A REPORT

ON

DRIVER DROWSINESS DETECTION SYSTEM USING FACIAL RECOGNITION TECHNIQUES

BY

Anjana Vasudevan 2015AAPS0061U ECE

BITS Pilani, Dubai Campus
Dubai International Academic City (DIAC)
Dubai, U.A.E

(JANUARY 2019 - MAY 2019)

A REPORT

ON

DRIVER DROWSINESS DETECTION SYSTEM USING FACIAL RECOGNITION TECHNIQUES

BY

Anjana Vasudevan 2015AAPS0061U ECE

Prepared in Partial Fulfillment of the
Project Course ECE F366

BITS Pilani, Dubai Campus
Dubai International Academic City (DIAC)
Dubai, U.A.E

BITS Pilani, Dubai Campus

Dubai International Academic City (DIAC)
Dubai, U.A.E

BITS Pilani, Dubai Campus

Dubai International Academic City (DIAC)
Dubai, U.A.E

Student Name: Anjana Vasudevan **Student ID**: 2015AAPS0061U

Discipline: Electronics and Communication Engineering

Title of the Project: DRIVER DROWSINESS DETECTION SYSTEM USING FACIAL RECOGNITION TECHNIQUES

Name of the Faculty: Dr. Vilas Gaidhane

Date of Start: 04-Jun-2017

Key Words: Drowsiness, detection, image processing, tracking

Project Areas: Image Processing, Machine Learning

Abstract: Driver fatigue is one of the causes of road accidents and fatalities as it results in lack of attention to the road conditions. This paper presents a drowsiness detection system which will alert the driver about their exhaustion. It will focus on the algorithm and the implementation of the system, its scope and limitations. The system first recognizes the drivers' face, tracks their eye and mouth movement to check if the eye is closed and (or) yawning occurs for a prolonged period of time, in terms of frames, and raises an alarm.

.

Signature of the Student Signature of Faculty

Date: 15-May-2019 Date: 15-May-2019

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 - INTRODUCTION

1.1 ABOUT THE PROBLEM

Drowsiness is the tendency to feel lethargic or tired, which results in distraction from the road conditions while driving. This lack of concentration on the roads can result in possible accidents. Drowsiness reduces alertness of the driver, which leads to reduced judgement and therefore increases the risk of accidents. Drowsiness is ranked fourth in terms of causes of accidents worldwide. In the United States alone, drowsiness related accident fatalities accounted for about 7% of the total deaths associated with road accidents[1]. In India, around 1800 deaths occurred due to driver drowsiness, out of the total 180,000 deaths in 2016 [2].

Risks and fatalities due to drowsiness can be significantly reduced by awareness and systems which can alert the driver of possible drowsy feeling. The system can be configured in the vehicle such that it does not interfere the concentration of the driver, and at the same time monitor the person to assess if he / she is drowsy or not. In this way the lack of vigilance of the driver can be tracked and can possibly reduce the risk of accidents due to drowsiness.

1.2 SYSTEMS AVAILABLE TO TRACK DROWSINESS

There are two types of systems available, that can detect driver drowsiness. These are:

1. Intrusive
2. Non - intrusive

Intrusive systems, as the term suggests, uses data that can be collected only from the drivers' body to evaluate drowsiness. These include monitoring the pulse rate, heartbeat, analyzing the Electroencephalogram (EEG), etc. These systems estimate drowsiness based on how relaxed the heart is - the human heart beat rate considerably reduces during resting period [3]. These systems are not preferred as they affect the concentration of the driver due to the bulky nature of the system (in terms of wiring).

Non-intrusive system also use data from the drivers' body to evaluate drowsiness. The difference is that these systems use parameters that can be collected without interfering with drivers' vigilance. Automobile companies such as Nissan [4] and Volkswagen [5] estimate drowsiness based on the steering patterns of the driver. If there is an abnormality in the steering control and mechanism, the inbuilt driver assistance systems in the cars send a warning to the driver and automatically reduce the speeds to safe levels.

In addition to monitoring steering behavior, monitoring the drivers’ facial features is a popular method to estimate drowsiness. This approach is easier as it is less intrusive and does not affect the vigilance of driver in anyway. There is considerable amount of work done for tracking facial features to evaluate drowsiness. The Viola Jones Algorithm uses Haar like features to extract features and uses a classifier to classify various regions in the image. This is used to classify drowsy eyes from non - drowsy eyes [6].

Another method of evaluating drowsiness is by estimating percentage eye closure (PERCLOS). This involves segmentation and thresholding the image to detect the region and evaluate drowsiness (as the color composition of image of an active person differs from that of a drowsy person - especially the eye area) [7]. Also the numeral features and difference in brightness of pixels in the eye region have been used to track drowsiness [8]. Hough transforms have also been used to track eyes and estimate drowsiness as they are easier implement an build [9] .

Machine learning techniques have also been used to develop drowsiness detection systems. A convolutional neural network uses the input parameters such as the eye region and mouth region images for training and accurately estimate drowsiness [10]. Support Vector Machines (SVMs) have been used in classifying if a person is drowsy or not, but they have been tested on EEG signals [11].

1.3 OBJECTIVE

Based on the review of the above systems available, a drowsiness detection system implemented in this report is developed by using a combination of image processing and machine learning techniques. The system aims at continuously monitoring the driver for assessing drowsiness. The system has to meet the following specifications:

- The system will have to communicate if there are any visible signs of drowsiness.
- The system will have to raise an alarm if these signs persist for longer periods of time.
- The system must be robust and must function in limited processing capability.
- The system must yield high accuracy in assessing drowsiness.

In order to fulfill the above criteria, the method proposed by Vahid Kazemi and Josephine Sullivan [12] is used for identifying facial features and assessing if the person is drowsy or not. The detailed description of the method, along with design and testing is provided in detail in the following sections.

