DRIVER FATIGUE DETECTION

A Online Summer Internship project report submitted to the ***JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD*** *in partial fulfilment of the requirements for the award of the degree of*

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

A. Akshitha (18071A0502)

Ch. Sankranthi (18071A0510)

K. Sahithi Latha (18071A0528)

S. Shanmukha Anurag (18071A0551)

Under the Guidance Of

Mrs. Vasundhara

(Assistant Professor, VNR VJIET)

VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTE, NAAC ACCREDITED WITH 'A++' GRADE, NBA ACCREDITED, APPROVED BY AICTE, NEW DELHI, AFFILIATED TO JNTUH)

August 2020

VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute, NAAC Accredited With 'A++' Grade, NBA Accredited, Approved by AICTE, New Delhi, Affiliated to JNTUH)

TAMASŌMA JYOTIRGAMAYA

CERTIFICATE

This is to certify that Miss. A. Akshitha (18071A0502), Miss. Ch. Sankranthi (18071A0510), Miss. K. Sahithi Latha (18071A0528) and Mr. S. Shanmukha Anurag (18071A0551) have successfully completed their Online Summer Internship project work at CSE Department of VNR VJIET, Hyderabad entitled **"DRIVER FATIGUE DETECTION SYSTEM"** in partial fulfilment of the requirements for the award of B.Tech degree during the academic year 2020-2021.

This work is carried out under my supervision and has not been submitted to any other University/Institute for award of any degree/diploma.

Mrs. Vasundhara,	Dr. B.V. Kiranmayee
Project Guide	Associate Professor and Head
Designation	CSE Department
CSE Department	VNRVJIET.
VNRVJIET.	

DECLARATION

This is to certify that the project work entitled “**DRIVER FATIGUE DETECTION**” submitted in VNR Vignana Jyothi Institute of Engineering & Technology in partial fulfilment of requirement for the award of Bachelor of Technology in Computer Science and Engineering is a bonafide report of the work carried out by us under the guidance and supervision of Mrs. N Vasundhara (Associate Professor), Department of CSE, VNRVJIET. To the best of our knowledge, this report has not been submitted in any form to any university or institution for the award of any degree or diploma.

A. Akshitha	**Ch. Sankranthi**	**K. Sahithi Latha**	**S. Shanmukha Anurag**
18071A0502	18071A0510	18071A0528	18071A0551
II B. Tech-CSE	II B. Tech-CSE	II B. Tech-CSE	II B. Tech-CSE
VNR VJIET	VNR VJIET	VNR VJIET	VNR VJIET

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who activated it, without it would ever never have come into existence. To them we lay the words of gratitude imprinting within us.

We are indebted to our venerable principal **Dr. C. D. Naidu** for this unflinching devotion, which lead us to complete this project. The support, encouragement given by him and his motivation lead us to complete this project.

We express our thanks to internal guide **Mrs. N. Vasundhara** and also Head of the department **Mrs. B. V. Kiranmayee** for having provided us a lot of facilities to undertake the project work and guide us to complete the project.

We express our sincere thanks to our faculty of the department of **Computer Science and Engineering** and the remaining members of our college **VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY** who extended their valuable support in helping us to complete the project in time.

A. Akshitha (18071A0502)

Ch. Sankranthi (18071A0510)

K. Sahithi Latha (18071A0528)

S. Shanmukha Anurag (18071A0551)

ABSTRACT

The main idea behind this project is to develop a non-intrusive system which can detect fatigue of the driver and issue a timely warning. Each year, millions of people lose their life due to traffic accidents around the world.

In a study by the National Transportation Research Institute (NTSRB), in which 107 random car accidents had been selected, driver fatigue accounted for 58% of all the accidents. The main cause of the fatigue is sleeplessness or insomnia.

The obvious solution to prevent or at least decrease fatigue related accidents, is to ensure that the driver rests frequently. However, the simple fact of the matter is that frequent resting periods cannot be effectively enforced, and as a result there is a need for a system to monitor the level of driver fatigue in real-time.

Drowsiness detection can be done in various ways according to the researchers. The most accurate technique towards the driver fatigue detection is detection of brain waves and heart rate. Also, different techniques based on behaviour can be used which are natural and non-intrusive like closing and visual behaviour of eyes and other ways like yawning and head lowering.

By monitoring the driver's movements using a camera and developing an algorithm we can detect symptoms of driver fatigue early enough to avoid an accident. So, this project will be helpful in detecting driver fatigue in advance and will gave a warning output in form of sound and vibration.

Finally, simulation results show that the proposed fatigue monitoring system detects driver fatigue probability more exactly and robustly. Hence this system will be helpful in preventing many accidents, and consequently save money and reduce personal suffering.

For implementing this system several OpenCV libraries are used including Haar-cascade. The entire system is implemented using Raspberry-Pi.

INDEX

CONTENTS: Page No.

CHAPTER 1: INTRODUCTION

1.1 Need for the project 2
1.2 Introduction to Driver Fatigue Detection 4

CHAPTER 2: EXISTING SYSTEM AND PROPOSED SYSTEM

2.1 Existing System 5
    2.1.1 Study on the Detection of Locomotive Driver Fatigue Based on Image 6
2.2 Proposed System 7
    2.2.1 Driver Fatigue Detection Based on Eye Tracking 7
    2.2.2 A New Real-Time Eye Tracking for Driver Fatigue Detection 7
    2.2.3 Eye Tracking and Dynamic Template Matching 8
    2.2.4 A Novel Real-time Face Tracking Algorithm 9
    2.2.5 Driver Fatigue Detection through Pupil Detection and Yawning Analysis 10

CHAPTER 3: FEASIBILITY STUDY

3.1 Feasibility 11

CHAPTER 4: SYSTEM ANALYSIS

4.1 Interface Requirements 13
    4.1.1 User Interface 13
    4.1.2 Hardware Interface 14
4.2 Hardware and Software Requirements 14
    4.2.1 Hardware 14
    4.2.2 Software 14

CHAPTER 5: SOFTWARE DESIGN

5.1 UML Diagram 15
5.1.1 Definition 15
5.1.2 UML is a Language 15
5.1.3 UML Specifying 15
5.1.4 UML Visualisation 15
5.1.5 UML Constructing 15
5.1.6 Building Blocks of UML 16
5.1.6.1 Things 16
5.1.6.2 Relationships 16
5.1.6.3 UML Diagrams 16
5.1.6.3.1 Structural Diagrams 16
5.1.6.3.2 Behavioural Diagrams 17
5.2 UML Diagrams 17
5.2.1 Use Case Diagram 17
5.2.2 Sequence Diagram 18
5.2.3 Collaboration Diagram 19
5.2.4 Activity Diagram 20

CHAPTER 6: IMPLEMENTATION

6.1 Face detection 21
6.2 Pupil detection and eye state analysis 22
6.3 Yawning Analysis 23
6.4 Inferences 23
6.5 Code Snippets 24

CHAPTER 7: TESTING

7.1 Introduction to Testing 31
7.1.1 Testing Objective 31
7.1.2 Levels of Testing 32
7.2 Testing Strategies 32

7.2.1 Unit Testing 32
7.2.2 Integration Testing 34
7.2.3 Validation Testing 34
7.2.4 System Testing 34
7.2.5 Output Testing 35
7.2.6 Acceptance Testing 35
7.3 Test Approach 35
7.4 Validation 36
7.5 Test Case Results 36

CHAPTER 8: CONCLUSION AND FUTURE SCOPE

8.1 Conclusion 37
8.2 Future Scope 38
8.2.1 Skin Segmentation 38
8.2.2 Eye Binarization 38
8.2.3 Mouth Binarization 39

CHAPTER 9:BIBILOGRAPHY

9.1 References 40

LIST OF FIGURES

CONTENT	PAGE.NO
2.1 Flowchart representing the proposed approach	10
4.1 Functional Block Diagram of System Analysis	13
5.1 Use Case Diagram for Drowsy Detector	17
5.2 Sequence Diagram for Drowsy Detector	18
5.3 Collaboration Diagram for Drowsy Detector	19
5.4 Activity Diagram	20
6.1 Face Detection	21
6.2 Pupil Detection and Eye State Analysis	22
6.3 Yawning Analysis	23
7.1 Levels of Testing	32
8.1 Eye Binarization	39

CHAPTER 1

INTRODUCTION

This chapter outlines the need for fatigue detection system in vehicles and the overview of the real time image processing system designed in this project. Driver fatigue is a significant factor in many vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. There has been ample development in the field of safety in case of accidents in the form of air bags, etc. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects. The aim of this project is to develop a prototype drowsiness detection system. This system will primarily measure and record the real time features of the driver or the driving pattern and continuously evaluate them based on the levels predetermined to indicate fatigue. The driver may show signs of fatigue in numerous ways.

