Chapter 1

Introduction

Driver Drowsiness detection is the task of identifying a drowsy driver and generating an alert to indicate the drowsiness to avoid accidents. After drunk driving, driver's fatigue is the second largest cause behind the accidents. Thereby, this is an important issue that needs to be addressed. The proposed work intends to develop a feasible, fast and accurate system that detects driver drowsiness as well as driver distraction and sets off an alarm if the driver is found to be drowsy or distracted, thereby mitigating road accidents.

1.1 Description

The system uses live video stream from a camera while the model detects facial features like eye aspect ratio (EAR), mouth aspect ratio (MAR), yawning, head tilts to monitor driver's state etc to yield an output. The approach is essentially based on OpenCV and Machine Learning, implemented on embedded systems so as to depict real life scenarios. This work makes use of OpenCVs Dlib library that makes use of Haar-like features to detect facial coordinates. These are further used to calculate the EAR and MAR. When the values surpass the thresholds of any of the respective parameters, an alarm is generated.

1.2 Problem Formulation

As of the current scenario, the advancement in technology has led to significant advances in the automobile industry- from cruise control and autonomous driving cars to post-crash help for the victims. But a little attention has been paid towards taking steps to ensure accidents from taking place in the first case. Some solutions to this problem do exist but are found to be mostly in the high end luxury vehicles thus being far from the reach of a normal human being. In India, approximately five lakh people are victims of road accidents every year. Predictably, 78% of the accidents owe their cause to the driver's negligence. Digging deeper, after drunk driving, driver's fatigue is the second largest cause behind the accidents. Thereby, this is an important issue that needs to be addressed. This proposed work aims to build a Drowsiness detection model that will determine whether a driver is drowsy or distracted (A driver is said to be distracted if his eyes are not on the road). It will generate an alarm to alert the user in case he is drowsy or distracted and will generate no signal in the second case. The system intends to provide a fast, cost-effective and viable solution to the problem in hand.

1.3 Motivation

Every year roughly of the five lakh Indians involved in road accidents, a mighty 1.2 lakh lose their lives. Although the data available does not explicitly mention the number of accidents due to drowsiness of the driver; it is believed to be the most prominent one behind drunk driving incidents. The effects of drowsiness are similar to alcohol - it will make your driving inputs (steering, acceleration, braking) poorer, destroy your reaction times and blur your thought processes. The AAA Foundation for Traffic Safety says that 20% of all fatal accidents in the USA are due to drowsiness![5] One can only imagine what the statistics are like for India which has a higher road accident rate.

Research shows that drivers who are deprived of more than four hours of sleep are 10.2 times more accident prone. Some say that if one has been awake for more than 24 hours, he is a more dangerous driver than a drunk one. Essentially, one could nod off without realising it if he/she is terribly fatigued. Moreover, there's no way of catching sleepy drivers (like how cops nab drunks with breathalysers). No one has ever been put behind the bars for drowsy driving. On a pertinent note, 9 out of 10 USA cops admitted to pulling over drivers on suspicion of drunk driving, but the drivers were actually drowsy. NHTSA's (The National Highway Traffic Safety Administration) census of fatal crashes and estimates of traffic-related crashes and injuries rely on police and hospital reports to determine the incidence of drowsy-driving crashes. NHTSA estimates that in 2017, 91,000 police-reported crashes involved drowsy drivers. These crashes led to an estimated 50,000 people injured and nearly 800 deaths.[4] But there is broad agreement across the traffic safety, sleep science, and public health communities that this is an underestimate of the impact of drowsy driving.

The development of drowsiness detection technologies is both an industrial and academic challenge. Particularly in India, the automobile industry has been rather negligent in providing sufficient safety measures in their vehicles.

1.4 Proposed Solution

The system will take live video feed of the driver as the input. Next the videoframes will be converted into images and processed further for analysis. The frontal face detector of the dlib library is then applied on the image to detect the face. It returns a 68 set of points on the face. These points will then be used to get the coordinates of eyes and mouth. Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) will then be calculated from these points. If these ratios cross a certain threshold then the results will be generated accordingly. If the driver is found to be drowsy or distracted then an alarm will be generated thus alerting the driver.

1.5 Scope of the Project

Once the system is up and running, it will be successfully able to examine the state of a driver and if the driver is found to be in a drowsy or distracted state it will alert the driver by setting off an alarm. The system looks to provide a safe driving experience for the driver as well as the pedestrians with an aim to reduce the number of accidents caused due to the negligence of the driver. The system can be further developed by introducing a module that aims to prevent drunk drivers from driving. Another addition to the system can be to integrate it with a post-crash help system that alerts the nearby police stations, hospitals and the family members of the driver in case of an accident.

Chapter 2

Literature Review

Drowsiness detection can be divided into three categories:

1. Vehicle based
2. Behavioral based
3. Physiological based

Vehicle based measures include a number of metrics including deviations from lane position, movement of the steering wheel, pressure on the acceleration pedal, etc. are constantly monitored and any change in these that crosses a specified threshold indicates a significantly increased probability that the driver is drowsy.

Behavioral based measures including yawning, eye closure, eye blinking, head pose, etc. is monitored through a camera and the driver is alerted if any of these drowsiness symptoms are detected.

Physiological based measures include the correlation between physiological signals ECG (Electrocardiogram) and EOG (Electrooculogram). Drowsiness is detected through pulse rate, heartbeat and brain information.

In this project, we intend to focus on the behavioral aspect of drowsiness detection. The literature review on the same is studied and referred for this project.

Viola P, Michael Jones divide the analysis into three stages. The first stage makes use of an image representation called Integral which allows our detector to compute images easily. The second stage is based on Adaboost which makes use of very few critical features from a larger set and provides extremely efficient classifiers. The third stage combines more complex classifiers as cascades which discards the background region of the image. The detector runs at 15 frames per second without resorting to image differencing or skin color detection. This method is much more efficient as compared to other algorithms since it is able to detect faces rapidly [1].

Ching-Hua Weng, Ying-Hsiu Lai, Shang-Hong Lai make use of this methodology which uses two continuous Hidden Markov Models constructed on top of the Deep Belief

Network.[2] It extracts high-level facial and head feature representations to recognize drowsiness-related symptoms. It makes use of Baum-Welch and Forward-Backward Algorithm which provides an accuracy of 85.39%.

Sanghyuk Park, Fei Pan, Sunghun Kang and Chang D. Yoo makes use of a layout which consists of three layers.[3] AlexNet, VGG-Face Net, Flow Image Net. Alex Net is tuned to learn facial features during drowsiness. VGG Face Net is trained to discover facial features. Flow image net examines behavioural features. The layers are Independently tuned and then averaged for final classification by SoftMax classifier. The two different fusion strategies applied - **Independently Averaged Architecture (IAA)** where the probability distribution is integrated and averaged. **Feature Fused Architecture (FFA)** – integrated such that the fc7 layer is concatenated and based on this, the video is classified as per the four classes using SVM. The accuracy achieved by DDD-IAA is 73.06% and DDD-FFA (70.81%)

Chapter 3

System Analysis

3.1 Functional Requirements:

3.1.1 Interface Requirements:

1. Input video of a driver.

3.1.2 Business Requirements:

1. The camera must be well adjusted and the driver must be in frame.
2. Clicking the start button must initiate the detection mechanism.

3.2 NonFunctional Requirements:

3.2.1. Performance

The computer running the backend modules must have a powerful CPU or GPU for processing the input fast, and generating quick responses, so that the user won't experience any delay.

3.2.2. Reliability

System should receive the input without any error, and should recognize the face and extract the entity features successfully. While continuously performing face feature detection and analysis, the model should generate the response such that it determines whether the driver is drowsy or not through alarm.

3.2.3. Usability

The system is easy to handle and navigates in the most expected way with almost no delays. The system interacts with the user in a very friendly manner making it easier for the users to use the system.

3.3 Hardware and Software Requirements

3.3.1 Software requirements:

1. Anaconda 3/ Google Colab

2. Python 3

3.3.2 Minimum Hardware Requirements:

1. Logitech camera recording at 720p.
2. After the model is developed and fitted by using facial landmark coordination, it will be evaluated by a computer with the following minimum requirements: Intel Core i7-7500U, 8 GB RAM, Intel GMA HD 2 GB.

3.4 Use-Case Diagram and Description

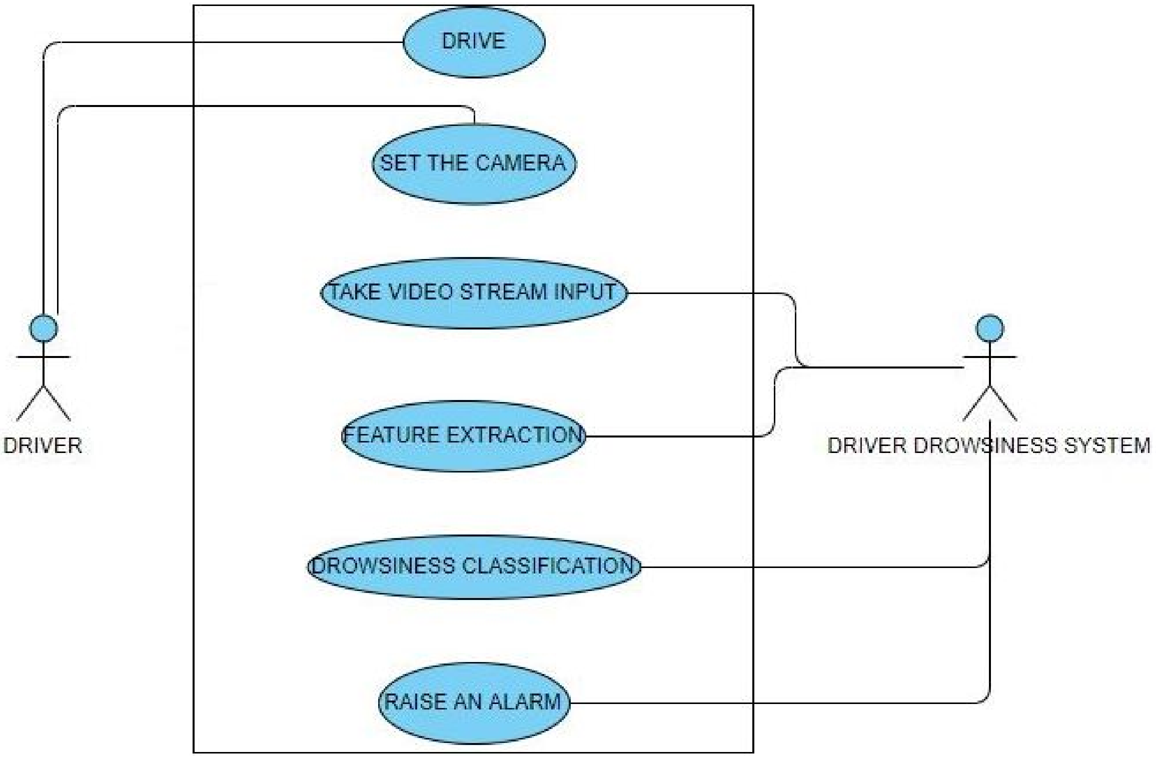


Fig 3.4 Use Case Diagram

Description : To show that the system enable driver drowsiness classification

Summary : The ultimate goal of the driver drowsiness system is to determine whether the Driver is drowsy or not based on the input video stream.

