A project report on

DROWSINESS DETECTION & ALERT SYSTEM

Submitted in partial fulfillment of the requirements for the Degree of

B. Tech in Computer Science & System Engineering

by

Borneel Bikash Phukan (1728207)
Saptami Das (1728208)
Anmol Sharma (1728219)
Neelay Chaudhury (1728221)
Vishaka Sinha (1728237)

under the guidance of

Prof. Tanmaya Swain

School of Computer Engineering

Kalinga Institute of Industrial Technology
Deemed to be University
Bhubaneswar

December 2020

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
Deemed to be University U/S 3 of the UGC Act, 1956

CERTIFICATE

This is to certify that the project report entitled " **Drowsiness Detection & Alert System"** submitted by

Borneel Bikash Phukan	**1728207**
Saptami Das	**1728208**
Anmol Sharma	**1728219**
Neelay Choudhury	**1728221**
Vishakha Sinha	**1728237**

in partial fulfillment of the requirements for the award of the **Degree of Bachelor of Technology** in **Computer Science & System Engineering** is a bonafide record of the work carried out under my guidance and supervision at School of Computer Engineering, Kalinga Institute of Industrial Technology, Deemed to be University.

Signature of Supervisor

Prof. Tanmaya Swain
Assistant Dean
KIIT University, Bhubaneswar

....

The Project was evaluated on 04/12/2020

ACKNOWLEDGEMENTS

We are profoundly grateful to Prof. Tanmay Swain for his expert guidance and encouragement throughout to see that this project rights its target since its commencement to its completion. The work is a team effort minus which the completion of this project was not possible.

BORNEEL BIKASH PHUKAN
SAPTAMI DAS
ANMOL SHARMA
NEELAY CHOUDHURY
VISHAKHA SINHA

ABSTRACT

According to a WHO report on road accidents in India, sleep deprived drivers are responsible for 40% of all the road accidents in India including thousands of death every year. Several laws and restrictions have been enforced prevent such accidents but with minimal effect. Therefore companies like Tesla, BMW and Audi had developed their own auto-pilot mode which assists sleep deprived drivers. However huge cost and maintenance often deprives the middle class Indian drivers from availing such features. Therefore this project is aimed at developing an innovative and affordable approach to drowsiness/sleep detection and alert system which uses facial landmark detection and requires no external hardware but only a dashboard camera and a computing device.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 PREFACE

Indian roads are considered to be the most crowded in the world and in such crowded streets and highways, it is quite common for drivers occasionally falling asleep on the wheels. Such incidents often results in fatal accidents and according to WHO report, it is estimated that around 15000 people looses their lives every year as a result of road accidents. Rules and regulations are often ignored and people prefer the completion of their own tasks rather than their own and their passenger's safety. Even though numerous automobile industries had developed multiple accident averting systems like auto pilot mode, air bags etc. However these features are not affordable by the middle class Indian drivers and as safety should always be a prime concern, this project aims to provide the safety feature in which the system automatically detects the sleep deprivation state in the driver and raises an alarm to alert of impending disaster. The project is a demonstration of safety system and in the near future this project has the scope of seamlessly connecting multiple safety features like deacceleration, auto-pilot etc.

The following project is an OpenCV project and makes use of multiple OpenCV features and features developed in the former minor project. There are also multiple new external features like dlib and playsound which were developed in the last project and used here for facial landmark detection and various other tasks. Henceforth this project is an amalgam of new ideas and important insights from the former project.

The major concept behind this project is the detection of a human face, extract the landmarks (features) around the eyes and determine the eye aspect ratio and count the time upto which the eye aspect ratio is below a particular threshold and raise the alarm if the conditions are fulfilled.

All the concepts and the code implementation will be discretely explained in subsequent upcoming sections in this project report.

1.2 Related Works

Vehicle Control and Drowsiness Warning System

In 2002, the Swedish National Road and Transport Research Institute developed the first drowsiness detection system for commercial purpose. This system used a camera fitted on the steering wheel and the output channel involved the vehicle braking system and the signal lights. In case of any discrepancies in the driver's face, the system immediately switched on the brakes. However due to the use of primitive facial recognition system, the prototype was highly unreliable and therefore commercial production was cancelled. Nonetheless, this prototype paved the way to future research and safe implementation of the vehicle safety system.