CHAPTER 2 - SYSTEM DESIGN AND OVERVIEW

2.1 OVERVIEW OF THE SYSTEM

The system is implemented in three parts:

- Face detection
- Feature Extraction
- Estimation of drowsiness

The face detection algorithm is implemented by using the Local Binary Patterns method.The extraction of features is done using the method proposed by Vahid Kazemi and Josephine Sullivan. The drowsiness is estimated as described in the flow diagram:

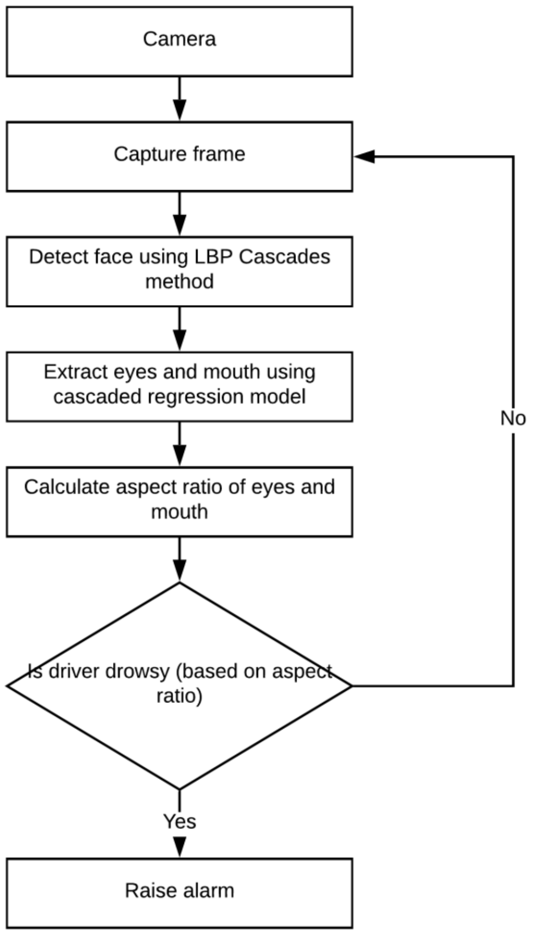
Camera
Capture frame
Detect face using LBP Cascades method
Extract eyes and mouth using cascaded regression model
Calculate aspect ratio of eyes and mouth
Is driver drowsy (based on aspect ratio)
No
Yes
Raise alarm

Figure 2.1 Flow diagram of the drowsiness detection system

2.2 INFORMATION ON THE TOOLS AND INTERFACE USED

The drowsiness is estimated based on aspect ratio of the mouth and eyes, after extracting the necessary features. The system was coded in Python with the help of two built in libraries from Python, dlib and OpenCV.

OpenCV stands for Open source Computer Vision and is a library designed for simplifying and applying image processing and machine learning concepts in real time applications. It was developed by Gary Bradski in Intel in the year 1999 [13]. It is written in C++ and has around 2500 algorithms optimized and being used for the same. It is a reliable tool for both government and private institutions for implementing face recognition techniques. It works well on all operating systems (Windows, Mac, Linux, Android) and supports programming software such as Python, C++, MATLAB [14].

Dlib is a library dedicated for implementing machine learning techniques in deploying applications. It was developed in the year 2002 and is written in C++. It has algorithms used primarily for machine learning and facial recognition problems [15]. The implementation of the feature extraction (as mentioned in the overview) is available in this library, which requires training for accurate results. It works well on Windows, Mac OS and Linux distributions and can be deployed in C, C++ and Python [16].

The detector was implemented and tested on a 64-bit Dell Inspiron Laptop with a processor speed of 2 GHz. The webcam that comes prebuilt with the Laptop has a frame resolution of 1280 $\times$ 720 (HD) pixels at 30 fps (frames per second) and a diagonal viewing angle of 74 degrees, which ensures high quality results of testing the system.

2.3 DETECTING THE FACE

The algorithm used to detect the face was the Local Binary Patterns algorithm. It was proposed by T. Ojala, M. Pietikäinen, and D. Harwood in 1994 [17]. The algorithm helps in differentiating the texture composition of an image, thereby classifying various regions / sections available in the image (this demarcation of regions obvious to the human eye). The steps followed for classifying texture in the image (using the LBP approach) are:

1. Divide the image or region of interest (ROI) into smaller areas called cells (a neighborhood of 8$\times$8, 16$\times$16 pixels).
2. Compare a pixel in the cell with its neighbors (in case of corner cells, pad zeroes to the neighbor pixels and then compare) in a clockwise or anticlockwise fashion. If the pixel value of center pixel is greater than the pixel value of neighbor pixel, encode 1, else encode zero to a vector V of size same as the cell. The resultant vector is an 8 to 10 digit binary number for a pixel (see Figure 2.2).
3. Compute the histogram, over the cell, of the frequency of the occurrences of 0s and 1s in the cell. This gives a feature vector for that cell.

4. Normalize all the feature vectors (of all cells) and compute another histogram for the entire window for the same quantity (frequency of zeroes and ones). This gives the feature vector for the entire ROI or image.

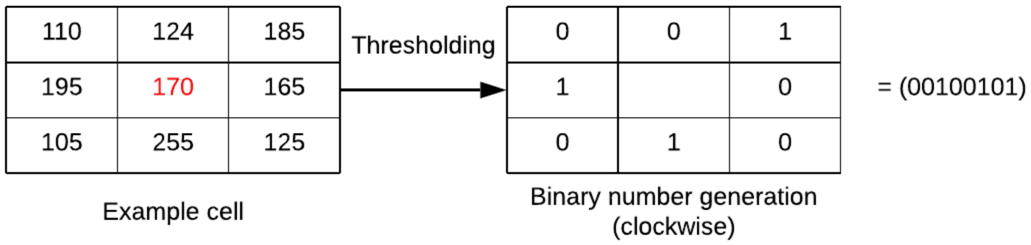

Figure 2.2 Thresholding operation of a pixel in cell

The resultant feature vectors for an image can be used to classify various regions using a suitable classifier or machine learning algorithm, such as Support Vector Machines, extreme learning machines. The LBP method, along with a Machine learning algorithm makes it a robust a computationally effective algorithm to classify regions in the images.

In OpenCV, a dedicated LBP cascade classifier is available, which utilises the LBP method to classify regions in the images or detect human faces in the images. The classifier is implemented as follows:

1. Divide the image into different regions.
2. LBP Labelling, which involves generating the binary numbers for each cell using thresholding (as in figure 2.2).
3. Generating feature vectors for each cell.
4. Using Adaboost learning to generate classifiers that remove redundant feature vectors.
5. Using the classifiers generated in the above step to classify the regions in the images.