Table 3.4.1Use case description for driver.

Actor	Driver
Brief description	The driver is the subject for classification. His video stream is the input served to the system based on which drowsiness classification will be performed.
Preconditions	The user must set the system camera perfectly in order to provide it the best frame.
Postconditions	If the driver is drowsy, he will be warned with an alarm so that he stays alert.
Flow of events	1. Driver sets the camera 2. The system takes video stream input 3. System performs drowsiness classification 4. Driver is alerted accordingly with an alarm
Priority	Driver has the highest priority
Non behavioral requirements	Stable camera
Assumptions	The driver drives authentically without misrepresenting his facial features.
Alternatively flows and exceptions	There are currently no parameters values uploaded.

Table 3.4.2 Use case description for driver drowsiness detection

Actor	Driver Drowsiness System
Description	The system essentially classifies the driver as drowsy or not and generates and alarm.
Non behavioral requirements	Stable camera on desktop
Preconditions	The system must be initially trained with the training data set of video streams. The trained model then will analyse the driver's video stream for classification.
Postconditions	After the alert is generated the system, the system goes back.

.

Chapter 4

Analysis Modeling

4.1 Activity diagram

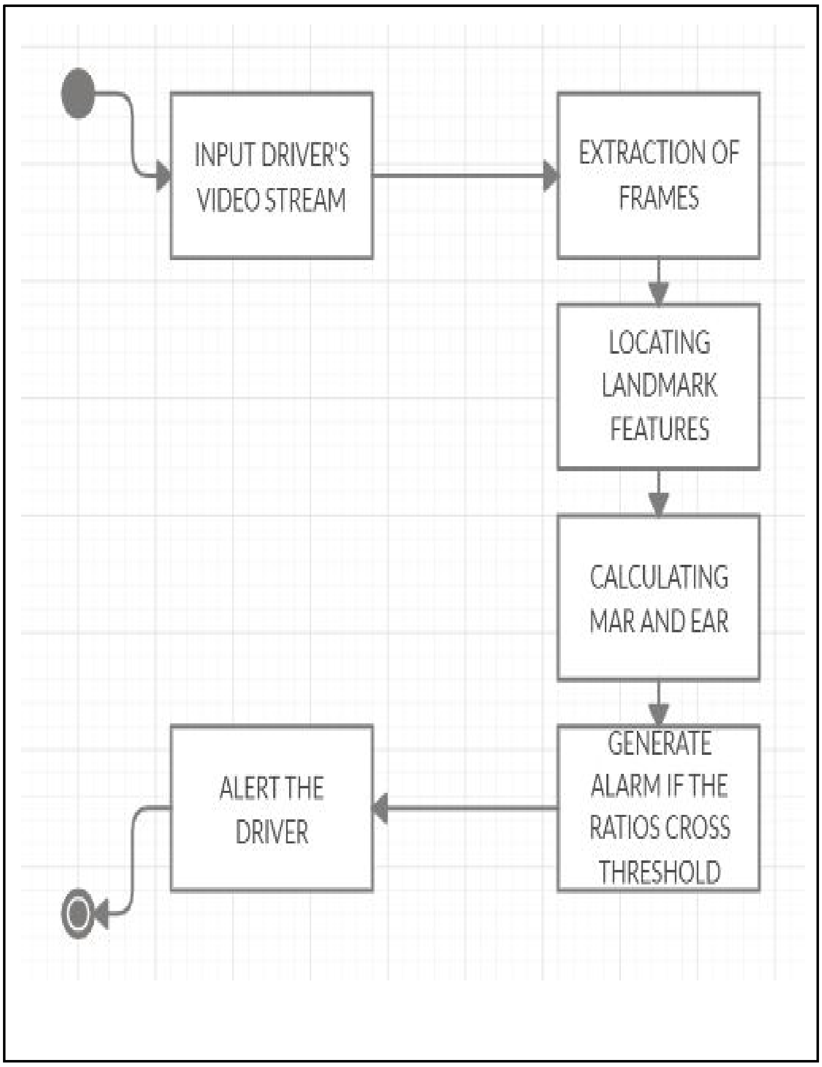

INPUT DRIVER'S VIDEO STREAM
EXTRACTION OF FRAMES
LOCATING LANDMARK FEATURES
CALCULATING MAR AND EAR
GENERATE ALARM IF THE RATIOS CROSS THRESHOLD
ALERT THE DRIVER

Fig 4.1. Activity Diagram for Drowsiness Detection.

4.2 Class diagram

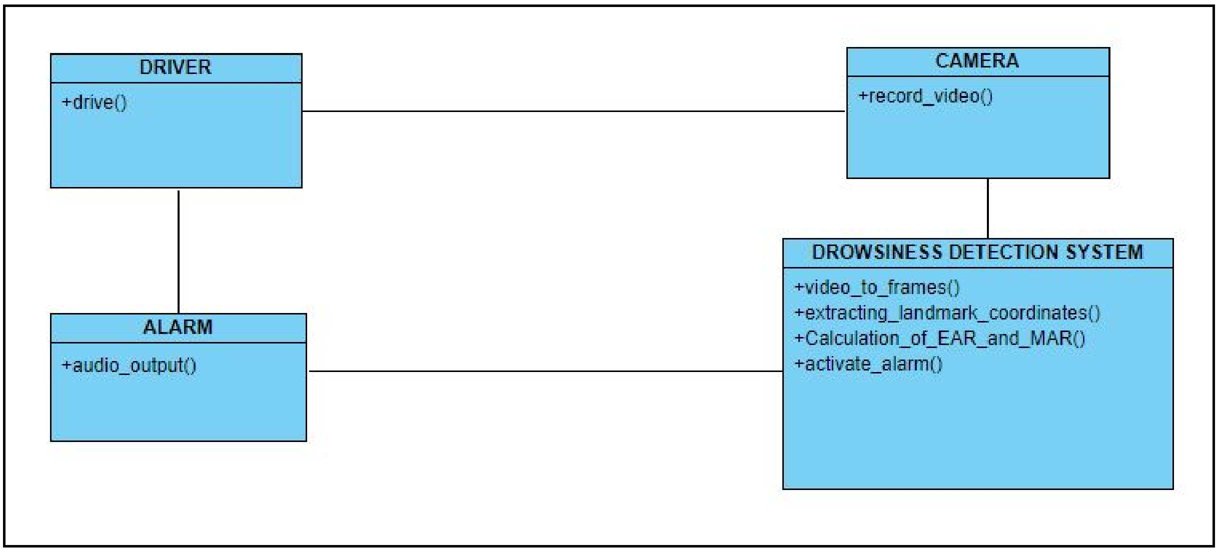

DRIVER
+drive()
CAMERA
+record_video()
ALARM
+audio_output()
DROWSINESS DETECTION SYSTEM
+video_to_frames()
+extracting_landmark_coordinates()
+Calculation_of_EAR_and_MAR()
+activate_alarm()

Fig 4.2. Class Diagram for Driver Drowsiness Detection.

4.3 Data Flow Diagram

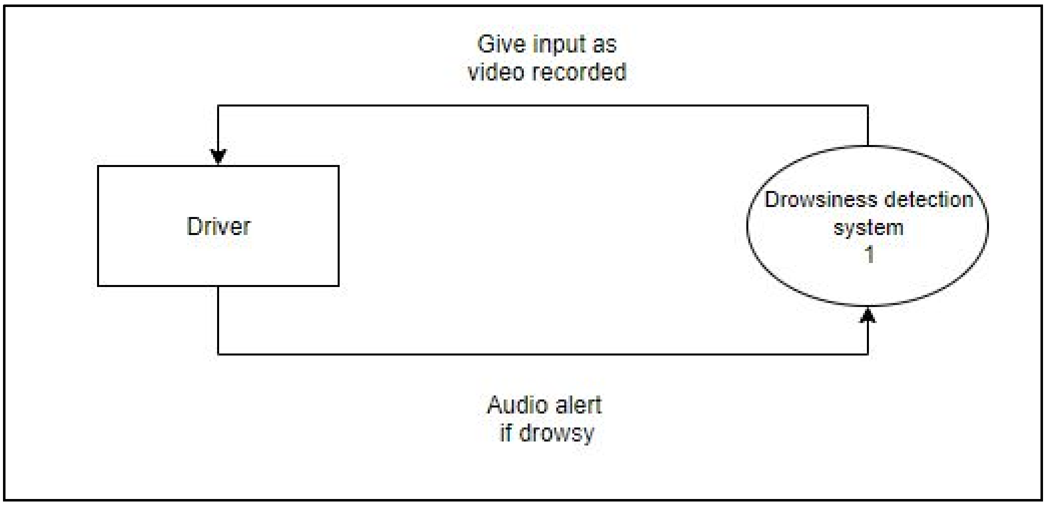


Fig 4.3.1 Data Flow Diagram Level 0

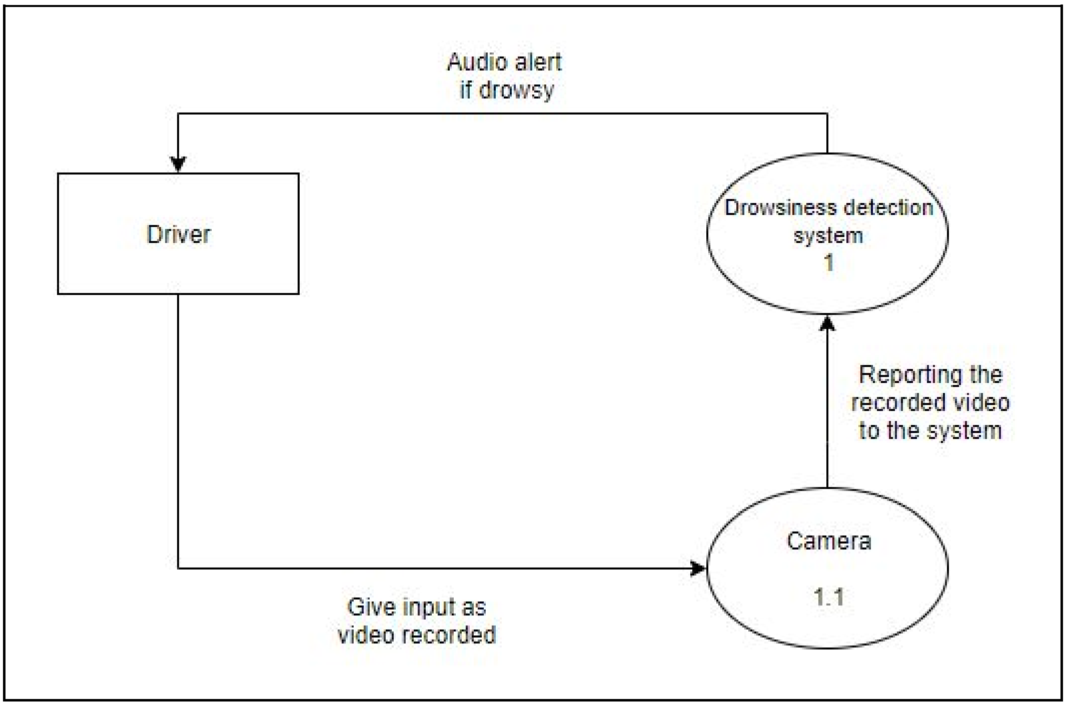

Audio alert if drowsy
Driver
Drowsiness detection system 1
Reporting the recorded video to the system
Camera 1.1
Give input as video recorded