The project surveys different methods of eye detection and checking whether the detected eyes are open or closed. Based on the number of times the eyes are found to be closed, it can be determined whether the person is drowsy or not. Another method of fatigue detection is the evaluation of the features of the mouth. If the driver yawns, showing signs of fatigue, it can be used to trigger the alarm system. This should be used to remind the driver in the form of an appropriate alarm to either park the car or consciously regain the composure to drive safely. Fatigue detection in the non-intrusive form may be done efficiently by increasing the number of parameters on which the driver is being monitored. This will lead to a complex algorithm which can be easily flexible to detect fatigue. Such a driver fatigue detection system has tremendous scope in the car industry. There are millions of cars being manufactured every year and it is a universally accepted fact that fatigue among drivers is a potent accident factor. However, the proposed prototype that this project aims at creating shall be far more inexpensive as compared to that model, thus creating a bigger market for this product. Introduction of such a system in long distance vehicles and in public transport would reduce the number of accidents.

1.1 Need for project

Several studies have investigated the relationship between driver fatigue and crash risk and have attempted to quantify the risk increase. In a case control study of New Zealand drivers, Connor et al compared 571 crash-involved drivers with 588 non-crash involved drivers driving in the same area and at the same times. Driver variables were taken from accident registration and additional interviews. Considering possible confounding variables (gender, age, socio-economic status, annual kilometres, speed, road type), they found a strong relationship between acute fatigue (based on loss of sleep the night before) and crash involvement.

In a case-control study, Cummings et al compared crash-involved drivers with a similar group of non-crash involved drivers at the same location, direction, time, and day. They found the crash risk was fourteen times higher for drivers who had reported to have almost fallen asleep behind the wheel (95 percent confidence interval 1.4-147). The data collected in The 100 Car Naturalistic Driving Study shows that driving while fatigued increases a driver’s risk of involvement in a crash or near-crash by nearly four times. Studies of professional drivers (bus, lorry, truck) show that it takes around 9 or 10 hours of driving, or 11 hours of work, before crash risk starts to rise. Hamelin found that after 11 hours of work span the crash risk doubles. The effect of task duration is practically always entangled with the effects of the time of day and sometimes also with the length of time awake and previous lack of sleep. The duration of a trip may be of lesser importance compared to these other factors - many fatigue-related accidents occur after driving for only a few hours. Short trips can also end up in fatigue-related crashes because time of day and long and irregular working hours are stronger predictors of fatigue than time spent driving. Connor et al. also note blind spots in the research on driver fatigue. The association of non-medical (lifestyle) determinants of fatigue with crash has not been the subject of thorough research.

There is still a lack of knowledge concerning the contribution of increasing total hours of work, and shift schedules to driver fatigue. Whereas research into fatigue and sleep apnoea in truck drivers has led to awareness of these problems and some modification of work conditions, occupationally induced fatigue in potentially much larger numbers of commuters has received little attention. Drowsiness affects mental alertness, decreasing an individual’s ability to operate a vehicle safely and increasing the risk of human error that could lead to fatalities and injuries. Furthermore, it has been shown to slow reaction time, decreases awareness, and impairs judgment. Long hours behind the wheel in monotonous driving environments make truck drivers particularly prone to drowsy-driving crashes. Operational requirements are diverse, and factors such as work schedules, duty times, rest periods, recovery opportunities, and response to customer needs can vary widely. While these challenges preclude a single, simple solution to the problem, there is reason to believe that driver drowsiness can nevertheless be effectively managed, thus resulting in a significant reduction in related risk and improved safety. Addressing the need for a reduction in crashes related to driver drowsiness in transportation will require some innovative concepts and evolving methodologies.

Within any comprehensive and effective fatigue management program, an on-board device that monitors driver state in real time may have real value as safety net. Sleepy drivers exhibit certain observable behaviour, including eye gaze, eyelid movement, pupil movement, head movement, and facial expression. Non-invasive techniques are currently being employed to assess a driver's alertness level through the visual observation of his/her physical condition using a remote camera and state-of-the-art technologies in computer vision. Recent progress in machine vision research and advances in computer hardware technologies have made it possible to measure head pose, eye gaze, and eyelid movement accurately and in real time using video cameras. The alertness monitoring technologies monitor - usually on-line and in real time - bio-behavioural aspects of the operator; for example, eye gaze, eye closure, pupil occlusion, head position and movement, brain wave activity, and heart rate. To be practical and useful as driver warning systems, these devices must acquire, interpret, and feed-back information to the operator in real world driving environments. As such, there exists a need and, thus, ongoing efforts are underway, to validate operator-based, on-board fatigue monitoring technologies in a real-world naturalistic driving environment. In view of the need for non-intrusive relatively inexpensive modules for fatigue detection in vehicles, this project has been formulated to monitor the driver's actions and reactions, check for fatigue and on detection of fatigue or drowsiness, stimulate the appropriate plan of action, which could be an alarm or deceleration of the vehicle, etc.

1.2 Face Detection using Image Processing

Image processing is a physical process used to convert an image signal into a physical image. The image signal can be either digital or analogue. The actual output itself can be an actual physical image or the characteristics of an image. In this project, it is required to detect the face of the driver and then detect the driver’s eyes and mouth to check whether he is blinking and yawning, respectively. We have used image processing to synthesize and analyse the real time captured images of the driver from the camera to detect the face and use them for probable fatigue detection. There have been many varied ways to detect the human face that have been tested positively till now in various parts of the world. However, the aim of this project is to have fast and efficient detection of face and features of the face and hence real time face detection is necessary.

CHAPTER 2

EXISTING SYSTEM AND PROPOSED SYSTEM

2.1 Existing System

In the existing system, we must wear a glass and an IR sensor is fixed in that glass to sense the drowsiness of a person. By using this for a prolonged period, eyes may get affected due to the passage of IR rays.

Several other works have been done in the field of driver abnormality monitoring and detection systems using a wide range of methods. Among the possible methods, the best techniques are the ones based on human physiological phenomena. These techniques can be implemented by measuring brain waves (EEG), heart rate (ECG) and open/closed state of the eyes. The former two methods, though being more accurate, are not realistic since sensing electrodes to be attached directly onto the driver's body and hence be annoying and distracting the driver. The latter technique based on eye closure is well suited for real world driving conditions, since it can detect the open/closed state of the eyes non-intrusively using a camera. Eye tracking based drowsiness detection systems have been done by analysing the duration of eye closure and developing an algorithm to detect the driver's drowsiness in advance and to warn the driver by in-vehicle alarms.

Other problems related with existing systems are:

1) For identifying the driver's fatigue level, detecting the eye state is more important than detecting the eye-steering correlation.
2) Correlation between the eye-steering is complex and might produce wrong results in undetermined cases.
3) Steering movements consists of large calculations and graphs and it is difficult to merge with eye movements to all types of roads. Lane determination is not necessary for driver distractions.

2.1.1 Study on the Detection of Locomotive Driver Fatigue Based on Image

By using a non-intrusive machine vision-based concepts, drowsiness of the driver detected system is developed. Many existing systems require a camera which is installed in front of driver. It points straight towards the face of the driver and monitors the driver’s eyes to identify the drowsiness. For large vehicle such as heavy trucks and buses this arrangement is not pertinent. Bus has a large front glass window to have a broad view for safe driving. If we place a camera on the window of front glass, the camera blocks the frontal view of driver, so it is not practical. If the camera is placed on the frame which is just about the window, then the camera is unable to detain the anterior view of the face of the driver correctly. The open CV detector detects only 40% of face of driver in normal driving position in video recording of 10 minutes. In the oblique view, the Open CV eye detector (CV-ED) frequently fails to trace the pair of eyes. If the eyes are closed for five successive frames the system concludes that the driver is declining slumbering and issues a warning signal [4]. Hence existing system is not applicable for large vehicles. To conquer the problem of existing system, new detection system is developed in this project work.

Steering pattern monitoring

Primarily uses steering input from electric power steering system. Monitoring a driver this way only works as long a driver actually steers a vehicle actively instead fo automatic lane keeping system

Vehicle position in lane monitoring

Uses lane monitoring camera. Monitoring a driver this way only works as long a driver steers a vehicle actively instead of an automatic lane-keeping system.

Driver eye/face monitoring

Uses computer vision to observe the driver's face, either using a built-in camera or on mobile devices.

Physiological measurement

Requires body sensors to measure parameters like brain activity, heart rate, skin conductance, muscle activity.

2.2 Proposed System

2.2.1 Driver Fatigue Detection Based on Eye

This paper gives a method of detection of driver fatigue based simply on eye-tracking. The first and foremost step is to detect the face. Once that is done, the image is binarized. The image is then scanned to detect gray scale value zero (0) - which indicates the eye region. Projection charts are now created which depict the number of pixels per line. The two crests observed in the chart indicate eyes. For more accurate position of eyes, a unique way of calculating slope of lines is presented. This helps ascertain the boundaries of the eye region. Once eyes have been detected, the next step is to find out if the driver is displaying signs of fatigue. One of the easiest methods to do so is to calculate the number of pixels with gray scale value zero (0) - they indicate presence of eyes and eyelashes. The value, say n, will change depending on eye state. If the driver is dozing, n will vary very slowly i.e. a gradual change in the value of n will be seen. While if the driver blinks, n will change, however the change will be very fast and can thus be disregarded. The best aspect of this algorithm is that it has been implemented in VC 6.0 on a windows XP platform, like our needs. It is a time saving method and is known to give good results. So, considering this fact, it will be useful to us as we are making a real-time system which must respond quickly and accurately.

