**DRIVER**

**DROWSINESS DETECTION SYSTEM**

Project Report Submitted in Complete fulfilment

Of The Requirements for the degree

                            Of

**Maulana   Abul   Kalam   Azad   University   of   Technology**

By

Raman Agarwal   Roll no  54, WBUT  Roll No. 10900316078

Romik Roy   Roll  no  49, WBUT  Roll No. 10900316073

Rahul Kumar Verma Roll  no 56, WBUT  Roll No. 10900316080

Shagun Priya Roll  no  37, WBUT  Roll No. 10900316061

Under the guidance of

Prof. Niladri Shekhar Mishra

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

****

**NETAJI SUBHASH ENGINEERING COLLEGE TECHNO CITY, GARIA, KOLKATA – 700 152**

**The academic year of pass out 2019-2020**

**CERTIFICATE**

This is to certify that this project report titled **DRIVER DROWSINESS DETECTION SYSTEM** submitted in complete fulfillment of requirements for the award of the degree Bachelor of Technology (B.Tech) in Electronics & Communication Engineering of Maulana Abul Kalam Azad University of Technology is a faithful record of the original work carried out by,

Raman Agarwal   Roll no  54, WBUT  Roll No. 10900316078

Romik Roy   Roll  no  49, WBUT  Roll No. 10900316073

Rahul Kumar Verma Roll  no 56, WBUT  Roll No. 10900316080

Shagun Priya Roll  no  37, WBUT  Roll No. 10900316061

under   my   guidance   and   supervision.

It is further certified that it contains no material, which to a substantial extent has been submitted for the award of any degree/diploma in any institute or has been published in any form, except the assistance drawn from other sources, for which due acknowledgment has been made.

DATE…………

**Guide’s signature**

**Niladri Shekhar Mishra**

**Head   of   the   Department**

Electronics   &   Communication   Engineering

NETAJI SUBHASH ENGINEERING COLLEGE

TECHNO CITY, GARIA, KOLKATA – 700 152

**DECLARATION**

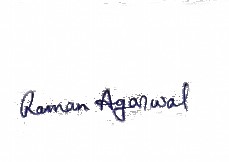
           We hereby declare that this project report titled

**DRIVER DROWSINESS DETECTION SYSTEM**

is our own original work carried out as an undergraduate student in Netaji Subhash Engineering College except to the extent that assistance from other sources is duly acknowledged.

All sources used for this project report have been fully and properly cited. It contains no material which to a substantial extent has been submitted for the award of any degree/diploma in any institute or has been published in any form, except where due acknowledgment is made.

Student’s Names: Signature: Date:

Raman Agarwal  14/06/2020

Romik Roy  14/06/2020

Rahul Kumar Verma WhatsApp Image 2020-05-10 at 1.59.09 PM 14/06/2020

Shagun priya large-white-page-desktop-background (1) 14/06/2020

**CERTIFICATE   OF   APPROVAL**

We   hereby   approve   this   dissertation   titled

**DRIVER DROWSINESS DETECTION SYSTEM**

                                                  Carried out by

Raman Agarwal   Roll no  54, WBUT  Roll No. 10900316078

Romik Roy   Roll  no  49, WBUT  Roll No. 10900316073

Rahul Kumar Verma Roll  no 56, WBUT  Roll No. 10900316080

Shagun Priya Roll  no  37, WBUT  Roll No. 10900316061

                                            under the guidance of

                                Prof. Niladri Shekhar Mishra

of  Netaji  Subhash  Engineering  College,Kolkata in complete fulfillment of requirements for the award of the degree Bachelor of Technology (B.Tech) in Electronics and Communication Engineering of Maulana Abul Kalam Azad University of Technology.

DATE…………..

**Examiner’s Signature :**

**1 ………………………………**

**2 ………………………………**

**3 ………………………………**

**4 ………………………………**

**5 ………………………………**

**Acknowledgement**

We feel thankful that we have been given the opportunity to prepare an insightful project on the topic- **DRIVER DROWSINESS DETECTION SYSTEM**.We would like to express our sincere gratitude towards our teacher prof.Niladri Shekhar Mishra who has guided us continuously and given his valuable time and advice. We extend our appreciation to him for sharing his wisdom with us throughout the course of this project.

This project inculcated in us the sense of team-work and also encouraged us to do deep research which cleared all the bars of doubts and confusion.

Last but not least, we are grateful to everyone who is associated with our project.

Raman Agarwal

Romik Roy

Rahul Kumar Verma

Shagun Priya

Date : 14/06/2020

**ABSTRACT**

In recent years driver fatigue is one of the major causes of vehicle accidents in the world. A direct way of measuring driver fatigue is measuring the state of the driver i.e. drowsiness. So it is very important to detect the drowsiness of the driver to save life and property. This project is aimed towards developing a software of drowsiness detection. This software captures image continuously and measures the state of the eye according to the specified algorithm and gives warning if required.

Though there are several methods for measuring the drowsiness but this approach is completely non-intrusive which does not affect the driver in any way, hence giving the exact condition of the driver. For detection of drowsiness the per closure value of eye is considered. So when the closure of eye exceeds a certain amount then the driver is identified to be sleepy. For developing this software several OpenCv libraries, dlib library, SciPy library, imutils library, python and a laptop webcam are used.

**CONTENTS:**

Page No.

1. CHAPTER 01

1.1 Introduction 1-2

1. CHAPTER 02
   1. Drowsiness 3
   2. Measures for detection of Drowsiness 3-6
   3. Proposed Method 6
   4. Drowsiness detection approach 7
2. CHAPTER 03
   1. Process steps 8
   2. Dividing into frames 8
   3. Face detection 9
   4. Eye detection 9
   5. State of the eye 9-10
   6. Facial landmarks location 10
   7. Understanding dlib facial landmark detector 11
   8. Calculation of eye-aspect-ratio 11-12
3. CHAPTER 04
   1. Programming language and packages description 13

4.1.1 Python 13

4.1.2 NumPy 13

4.1.3 Pygame 13-14

4.1.4 OpenCV 14

4.1.5 SciPy 14

4.1.6 Imutils package 15

1. CHAPTER 05
   1. Source Code 16-18
   2. Source Code summarization 19-22
2. CHAPTER 06 23-25
   1. Output/Result 23-25
3. CHAPTER 07
   1. Conclusion 26-27

7.1.1 Limitations 26-27

7.1.2 Future scopes 27

8. BIBLIOGRAPHY 28

**Chapter 01**

**1.1 Introduction**

The attention level of driver degrade because of less sleep, long continuous driving or any other medical condition like brain disorders etc. Several surveys on road accidents says that around 30 percent of accidents are caused by fatigue of the driver. When driver drives for more than normal period for human then excessive fatigue is caused and also results in tiredness which drives the driver to sleepy condition or loss of consciousness. Drowsiness is a complex phenomenon which states that there is a decrease in alerts and conscious levels of the driver. Though there is no direct measure to detect the drowsiness but several indirect methods can be used for this purpose.