Fig 4.3.2 Data Flow Diagram Level 1(a)

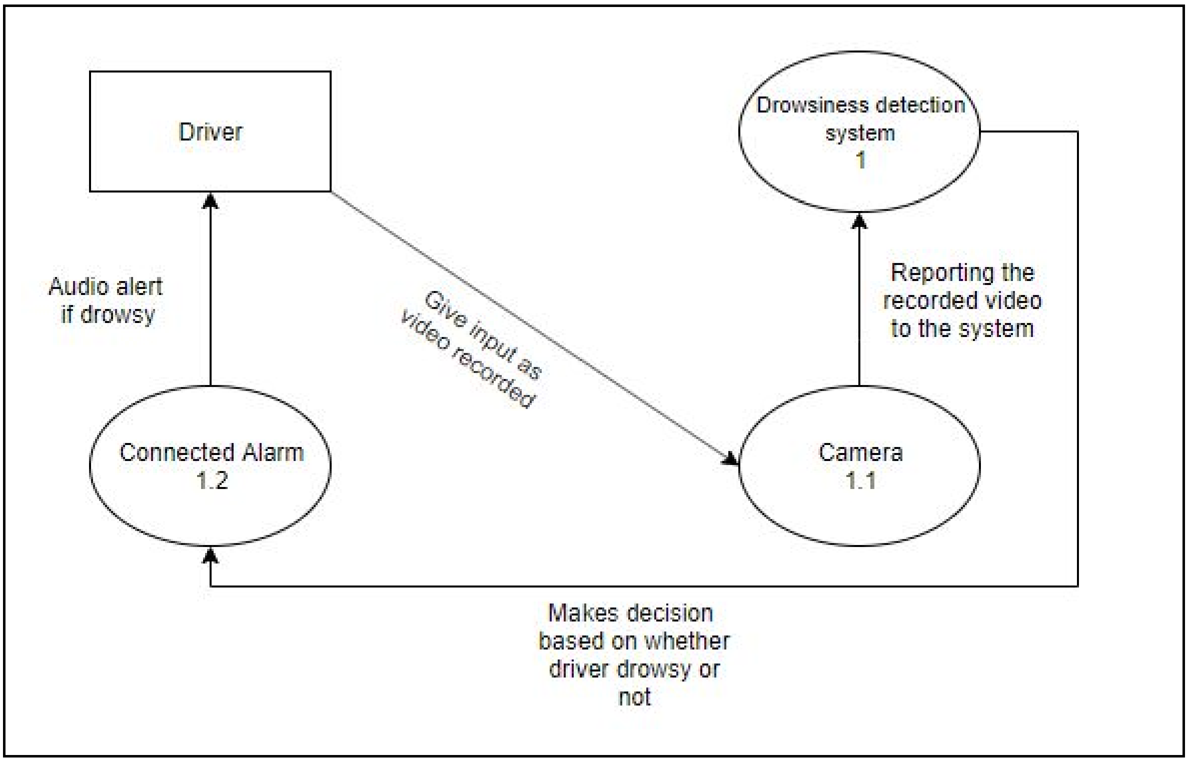
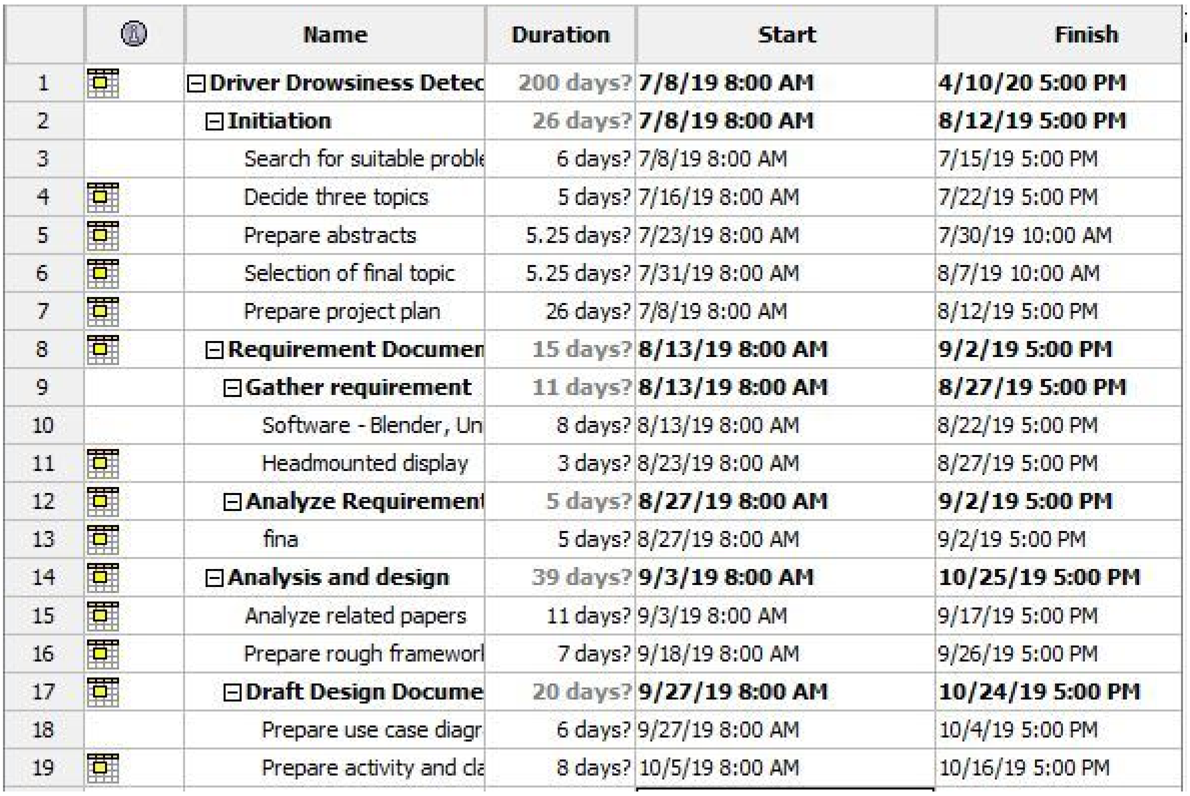

Driver
Drowsiness detection system
1
Audio alert if drowsy
Give input as video recorded
Reporting the recorded video to the system
Connected Alarm
1.2
Camera
1.1
Makes decision based on whether driver drowsy or not

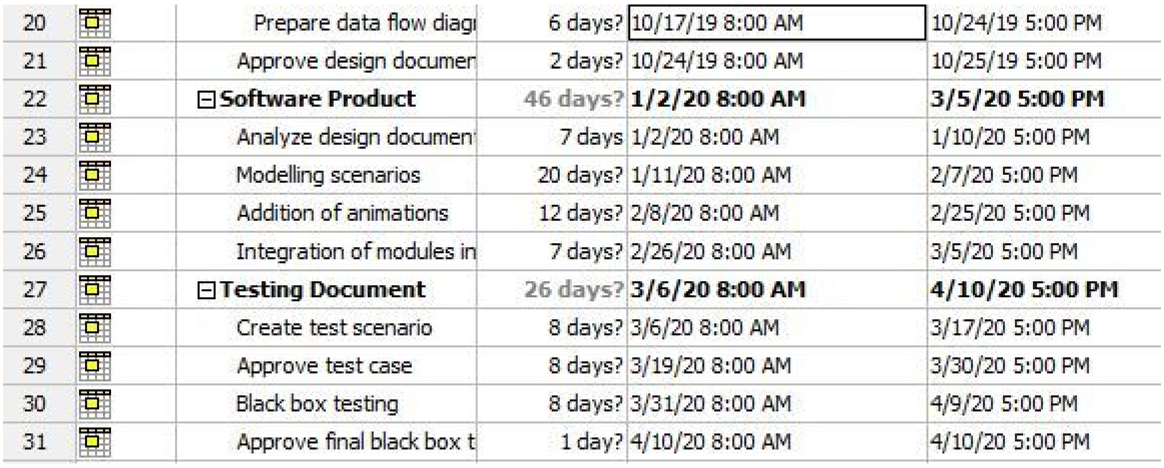
Fig 4.3.3 Data Flow Diagram Level 1(b)

4.4 Timeline chart



	ⓘ	Name	Duration	Start	Finish
1	▦	⊟ **Driver Drowsiness Detec**	200 days?	**7/8/19 8:00 AM**	**4/10/20 5:00 PM**
2		⊟ **Initiation**	26 days?	**7/8/19 8:00 AM**	**8/12/19 5:00 PM**
3		Search for suitable proble	6 days?	7/8/19 8:00 AM	7/15/19 5:00 PM
4	▦	Decide three topics	5 days?	7/16/19 8:00 AM	7/22/19 5:00 PM
5	▦	Prepare abstracts	5.25 days?	7/23/19 8:00 AM	7/30/19 10:00 AM
6	▦	Selection of final topic	5.25 days?	7/31/19 8:00 AM	8/7/19 10:00 AM
7	▦	Prepare project plan	26 days?	7/8/19 8:00 AM	8/12/19 5:00 PM
8	▦	⊟ **Requirement Documen**	15 days?	**8/13/19 8:00 AM**	**9/2/19 5:00 PM**
9		⊟ **Gather requirement**	11 days?	**8/13/19 8:00 AM**	**8/27/19 5:00 PM**
10		Software - Blender, Un	8 days?	8/13/19 8:00 AM	8/22/19 5:00 PM
11	▦	Headmounted display	3 days?	8/23/19 8:00 AM	8/27/19 5:00 PM
12	▦	⊟ **Analyze Requirement**	5 days?	**8/27/19 8:00 AM**	**9/2/19 5:00 PM**
13	▦	fina	5 days?	8/27/19 8:00 AM	9/2/19 5:00 PM
14	▦	⊟ **Analysis and design**	39 days?	**9/3/19 8:00 AM**	**10/25/19 5:00 PM**
15	▦	Analyze related papers	11 days?	9/3/19 8:00 AM	9/17/19 5:00 PM
16	▦	Prepare rough framewor	7 days?	9/18/19 8:00 AM	9/26/19 5:00 PM
17	▦	⊟ **Draft Design Docume**	20 days?	**9/27/19 8:00 AM**	**10/24/19 5:00 PM**
18		Prepare use case diagr	6 days?	9/27/19 8:00 AM	10/4/19 5:00 PM
19	▦	Prepare activity and cla	8 days?	10/5/19 8:00 AM	10/16/19 5:00 PM