The AdaBoost (adaptive booster) is a machine learning algorithm developed by Freund and Schapire in 1996 [18]. It converts a set of weak classifiers into a strong classifier. A booster is of the form:

$$F_t(x) = \sum_{t=1}^{T} F_t(x)$$

Where each f_t in summation is a weak classifier that takes an input and returns a value. The adaboost algorithm involves the following steps:

1. Take a set of classifiers, m, and select the classifier with the lowest error:

$$\epsilon_m = E_{w_m}\left[1_{y \neq f(x)}\right]$$

2. Calculate the weights for the m classifiers as shown

$$\theta_m = \frac{1}{2} ln(\frac{1-\epsilon_m}{\epsilon_m}).$$

3. Update the weights for each point in a given dataset associated with the classifiers:

$$w_{m+1}(x_i, y_i) = \frac{w_m(x_i, y_i) exp[-\theta_m y_i f_m(x_i)]}{Z_m},$$

   Where Z is the normalization coefficient that maintains the net sum of all weights to be 1.

The main advantage of the LBP algorithm in combination with the classifier is the reduced amount of computation and processing time, which makes it easier for implementing in remote embedded systems. It is also more responsive to illumination changes than any other face recognition systems such as the Haar classifier and dlib face predictor. Despite being accurate, it lacks precision in mapping the face region in the image wherein some regions of the face are cropped out.

The LBP algorithm is initialised by the following commands:

```
FACE_DETECTOR = "lbpcascade_frontalface.xml"
lbp_cascade = cv2.CascadeClassifier(FACE_DETECTOR)
# Detect faces
 rects = lbp_cascade.detectMultiScale(im, scaleFactor=1.1, minNeighbors=5)
```

The detectMultiScale command detects the number of faces present in the given region of interest, given the scaling range and no. of neighbourhood cells in the LBP algorithm (for generating the feature vector).

2.4 MAPPING OUT THE FACIAL REGIONS

The facial regions are mapped out using the Histogram of Gradients technique. The algorithm, proposed by Navneet Dalal and Bill Triggs [19], involves the following steps:

1. Preprocess the test image by normalizing its colour components, by using a contrast stretching technique - here the power law transform is used in the algorithm.
2. Calculate the gradient of the preprocessed image using edge detection operators, such as Canny edge detector and Sobel operator.
3. Divide the gradient image into cells (similar to the division of image to cells in the LBP algorithm).
4. Construct a histogram of pixel value -vs- angle of orientation for each cells of the gradient image.
5. Contrast normalize the histogram over 2 to 3 cells at a time (called blocks).

The resultant vector from the normalization is classified using Support Vector Machines (as per the method proposed by Dalal and Triggs). This in turn gives the outline of the shape of face.

The accuracy of the face outline is improved by employing a cascade of regressors, as proposed by Joseph Sullivian and Vahid Kazemi [12]. Here a regressor model is trained using the outline of the face, which will be referred to as the shape henceforth, as the initial input and a cascade of regressors. The regressor learns from the initial input and output of the previous iteration (the previous shape), with the help of the learning rate ($0 < \alpha < 1$) and returns a new, updated shape as per the equation:

$$S(t+1) = S(t) + r_t(I, \; S(t))$$

Where S is the shape vector, t is the iteration and r is the regressor model. The initial estimate S(0) is obtained from the Histogram of Gradients technique. The regressors are trained using the Gradient tree boosting algorithm. Like the Adaboost algorithm (which trains weak classifiers to strong one). The training of the regressors is done as follows:

1. Initialise
   $$F_0(I, S(t)) = \sum_{i=1}^{n} (\Delta S_i(t) - \gamma)^2$$
   ,
   where gamma is the error correction term and ΔS is the error from the previous iteration.
2. For k epochs, the regressor r is given by:
   $$r_{ik} = \Delta S_i(t) - f_{k-1}(I_i, \; S_i(t))$$
3. Fit a regression tree to the regressor r, and update f(I, S) with the help of gamma and learning rate.

Once the regressors are trained, they are applied to the image and and the shape and learnt until the desired accuracy is obtained.

The histogram of gradients technique along with the cascade of regressor model comes pre-built in the dlib library of Python. The face predictor is implemented based on the Sullivian Kazemi method. The facial landmarks are approximated and reduced to 68 facial landmark points as shown in the figure below. This model has been tested on 300 test images (iBug 300W dataset) and had an accuracy of 99.3% as described in the paper proposed by Sagonas et. al. [20].

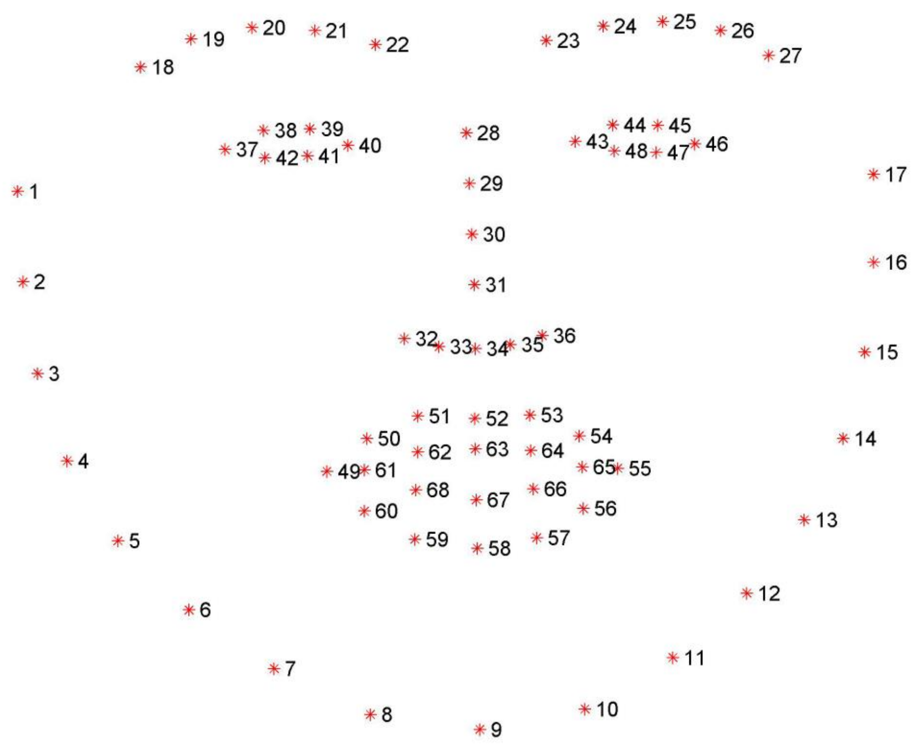
Figure 2.3 68 point facial landmarks diagram