In chapter 2, in initial sections different types of methods for measuring the drowsiness of the driver are mentioned which includes Vehicle based measures, Physiological measures, Behavioral measures. Using those methods an intelligence system can be developed which would alert the driver in case drowsy condition and prevent accidents. Advantages and dis advantages corresponding to each and every system is explained. Depending on advantages and disadvantages the most suitable method is chosen and proposed.

In chapter 3, all theoretical explanation is written line by line that shows the whole process of building the model/software. It is started with  capturing the video through the webcam, dividing the captured video into frames, then the face detection process gets started, after that the eye detection process is initiated automatically, then detecting the state of eyes, with this the facial landmarks are detected and then the calculation of eye aspect ratio is mentioned.

In chapter 4, programming language and the scientific libraries which have been used are written briefly, for building this software we used Python language, SciPy,OpenCV, imutils, NumPy and Pygame libraries.

In chapter 5, The source code and for making the source code understandable we have written the whole source code’s summary. Source code is also the backend of the software.

In chapter 6, the output or result of this model/software is given. The outputs are the images which show the eye is open i.e the person is not in drowsy state and also when the eyes are closed it shows the person is in drowsy state by making an alarm.

In chapter 7, the conclusion, limitations, and the future scopes of the model are written.

**Chapter 02**

**2.1 Drowsiness**

Drowsiness is defined as a decreased level of awareness portrayed by sleepiness and trouble in staying alert but the person awakes with simple excitement by stimuli. It might be caused by an absence of rest, medicine, substance misuse, or a cerebral issue. It is mostly the result of fatigue which can be both mental and physical. Physical fatigue, or muscle weariness, is the temporary physical failure of a muscle to perform ideally. Mental fatigue is a temporary failure to keep up ideal psychological execution. The onset of mental exhaustion amid any intellectual action is progressive, and relies on an individual's psychological capacity, furthermore upon different elements, for example, lack of sleep and general well-being. Mental exhaustion has additionally been appeared to diminish physical performance. It can show as sleepiness, dormancy, or coordinated consideration weakness.

In the past years according to available data driver sleepiness has gotten to be one of the real reasons for street mishaps prompting demise and extreme physical injuries and loss of economy. A driver who falls asleep is in an edge of losing control over the vehicle prompting crash with other vehicle or stationary bodies. Keeping in mind to stop or reduce the number of accidents to a great extent the condition of sleepiness of the driver should be observed continuously.

**2.2 Measures for detection of Drowsiness**

The study states that the reason for a mishap can be categorized as one of the accompanying primary classes: (1) human, (2) vehicular, and (3) surrounding factor. The driver's error represented 91% of the accidents. The other two classes of causative elements were referred to as 4% for the type of vehicle used and 5% for surrounding factors. Several measures are available for the measurement of drowsiness which includes the following:

1. Vehicle based measures.
2. Physiological measures.
3. Behavioral measures .
4. **Vehicle based measures.**

Vehicle-based measures survey path position, which monitors the vehicle's position as it identifies with path markings, to determine driver weakness, and accumulate steering wheel movement information to characterize the fatigue from low level to high level. In many research project, researchers have used this method to detect fatigue, highlighting the continuous nature of this non-intrusive and cost-effective monitoring technique.

This is done by: 1. Sudden deviation of vehicle from lane position. 2.Sudden movement of steering wheels. 3. Pressure on acceleration paddles.

For each measures threshold values are decided which when crossed

indicated that driver is drowsy.

**Advantages:** 1. It is noninvasive in nature.

2. Provides almost accurate result.

**Disadvantages:** 1. Vehicle based measures mostly affected by the geometry of road which sometimes unnecessarily activates the alarming system.

2. The driving style of the current driver needs to be learned and modeled for the system to be efficient.

3. The condition like micro sleeping which mostly happens in straight highways cannot be detected.

1. **Physiological measures.**

Physiological measures are the objective measures of the physical changes that occur in our body because of fatigue. These physiological changes can be simply measure by their respective instruments as follows:

ECG (electro cardiogram)

EMG (electromyogram)

EOG (electro occulogram)

EEG (electroencephalogram)

**Monitoring Heart Rate:** An ECG sensor can be installed in the steering wheel of a car to monitor a driver's pulse, which gives a sign of the driver's level of fatigue indirectly giving the state of drowsiness. Additionally the ECG sensor can be introduced in the back of the seat.

**Monitoring Brain Waves:** Special caps embedded with electrodes measures the brain waves to identify fatigue in drivers and report results in real time. Then each brain waves can be classified accordingly to identify drowsiness.

**Monitoring muscle fatigue:** As muscle fatigue is directly related to drowsiness. We know during fatigue the pressure on the steering wheel reduces and response of several muscle drastically reduces hence it can be measured by installation of pressure sensors at steering wheel or by measuring the muscle response with applied stimuli to detect the fatigue.

**Monitoring eye movements:** Invasive measurement of eye movement and eye closure can be done by using electro occulogram but it will be very uncomfortable for the driver to deal with. Though this method gives the most accurate results regarding drowsiness. But it requires placement of several electrodes to be placed on head, chest and face which is not at all a convenient and annoying for a driver. Also they need to be very carefully placed on respective places for perfect result.

1. **Behavioral measures.**

Certain behavioral changes take place during drowsing like

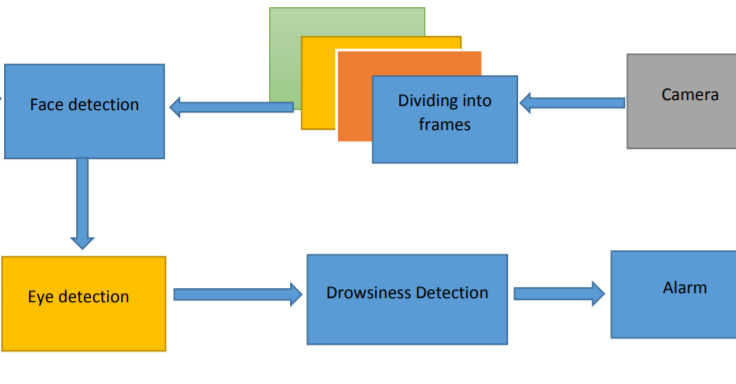
1. Yawning.
2. Amount of eye closure.
3. Eye blinking.
4. Head position.