Fig 4.4.1 Timeline chart (1)



20	▦	Prepare data flow diag	6 days?	10/17/19 8:00 AM	10/24/19 5:00 PM
21	▦	Approve design documen	2 days?	10/24/19 8:00 AM	10/25/19 5:00 PM
22	▦	⊟ **Software Product**	46 days?	**1/2/20 8:00 AM**	**3/5/20 5:00 PM**
23	▦	Analyze design documen	7 days	1/2/20 8:00 AM	1/10/20 5:00 PM
24	▦	Modelling scenarios	20 days?	1/11/20 8:00 AM	2/7/20 5:00 PM
25	▦	Addition of animations	12 days?	2/8/20 8:00 AM	2/25/20 5:00 PM
26	▦	Integration of modules in	7 days?	2/26/20 8:00 AM	3/5/20 5:00 PM
27	▦	⊟ **Testing Document**	26 days?	**3/6/20 8:00 AM**	**4/10/20 5:00 PM**
28	▦	Create test scenario	8 days?	3/6/20 8:00 AM	3/17/20 5:00 PM
29	▦	Approve test case	8 days?	3/19/20 8:00 AM	3/30/20 5:00 PM
30	▦	Black box testing	8 days?	3/31/20 8:00 AM	4/9/20 5:00 PM
31	▦	Approve final black box t	1 day?	4/10/20 8:00 AM	4/10/20 5:00 PM

Fig 4.4.2 Timeline chart (2)

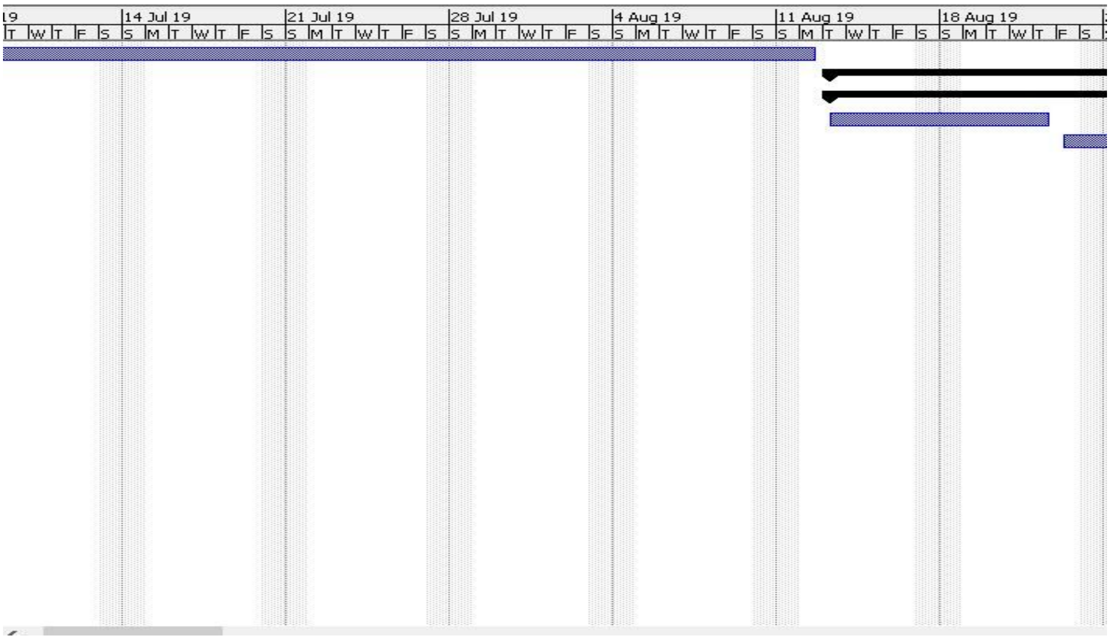


Fig 4.4.3 Timeline chart (3)

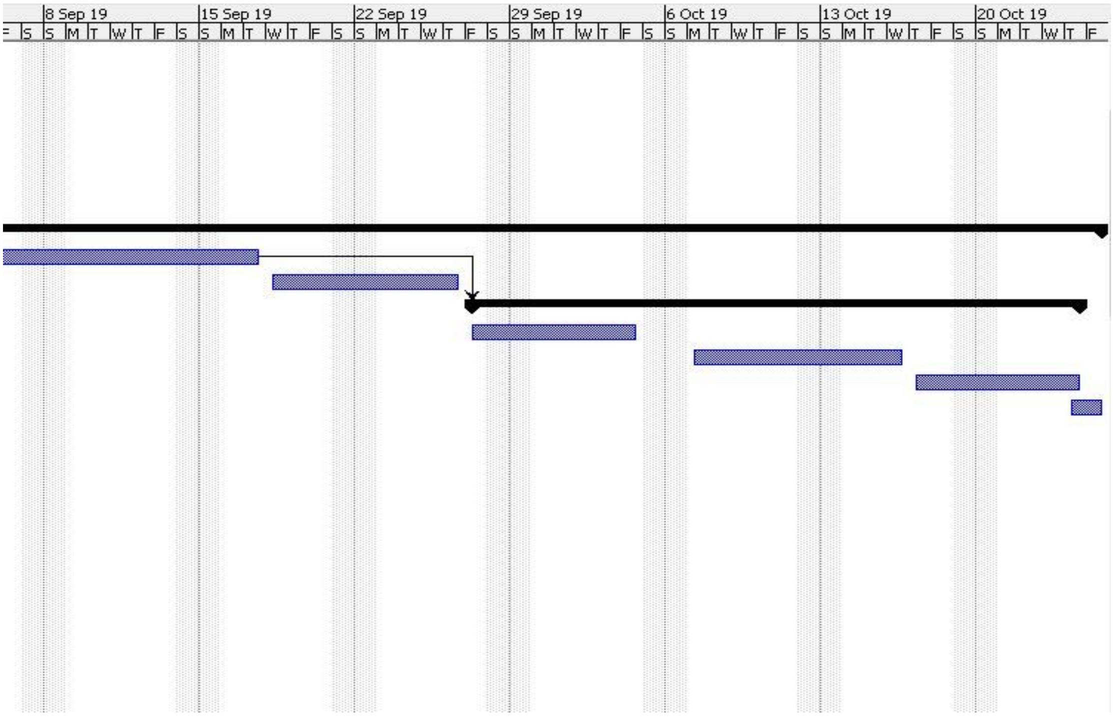


Fig 4.4.4 Timeline chart (4)

Chapter 5

Design

5.1 Architecture Design

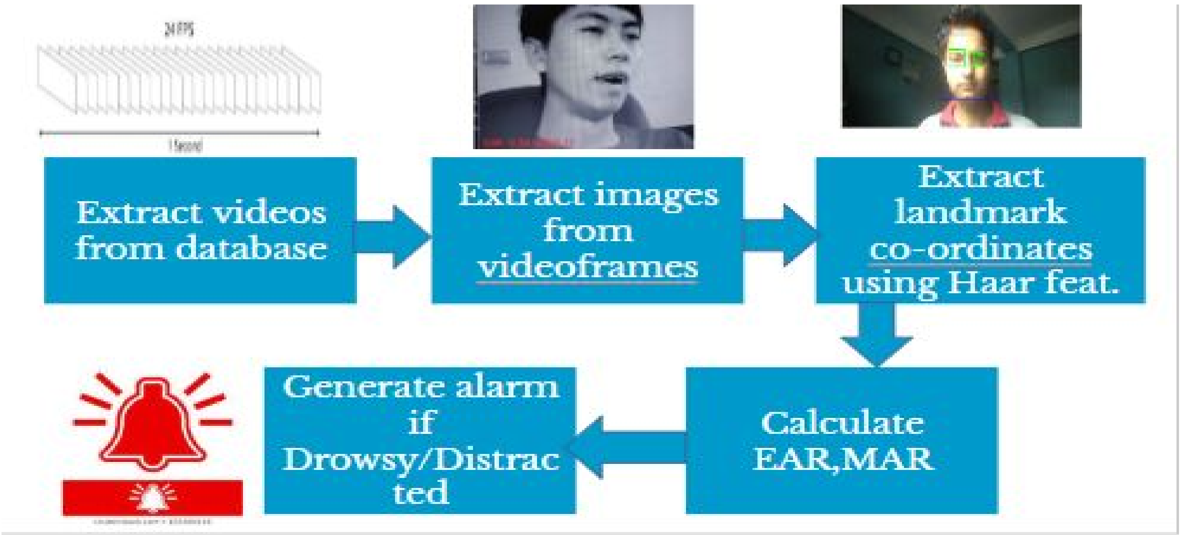

24 FPS
1 Second
Extract videos from database
Extract images from videoframes
Extract landmark co-ordinates using Haar feat.
Calculate EAR,MAR
Generate alarm if Drowsy/Distracted

Fig 5.1 Architecture Design

- **Monitor live video of the driver**- This video serves as the input to the system.
- **Extracting images from video frames**- The images are extracted from the video at a rate of 30 frames per second.
- **Extracting landmark coordinates**- The Dlib library is used for this purpose, it gives 68 (x,y) coordinates to help in extraction of facial features. Dlib uses OpenCV's built-in Haar feature-based cascades to detect facial landmarks. Paul Viola and Michael Jones proposed this efficient object detection method that is based on machine learning.

Detecting facial landmarks is therefore a two step process:

Step #1: Localize the face in the image.

Step #2: Detect the key facial structures on the face ROI.

- **Calculate EAR,MAR** - The dlib face landmark detector will return a shape object containing the 68 *(x, y)*-coordinates of the facial landmark regions.

  Provides lists for various features of the face : eyes, mouth etc

  EAR (Eye Aspect Ratio) - vertical eye distances / horizontal distance

  MAR (Mouth Aspect Ratio) - distance between the lower side of upper lip and the upper side of lower lip

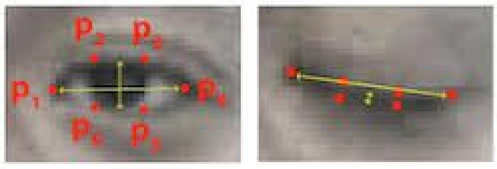
- **Generate an Alarm -** An audio Alarm is set off in case the driver is found to be drowsy.