The facial landmarks are loaded in system for ease of mapping using the following set of commands:

```
SHAPE_PREDICT = "shape_predictor_68_face_landmarks.dat"
predictor = dlib.shape_predictor(SHAPE_PREDICT)
# Detect face using dlib's facial detector
detector = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
shape1 = predict(gray, dlib_rect) # where dlib_rect contains info on the no. of
faces
```

The 68 landmarks can be converted to various data formats for manipulation and mapping facial regions. A real-time landmark marking is shown in the figure below:

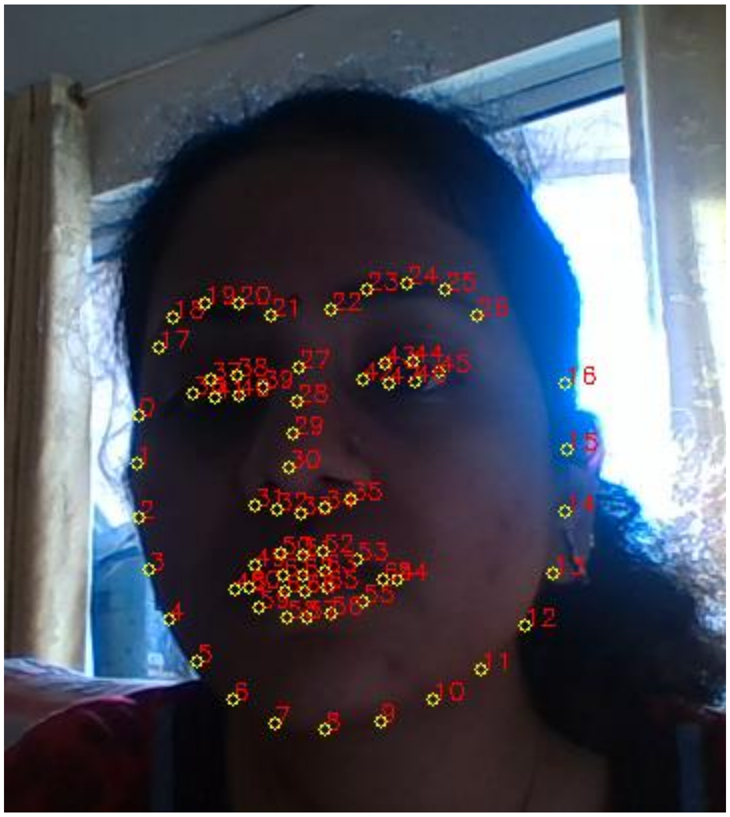
Figure 2.4 Real time visualisation of the 68 point facial landmarks

2.5 CALCULATING ASPECT RATIO AND ESTIMATING DROWSINESS

Once the the landmarks are obtained, the features - the eyes and mouth region are extracted. The aspect ratio is calculated using the traditional Euclidean distance formula:

$$d(p,\ q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

where i is the i th coordinate in n coordinate system.

From the 68 point facial landmark model, there are twelve points that map the eyes, with points for each side. Four points lie along the vertical (2 points forming a line), and two lie along the horizontal axis. The distance between these set of points is calculated using the Euclidean formula, and the aspect ratio is calculated as:

$$EAR = \frac{d_H}{d_v} = \frac{V_1 + V_2}{2H}$$

, where V1 and V2 are the set of vertical distances between those four points. The average aspect ratio for both the eyes is taken while estimating drowsiness.

For the mouth, the yawning is observed for monitoring the drowsiness. A total of twenty points are used to map the mouth, as shown in the figure:

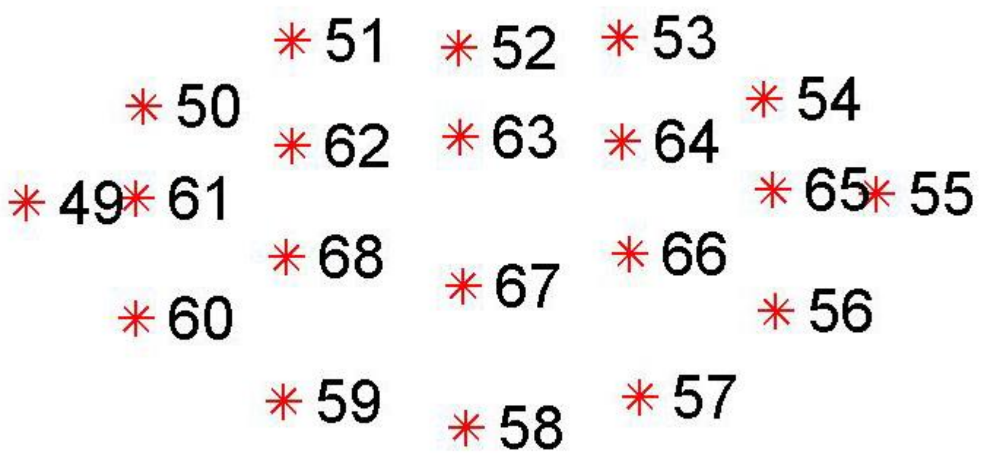

Figure 2.5 Facial landmarks marking the yawning region

In the figure above the outermost points that mark the lips are used for estimating drowsiness (distance between points 52 and 58). The condition that decides drowsiness is a combination of the aspect ratio of eye and mouth as shown:

```
if eye_closed >= 40 or yawns >= 5:
       cv2.putText(image, "You are drowsy, quit driving", (50, 50),
                   cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
       # playsound("Alarm.mp3")
```

The source code for the drowsiness detector can be found in the appendix.