**2.3 Proposed Method**

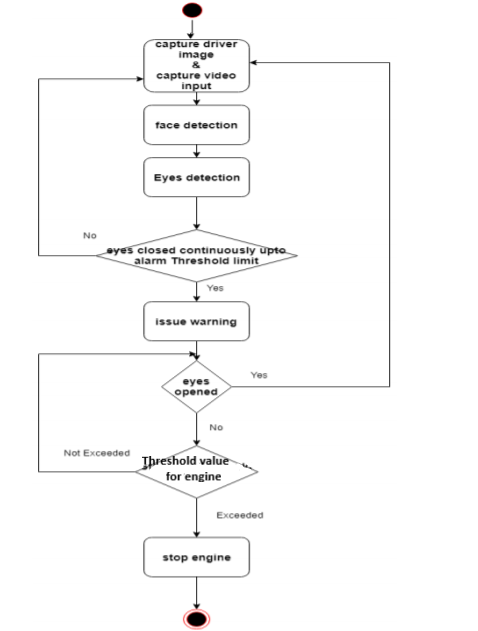
Among all these four strategies, the most precise technique depends on human physiological measures [1]. This procedure is executed in two ways: measuring changes in physiological signs, for example, brain waves, heart rate, and eye flickering; and measuring physical changes, for example, sagging posture, inclining of the driver's head and the open/shut conditions of the eyes [1]. In spite of the fact that this procedure is most precise, it is not reasonable, since detecting electrodes would need to be put straightforward onto the driver's body, and thus be irritating and diverting to the driver. Also, long time driving would bring about sweat on the sensors, reducing their capacity to screen precisely.

Hence this approach will be mostly focusing on amount of eye closure also called **(PERCLOS)** percentage of closure as it provides the most accurate information on drowsiness. It is also non-intrusive in nature, hence does not affect the state of the driver and also the driver feels totally comfortable with this system. Environmental factors like road condition does not affect this system. The case of micro nap is also detected according the given threshold value. The development of this system/software includes face identification and tracking, detection and location of the human eye, human eye tracking, eye state detection, and driver fatigue testing. The key parts of the detection framework fused the detection and location of human eyes and driver fatigue testing. The improved technique for measuring the PERCLOS estimation of the driver was to compute the proportion of the eyes being open and shut with the aggregate number of frames for a given period.

**2.4 Drowsiness detection approach**



**Fig. 1: Process diagram of Nap detection and alert system**

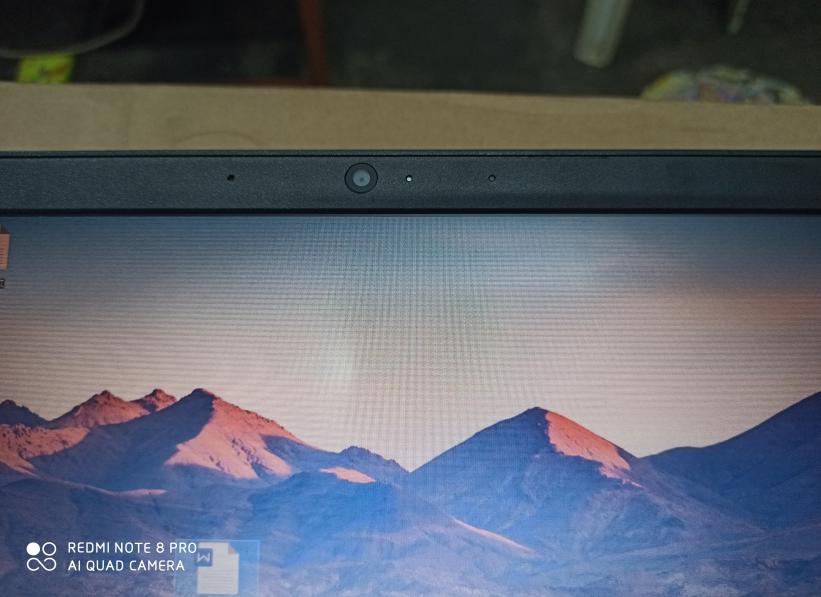
****

**Fig. 2: A Flow chart for Nap detection and alert system.**

**Chapter 03**

**3.1 Process steps:**

Utilizing a web camera introduced inside the automobile we can get the picture of the driver. Despite the fact that the camera creates a video clip, we have to apply the developed algorithm on each edge of the video stream. This paper is only focused on the applying the proposed mechanism only on single frame. But the used camera is an inbuilt web camera with a frame rate of 30 fps in VGA mode. Lenevo laptop web- Camera is used for this process is shown in figure 2.



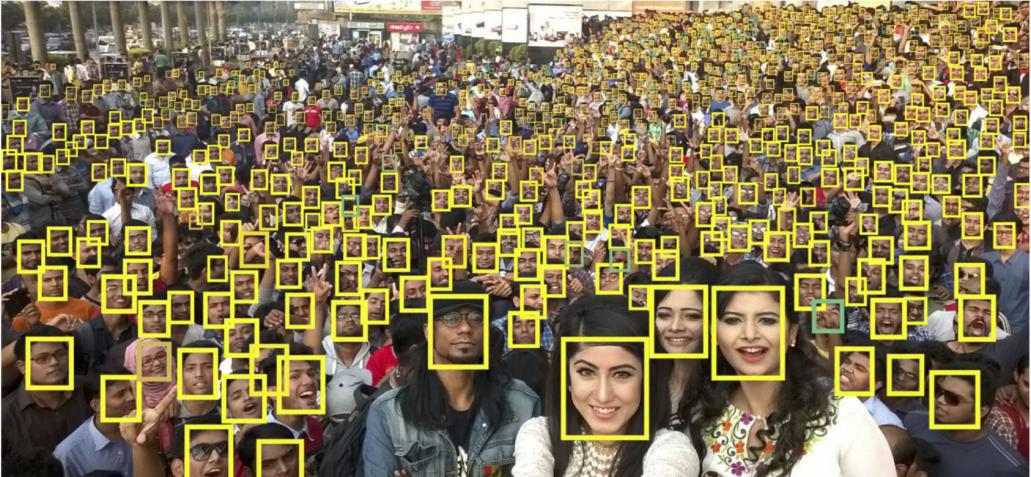
**Figure 2: Camera used for implementing drowsiness detection system**

**3.2 Dividing into Frames:**

We are dealing with real time situation where video is recorded and has to be processed. But the processing or application of algorithm can be done only on an image. Hence the captured video has to be divided into frames for analyzing.

**3.3 Face Detection:**

In this stage we detect the region containing the face of the driver. A specified algorithm is for detection of face in every frame. By face detection we means that locating the face in a frame or in other words finding location of facial characters through a type of technology with the use of computer. The frame may be any random frame. Only facial related structures or features are detected and all others types of objects like buildings, tree, bodies are ignored.



**Figure 3: Faces detection in the given frame ignoring all other objects**

**3.4 Eye Detection:**

After successful detection of face eye needs to be detected for further processing.