Microsleep detection by Bosch Mobility Solution

Robert Bosch recently developed an integrated microsleep and fatigue detection system that comes preloaded in smart vehicle systems. The device has the ability to analyse 70 facial features in human to determine the type of issue being faced by the driver and accordingly change the vehicle speed, the destination and search for nearby gas stations, etc. However, this system is only sold integrated with the vehicles and therefore is not meant for the middle class drivers owning standard family vehicles.

Sleep sensors by Plessey Semiconductors

In 2017, Plessey Semiconductors had developed a sleep sensor that seemlessly fits with the vehicle dashboard console and monitors the facial features of the driver. In case of any anomalous expression involving microsleep or drowsiness, the dashboard opens up to serve coffee along with a bell alarm. However the major drawback was the extensive use of hardware elements which means increased cost and less affordability.

CHAPTER 2

BACKGROUND AND CONCEPTS

2.1 Requirement specification and Analysis

2.1.1 Hardware Requirements

Processor – We are using Intel® Core(TM) i5-7200U CPU @ 2.50GHz 2.71GHz processor for the purp speed and better load management that will be needed during the embeddings extraction and training of th model. Processor is a very important factor when it comes to generating accurate results.

RAM – Higher the RAM faster is the process in fetching the data from the secondary memory. The RAM reduces the efforts of fetching data from the secondary memory everytime the code runs an iteration procedure.

2.1.2 Software Requirements

IDE – A Python specific Integrated Development Environment (IDE) like VSCode, PyCharm or Spyder has been used for facilitated coding of the entire system. The IDE of choice depends upon the individual using it.

Python – The programming language plays a very important role in every development procedures. The reason for choosing python over other languages is due to the wider acceptability and larger community associated with it. Libraries like OpenCV, Numpy etc all are supported by Python which makes development easier.

OpenCV – OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. In this project, we are using OpenCV for the purpose of detecting faces from the camera and drawing the bounding boxes.

Argparse – The Argument Parser is a special tool in Python that is used for taking command line arguments during runtime. This method makes the entire process of taking components easier and thereby increases flexibility of the program.

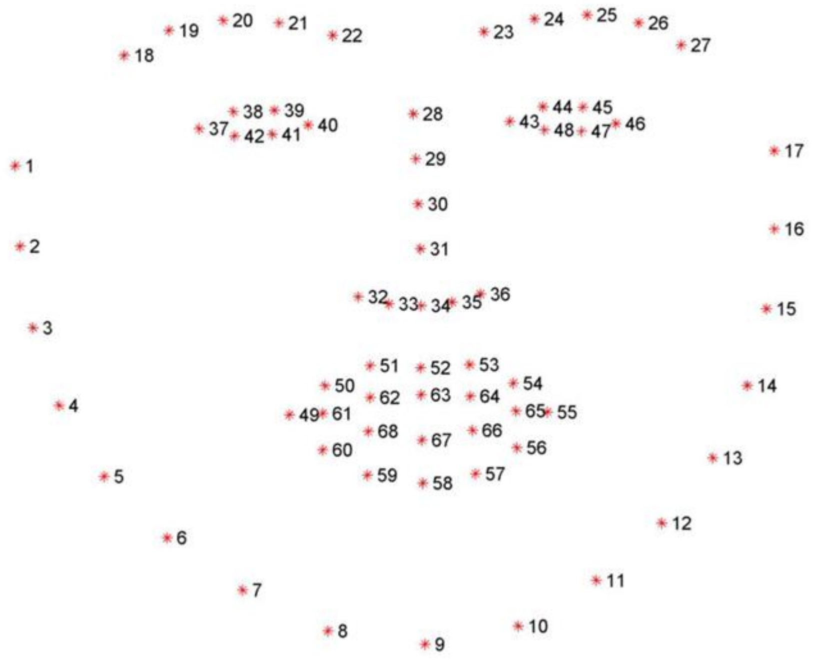
Numpy – Numpy is a special python tool that is used for performing numerical and scientific computation. In the field of machine learning, this is extremely important as it ensures that the program does not face any space or time complexity issues.

2.2.1 Concept of dlib

Dlib is a machine learning framework that is used to carry out various face recognition and facial landmark extraction tasks in real-time. The framework is originally written in C++ but has multiple Python binding which makes it versatile as well as faster to implement in low grade systems. In this context, we shall use the dlib to localize and extract the eye features from the detected human face. Let us understand how exactly dlib detects these facial features.