Chapter 6

Implementation

The project deals with the problem of binary classification that is basically to detect whether the driver is drowsy or not. The classification part in this project is done with the help of calculation of Eye aspect ratio (EAR) and mouth aspect ratio (MAR). These ratios are calculated by locating essential features on the face. The Dlib's facial landmark recognizer is used for the localization of 68 landmark facial features. Contours are built over the eyes and the mouth. The motion of eyes and mouth and the amount of visibility are the determining factors for the ratios.

EAR - The Eye Aspect Ratio lies within a certain range in normal conditions but when a person is drowsy it rapidly falls to a very low value. The average EAR of a driver can be calculated over a period of time and if the eyes close to about 0.3*EAR the person is classified as drowsy.



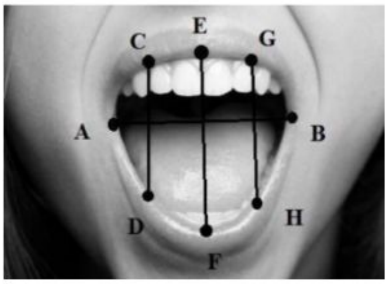
$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Fig : Eye Aspect Ratio formula and working

$$\text{EAR}= \frac{||p2\text{-}p6|| + ||p3\text{-}p5||}{2||p1 - p4||} \quad \ldots\ldots\ldots\ldots\ldots(1)$$

MAR- The mouth aspect ratio is defined as the ratio of the horizontal and vertical distances of the mouth. MAR is used to detect if a person is yawning. Yawning is differentiated from talking or eating by using the time factor. If the MAR exceeds a certain threshold value for more than a specific amount of time then it is certain that he is yawning.

$$\text{MAR} = \frac{|EF|}{|AB|} \quad \ldots\ldots\ldots\ldots\ldots(2)$$



$$\text{MAR} = \frac{|\text{EF}|}{|\text{AB}|}$$

Fig : Mouth Aspect Ratio formula and working

6.1 Computer Vision (openCV (Python))

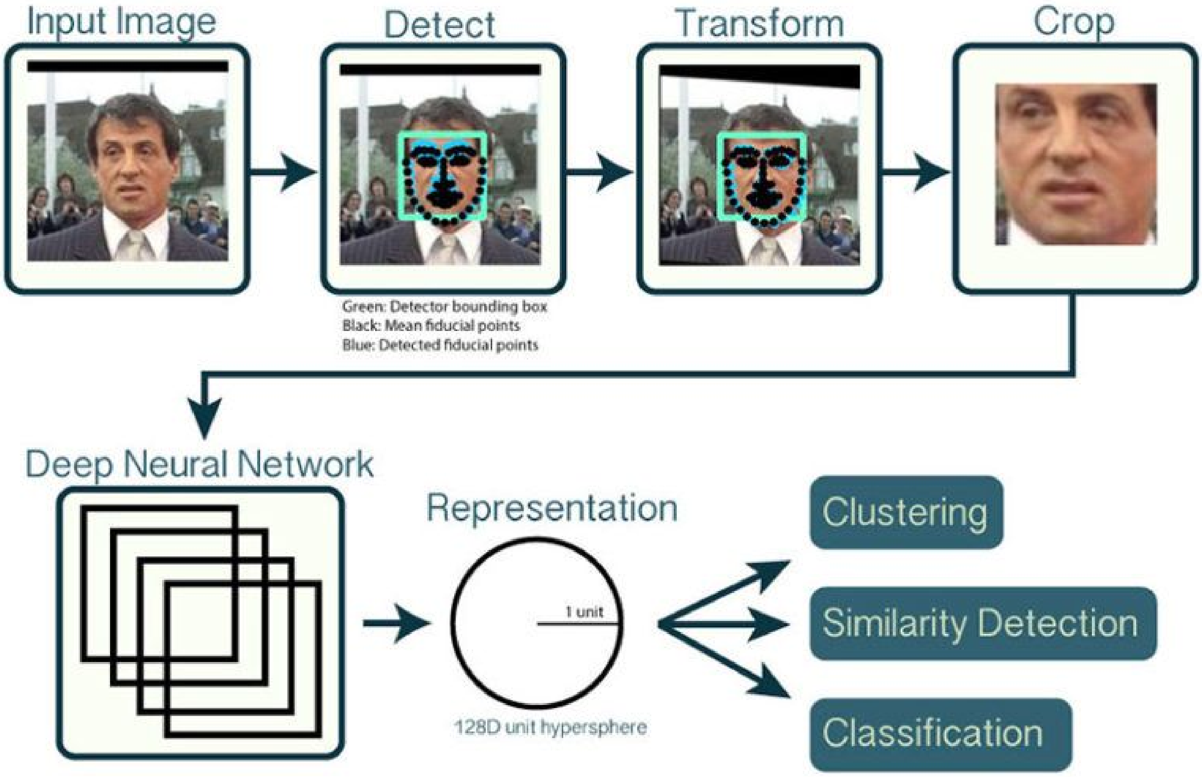


Fig 6.1.1 Sample application of opencv

WORKING OF OPENCV DETECTION:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.[7]

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

6.2 Haar Like Features

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

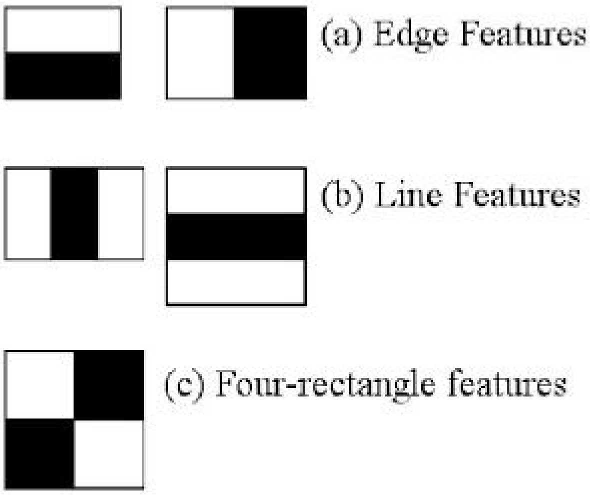


Fig 6.2.1 Haar like features

Now, all possible sizes and locations of each kernel are used to calculate lots of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker.

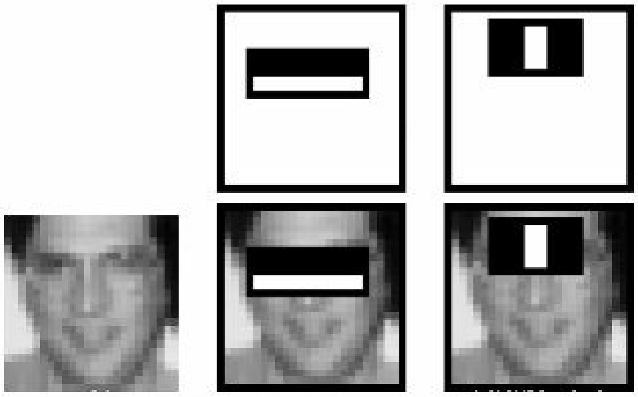


Fig 6.2.2 Haar like features for face detection

We apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found).

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features.

In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. (Normally the first few stages will contain very many fewer features). If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

6.3 Dlib Library

DLib is an open source C++ library implementing a variety of machine learning algorithms, including classification, regression, clustering, data transformation, and structured prediction. It is majorly used for facial landmark detection. The frontal face detector in dlib works really well. It is simple and just works out of the box. This detector is based on histogram of oriented gradients (HOG) and linear SVM. Although the HOG based face detector works quite well for frontal faces, its performance is not good when it comes to detecting faces at odd angles. Thus, the positioning of the camera in the car will play a crucial role in driving the output. Meanwhile, the CNN based detector is capable of detecting faces at almost all angles. Unfortunately it is not suitable for real time video. It is meant to be executed on a GPU. To get the same speed as the HOG based detector you might need to run on a powerful Nvidia GPU.

The frontal face detector of dlib uses haar like features to detect a face and returns 68 landmark coordinates as shown in the figure.