In our method eye is the decision parameter for finding the state of driver. Though detection of eye may be easier to locate, but it is really quite complicated. At this point it performs the detection of eye in the required particular region with the use of detection of several features. When eye detection is done then the result is matched with the reference or threshold value for deciding the state of the driver.

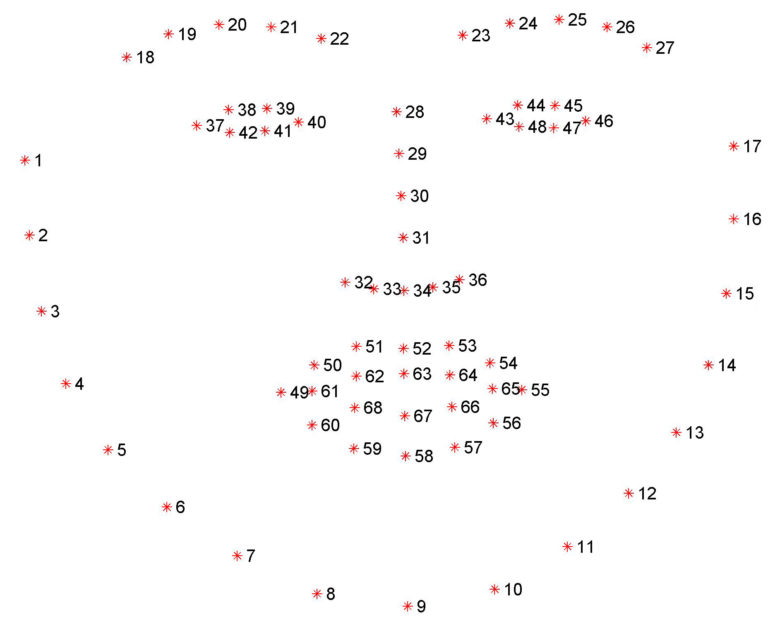
**3.5 State of eye:**

In this stage, we find the actual state of the eye that if it is closed or open or semi closed or open. The identification of eyes status is most important requirement. We channelize a warning message if we obtain that the eyes are in open state or semi open state up to a particular threshold value. If the system detects that the eyes are open then the steps are repeated again and again until it finds a closed eye.

**3.6 Facial landmarks location:**

Once the face gets detected, then the algorithm locates all the important features of the face. Facial landmarks are used to localize and represent salient regions of the face, such as

* Eyes
* Eyebrows
* Nose 
* Mouth
* Jawline



**Figure 5: Facial landmarks around the face.**

**3.7 Understanding dlib facial landmark detector :**

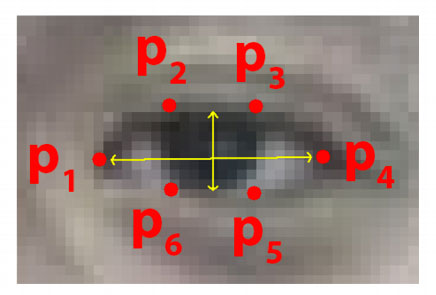
The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x,y)- coordinate that map to facial structures on the face as shown in figure 5.

The program uses a facial training set to understand where certain points exist on facial structures. The program then plots the same points on the region of interests in other images, if they exist. The program uses priors to estimate the probable distance between key points.

These annotations are part of the 68 points dataset which the dlib facial landmark predictor was trained on. The dlib face landmark detector will return a shape object containing the 68 *(x, y)*-coordinates of the facial landmark regions.

**3.8 Calculation of Eye-Aspect-ratio:**

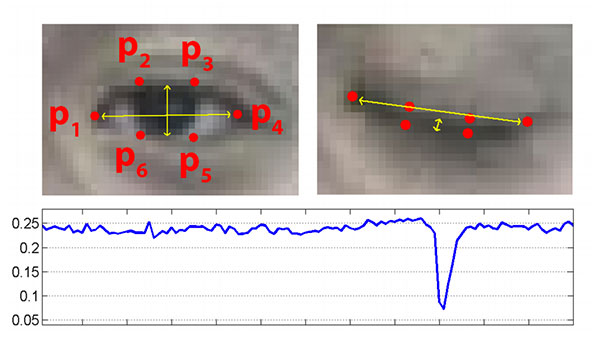
The Eye Aspect Ratio is a constant value when the eye is open, but rapidly falls to 0 when the **eye** is closed. For eye blinks we need to pay attention to points 37-46, the points that describe the eyes.



**Figure 6 : Eye facial landmarks**

The equation to find out the eye aspect ratio=

𝐸𝐴𝑅= ||𝑝2−𝑝6||+||𝑝3−𝑝5|| / |2||𝑝1−𝑝4||



**Figure 7 : A person’s Eye Aspect ratio over time**

A program can determine, if a person’s eyes are closed and if the Eye Aspect Ratio falls below a certain threshold. If these conditions are met, an alarm is issued continuously until the driver opens his eyes, which means he is awake. Even after alerting the driver through the alarm, if he is not responding and sleep threshold limit is exceeded then we are going to stop the vehicle after giving the signal to vehicles around but in this project we are focusing on the alarm alert.

Thereby, as drowsiness of the driver is detected and produced an alert in the initial stage itself, helps in minimizing the road accidents caused due to drowsiness of the driver.

**Chapter 04**

* 1. **Programming language and packages**

**Description**

**4.1.1 Python :**

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

**4.1.2 NumPy :**

NumPy is a general-purpose array-processing package and is the fundamental package needed for scientific computing with Python.. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones-

* A powerful N-dimensional array object.
* Sophisticated (broadcasting) functions.
* Tools for integrating C/C++ and Fortran code.
* Useful linear algebra, Fourier transform, and random number capabilities.

**4.1.3 Pygame**

**Pygame** is a set of Python modules designed for writing video games and allows you to create fully featured games and multimedia programs in the python language.

It consists of computer graphics and sound libraries designed to be used with the Python programming language.

Pygame is highly portable and runs on nearly every platform and operating system.

In this project we used Pygame for implementing the alarm sound.

**4.1.4 OpenCV**

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today’s systems.

By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

Applications of OpenCV:-

* face recognition
* Vehicle counting on highways along with their speeds
* Street view image stitching
* Video/image search and retrieval
* Robot and driver-less car navigation and control
* object recognition
* Medical image analysis
* TV Channels advertisement recognition

**4.1.5 SciPy**

SciPy, a scientific library for Python is an open source, for mathematics, science and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The main reason for building the SciPy library is that, it should work with NumPy arrays. It provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization

In this project we are using this library for the calculating the Euclidean distance between facial landmarks points in the eye aspect ratio calculation.