The dlib library uses Convolutional Neural Network to train and save the model and then uses histogram oriented gradient and linear SVM to detect human faces and determine the probability of a human face in real-time. The facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 point mappings were obtained by training a shape predictor on the labeled iBUG 300-W dataset. These facial regions can be accessed with simple python indexing.

Below we can visualize each of these 68 coordinates map to:

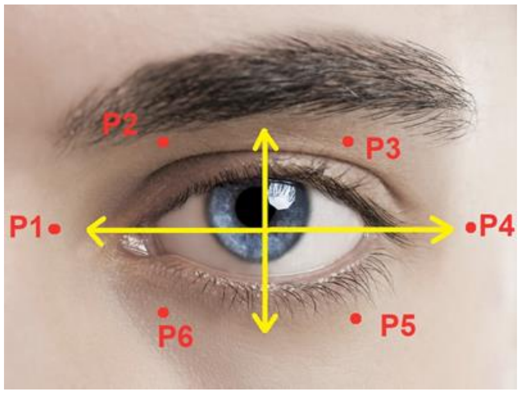


2.2.2 Eye Aspect Ratio (EAR)

The Eye Aspect Ratio is the ratio of distance between the vertical and horizontal eye landmarks or simply the ratio between the vertical distance of the eye and the horizontal distance of the eye. To calculate the distances, we use Euclidean distance formula between the two corresponding points, twice. The euclidean distance formula between two points is given below:

$$d(p,q) = \sqrt{(p1 - q1)2 + (p2 - q2)2}$$

The machine needs to measure the distance in real time in order to constantly keep track of the changing EAR. The EAR value of an open eye is any floating point variable in the range 0 to 1 and the EAR value of a closed eye is 0 or near to 0. The points of reference for calculating the euclidean distances are given below:



Once the distances has been measured, the next step is to calculate the EAR. The formula for EAR is given below:

$$EAR = \frac{\|P2 - P6\| + \|P3 - P5\|}{2\|P1 - P4\|}$$

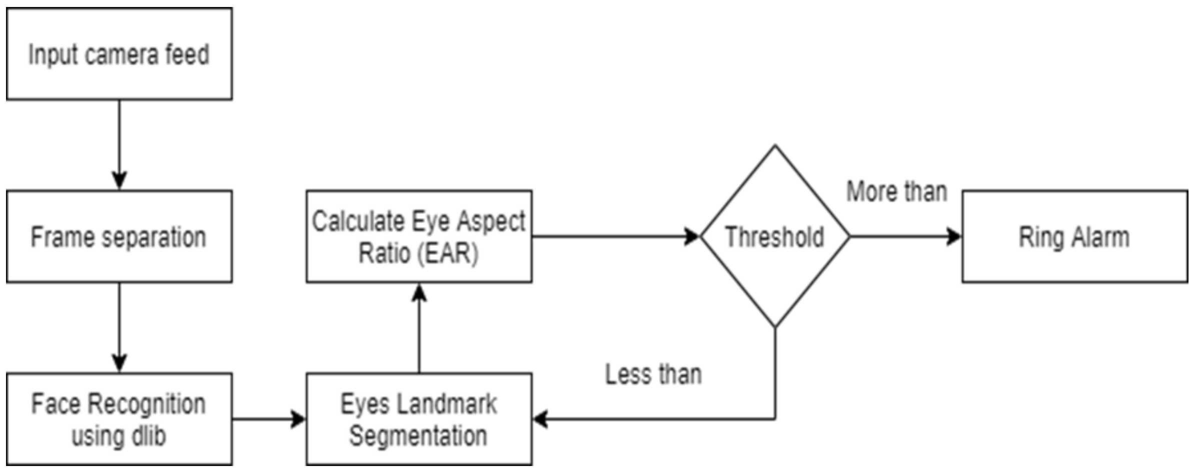
Finally the average of the EAR of the two eyes are calculated and implemented. For those EAR falling below 0.3, and lasting more than 5 seconds, the machine determines it as closed and for those more than 0.5 and lasting less than 5 seconds, the machine determines it as open. For a fully opened eye, EAR = 1, for closed eye, EAR = 0.