2.2.2 A New Real-Time Eye Tracking for Driver Fatigue Detection

The Unscented Kalman filter (UKF) is proposed to track eyes for Driver fatigue detection. Haar algorithm is firstly used to locate the face. Secondly, the geometric properties and projection technique are used for eye location. Thirdly, the UKF is used for real time eye tracking method because UKF is of good tracking performance for quick moving target. Finally, driver fatigue can be detected whether the eyes are closed over 5 consecutive frames using vertical projection matching. Face Detection-Harr algorithm has good robustness in terms of head motions. Variable lighting conditions, the change of hair and having glasses. The response, in Haar algorithm, is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature. Eye Detection-According to the normal position of the eyes in driver face, it is reasonable to assume that the possible location of eyes will be in the block of the upper three fifths of the driver face region, After defining the eye region, we can use geometric properties and projection technique for eye detection. Because the value of pixels in eye region is relatively lower than other region, we can calculate the pixels value for eye detection. Performing horizontal and vertical projection on the eye region map, the exact position of the eye can be located at the peak. There will be two peaks which will be detected, one will represent the projection of eyebrow. After locating the left and right eye we use the bounding box to enclose the eye region.UKF FILTER BASED EYE TRACKING- The unscented Kalman filter proposed by Julier and Uhlmann can not only avoid the use of Jacobian matrices, simplifying the implementation of the filter and potentially reducing its complexity, but it also provides more accurate estimation result, reducing the potentially instability issues that could arise with the Kalman filter when the linearization is not a good approximation.

2.2.3 Driver Fatigue Detection Based on Eye Tracking and Dynamic Template Matching

This paper presents an idea for detection of face as well as the eyes. Face detection is achieved by the characteristic of skin colours. Since skin colours have quite stable distribution in some colour models, face detection based on skin colour is popular. The HSI colour model is preferred since intensity changes should not create an obstacle in face detection. In this model, intensity is detached from the hue of a colour. From face images in a test video, it was observed that H values of skin colour fall in the range of [10 degrees, 45 degrees]. So, the face region can now be separated from the image if it crosses this threshold. Next, the eyes must be located. Here, it is assumed that the possible location of the eyes will be in the block of the upper two fifths of the face region. Once the eye region is defined, it is converted to grey. Now, in this paper, it is said that it consists of dynamic tracking of the eyes. It means that at time t=0, there is no stored eye template. The first eye template obtained becomes the basis for comparison for the next eye search. Thus, this method ensures that memory requirements are kept to a minimum. For exact location of eyes, horizontal projection is performed. The eyes are regarded as connected components for rectangular boxes to be drawn around them. Hence, the template is ready. For fatigue detection, eyeballs which appear bright are obtained by inverting the image. By observation, the saturation values of eyeball pixels fall in the range of 0.00 to 0.14. Eyeball pixels are checked for detection of open or closed eyes. If eyes are closed over five consecutive frames, then the driver is said to be dozing.

2.2.4 A Novel Real-time Face Tracking Algorithm for Detection of Driver Fatigue

The input face images are pre-processed to gray level images by using generalized Laplacian. Two eyes and mouth are labelled in binary image. According to pose of the driver, the system forms either isosceles feature triangle or right feature triangle to acquire potential face region. During the tracking of the video sequence, the confusing triangle is removed effectively.

Image Processing- RGB image is converted to Y, Cr and Cb colour space. The eye regions have low intensity (Y), low red chrominance (Cr), and high blue chrominance (Cb), when compared to the forehead region of the face. Using this fact, we will pre-process the input image to a gray level image. The PDF of gray-level differences between neighbouring pixels is well approximated by a generalized Laplacian.

Rules for Triangle Formation-For frontal images, we could search the potential face regions that are obtained from the criteria of" the combination of two eyes and one 10 mouth (isosceles triangle)". For side-view images, we use the rationale of" the combination of one eye, one ear hole, and one mouth (right triangle)", which is called the Feature Triangle. For Isosceles triangle- Statistics shows that the Euclidean distance between two eyes is about 90-110 percent of the Euclidean distance between the centre of the right/left eye and the mouth. Using this data, we can form an isosceles triangle on the frontal image. We then apply matching rules to verify whether the right-angled triangle and Isosceles triangle so formed is the required one which can be used for further processing. Most noise components can be removed after the above triangle verification process. Skin Colour Ratio (SCR) defined for arbitration of confusing triangle where if a confusing combination of triangle is encountered it can be eliminated using the above factor which on area of triangle and number of skin coloured pixels within the enclosed triangle. Steps for above algorithm are as follows

1. Initialize the image and convert to Y, Cr, Cb
2. Apply feature triangle rules to form isosceles or right-angle triangle

3. Verify the triangle

4. Eliminate Noise Use SCR ratio for arbitration of confusing triangle to ensure no false triangle detection.

2.2.5 Driver Fatigue Detection through Pupil Detection and Yawning Analysis

Basic parameters that can be used to evaluate alertness of driver:

1. Drivers’ physiological parameters like brain waves, heart rate, pulse rate and respiration

2. Driving performance like changes of the steering wheel, vehicle lateral position, driver’s grip force on the steering wheel, acceleration, vehicle speed, acceleration, braking and turning angle.

3. Non-intrusive way, by detecting the eyes- open or close and rate of opening and closing, and mouth- frequency of yawning

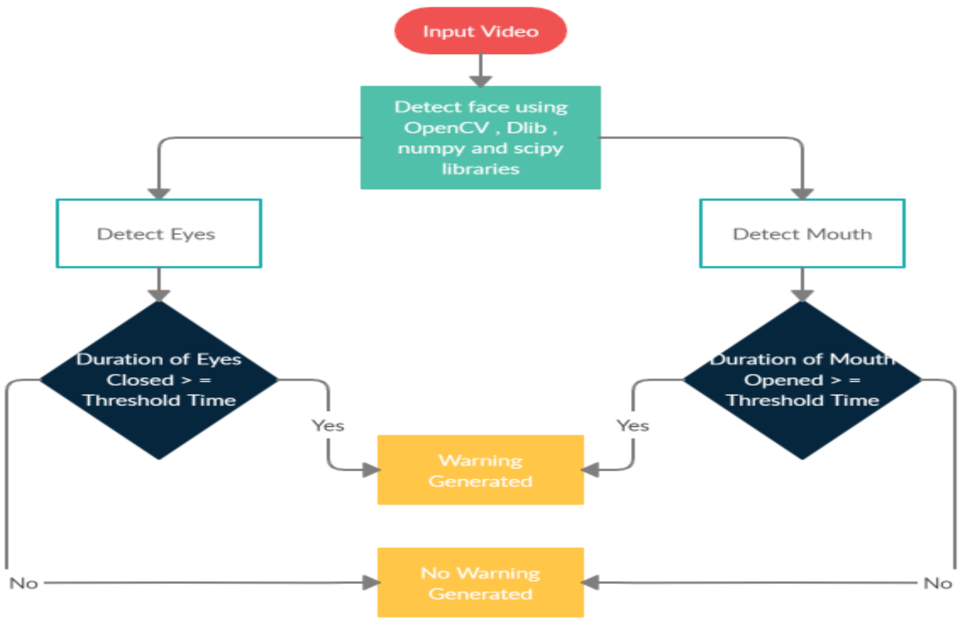
Input Video
Detect face using OpenCV , Dlib , numpy and scipy libraries
Detect Eyes
Detect Mouth
Duration of Eyes Closed > = Threshold Time
Duration of Mouth Opened > = Threshold Time
Yes
Yes
Warning Generated
No
No Warning Generated
No

Fig: 2.1 Flowchart representing the proposed approach for Driver Fatigue Detection

CHAPTER 3

FEASIBILITY STUDY

Fatigue detection is not an easy task. It requires considering many factors.
Using a video system for this purpose can be a good solution. This system would allow for precise detection of a fatigue in real time. The speed of such a system is very important because even slight delays in the operation of such a system could be fatal (excessive reaction of the system while traveling along the highway).
An important issue in the design of the vision-based driver fatigue detection system is the right choice of the analysed symptoms of fatigue. In a situation, where it is not possible to monitor all potential symptoms of a fatigue, it should be limited to the detection of the most important ones such as: closing the eyelids, slow the eye movements, yawning and drooping a head.
The basis of the fatigue detection system are the algorithms responsible for detecting facial features and their motion. There are many methods that allow detecting individual facial elements. They are based both on the vector operations and the pattern classification. Particular methods are based on an image filtering in complex space or an image processing in spatial-frequency domain. Some methods are very effective in detecting characteristic facial features, but sensitive to changing lighting conditions.