**4.1.6 Imutils package**

Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

**Chapter 05**

**5.1 Source Code :**

***#Import necessary libraries***

***from* scipy.spatial *import* distance**

***from* imutils *import* face\_utils**

***import* numpy *as* np**

***import* pygame *#For playing sound***

***import* time**

***import* dlib**

***import* cv2**

***#Initialize Pygame and load music***

**pygame.mixer.init()**

**pygame.mixer.music.load('audio/alert.wav')**

***#Minimum threshold of eye aspect ratio below which alarm is  triggerd***

**EYE\_ASPECT\_RATIO\_THRESHOLD = 0.3**

***#Minimum consecutive frames for which eye ratio is below threshold for alarm to be triggered***

**EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES = 15**

***#Counts no. of consecutive frames below threshold value***

**COUNTER = 0**

***#Load face cascade which will be used to draw a rectangle around detected faces.***

**face\_cascade = cv2.CascadeClassifier("haarcascades/haarcascade\_frontalface\_default.xml")**

***#This function calculates and return eye aspect ratio***

**def eye\_aspect\_ratio(eye):**

**A = distance.euclidean(eye[1], eye[5])**

**B = distance.euclidean(eye[2], eye[4])**

**C = distance.euclidean(eye[0], eye[3])**

**ear = (A+B) / (2\*C)**

***return* ear**

***#Load face detector and predictor, uses dlib shape predictor file***

**detector = dlib.get\_frontal\_face\_detector()**

**predictor = dlib.shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat')**

***#Extract indexes of facial landmarks for the left and right eye***

**(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS['left\_eye']**

**(rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS['right\_eye']**

***#Start webcam video capture***

**video\_capture = cv2.VideoCapture(0)**

***#Give some time for camera to initialize(not required)***

**time.sleep(2)**

***while*(True):**

***#Read each frame and flip it, and convert to grayscale***

**ret, frame = video\_capture.read()**

**frame = cv2.flip(frame,1)**

**gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)**

***#Detect facial points through detector function***

**faces = detector(gray, 0)**

***#Detect faces through haarcascade\_frontalface\_default.xml***

**face\_rectangle = face\_cascade.detectMultiScale(gray, 1.3, 5)**

***#Draw rectangle around each face detected***

***for* (x,y,w,h) *in* face\_rectangle:**

**cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)**

***#Detect facial points***

***for* face *in* faces:**

**shape = predictor(gray, face)**

**shape = face\_utils.shape\_to\_np(shape)**

***#Get array of coordinates of leftEye and rightEye***

**leftEye = shape[lStart:lEnd]**

**rightEye = shape[rStart:rEnd]**

***#Calculate aspect ratio of both eyes***

**leftEyeAspectRatio = eye\_aspect\_ratio(leftEye)**

**rightEyeAspectRatio = eye\_aspect\_ratio(rightEye)**

**eyeAspectRatio = (leftEyeAspectRatio + rightEyeAspectRatio) / 2**

***#Use hull to remove convex contour discrepancies and***

***draw eye shape around 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)**

***#Detect if eye aspect ratio is less than threshold***

***if*(eyeAspectRatio < EYE\_ASPECT\_RATIO\_THRESHOLD):**

**COUNTER += 1**

***#If no. of frames is greater than threshold frames,***

***if* COUNTER >= EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES:**

**pygame.mixer.music.play(-1)**

**cv2.putText(frame, "You are Drowsy", (150,200), cv2.FONT\_HERSHEY\_SIMPLEX, 1.5, (0,0,255), 2)**

***else*:**

**pygame.mixer.music.stop()**

**COUNTER = 0**

***#Show video feed***

**cv2.imshow('Video', frame)**

***if*(cv2.waitKey(1) & 0xFF == ord('q')):**

***break***

***#Finally when video capture is over, release the video capture  and destroyAllWindows***

**video\_capture.release()**

**cv2.destroyAllWindows()**

**5.2 Source Code summarization :-**

The above code detects whether a person is drowsy or not using webcam video feed as input and performing eye aspect ratio calculations.

Necessary inbuilt libraries are imported which is used for various different functions.

**Pygame** is a cross platform set of python modules. It includes sound libraries designed to be used with the python programming language. We have initialized the pygame and loaded that music. The music works as an alarm for the person in the drowsiness.

We have set a threshold eye aspect ratio. If the eye aspect ratio goes below the threshold then the alarm is triggered. We have also initialized eye aspect ratio consecutive frames which defines the minimum number of times the eye aspect ratio is below the threshold. A variable counter is taken which counts the total number of consecutive frames below the threshold.

Then, we use a cascade classifier. It is an effective object detection method. Here in the parenthesis we have sent the data of the haarcascade frontal face. It is used to draw a rectangle around the detected face.

A function called eye\_aspect\_ratio is used to calculate the eye aspect ratio. This is done by calculating the euclidean distance between the x and y coordinates of the eye.

**Detector** is used to detect the face of a person.

Shape **predictors**, also called landmark predictors, are used to predict key (x, y)-coordinates of a given “shape”. The most common, well-known shape predictor is dlib's facial landmark predictor used to localize individual facial structures, including the eyes.

Indexes are extracted for the **facial landmarks** of the left and right eye. Advantages of using facial landmarks, especially for face alignment, face swapping, and extracting various facial structures.

Video\_capture is used for capturing video from cameras or for reading video files and image sequences. To capture a video, you need to create a **VideoCapture** object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected (as in our case). So we simply pass 0.

We have used the **python sleep** function to halt the execution of the program for a given **time** in seconds so that the camera which we use gets the time to initialize.

Each frame is converted to **gray scale**. Color video file size is large enough while uploading to the server in the current bandwidth available. So, file size is reduced by converting it to grayscale video in the OpenCV.

**cv2.rectangle()** method is used to draw a rectangle on any image.

Syntax: cv2.rectangle(image,start\_point,end\_point,color,thickness)

Parameters:

Image: It is the image on which rectangle is to be drawn.

Start\_point: It is the starting coordinates of the rectangle. The coordinates are represented as tuples of two values i.e. (X coordinate value, Y coordinate value).

End\_point: It is the ending coordinates of the rectangle. The coordinates are represented as tuples of two values i.e. (X coordinate value, Y coordinate value).

Color: It is the color of the border line of the rectangle to be drawn. For BGR, we pass a tuple. Eg: (255,0,0) for blue color.

Thickness: It is the thickness of the rectangle border line in px. Thickness of -1 px will fill the rectangle shape by the specified color.

We detect the facial points and we fetch the coordinates of the left eye and right eye. Eye aspect ratio of both left and right eye is detected. The resultant **eye aspect ratio** is the average of left eye aspect ratio and right eye aspect ratio.

We use the **hull** to remove the convex contour discrepancies and draw eye shape around the eyes. Hull means the exterior or the shape of the object. The convex hull of a shape or a group of points is a tight fitting convex boundary around the points.

To draw the contours, **cv2.drawContours** function is used. It can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc.

We check the **condition**, if eye aspect ratio is less than eye aspect ratio threshold then counter is incremented by 1. Now, if the counter is greater than equal to eye aspect ratio consecutive frames then alarm alert is triggered.