CHAPTER 3 - RESULTS AND CONCLUSION

3.1 TESTING RESULTS

The system was tested using the built in web camera of the Dell Inspiron laptop with a video resolution of 1280 × 720 (HD) pixels at 30 fps (frames per second). It was tested in the following conditions. The resultant frames are displayed as shown:

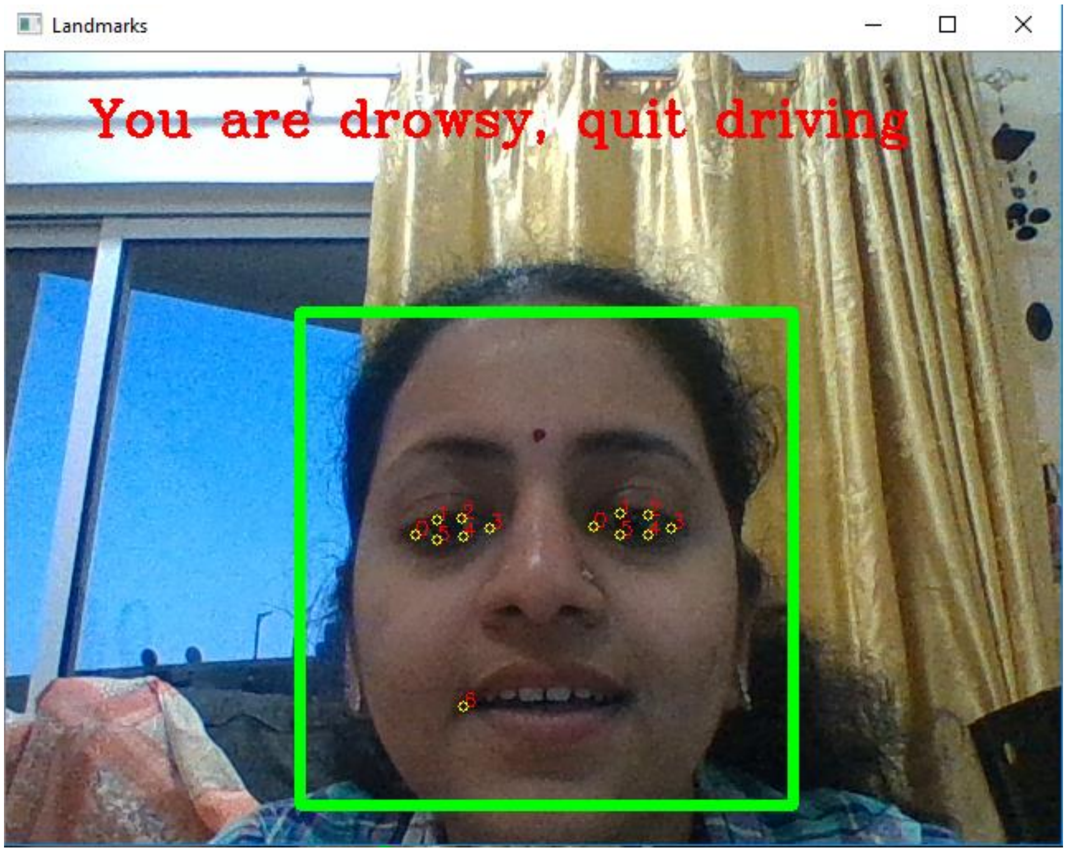

Figure 3.1 Testing under good illumination, without glasses

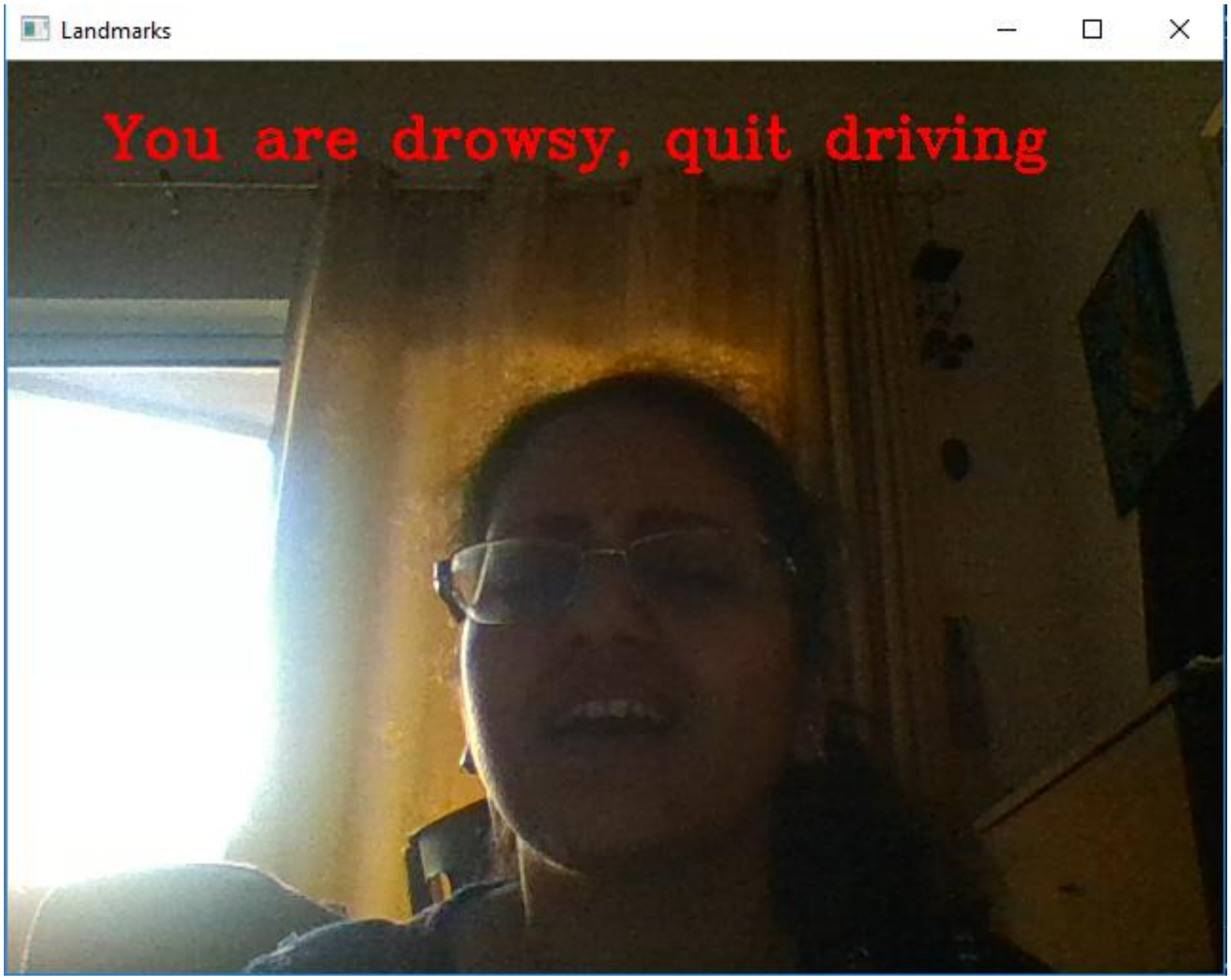

Figure 3.2 Testing under poor illumination, with glasses.