The method based on neural networks is used for processing input data. Neural networks are used for identification and classification of pattern data, and therefore they are also used in face detection and recognition systems. Gabor filters are one of the most commonly used methods for representing facial features, using complex functions. Frequency-spatial methods are based on frequency analysis of the image in conjunction with the methods based on a geometric model. Frequency-spatial methods allow for the proper isolation of 44 the characteristic facial features and minimizing the influence of lighting conditions during the acquisition.

In vision-based systems it is important to correctly identify the specific elements as well as to analyse their movement. Common methods used to detect a motion in video systems are differential and gradient methods. Differential methods determine the difference between the subsequent image frames. This allows determining the brightness level in the grayscale or the colour intensity of the pixel during the frame changes. So, the movement of the object can be detected (this is related to the change in the brightness of the pixels that appear next to each other in the image). This is a simple way to detect a movement, however, its implementation can be tedious. One of the limitations of this method is to have the stationary background, the lighting should be constant, and the noise in the film should be reduced to a minimum (otherwise, the algorithm may work not properly).

Additionally, to improve a movement detection, the moving object should contrast with the background. Gradient methods rely on the optical flow. They use spatial and temporal derivatives of the consecutive video frames. In order to make an effective use of this group of the methods, the following conditions should be met: invariability of the light, a small displacement of moving objects in one sequence and spatial coherence of the contiguous dots. Two most popular gradient algorithms are Lucas-Kanade and Horn-Schunk algorithm. The principle of operation of the first algorithm is a characteristic assumption: the brightness of the dots in the image is unchanged over the time, the movement of the frames is constant.

When designing a video system that records moving objects, one should choose algorithms that are resistant to interference. Interference occurrence may disturb the processing and the analysis of the data, which may lead to misinterpretation by the system. If the selected methods are susceptible to interference, then the system will not analyse the movement of the elements. It may lead to a kind of dynamic "jumps" of the system between the observed objects. The next procedure is to register and analyse the movement of the classified features. It should be remembered that real-time systems require a rapid response to the changes in observed objects. For example, if the eyelid is closed for a long time, the system response: "eyelids close" should be immediate. Any delay in the identification of a fatigue can have catastrophic consequences. If the system does not respond in time to the driver's microsleep, an accident may occur.

CHAPTER 4

SYSTEM ANALYSIS

In order to enhance the accuracy rate of detection to the fatigue state of the pilot, this system extracts four state variables from the eye condition: It contains the frequency of blink, the average degree of opening eyes, the eye stagnation time and the longest time of closing eyes . According to the parameter value of the pilot’s sober condition by statistics, it can make the corresponding judgment by the fatigue state of the pilot.

Considered the changing driving environment, the detection system must work normally in the night or the situation of inadequate lighting, therefore, this system adopts the camera with light source(automatic opening when the light is inadequate) to gather sequence image, in order to reduce the disturbance from the external environment. When the system reads in the frame image, it carries on the denoising and the image intensification process to the image first, and then obtains two real eye points. Afterward, it adopts the target tracking method to track the already targeted eye point. At last, it can calculate the area of the eyes and make the judgment and the early warning to the fatigue state of the pilot. The functional block diagram of the system is shown in the following Figure 4.1

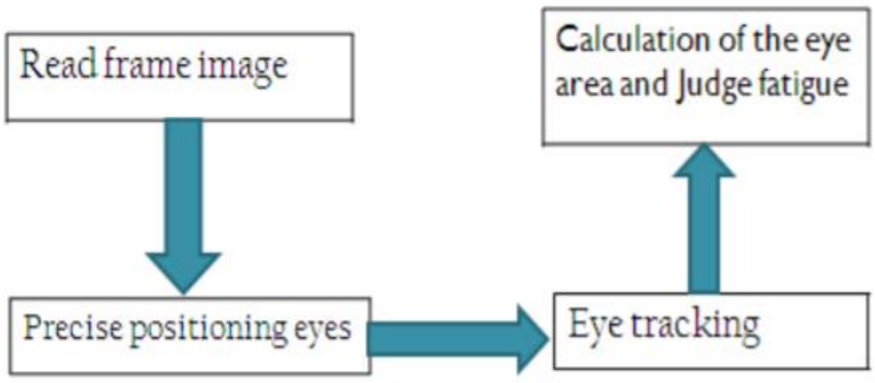

Fig 4.1 Functional Block Diagram of System Analysis

4.1 Interface Requirements:

4.1.1 User Interface:

Every user may not be skilled at handling the interfaces. Hence the product that we developed used a simple and easy to use GUI Input from user is via keyboard.

4.1.2 Hardware Interface:

The minimum requirements that are required to interact with a simple GUI are well enough to support this product.

4.1.3 Software Interface: This product is developed in Windows 10 environment using PYTHON. The toolboxes used to develop this product are Image Processing Toolbox and Computer Vision System Toolbox. This project is completely implemented using these two toolboxes.

4.2 Hardware and Software Requirements

4.2.1 Hardware Requirements:

Processor: Intel® Core™ i3 2.53 GHz / Above

RAM: RAM 2 GB / Above HDD: 120 GB / Above

4.2.2 Software Requirements:

Operating System: Windows XP and above

Developing Environment: PYTHON 3.6

CHAPTER 5

SOFTWARE DESIGN

5.1 UML Diagrams:

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. UML is a very important part of developing objects-oriented software and the software development process.

UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

5.1.1 Definition:

UML is a general-purpose visual modelling language that is used to specify, visualize, construct, and document the artifacts of the software system.

5.1.2 UML is a language:

It will provide vocabulary and rules for communications and function on conceptual and physical representation. So, it is modelling language.

5.1.3 UML Specifying:

Specifying means building models that are precise, unambiguous, and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

5.1.4 UML Visualization:

The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

5.1.5 UML Constructing:

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

5.1.6 Building blocks of UML:

The vocabulary of the UML encompasses 3 kinds of building blocks

1. Things
2. Relationships
3. Diagrams

5.1.6.1 Things:

Things are the data abstractions that are first class citizens in a model. Things are of 4 types Structural Things, Behavioural Things, Grouping Things, Annotational Things.

5.1.6.2 Relationships:

Relationships tie the things together. Relationships in the UML are Dependency, Association, Generalization, Specialization.

5.1.6.3 UML Diagrams:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships). There are two types of diagrams, they are:

1.Structural Diagrams

2.Behavioural Diagrams

5.1.6.3.1 Structural Diagrams:

The UML ‘s four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. I can View the static parts of a system using one of the following diagrams. Structural diagrams consist of Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

5.1.6.3.2 Behavioural Diagrams:

The UML's five behavioural diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML's behavioural diagrams are roughly organized around the major ways which can model the dynamics of a system. Behavioural diagrams consist of Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram.

5.2 UML DIAGRAMS

5.2.1 Use Case Diagram:

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, associations between actors and users. In general, it shows a set of use cases and actors and their relationships. The creation of a use case model is an excellent vehicle for elicitation of functional requirements.

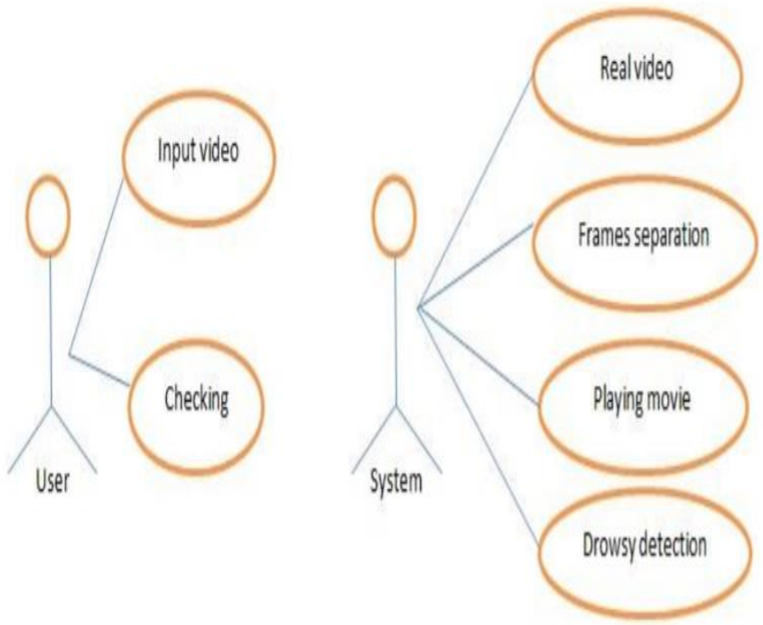

Fig 5.1 Use Case Diagram for Drowsy Detector

5.2.2 Sequence Diagram

Sequence diagram are an easy and intuitive way of describing the behaviour of a system by viewing the interaction between the system and its environment. A sequence diagram has two components are vertical dimension represents time, the horizontal dimension represents different object. The vertical line is called object's lifeline