**cv2.putText()** method is used to draw a string on any image.

Syntax: cv2.putText(image,text,org,font,fontScale,color,[thickness,[lineType[bottomLeftOrigin]]])

Parameters:-

Image: It is the image on which rectangle is to be drawn.

text : Text string to be drawn.

Org: It is the coordinates of the bottom-left corner of the text string in the image. The coordinates are represented as tuples of two values i.e. (X coordinate value, Y coordinate value).

Font: It denotes the font type. Some of the font types are FONT\_HERSHEY\_SIMPLEX, FONT\_HERSHEY\_PLAIN, etc.

Font scale: Font scale factor that is multiplied by the font-specific base size.

Color: It is the color of text string to be drawn. For BGR, we pass a tuple. Eg: (255,0,0) for blue color.

Thickness: It is the thickness of the line in px.

lineType: This is an optional parameter. It gives the type of the line to be used.

bottomLeftOrigin: This is an optional parameter. When it is true, the image data origin is at the bottom-left corner. Otherwise, it is at the top-left corner.

**cv2.imshow()** method is used to display an image in a window.

Syntax: cv2.imshow(window\_name,image)

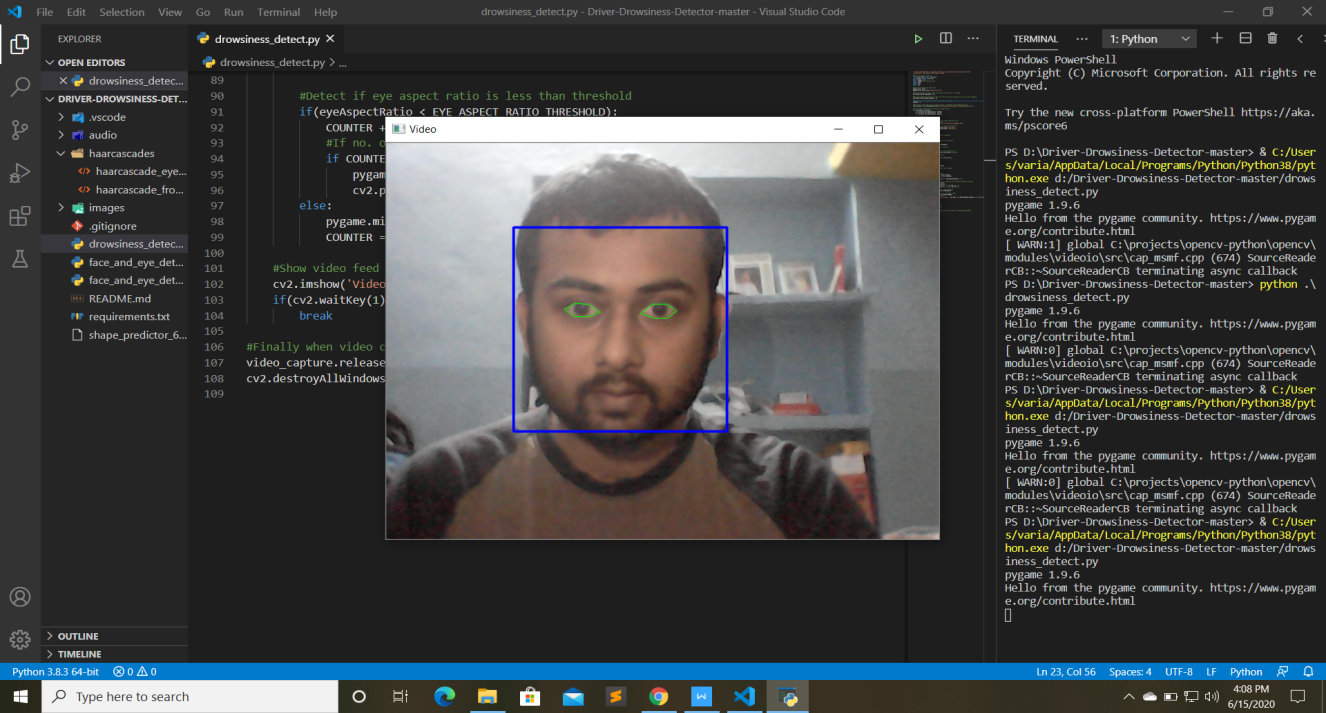
Parameter:

Window\_name: A string representing the name of the window in which image to be displayed.

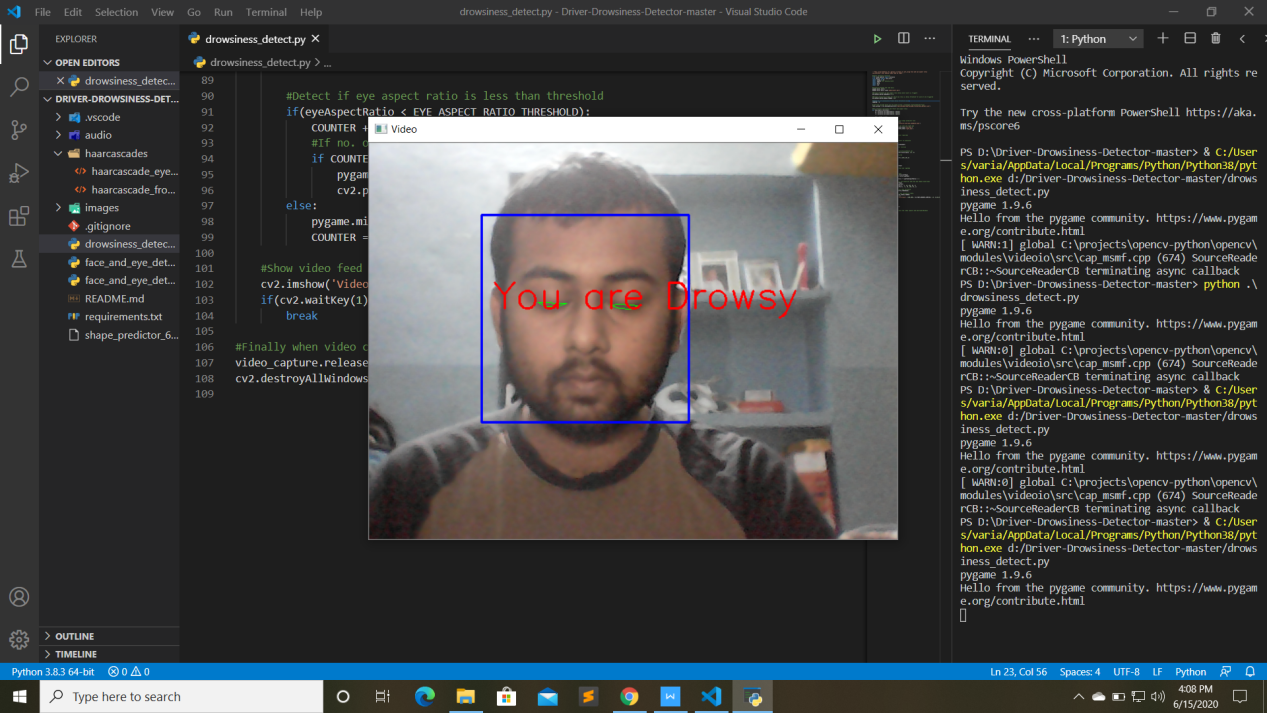
Image: It is the image to be displayed.