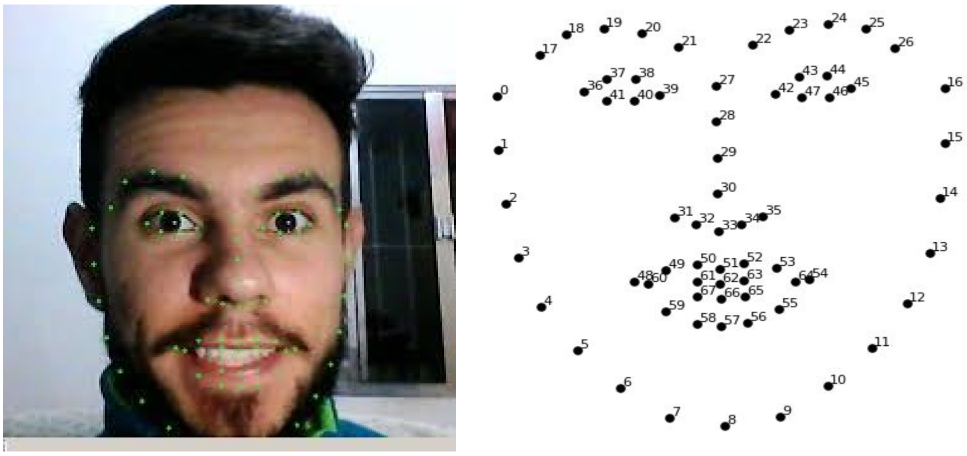


Fig 6.3.1 Dlib face detection

CODE-

```
from parameters import *
from scipy.spatial import distance
from imutils import face_utils as face
from pygame import mixer
import imutils
import time
import dlib
import cv2

# Some supporting functions for facial processing

def get_max_area_rect(rects):
    if len(rects)==0: return
```

```
        areas=[]
        for rect in rects:
            areas.append(rect.area())
        return rects[areas.index(max(areas))]

    def get_eye_aspect_ratio(eye):
        vertical_1 = distance.euclidean(eye[1], eye[5])
        vertical_2 = distance.euclidean(eye[2], eye[4])
        horizontal = distance.euclidean(eye[0], eye[3])
        return (vertical_1+vertical_2)/(horizontal*2) #aspect ratio of eye

    def get_mouth_aspect_ratio(mouth):
        horizontal=distance.euclidean(mouth[0],mouth[4])
        vertical=0
        for coord in range(1,4):
            vertical+=distance.euclidean(mouth[coord],mouth[8-coord])
        return vertical/(horizontal*3) #mouth aspect ratio


    # Facial processing

    def facial_processing():
        output_labels=[]
        mixer.init()
        distracton_initlized = False
        eye_initialized      = False
        mouth_initialized    = False

        detector    = dlib.get_frontal_face_detector()
        predictor   = dlib.shape_predictor(shape_predictor_path)

        ls,le = face.FACIAL_LANDMARKS_IDXS["left_eye"]
        rs,re = face.FACIAL_LANDMARKS_IDXS["right_eye"]
                                            path_to_video=r"C:\Users\urjit\DDD\Training
Dataset\001\noglasses\sleepyCombination.avi"
```

```
    cap=cv2.VideoCapture(0)
     ######
     no_frame_not_detected=0
     fps_couter=0
     fps_to_display='initializing...'
     fps_timer=time.time()
     while True:
         ret , frame=cap.read()
         if ret==False:
             break
         frame=cv2.resize(frame,(640,480))
         frame_class=0
         fps_couter+=1
         frame = cv2.flip(frame, flipCode=1)
         if time.time()-fps_timer>=1.0:
             fps_to_display=fps_couter
             fps_timer=time.time()
             fps_couter=0
                         cv2.putText(frame,   "FPS   :"+str(fps_to_display),   (frame.shape[1]-100,
frame.shape[0]-10),
                     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)



         #frame = imutils.resize(frame, width=900)
         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


         rects = detector(gray, 0)
         rect=get_max_area_rect(rects)


         if rect!=None:


             distracton_initlized=False


             shape = predictor(gray, rect)
```

```
        shape = face.shape_to_np(shape)

        leftEye = shape[ls:le]
        rightEye = shape[rs:re]
        leftEAR = get_eye_aspect_ratio(leftEye)
        rightEAR = get_eye_aspect_ratio(rightEye)

        inner_lips=shape[60:68]
        mar=get_mouth_aspect_ratio(inner_lips)

        eye_aspect_ratio = (leftEAR + rightEAR) / 2.0
        #eye_aspect_ratio=rightEAR
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (255, 255, 255), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (255, 255, 255), 1)
        lipHull = cv2.convexHull(inner_lips)
        cv2.drawContours(frame, [lipHull], -1, (255, 255, 255), 1)

         cv2.putText(frame, "EAR: {:.2f} MAR{:.2f}".format(eye_aspect_ratio,mar), (10,
frame.shape[0]-10),\
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

              if((eye_aspect_ratio < EYE_DROWSINESS_THRESHOLD) or (mar >
MOUTH_DROWSINESS_THRESHOLD)):
          frame_class=1

        if (eye_aspect_ratio < EYE_DROWSINESS_THRESHOLD ):
          #output_labels.append(1)

          if not eye_initialized:
            eye_start_time= time.time()
            eye_initialized=True

          if time.time()-eye_start_time >= EYE_DROWSINESS_INTERVAL:
              alarm_type=0


              cv2.putText(frame, "YOU ARE SLEEPY...\nPLEASE TAKE A BREAK!",
(10, 20),
                  cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)


                        if  not  distracton_initlized  and  not  mouth_initialized  and  not
mixer.music.get_busy():
              mixer.music.load(alarm_paths[alarm_type])
              mixer.music.play()
        else:
          #output_labels.append(0)
          eye_initialized=False
          if not distracton_initlized and not mouth_initialized and mixer.music.get_busy():
            mixer.music.stop()



        if mar > MOUTH_DROWSINESS_THRESHOLD:
          #output_labels.append(1)

          if not mouth_initialized:
            mouth_start_time= time.time()
            mouth_initialized=True

          if time.time()-mouth_start_time >= MOUTH_DROWSINESS_INTERVAL:
            alarm_type=0
            output_labels.append(1)
                    cv2.putText(frame,  "YOU  ARE  YAWNING...\nDO  YOU  NEED  A
BREAK?", (10, 40),
                  cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

            if not mixer.music.get_busy():
              mixer.music.load(alarm_paths[alarm_type])
              mixer.music.play()
        else:
            mouth_initialized=False
            if not distracton_initlized and not eye_initialized and mixer.music.get_busy():
                mixer.music.stop()



    else:


        frame_class=1
        alarm_type=1
        if not distracton_initlized:
            distracton_start_time=time.time()
            distracton_initlized=True

        if time.time()- distracton_start_time> DISTRACTION_INTERVAL:
            output_labels.append(1)

            cv2.putText(frame, "EYES ON ROAD", (10, 20),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

            if not eye_initialized and not mouth_initialized and not  mixer.music.get_busy():
                mixer.music.load(alarm_paths[alarm_type])
                mixer.music.play()

    cv2.imshow("Frame", frame)
    output_labels.append(frame_class)
    key = cv2.waitKey(5)&0xFF
    if key == ord("q"):
        break

print(output_labels)
print(len(output_labels))
cv2.destroyAllWindows()
cap.release()
```

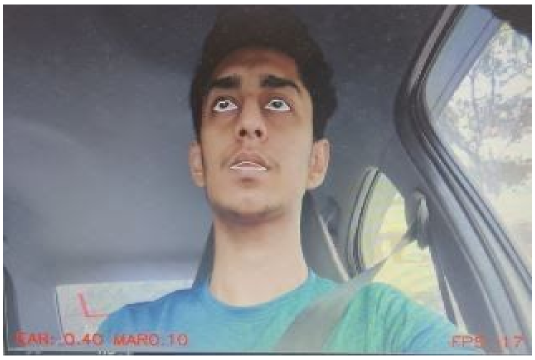
```
if __name__=='__main__':
        op=facial_processing()
    #print(op)
    #print(len(op))
```

Chapter 7

Testing

7.1 Test Cases

1. Good Lighting Conditions without Spectacles



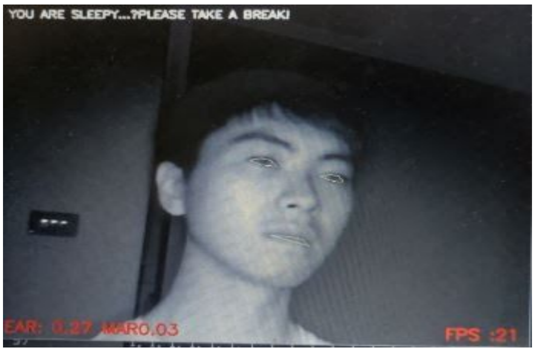
EAR: 0.40 MARO:10

FPS :17

Fig 7.1 Good Light without Spectacles

In this case, the system was tested under good lighting conditions and for driver without spectacles. The output was a satisfactory detection of eyes and mouth.

2. Bad Lighting Conditions without Spectacles



YOU ARE SLEEPY....?PLEASE TAKE A BREAK!

EAR: 0.27 MARO.03

FPS :21

Fig 7.2 Bad Lighting without Spectacles

In this case, the system was tested under bad lighting conditions and for driver without spectacles. The output was satisfactory detection of eyes and mouth. The image also shows “YOU ARE SLEEPY....PLEASE TAKE A BREAK” message which indicates driver’s drowsiness was detected in this frame.

3. Good Lighting Conditions with Spectacles

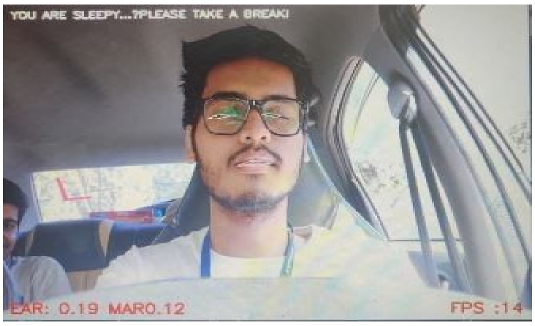


Fig 7.3 Good Lighting with Spectacles

In this case, the system was tested under good lighting conditions for driver with spectacles.The output was satisfactory detection of mouth but faulty detection of eyes due to reflection. The image also shows “YOU ARE SLEEPY....PLEASE TAKE A BREAK” message which indicates driver’s drowsiness was detected incorrectly.

4. Bad Lighting Conditions with Spectacles

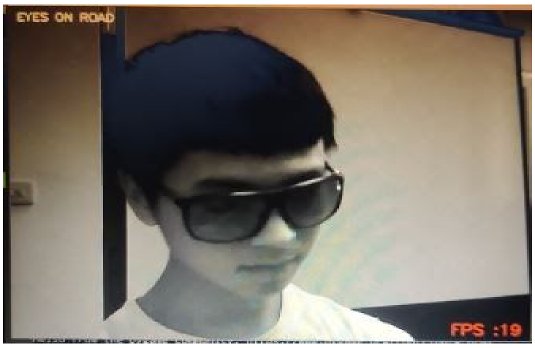


Fig 7.4 Bad lighting with Spectacles

In this case, the system was tested under bad lighting conditions for driver with spectacles.The output was failed detection of both eyes and mouth. The image also shows “EYES ON ROAD” message which indicates driver’s drowsiness was detected incorrectly.

Chapter 8

Results and Discussions

The system was tested in real time under different environmental conditions (day, midday, night) and speed upto 60km/h. The system worked pretty accurately in decent lighting conditions. The system accurately classified the driver to be drowsy or distracted and gave a prompt accordingly. It was able to differentiate between acts of talking and eating from yawning, also normal eye blinks were differentiated from eye closures due to drowsiness. Also looking at side mirrors was correctly differentiated from a distracted state. Sample test images are shown in figure 8.2 illustrating the same. The system was able to provide a prompt in due time detecting drowsiness before the complete closure of the eyes thus enabling the driver to take timely actions. The system was also tested on the NTHU dataset which is considered to be a state of art in driver drowsiness [6].