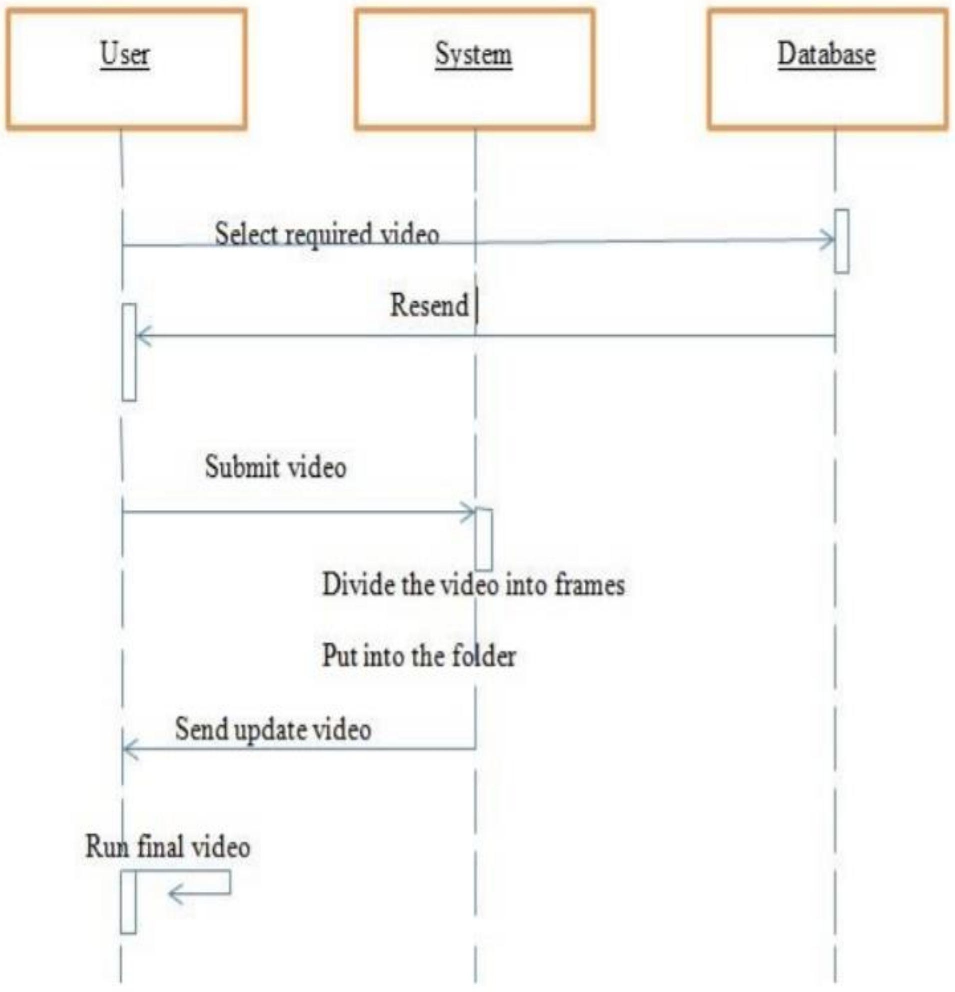

Fig 5.2 Sequence Diagram for Drowsy Detector

5.2.3 Collaboration Diagram

Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system. The collaboration diagram represents interactions among objects in terms of sequenced messages.

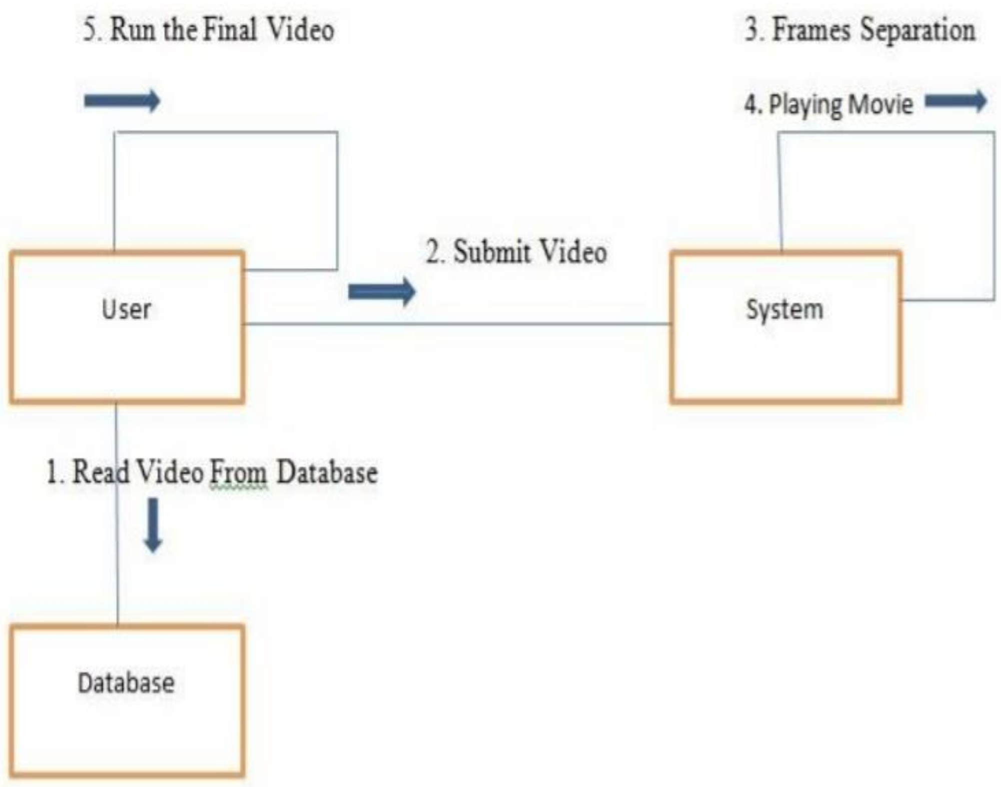

Fig 5.3 Collaboration Diagram for Drowsy Detector

5.2.4 Activity Diagram

The purpose of activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. An activity is shown as around box containing the name of the operation.

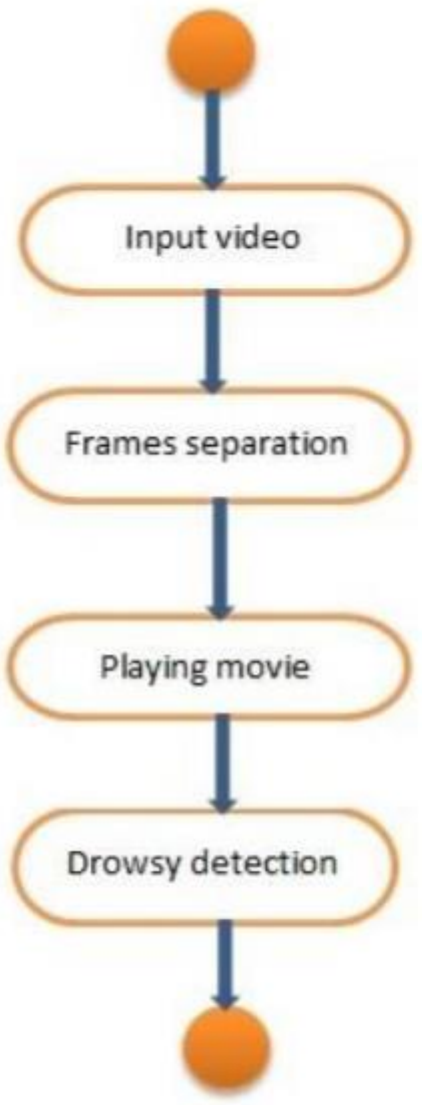

Fig 5.4 Activity Diagram for Drowsy Detector

CHAPTER 6

IMPLEMENTATION

6.1 Face detection

Using RGB values, luminance(Y), blue difference and red difference, Cb and Cr respectively are calculated from image. These values are then checked if they lie between the empirically calculated ranges of skin colour. This can later undergo binarization to create an image which is minus the background, thus detecting the face.