**Chapter 06**

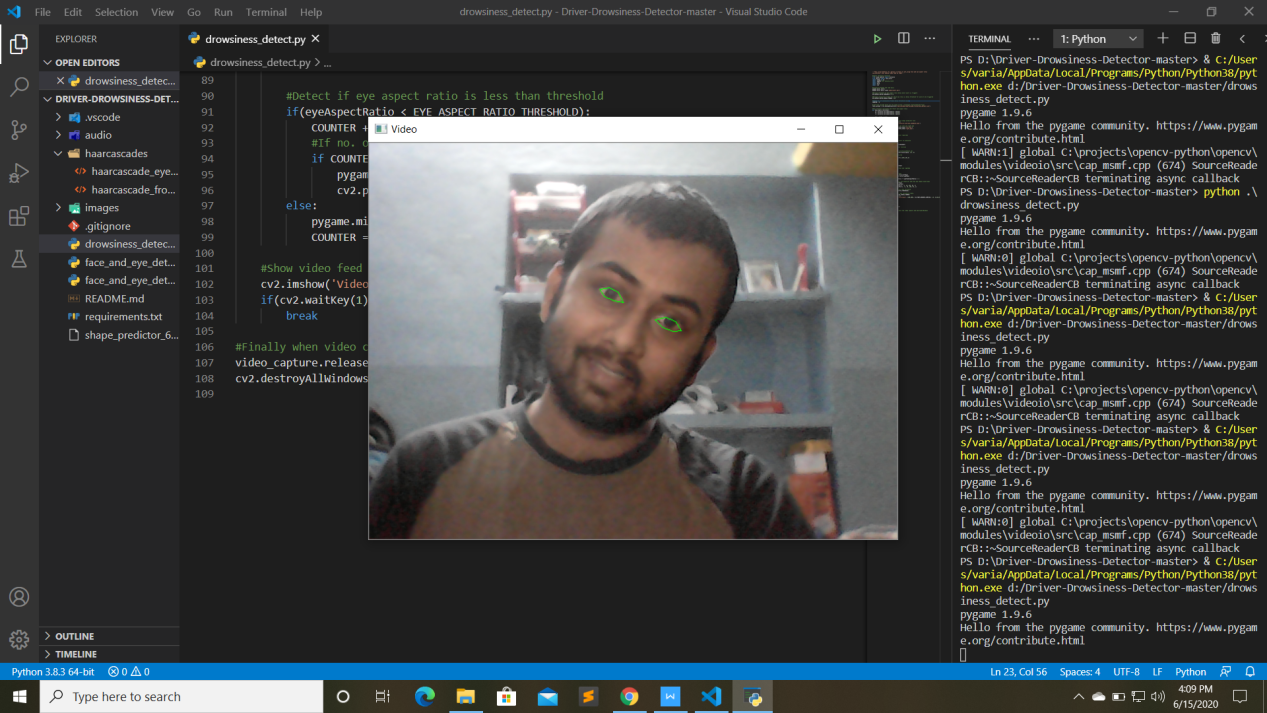
**6.1 Output/Result :-**

****

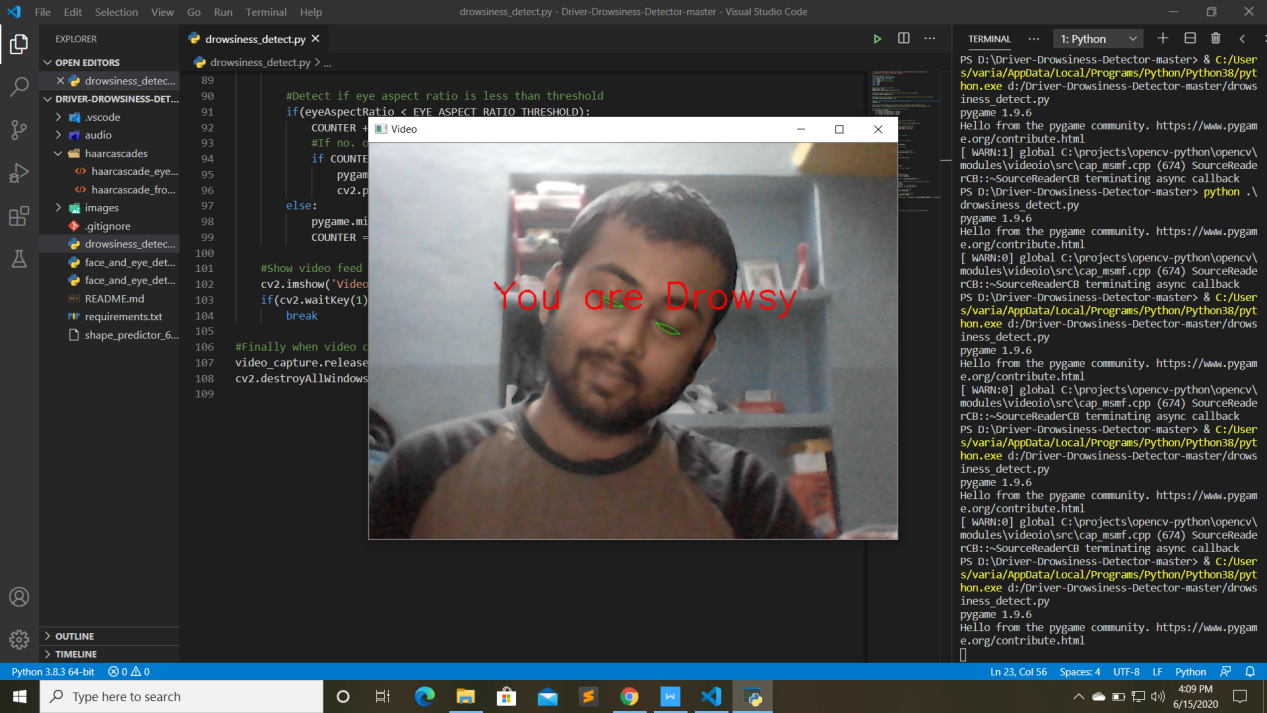
**Figure 8: Eye in open state with head position = straight.**