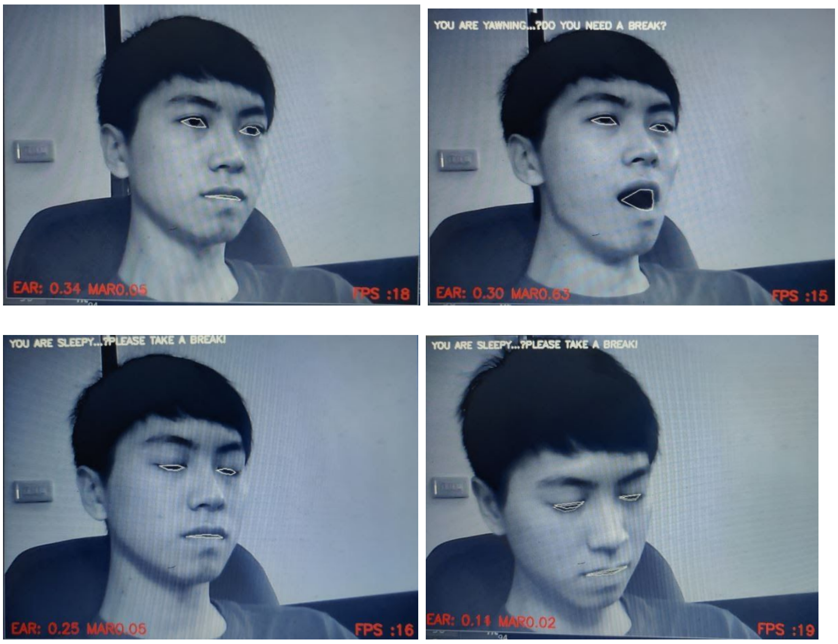
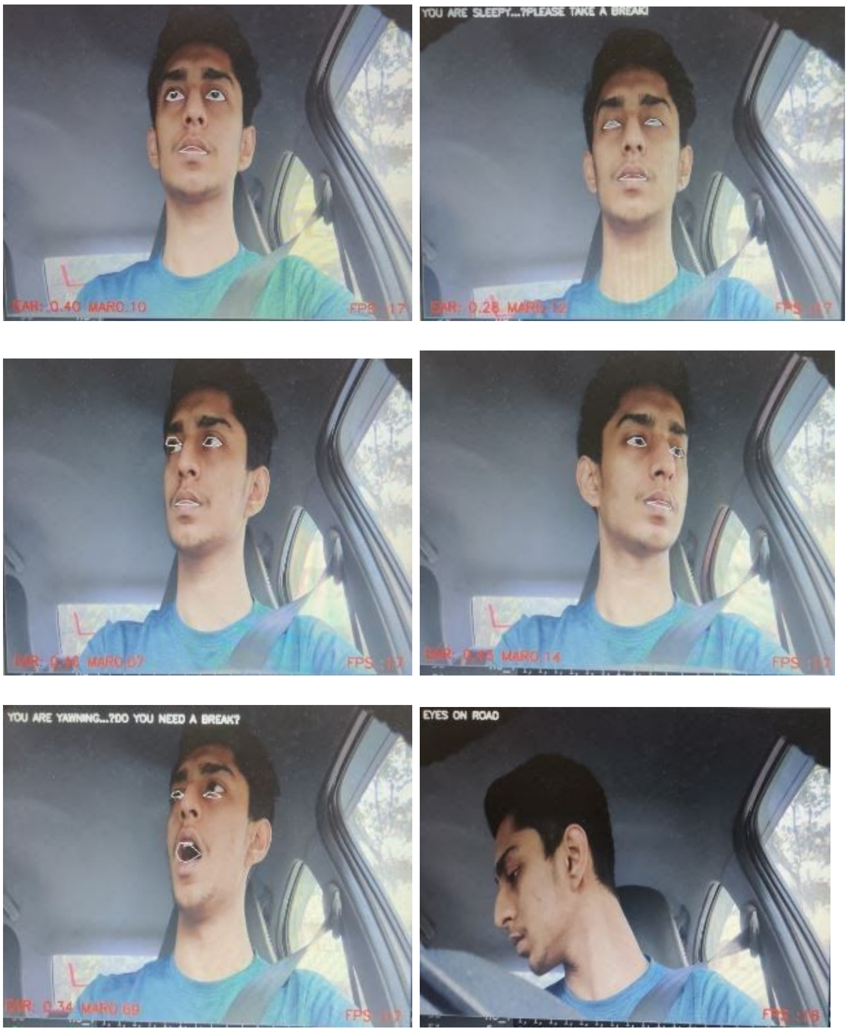

EAR: 0.34 MAR0.05
FPS :18
YOU ARE YAWNING...?DO YOU NEED A BREAK?
EAR: 0.30 MAR0.63
FPS :15
YOU ARE SLEEPY...?PLEASE TAKE A BREAK!
EAR: 0.25 MAR0.05
FPS :16
YOU ARE SLEEPY...?PLEASE TAKE A BREAK!
EAR: 0.11 MAR0.02
FPS :19

Fig 8.1 Testing on NTHU Dataset



YOU ARE SLEEPY...?PLEASE TAKE A BREAK!

YOU ARE YAWNING...?DO YOU NEED A BREAK?

EYES ON ROAD

Fig 8.2 Real-time testing

The systems results during night time are illustrated below.

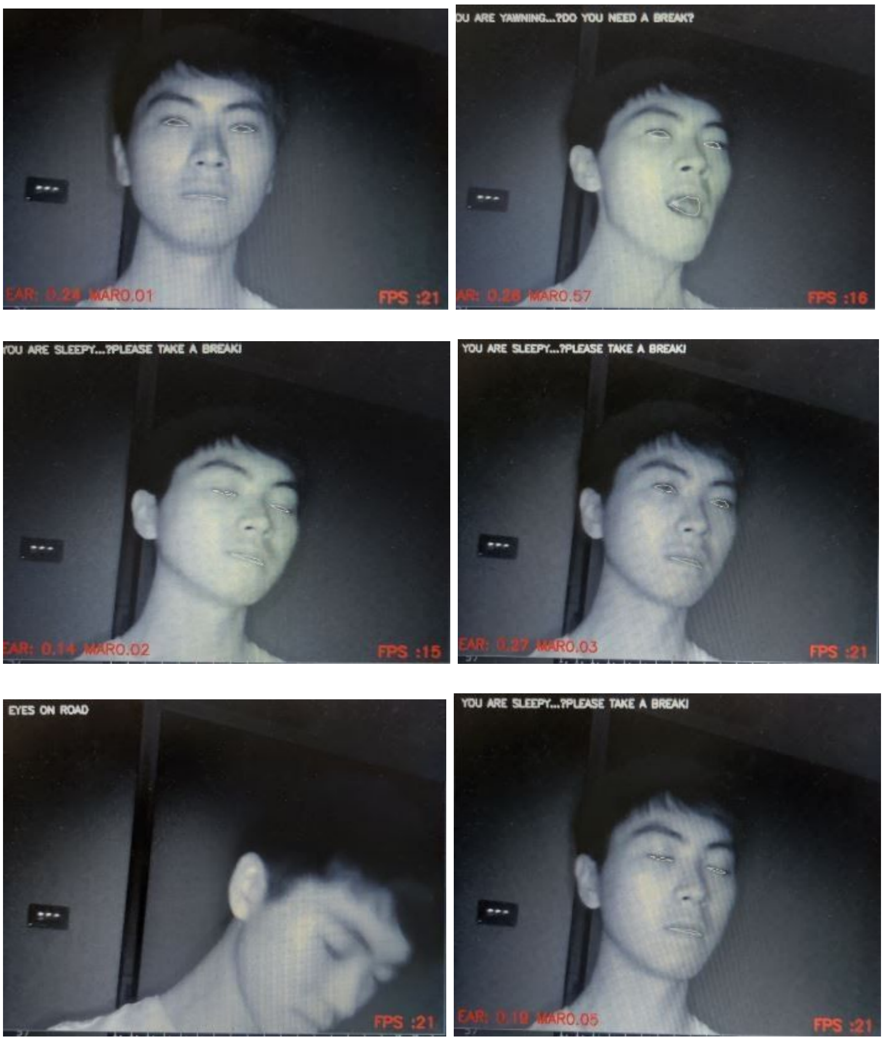

EAR: 0.24 MAR0.01
FPS :21
OU ARE YAWNING...?DO YOU NEED A BREAK?
AR: 0.28 MAR0.57
FPS :16
OU ARE SLEEPY...?PLEASE TAKE A BREAK!
EAR: 0.14 MAR0.02
FPS :15
YOU ARE SLEEPY...?PLEASE TAKE A BREAK!
EAR: 0.27 MAR0.03
FPS :21
EYES ON ROAD
FPS :21
YOU ARE SLEEPY...?PLEASE TAKE A BREAK!
EAR: 0.19 MAR0.05
FPS :21

Fig 8.3 NTHU dataset night conditions

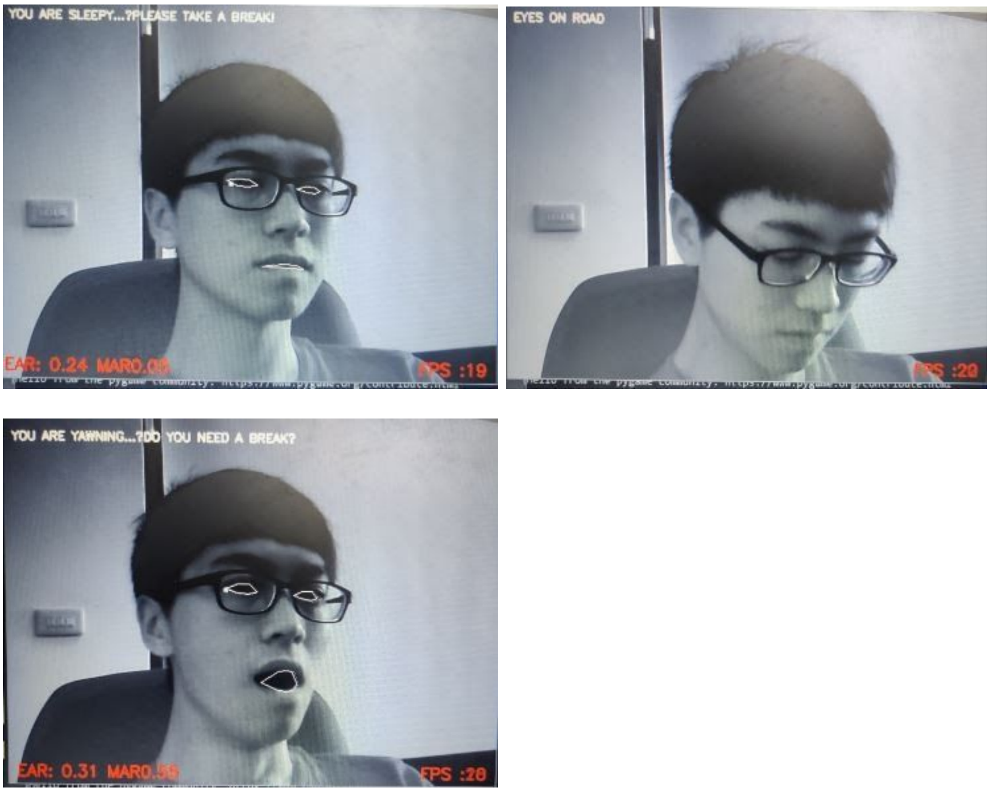

YOU ARE SLEEPY...?PLEASE TAKE A BREAK!
EAR: 0.24
FPS :19
EYES ON ROAD
FPS :20
YOU ARE YAWNING...?DO YOU NEED A BREAK?
EAR: 0.31
FPS :20

Fig 8.4 NTHU successful detection (glasses on)