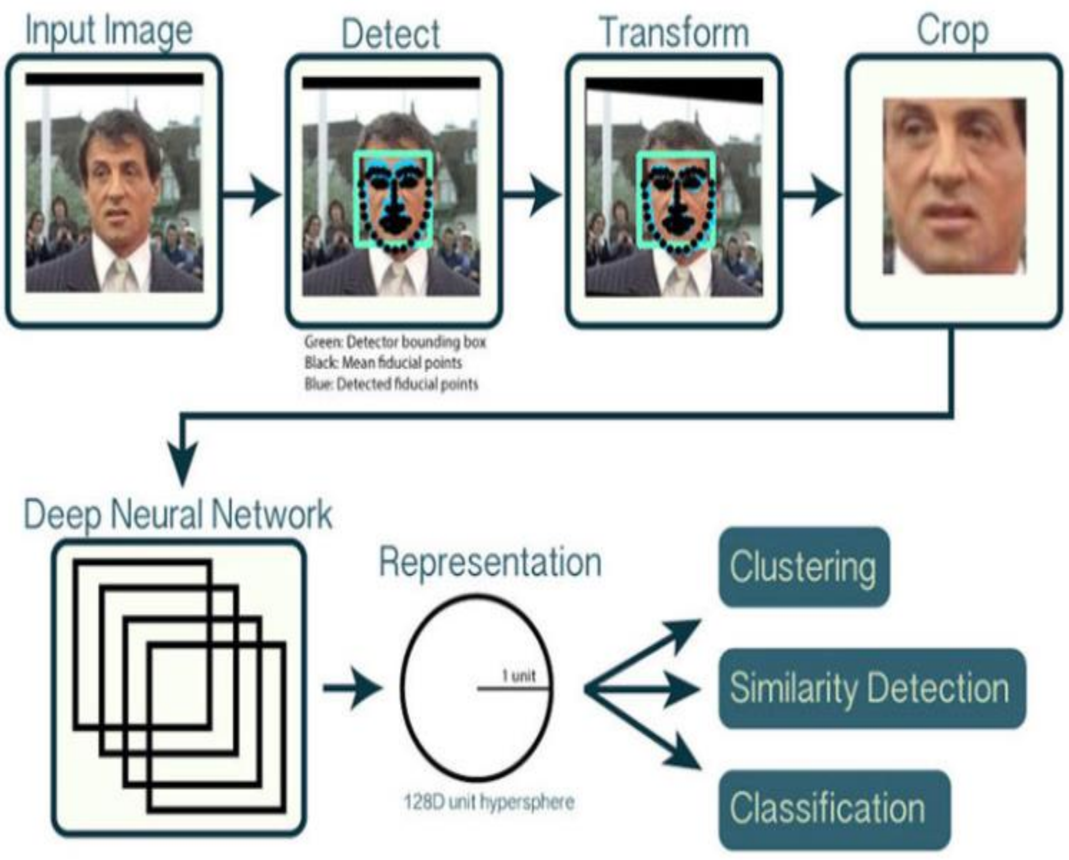

Fig 6.1 Face Detection

6.2 Pupil detection and eye state analysis

Using an IR illuminator, bright and dark image of the pupils are obtained and subtracted to get a difference image and the background is removed. PERCLOS (Percentage of Eyelid Closure Over the Pupil Over Time) principle is used to eye state analysis. f= [t3- t2] /[t4-t1] The f is the value of PERCLOS, t1, t2 are the time that eyes closed from the largest to 80 percent, from 80 percent to 20 percent; t3 is the time from 20percent closed to 20percent open, t4 is 20 percent open to 80 percent the amount of time spent. When used this method to measure state of eyes we use camera to get the image of driver's face, then we position eyes through image processing methods, at last we analysis and identify the image to confirm that the eyes are open or closed.

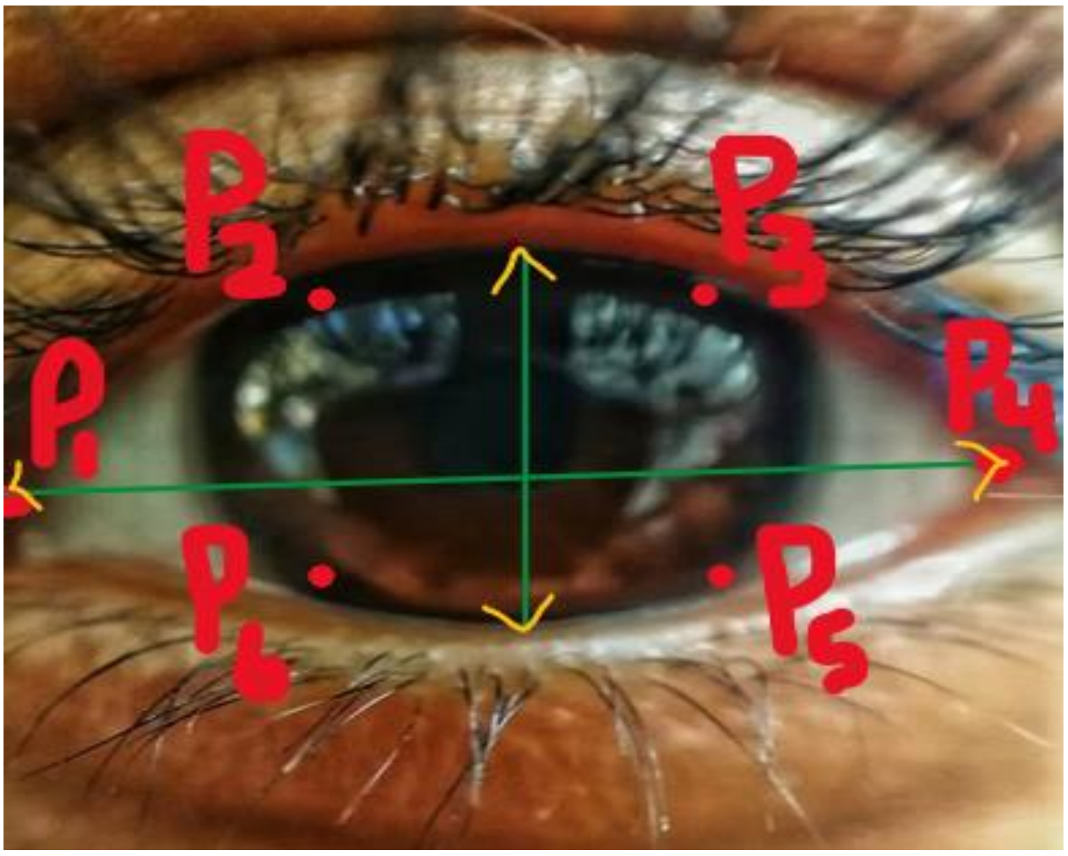

Fig 6.2 Pupil Detection and Eye State Analysis

6.3 Yawning Analysis

Yawning can be detected from the extent of openness of the mouth. If ratio of mouth height to width is above 0.5, the user is yawning and if more than 6 frames detect yawning, the system points fatigue.

Problems

Performance of the IR illuminator is good at night but reduces on bright days.

Yawning cannot be detected when the driver puts a hand on his face. Enditemize

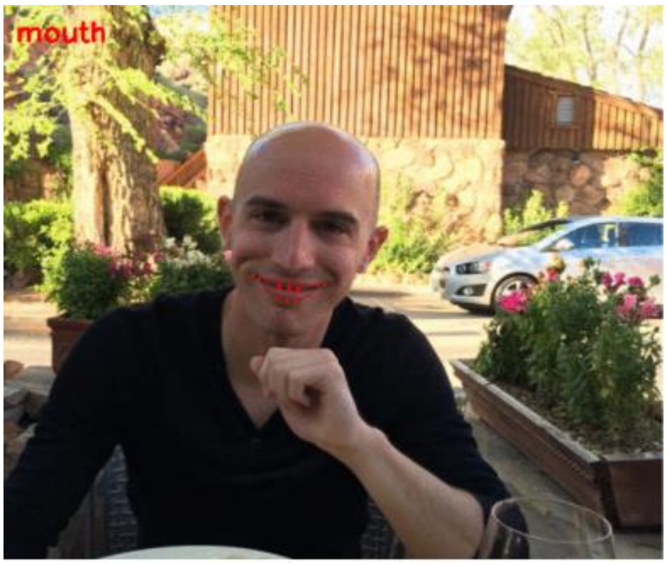

Fig 6.3 Yawning Analysis

6.4 Inferences

Based on the reviewed papers and after considering the advantages and drawbacks of various methods, Haar Classifier has been used to detect the face of the driver from the image and to further extract features like the eyes and mouth. Hough Transform is used detect whether the eyes are open or closed. These preferences were based on the memory limitations of the system and for real time operations

6.5 Code Snippets

List of libraries

```
from scipy.spatial import distance as dist
from imutils.video import Video Stream
from imutils import face_utils
from threading import Thread
import numpy as np
import playsound
import argparse
import imutils
import time
import dlib
import cv2
import matplotlib.gridspec as gridspec
from matplotlib import pyplot as plt
from collections import OrderedDict
import numpy as np
import cv2
```

Activating the alarm

```
# activating the alarm
def sound_alarm(path):
    playsound.playsound(path)
```

Calculating the eye aspect ratio

```
def cal_E_A_R(eye):
    # compute the euclidean distances between the two sets of
    # vertical eye landmarks (x, y)-coordinates
    dist_a = dist.euclidean(eye[1], eye[5])
    dist_b = dist.euclidean(eye[2], eye[4])

    # compute the euclidean distance between the horizontal
    # eye landmark (x, y)-coordinates
    dist_c = dist.euclidean(eye[0], eye[3])

    # compute the eye aspect ratio
    e_a_r = (dist_a + dist_b) / (2.0 * dist_c)

    # return the eye aspect ratio
    return e_a_r
```

Constructing Arguments to Parse in the Command Line Interface

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--landmark-identifier",
    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())

print("printing all the input arguments")
print("landmark-identifier ",args["landmark_identifier"])
print("alarm ",args["alarm"])
print("webcam ", args["webcam"]+1)
```

Defining Constants and Variables