**Figure 9: Eye in closed state with head position= straight.**



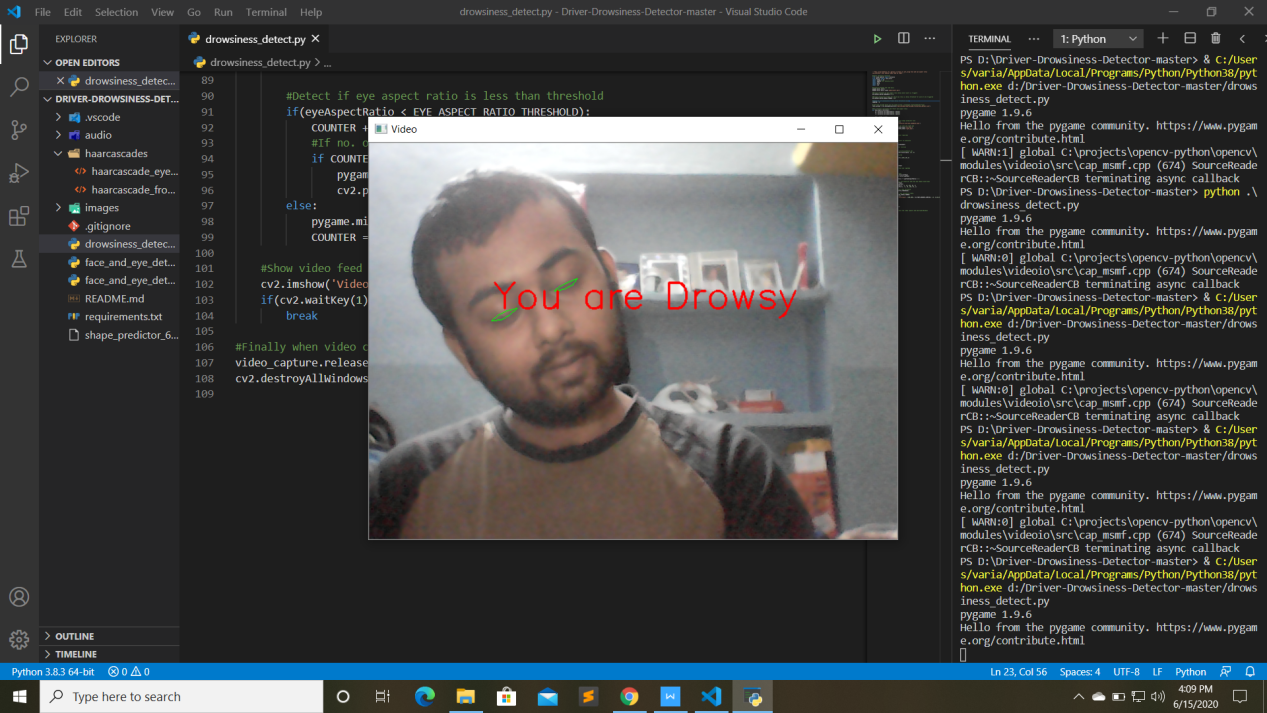
**Figure 10: Eye in open state with head position= Tilted (Right)**



**Figure 11: Eye in closed state with head position= Tilted (Right).**



**Figure 12: Eye in open state with head position= Tilted (Left).**



**Figure 13: Eye in closed state with head position= Tilted (Left).**

**Chapter 07**

**7.1 Conclusion :**

Implementation of drowsiness detection with OpenCV, dlib, Scipy, imutils libraries and all the coding platform is python programming was done which includes the following steps: Successful runtime capturing of video with camera.

Captured video was divided into frames and each frames were analyzed. Successful detection of face followed by detection of eye. If closure of eye for successive frames were detected then it is classified as drowsy condition else it is regarded as normal blink and the loop of capturing image and analyzing the state of driver is carried out again and again.

In this implementation during the drowsy state a message is shown to the screen and an alarm is activated. If the driver is not drowsy then the

message is not shown to the screen and the alarm is not activated.

**7.1.1 Limitations :**

**Dependence on ambient light:** The software developed for this purpose strongly depends on the ambient light condition or depends on certain minimum level of light condition otherwise it becomes very difficult to detect. To avoid this error we can use either LED light for better detection or we can use an infrared camera.

**Distance of camera from driver face:** For best result we have assumed and designed the code according to the fact that the distance between camera and face should be nearly close to each other. Hence the designed set up output may vary from vehicle to vehicle as different vehicle have different types of seat lengths.

**Use of spectacles:** In case the user uses spectacle then it is difficult to detect the state of the eye. As it hugely depends on light hence reflection of spectacles may give the output for a closed eye as opened eye. Hence for this purpose the closeness of eye to the camera is required to avoid light.

**Multiple face problem:** If multiple face arise in the window then the camera may detect more number of faces undesired output may appear. Because of different condition of different faces. So we need to make sure that only the driver face come within the range of the camera. Also the speed of detection reduces because of operation on multiple faces.

**7.1.2 Future scopes :-**

Our model/software is designed for detection of drowsy state of eye and give and alert signal or warning may be in the form of audio or any other means. But the response of driver after being warned may not be sufficient enough to stop causing the accident meaning that if the driver is slow in responding towards the warning signal then accident may occur. Hence to avoid this we can design and fit a motor driven system and synchronize it with the warning signal so that the vehicle will slow down after getting the warning signal automatically.

There following are couple of recommended areas for future

enhancements:

• **Standalone product:** It can be implemented as a standalone product, which can be installed in an automobile for monitoring the automobile driver.

• **Smart phone application:** It can be implemented as a smart phone application, which can be installed on smart phones. And the automobile driver can start the application after placing it at a position where the camera is focused on the driver.

**Bibliography**

[1] Karamjeet Singh, Rupinder Kaur, ”Physical and Physiological Drowsiness Detection Methods”, IJIEASR, pp.35-43, vol.2,2013.

[2] Jarek Krajewski, David Sommer, ”Steering Wheel Behavior Based Estimation Of Fatigue”, Fifth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design, Germany.

[3] Behnoosh Hariri, Shabnam Abtahi, Shervin Shirmohammadi, Luc Martel,” A Yawning Measurement Method to Detect Driver Drowsiness “.Vandna Saini et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014, 4245-4249

[4] K. C. Yowand, R. Cipolla, “Feature-based human face detection, “Image Vision Comput., vol.15, no.9, 1997, pp.713–735.

1. Xia Liu; Fengliang Xu; Fujimura, K., "Real-time eye detection an

tracking for driver observation under various light conditions,"

Intelligent Vehicle Symposium, 2002. IEEE , vol.2, no.,

pp.344,351 vol.2, 17-21 June 2002.

1. Xianghua Fan, et.al, "The system of face detection based on OpenCV," Control and Decision Conference (CCDC), 2012 24th

Chinese , vol., no., pp.648,651, 23-25 May 2012