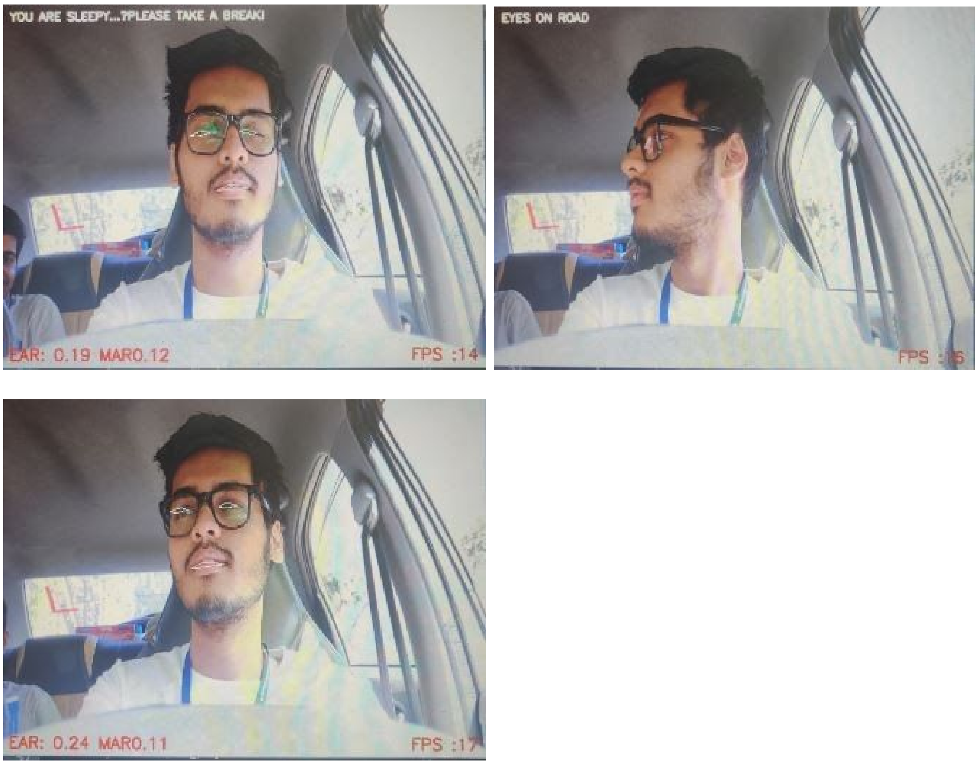


Fig 8.5 Real time successful detection (glasses on)

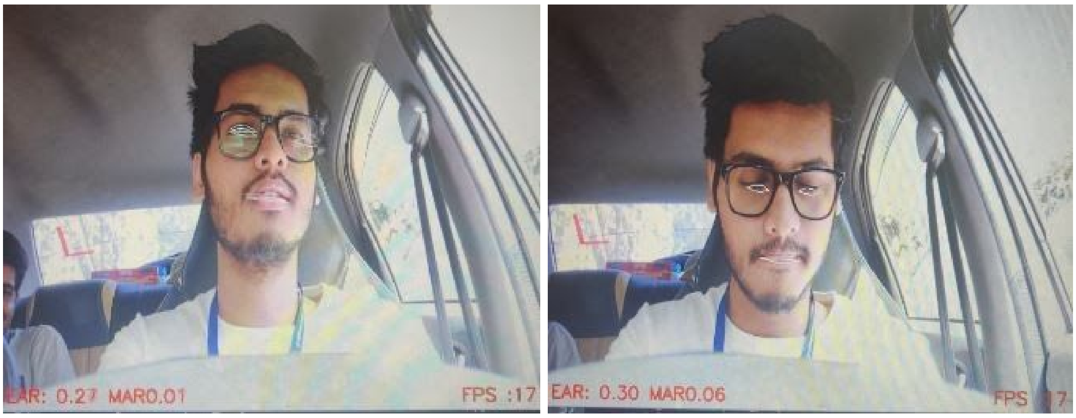


Fig 8.6 Real time faulty detection of eyes due to reflection

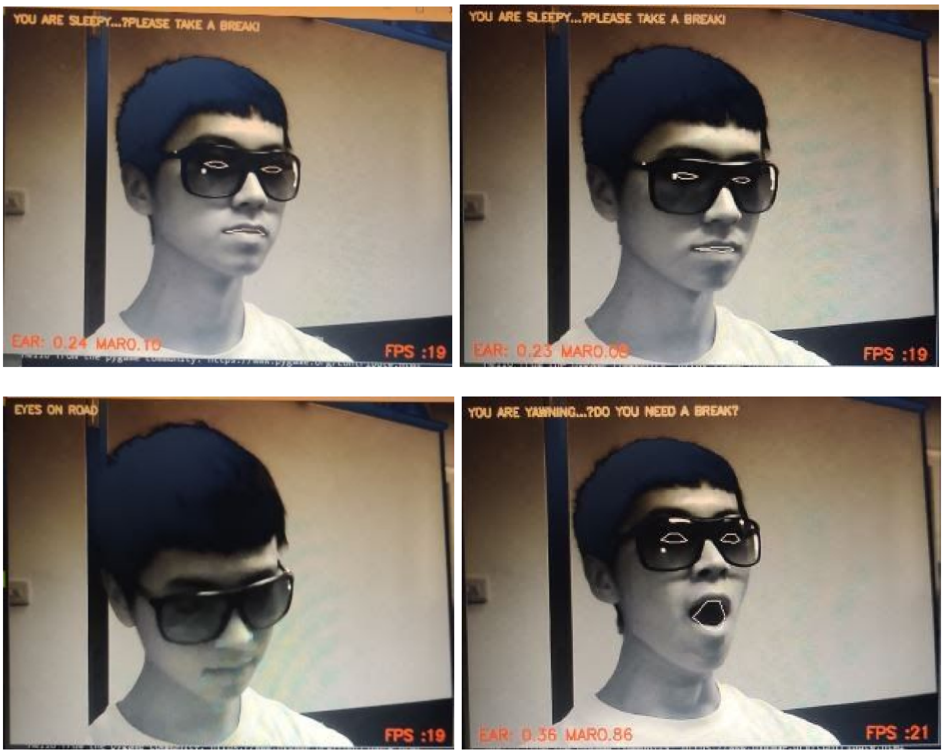
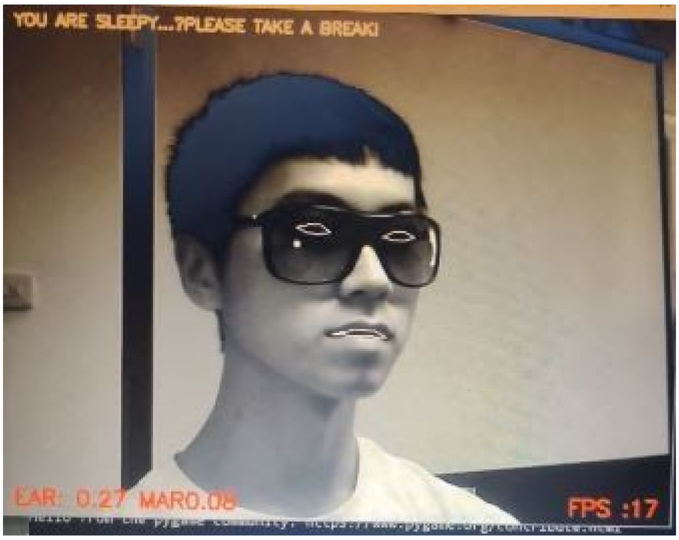

YOU ARE SLEEPY...?PLEASE TAKE A BREAK!
EAR: 0.24 MARO.10
FPS :19
YOU ARE SLEEPY...?PLEASE TAKE A BREAK!
EAR: 0.23 MARO.08
FPS :19
EYES ON ROAD
FPS :19
YOU ARE YAWNING...?DO YOU NEED A BREAK?
EAR: 0.36 MARO.86
FPS :21

Fig 8.7 NTHU successful detection (sunglasses)



YOU ARE SLEEPY...?PLEASE TAKE A BREAK!

EAR: 0.27 MAR0.08

FPS :17

Hello from the pygame community. https://www.pygame.org/contribute.html

Fig 8.8 NTHU faulty detection (sunglasses)

The system does not live upto the standards in case the driver has glasses on as it often falsely detects the eyes due to reflection on the glasses. Using MTCNN was also considered for facial detection as it is much more accurate then Haar cascade but the drawback of MTCNN was that the processing time was too slow on CPU and we could not compensate that for accuracy as speed of detection and processing is of primary importance in sceanrios like driving. Thus that idea was canned.

Chapter 9

Conclusion & Future Scope

Driver Drowsiness Detection is the task of identifying whether a driver is not in a proper state to drive the vehicle. The system is cost efficient, fast and reliable so that the driver is alerted within a few seconds if he is found to be drowsy. Driver drowsiness has been found to be among the leading causes of road accidents and by providing a feasible solution this work aims to reduce the number of accidents caused due to this reason. The system takes into consideration the behavioral characteristics of

the driver such as eye aspect ratio, yawning movements and head tilts that are signs of drowsiness.

The system is built to function in real time by taking the live video input stream of the driver and calculation of EAR and MAR simultaneously. If the threshold values for the ratios are surpassed then the driver is classified as drowsy or distracted. If found to be drowsy or distracted, the driver will be alerted within a few seconds by setting off an alarm.

The system can be improved further by introducing a hybrid approach, one which considers the vehicle based, behavioral and physiological aspects of the scenario. Considering all these parameters simultaneously is likely to yield more accurate results than the current system. Face mesh is a new development in the area of face detection and it should be considered in future projects for this purpose.

References

[1]Viola P, Micheal Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, *Accepted conference on computer vision and pattern recognition 2001.*

[2]Weng CH., Lai YH., Lai SH, “Driver Drowsiness Detection via a Hierarchical Temporal Deep Belief Network.”, *In: Chen CS., Lu J., Ma KK. (eds) Computer Vision – ACCV 2016 Workshops. ACCV 2016. Lecture Notes in Computer Science, vol 10118. Springer, Cham*

[3] Park, S., Pan, F., Kang, S., & Yoo, C. D. (2017). *Driver Drowsiness Detection System Based on Feature Representation Learning Using Various Deep Networks. Lecture Notes in Computer Science, 154–164.* doi:10.1007/978-3-319-54526-4_12

[4] Drowsy Driver NHTSA reports. (2018, January 08). [online] available: https://www.nhtsa.gov/risky-driving/drowsy-

[5] AAA Foundation for Traffic Study. [online] available: https://aaafoundation.org/prevalence-impact-drowsy-driving/

[6] Driver Drowsiness image dataset from NTHU [online] available:

http://cv.cs.nthu.edu.tw/php/callforpaper/datasets/DDD

[7] OpenCV [online] available:

https://github.com/opencv/opencv/tree/master/samples/python

Acknowledgement

We express our deep sense of gratitude to our respected and learned guide, Ms. Supriya Solaskar for her valuable help and guidance. We are immensely thankful for the encouragement given to us during our research. We are also indebted to our respected Head of Department, Dr. Kavita Sonawane, our respected Principal Dr. Sincy George and Director Bro. Jose Thuruthiyil for permitting us to utilize all the necessary facilities of the institution. We are also thankful to all the other faculty members of our department for their kind cooperation and help. Lastly, we would like to express our deep appreciation towards our classmates and our indebtedness to our parents for providing us the moral support and encouragement throughout the project.