```
# define two constants, one for the eye aspect ratio to indicate
# blink and then a second constant for the number of consecutive
# frames the eye must be below the threshold for to set off the
# alarm
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 48
EYE_AR_CONSEC_FRAMES_blink = 20

# initialize the frame counter as well as a Boolean used to
# indicate if the alarm is going off
COUNTER = 0
ALARM_ON = False

# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["landmark_identifier"])
# grab the indexes of the facial landmarks for the left and
# right eye, respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

Start the Video Streaming

```
# start the video stream thread
print("[INFO] starting video stream thread...")
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)
```

Map the indexing to the facial

```
# define a dictionary that maps the indexes of the facial
# landmarks to specific face regions
FACIAL_LANDMARKS_IDXS = OrderedDict([
    ("mouth", (48, 68)),
    ("right_eyebrow", (17, 22)),
    ("left eyebrow", (22, 27)),
    ("right_eye", (36, 42)),
    ("left_eye", (42, 48)),
    ("nose", (27, 35)),
    ("jaw", (0, 17))
])
```

Rectangle to Bounded Box

```
def rect_to_bb(rect):
    # take a bounding predicted by dlib and convert it
    # to the format (x, y, w, h) as we would normally do
    # with OpenCV
    x = rect.left()
    y = rect.top()
    w = rect.right() - x
    h = rect.bottom() - y
    # return a tuple of (x, y, w, h)
    return (x, y, w, h)
```

Initialise the list of coordinates

```
def shape_to_np(shape, dtype="int"):
    # initialize the list of (x, y)-coordinates
    coords = np.zeros((68, 2), dtype=dtype)
    # loop over the 68 facial landmarks and convert them
    # to a 2-tuple of (x, y)-coordinates
    for i in range(0, 68):
        coords[i] = (shape.part(i).x, shape.part(i).y)
    # return the list of (x, y)-coordinates
    return coords
```

Visualising Facial Landmarks

```python
def visualize_facial_landmarks(image, shape, colors=None, alpha=0.75):
    # create two copies of the input image -- one for the
    # overlay and one for the final output image
    overlay = image.copy()
    output = image.copy()
    # if the colors list is None, initialize it with a unique
    # color for each facial landmark region
    if colors is None:
        colors = [(19, 199, 109),
            (79, 76, 240),
            (230, 159, 23),
            (168, 100, 168),
            (158, 163, 32),
            (163, 38, 32),
            (180, 42, 220)]
    # loop over the facial landmark regions individually
    for (i, name) in enumerate(FACIAL_LANDMARKS_IDXS.keys()):
        # grab the (x, y)-coordinates associated with the
        # face landmark
        (j, k) = FACIAL_LANDMARKS_IDXS[name]
        pts = shape[j:k]
        # check if are supposed to draw the jawline
        if name == "jaw":
            # since the jawline is a non-enclosed facial region,
            # just draw lines between the (x, y)-coordinates
            for l in range(1, len(pts)):
                ptA = tuple(pts[l - 1])
                ptB = tuple(pts[l])
                cv2.line(overlay, ptA, ptB, colors[i], 2)
        # otherwise, compute the convex hull of the facial
        # landmark coordinates points and display it
        else:
            hull = cv2.convexHull(pts)
            cv2.drawContours(overlay, [hull], -1, colors[i], -1)
    # apply the transparent overlay
    cv2.addWeighted(overlay, alpha, output, 1 - alpha, 0, output)
    # return the output image
    return output
```

Main Function and Logic

```
# loop over frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=700)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # detect faces in the grayscale frame
    rects = detector(gray, 0)
    cv2.putText(frame, "Press 'q' to exit", (600, 550),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.7, (0, 255, 255), 2)
    # 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 = face_utils.shape_to_np(shape)
        # loop over the face parts individually
        for (name, (i, j)) in face_utils.FACIAL_LANDMARKS_IDXS.items():
            # clone the original image so we can draw on it, then
            # display the name of the face part on the image
            # loop over the subset of facial landmarks, drawing the
            # specific face part
            for (x, y) in shape[i:j]:
                cv2.circle(frame, (x, y),
                1, (0, 0, 255), -1)
            # extract the ROI of the face region as a separate image
            (x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
            roi = frame[y:y + h, x:x + w]
            roi = imutils.resize(roi, width=600,
                inter=cv2.INTER_CUBIC)
        # 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]
        left_EAR = cal_E_A_R(leftEye)
        right_EAR = cal_E_A_R(rightEye)
        # average the eye aspect ratio together for both eyes
        e_a_r = (left_EAR + right_EAR) / 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)
        cv2.drawContours(frame, [leftEyeHull], -1,
            (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -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 e_a_r < EYE_AR_THRESH:
            COUNTER += 1
            # if the eyes were closed for enough 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=sound_alarm,
                            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)
            elif COUNTER >= EYE_AR_CONSEC_FRAMES_blink:
                cv2.putText(frame, "YOU JUST BLINKED", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 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, "THE DRIVER IS AWAKE", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 0), 2)
        # draw the computed eye aspect ratio on the frame to help
        # with debugging and setting the correct eye aspect ratio
        # thresholds and frame counters
        cv2.putText(frame, "Eye Aspect Ratio: {:.2f}".format(e_a_r),
                (400, 30),
                cv2.FONT_HERSHEY_TRIPLEX, 0.7, (0, 255, 0), 2)
    # 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 clean up
cv2.destroyAllWindows()
vs.stop()
```

How to run the python file

In the command line interface type the following command

```
python [filename].py --landmark-identifier shape_predictor_68_face_landmarks.dat --alarm alarm.wav
```

For the program to successfully run a xml haarcascade file must be present in the same directory as the current executable file. And, for the alarm to turn on when the driver is drowsy an alarm file with the .wav extension has to be present in the same working directory

File Hierarchy

Project

```
--------- Drowsiness_Detect.py
--------- shape_predictor_68_face_landmarks.dat
--------- alarm.wav
```

CHAPTER 7

TESTING

7.1 Introduction to Testing:

Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results. In simple words testing is executing a system to identify any gaps, errors, or missing requirements in contrary to the actual desire or requirements. Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design, and coding. It represents interesting anomaly for the software. Testing is the process of detecting errors. It performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later during maintenance also. Generally, the testing phase involves the testing of the developed system using various test data. Preparation of the test data plays a vital role in the system testing. After preparing the test data the system under study was tested using those test data. While testing the system, errors were found and corrected by using the testing steps and corrections are also noted for future use. Thus, a series of testing is performed for the proposed system before the system was ready for the implementation. Thus, the aim of testing is to demonstrate that a program works by showing that it has no errors. The fundamental purpose of testing phase is to detect the errors that may be present in the program. Thus, testing allows developers to deliver software that meets expectations, prevents unexpected results, and improves the long-term maintenance of the application. Depending upon the purpose of testing and the software requirements, the appropriate methodologies are applied. Wherever possible, testing can also be automated.

7.1.1 Testing Objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, altogether, Testing is a process of executing a program with the intent of finding an error. A successful test is one that uncovers a yet undiscovered error. A good test case is one that has a high probability of finding error, if it exists. The software more or less confirms to the quality and reliable standards.

7.1.2 Levels of testing:

To uncover the errors, present in different phases we have the concept of levels of testing. The basic levels of testing are

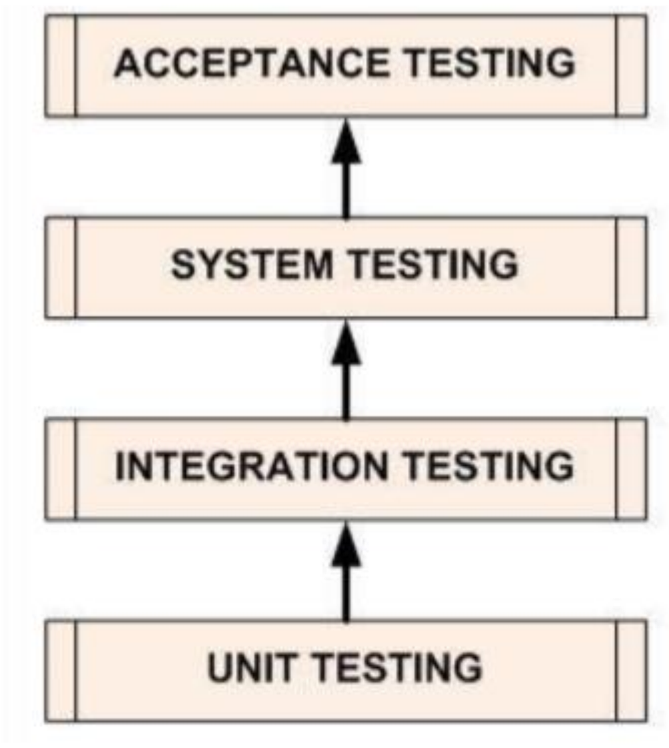

Fig 7.1 Levels of Testing

7.2 Testing Strategies:

7.2.1 Unit testing

Unit Testing is a level of the software testing process where individual units or components of a software or system are tested. It is also known as component testing, refers to tests that verify the functionality of a specific section of code. The purpose is to validate that each unit of the software performs as designed. All modules must be successful in the unit test before the start of the integration testing begins. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. Unit testing focuses verification effort on the similar unit of software design the form. This is known as form testing. In this testing step, each module is found to be working satisfactorily, as regard to the expected output from the module. Each module has been tested by giving different sets of inputs, when developing the module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user. Each module can be tested using the following two strategies:

White Box Testing:

White box testing is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. White box testing is testing beyond the user interface and into the nitty-gritty of a system. This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. We tested step wise every piece of code, taking care that every statement in the code is executed at least once. White-box testing can be applied at the unit, integration, and system levels of the software testing process. It is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test.