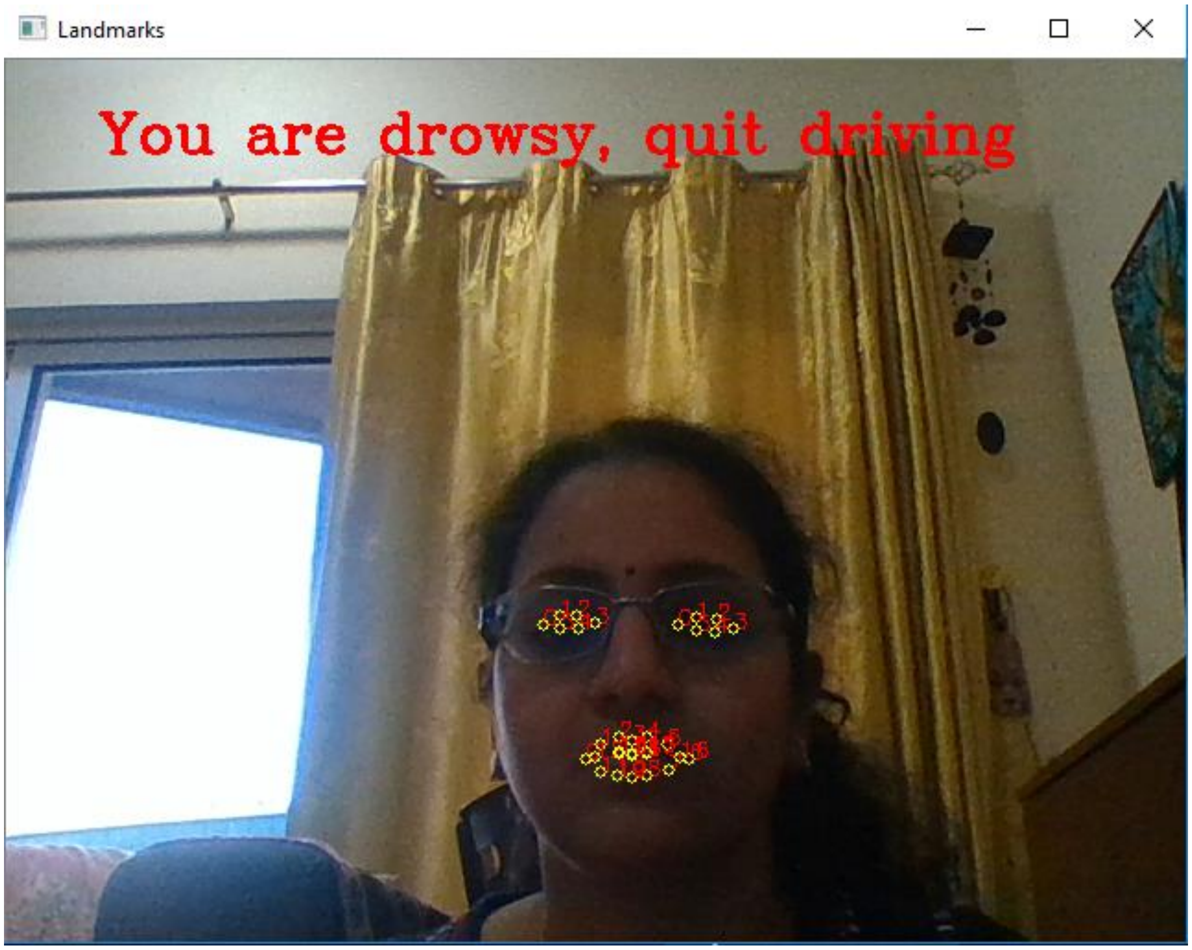

Figure 3.3 Testing under good illumination, with glasses

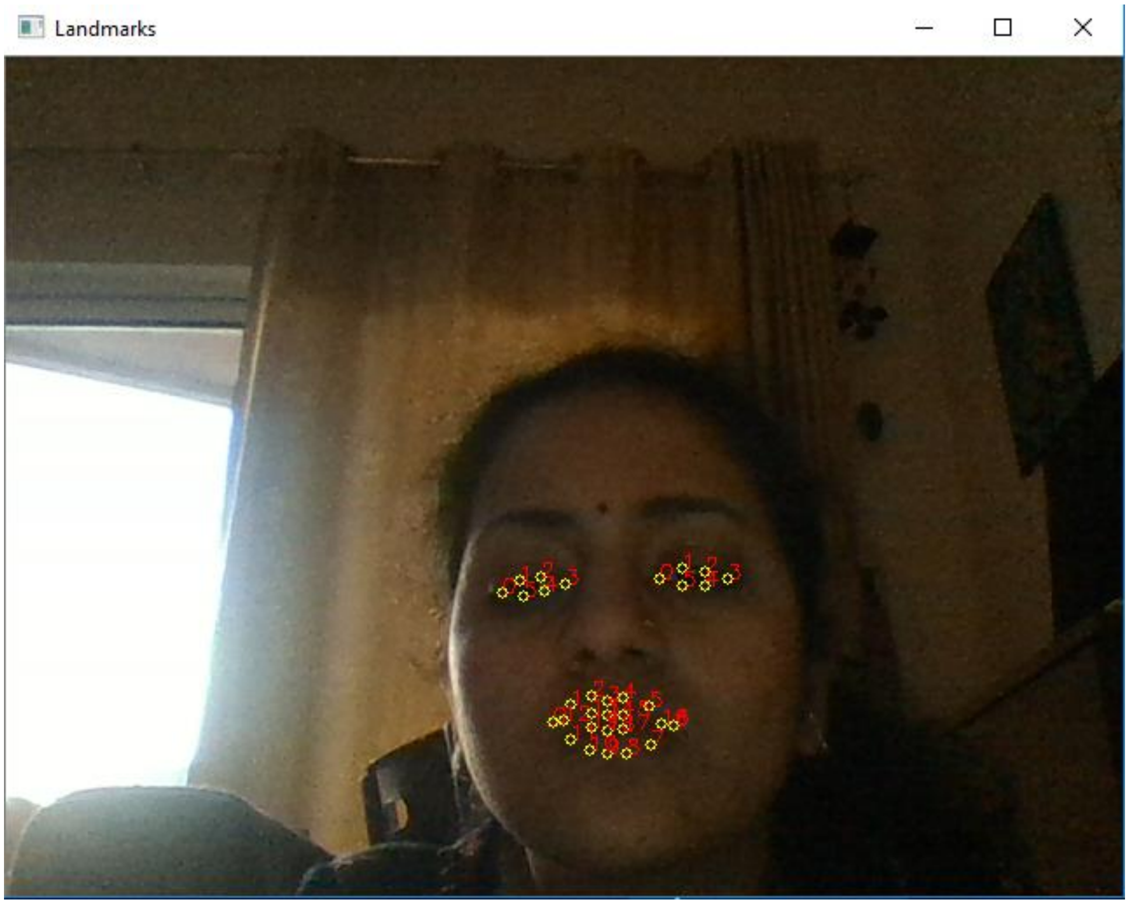

Figure 3.4 Testing under poor illumination, with glasses

From the above screenshots, it is clear that the detector accurately detects drowsiness when either of eyes or mouth is open, without glasses. The detector however fails to annotate facial landmarks nor does it detect drowsiness when glasses were used in combination with poor lighting conditions.

