068.

[13] M. A. Assari, M. Rahmati, Driver drowsiness detection using face expression recognition, in: Signal and Image Processing Applications(ICSIPA), 2011 IEEE International Conference on, IEEE, 2011,pp. 337–341.

[14] Dixon, C. Unobtrusive eyelid closure and visual of regard measurement system. In Proceedings of theConference on Ocular Measures of Driver Alertness, Washington, DC, USA, 26–27 April 1999.

[15] Lorente, M.P.S.; Lopez, E.M.; Florez, L.A.; Espino, A.L.; Martínez, J.A.I.; de Mi-guel, A.S. Explaining Deep Learning-Based Driver Models. *Appl. Sci.* **2021**, *11*, 3321. [CrossRef]

[16] Sipele, O.; Zamora, V.; Ledezma, A.; Sanchis, A.c. Advanced Driver's Alarms System through Multi-agent Paradigm. In Proceedings of the 2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE), Singapore, 3–5 September 2018; pp. 269–275. [CrossRef]

[17] Schiffman, H.R. *Sensation and Perception: An Integrated Approach*, 3rd ed.; John Wiley & Sons: Oxford, UK, 1990.

[18] King, D.E. Dlib-ml: A Machine Learning Toolkit. *J. Mach. Learn. Res.* **2009**, *10*, 1755–1758.

[19] Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 886–893. [CrossRef]

[20] Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.

[21] Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.

[22] Kazemi, V.; Sullivan, J. One millisecond face alignment with an ensemble of regression trees. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1867–1874. [CrossRef]

[23] Ishii, T.; Hirose, M.; Iwata, H. Automatic recognition of driver's facial expression by image analysis. J. Soc.Automot. Eng. Jpn. 1987,41, 1398–1403.

[24] iBUG 300-W dataset — Facial point annotations — https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

[25] Helen dataset — http://www.ifp.illinois.edu/~vuongle2/helen/

[26] Navneet Dalal and Bill Triggs — INRIA Rhone-Alps, ˆ 655 avenue de l'Europe, Montbonnot 38334, France, https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

[27] LFW dataset — Davis King, http://vis-www.cs.umass.edu/lfw/