The white box testing is also called Glass Box Testing. We have generated a list of test cases sample data, which is used to check all possible combinations of execution paths through the code at every module level. This testing has been uses to find in the following categories: Execute internal data structures to ensure their validity. Guarantee that all independent paths have been executed. Execute all logical decisions on their true and false sides.

Black Box Testing:

The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It treats the software as a "black box", examining functionality without any knowledge of internal implementation. This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level i.e., here the module will be treated as a black box that will take some input and generate output. Output for a given set of input combinations are forwarded to another module. This testing has been uses to find in the following categories: Initialization and termination errors. Incorrect or missing functions Performance errors

7.2.2 Integration Testing

Integration Testing is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. After the unit testing, we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. It works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

The testing of combined parts of an application to determine if they function correctly together is the main motto of integration testing. There are two methods of doing integration testing. They are Bottom-up integration testing and Top Down integration testing. All modules are combined in the testing step. Then the entire program is tested.

7.2.3 Validation Testing

At the culmination of the integration testing, the software is completely assembled as a package, interfacing errors have been uncovered and corrected and final series of software validation testing begins. Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements.

7.2.4 System Testing

System testing is a level of the software testing process where a complete, integrated system or software is tested. Once all the components are integrated, the application is tested rigorously to see that it meets quality standards. Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if software meets its requirements. Here entire project has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

7.2.5 Output Testing

After performing validation testing, the next steps are output testing of the proposed system since no system could be useful if it does not produce the desired output in the specified format. The output generated are displayed by the system under consideration or tested by asking the user about the format required by them. Here the output format is considered in two ways. One is on the screen and the other is on the printed form.

7.2.6 Acceptance Testing

Acceptance Testing is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. Acceptance test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behaviour of the system, the internal logic of program is not emphasized.

In this project we have collected some data and tested whether project is working correctly or not. Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once. The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met, and the user requirements are satisfied. User acceptance of a system is the key factor for the success of any system. The system under consideration was tested for user acceptance by constantly keeping in touch with the perspective system users at the time of developing and making changes whenever required. This is done about the following points. Input screen design Output screen design Menu driven system

7.3 Test Approach:

Testing can be done in two ways:

1.Bottom up approach

2.Top down approach

Bottom up Approach

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

Top down Approach

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

7.4 Validation

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

7.5 Test cases Results

Testing is the set of activities that can be planned and conducted systematically. The underlying motivation of program testing is to affirm software quality.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

We had a very successful result with an accuracy of 80 % using the above methods. To increase the accuracy to a certain extent, there is an another method called skin segmentation which helps to calculate the skin percentage in a frame which also helps us in taking a step forward for cases like head lowering and attention span of the driver on the road.

For this we binarize the video input and calculate the results as shown in the next few slides.

We must increase accuracy of our current model where we use multithreading to hold various aspects into account at the same time.

Limitations

Objects in the video, should be uniformly illuminated, else results can differ.

Changing distance of person from the camera can cause problems.

Head lowering can give abrupt results.

The algorithm does not work for the people sleeping with eyes open.

Face symmetry calculations are not same for everyone. The calculations considered are true for most of the people.

8.2 Future Scope

8.2.1 SKIN SEGMENTATION

An image which taken inside a vehicle includes the driver’s face. Typically, a camera takes images within the RGB model (Red, Green and Blue). However, the RGB model includes brightness in addition to the colors. When it comes to human’s eyes, different brightness for the same color means different color.

When analyzing a human face, RGB model is very sensitive in image brightness. Therefore, to remove the brightness from the images is second step. We use the YCbCr space since it is widely used in video compression standards.

Since the skin-tone color depends on luminance, we nonlinearly transform the YCbCr color space to make the skin cluster luma-independent. This also enables robust detection of dark and light skin tone colors. The main advantage of converting the image to the YCbCr domain is that influence of luminosity can be removed during our image processing.

In the RGB domain, each component of the picture (red, green and blue) has a different brightness. However, in the YCbCr domain all information about the brightness is given by the Y component, since the Cb (blue) and Cr (red) components are independent from the luminosity.

8.2.2 EYE BINARISATION

After the face is detected using Voila-Jones, the region containing the eyes and mouth must be separated.

To detect the coordinate from where the region of eye is starting certain calculations are done. After the rectangular window is extracted, we have considered that the eyes are located at (0.25 * height of window) from the top and (0.15 * width of window) from the left.

The size of window is (0.25 * height of window) in height and (0.68 * width of window) in width.

After the eyes are cropped the image is converted to YCbCr. The reason for conversion and way to convert is mentioned in “Skin Segmentation” column. Then image is converted to grayscale and ultimately to binary image by setting a threshold of (minimum pixel value + 10).

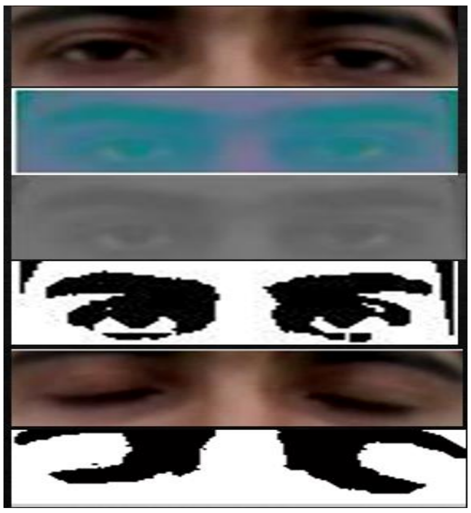

Fig 8.1 Eye Binarization

8.2.3 MOUTH BINARIZATION

After the face is detected using Voila-Jones, the region containing the eyes and mouth must be separated.

After the mouth is cropped the image is converted to YCbCr. The reason for conversion and way to convert is mentioned in “Skin Segmentation” column. Then image is converted to grayscale and ultimately to binary image by setting a threshold of (minimum pixel value + 10).

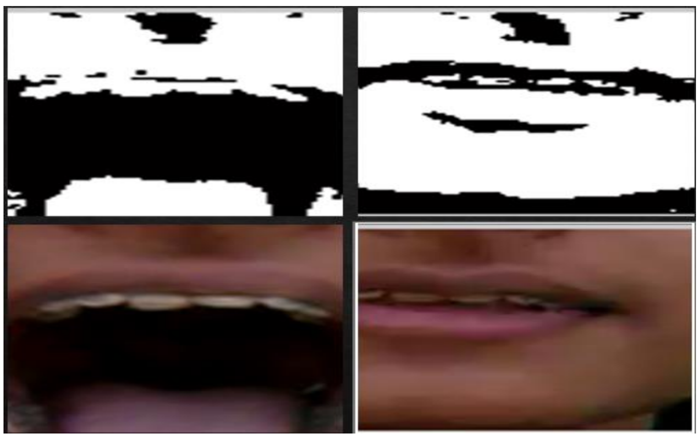

Fig 8.2 Mouth Binarization

CHAPTER 9

BIBLIOGRAPHY

References

[1]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.959.2211&rep=rep1&type=pdf

[2]https://www.ecse.rpi.edu/~cvrl/zhiwei/html/papers/itsc2004.pdf

[3]https://www.researchgate.net/publication/319464008_Driver_Drowsiness_Detection_Systems

[4] G. Hosseini, H. Hossein-Zadeh, A "Display driver drowsiness Warning system", International Conference of the road and traffic accidents, Tehran University, 2006.

[5]L. M Bergasa, J. u. Nuevo, M A. Sotelo, R Barea and E. Lopez, "Visual Monitoring of Driver Inattention," Studies in Computational Intelligence (SCI), 2008.

[6] Viola, Jones: Robust Real-time Object Detection, IJCV 2001 pages 1,3.

[7] C. Zhang, X Lin, R Lu, P.H. Ho, X Shen, "An efficient message authentication scheme for vehicular communications". IEEE Trans Veh TechnoI57(6):3357-3368.2008

[8] S. S. Manv M.S. Kakkasager J. Pitt, "MuItiagent based infonnation dissemination in vehicular ad hoc networks". Mobile Infonn Syst 5(4):363-389.2009.

[9]http://en.wikipedia.org/wiki/Haar-like_features