The system was also tested on the DROZY database (the ULg Multimodality Drowsiness database). The database consists of video recording of 14 subjects (3 males and 11 females)of performing 10 minute psychomotor vigilance tests (PVTs) subject to sleep deprivation induced by prolonged waking. The following results were obtained:

Subject Type	**Detecting Landmarks (Y / N)**	**Detecting Drowsiness (Y / N)**
1	Y	Y
3	N	Y
7	N	Y

9	Y	N
14	N	Y

3.2 CONCLUSION

A drowsiness detection system was developed by constantly monitoring the eyes and the mouth and was its performance was evaluated in different illumination conditions and situations. From the testing, it was evident that the detector performed better for situations where there were no glasses, irrespective of the illumination condition. However, considerable distortion was observed when the subject wore glasses, as seen in the figures above. The detector fails to annotate landmarks when it is subjected to poor illumination and the subject wore glasses.

In terms of system speed, the detector performed well on Windows systems without any hang ups or delays.

3.3 FUTURE SCOPE

The system can be modified to be deployed on an infrared camera to monitor situations in night driving, for better monitoring and and evaluation of drowsiness. As most drowsiness related accidents tend to happen at night, it is necessary that the system evaluates drowsiness even in the most difficult conditions. The 68 point landmark model can also be replaced with the 198 point facial landmark model, which improves the accuracy of estimating drowsiness.

The system can also be implemented on an embedded system platform such as Raspberry Pi for a readily available detector that is installed in a vehicle. Raspberry Pi typically comes pre-installed with Raspbian, an OS that is built using Python. Therefore, the detector will easily work in Raspberry Pi. It is also flexible, as it can also support other OS such as the Linux distributions and the Windows 10 IoT core, which in turn also support Python. Therefore a system can be rigged and deployed in the vehicles to monitor drowsiness.

REFERENCES

[1] “[No title].” [Online]. Available: https://www.ghsa.org/sites/default/files/2017-02/Drowsy%202016-U.pdf. [Accessed: 13-May-2019].

[2] “India way behind 2020 target, road accidents still kill over a lakh a year | India News - Times of India,” *The Times of India*, 11-May-2019. [Online]. Available: https://timesofindia.indiatimes.com/india/india-way-off-road-safety-targets-for-2020-road-accidents-still-kill-over-a-lakh-a-year/articleshow/65765549.cms. [Accessed: 13-May-2019].

[3] N. R. Pal *et al.*, “EEG-Based Subject- and Session-independent Drowsiness Detection: An Unsupervised Approach,” *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. 1. 2008.

[4] “Website.” [Online]. Available: Nissan, “Nissan’s ”Driver Attention Alert“ helps detect erratic driving caused by drowsiness and inattention,” 1 April 2015. [Online]. Available: https://nissannews.com/en-US/nissan/usa/releases/nissan-s-driver-attention-alert-helps-detect-erratic-driving-caused-by-drowsiness-and-inattention. [Accessed: 13-May-2019].

[5] “Website.” [Online]. Available: Volkswagen, “Driver Alert System,” [Online]. Available: https://www.volkswagen-newsroom.com/en/driver-alert-system-3932. [Accessed 20 April 2019]. [Accessed: 13-May-2019].

[6] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. .

[7] L. Lang and H. Qi, “The Study of Driver Fatigue Monitor Algorithm Combined PERCLOS and AECS,” *2008 International Conference on Computer Science and Software Engineering*. 2008.

[8] P. R. Tabrizi and R. A. Zoroofi, “Drowsiness Detection Based on Brightness and Numeral Features of Eye Image,” *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. 2009.

[9] T. D’Orazio, M. Leo, C. Guaragnella, and A. Distante, “A visual approach for driver inattention detection,” *Pattern Recognition*, vol. 40, no. 8. pp. 2341–2355, 2007.

[10] J. Yu, S. Park, S. Lee, and M. Jeon, “Driver Drowsiness Detection Using Condition-Adaptive Representation Learning Framework,” *IEEE Transactions on Intelligent Transportation Systems*. pp. 1–13, 2018.

[11] Masoud Alajmi , Fayas Asharindavida , Hedi Khammari , Irfan Ahmed and Mehedi Masud, “Effective Detection and Classification of Drowsiness using Clustering and Support Vector Machines,” *Journal of Engineering and Applied Sciences*, vol. 14, no. 6, pp. 1867–1874, 2019.

[12] V. Kazemi and J. Sullivan, “One millisecond face alignment with an ensemble of regression trees,” *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[13] “OpenCV | Willow Garage.” [Online]. Available: http://www.willowgarage.com/pages/software/opencv. [Accessed: 15-May-2019].

[14] “About.” [Online]. Available: https://opencv.org/about/. [Accessed: 15-May-2019].

[15] D. E. King, “Dlib-ml: A Machine Learning Toolkit,” *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, Jul. 2009.

[16] “dlib C++ Library.” [Online]. Available: http://dlib.net/. [Accessed: 15-May-2019].
[17] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” *Pattern Recognition*, vol. 29, no. 1. pp. 51–59, 1996.
[18] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1. pp. 119–139, 1997.
[19] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. .
[20] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “300 Faces in-the-Wild Challenge: The First Facial Landmark Localization Challenge,” *2013 IEEE International Conference on Computer Vision Workshops*. 2013.

APPENDIX - SOURCE CODE FOR THE DROWSINESS DETECTOR

```python
import numpy as np
import cv2
import dlib
import imutils
from imutils import face_utils
from scipy.spatial import distance as dist
# from playsound import playsound

# Load the models
FACE_DETECTOR = "lbpcascade_frontalface.xml"
SHAPE_PREDICT = "shape_predictor_68_face_landmarks.dat"
lbp_cascade = cv2.CascadeClassifier(FACE_DETECTOR)
predictor = dlib.shape_predictor(SHAPE_PREDICT)


def get_landmarks(im):
   # Detect faces
   rects = lbp_cascade.detectMultiScale(im, scaleFactor=1.1, minNeighbors=5)

   if len(rects) > 1:
       return "error"
   if len(rects) == 0:
       return "error"

   # Convert openCV rectangle to dlib rectangle
   for (x, y, w, h) in rects:
       dlib_rect = dlib.rectangle(int(x), int(y), int(x+w), int(y+h))

   return np.matrix([[p.x, p.y] for p in predictor(im, dlib_rect).parts()])

# Extract eye regions:
def eye_region(im, landmarks):

   left_eye = []
   left_eye = landmarks[36:42]
   im = annotate(im, left_eye)

   right_eye = []
   right_eye = landmarks[42:48]
   im = annotate(im, right_eye)
```

```python
    EAR_left = eye_aspect_ratio(left_eye)
    EAR_right = eye_aspect_ratio(right_eye)
    EAR = (EAR_left + EAR_right) / 2
    return im, EAR

# Annotate landmarks
def annotate(im, landmarks):

    for idx, point in enumerate(landmarks):
        pos = (point[0, 0], point[0, 1])
        cv2.putText(im, str(idx), pos,
                    fontFace=cv2.FONT_HERSHEY_SCRIPT_SIMPLEX,
                    fontScale=0.4,
                    color=(0, 0, 255))
        cv2.circle(im, pos, 3, color=(0, 255, 255))

    return im

# Calculate eye aspect ratio
def eye_aspect_ratio(parts):
    A = dist.euclidean(parts[0, 1], parts[3, 1])
    B = dist.euclidean(parts[1, 1], parts[4, 1])
    C = dist.euclidean(parts[2, 1], parts[5, 1])
    aspect_ratio = (A+B) / (2.0*C)
    return aspect_ratio

# Extract top lip points
def top_lip(landmarks):
    top_lip_pts = []
    for i in range(50, 53):
        top_lip_pts.append(landmarks[i])
    for i in range(61, 64):
        top_lip_pts.append(landmarks[i])
    top_lip_all_pts = np.squeeze(np.asarray(top_lip_pts))
    top_lip_mean = np.mean(top_lip_pts, axis=0)
    return int(top_lip_mean[:, 1])

# Extract bottom lip points
def bottom_lip(landmarks):
    bottom_lip_pts = []
    for i in range(65, 68):
        bottom_lip_pts.append(landmarks[i])
    for i in range(56, 59):
        bottom_lip_pts.append(landmarks[i])
    bottom_lip_all_pts = np.squeeze(np.asarray(bottom_lip_pts))
    bottom_lip_mean = np.mean(bottom_lip_pts, axis=0)
    return int(bottom_lip_mean[:, 1])

# Estimate yawning
def yawning(im, landmarks):
    lip = landmarks[48:68]
    im = annotate(im, lip)

    upper_lip = top_lip(landmarks)
    lower_lip = bottom_lip(landmarks)
    lip_distance = abs(upper_lip - lower_lip)

    return im, lip_distance

# Function for getting eye parameters
def eye_drowsy(im):
    landmarks_np = get_landmarks(im)

    if landmarks_np == "error":
        return im, 0

    im_landmarks, EAR = eye_region(im, landmarks_np)
    return im_landmarks, EAR

# Function for getting mouth parameters
def yawn(im):
    landmarks_np = get_landmarks(im)

    if landmarks_np == "error":
        return im, 0
    im_landmarks, lip_dist = yawning(im, landmarks_np)

    return im_landmarks, lip_dist


# Test in Webcam
cap = cv2.VideoCapture(0)
eye_closed = 0
yawns = 0
eye_status = False

while(True):
    # Capture frame-by-frame
    prev_eye_status = eye_status
    ret, frame = cap.read()
    im1, EAR = eye_drowsy(frame)
    image, lip_dist = yawn(frame)

    # Test drowsiness
    if EAR <= 0.25:
        eye_status = True
    else:
        eye_status = False

    if prev_eye_status == True and eye_status == True:
        eye_closed += 1

    if lip_dist >= 30:
        yawns += 1
        cv2.putText(image, "You are yawning", (50, 450),
                    cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)

    if eye_closed >= 40 or yawns >= 5:
        cv2.putText(image, "You are drowsy, quit driving", (50, 50),
                    cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
        # playsound("Alarm.mp3")

    # Display the resulting frame
    cv2.imshow('frame', frame)
    cv2.imshow('Landmarks', image)
    if cv2.waitKey(1) == 13:
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Driver drowsiness detection using Facial Recognition Techniques

ORIGINALITY REPORT

7%	3%	2%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Sheffield Student Paper	1%
2	Submitted to University of East London Student Paper	1%
3	Submitted to University of Liverpool Student Paper	1%
4	Submitted to Indian Institute of Technology, Madras Student Paper	1%
5	python.gurmezin.com Internet Source	<1%
6	Hongshuai Zhang, Zhiyi Qu, Liping Yuan, Gang Li. "A face recognition method based on LBP feature for CNN", 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017 Publication	<1%
7	cars.it.uu.se Internet Source	

<1%

8	Submitted to Universiti Sains Malaysia Student Paper	<1%
9	Submitted to Universiti Teknologi MARA Student Paper	<1%
10	www.pyimagesearch.com Internet Source	<1%
11	Submitted to University of Ballarat Student Paper	<1%
12	Submitted to Birla Institute of Technology and Science Pilani Student Paper	<1%
13	link.springer.com Internet Source	<1%
14	Lecture Notes in Computer Science, 2015. Publication	<1%
15	Jeremy N. Bailenson, Emmanuel D. Pontikakis, Iris B. Mauss, James J. Gross et al. "Real-time classification of evoked emotions using facial feature tracking and physiological responses", International Journal of Human-Computer Studies, 2008 Publication	<1%
	vbn.aau.dk	

16	Internet Source	<1%
17	www.jourlib.org Internet Source	<1%
18	www.robesafe.es Internet Source	<1%

Exclude quotes	On	Exclude matches	< 6 words
Exclude bibliography	On		