2.2.4 System Design

The entire model is built with special emphasis on computer vision. The dlib component is responsible for facial recognition followed by which the system focuses primarily on the eyes and calculates the Eye Aspect Ratio (EAR) with the help of the Euclidean distance formulae. The threshold value is the primary floating point value which determines if the alarm should go off or not. The basic structure or the blueprint of the project is given in section 2.2.1.

2.2.4 Blueprints and Components

The blueprint for the project is given below:


Input camera feed
Frame separation
Face Recognition using dlib
Eyes Landmark Segmentation
Calculate Eye Aspect Ratio (EAR)
Threshold
More than
Ring Alarm
Less than

CHAPTER 3

3. PROJECT ANALYSIS AND IMPLEMENTATION

The entire project is divided into 3 major parts. We shall implement and describe each parts in absolute details. The first part includes the import section where we'll import all the required libraries. The most important ones are imutils, threading, playsound, dlib and opencv.

```
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import playsound
import argparse
import imutils
import time
import dlib
import cv2
```

The next section includes the development of helper functions which shall be used as tools for the project. These helper functions are sound_alarm(), euclidean_distance(), eye_aspect_ratio().

Sound_alarm() - This single-lined function is triggered when the conditions for anomalous driver behaviour is detected.

```
def sound_alarm(path):
        playsound.playsound(path)
```

Euclidean_distance() - This function is used for calculating the euclidean distances between the landmarks of the eyes. This function is triggered for each frames of the video and calculates both vertical and horizontal landmarks of both the eyes. It returns a floating point value after calculation.

```
def euclidean_distance(point1, point2):
    return ((point1[0] - point2[0]) ** 2 + 550 / ((point1[1] + point2[1]) / 2) * (point1[1] - point2[1]) ** 2) ** 0.5
```

Eye_aspect_ratio() - The most cardinal function of all, this function is used for calculating the eye aspect ratio of the driver's eyes from each frames. It stores the two vertical landmarks and one horizontal landmark in three variables respectively and then inserts it into the EAR formula mentioned above.

```
def eye_aspect_ratio(eye):
    # vertical landmarks
    A = euclidean_distance(eye[1], eye[5])
    B = euclidean_distance(eye[2], eye[4])
    # horizontal landmark
    C = euclidean_distance(eye[0], eye[3])
    # compute the eye aspect ratio
    aspect_ratio = (A + B) / (2.0 * C)
    return aspect_ratio
```

Next we set up the resources for the model such as the facial landmarks, alarm tune file, eye threshold, timer variable etc.

```
shape_predictor = "./shape_predictor_68_face_landmarks.dat"

alarm_path = "./alarm.wav"

EYE_AR_THRESH = 0.3

EYE_AR_CONSEC_FRAMES = 48

COUNTER = 0

ALARM_ON = False
```

Next we initialize dlib's face detector (HOG-based) and then create the facial landmark predictor.

```
detector = dlib.get_frontal_face_detector()

predictor = dlib.shape_predictor(shape_predictor)
```

Then we grab the indexes of the facial landmarks for the left and right eye, respectively

```
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]

(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

Next we shall construct the framework for the OpenCV along with setting the required conditions.

```
video = VideoStream(0).start()

time.sleep(1.0)

while True:

frame = video.read()

frame = imutils.resize(frame, width=450)

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    rects = detector(gray, 0)
    # loop over the face detections
    for rect in rects:
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        # extract the left and right eye coordinates
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        #average the eye aspect ratio together for both eyes
        ear = (leftEAR + rightEAR) / 2.0
        Compute the convex hull for the left and right eye, then
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
```

Check to see if the eye aspect ratio is below the blink threshold,
If so, increment the blink frame counter

```
        if ear < EYE_AR_THRESH:
            COUNTER += 1
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
        if not ALARM_ON:
            ALARM_ON = True
```

Check to see if an alarm file was supplied, and if so, start a thread to have the alarm
sound played in the background

```
        if alarm_path != " ":
            t = Thread(target=sound_alarm, args=alarm_path)
            t.deamon = True
            t.start()
            cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

```
otherwise, the eye aspect ratio is not below the blink threshold, so reset the counter and alarm
        else:
            COUNTER = 0
            ALARM_ON = False
    draw the computed eye aspect ratio on the frame to help
        with debugging and setting the correct eye aspect ratio thresholds and frame counters
        cv2.putText(frame, "Eye Aspect Ratio: {:.2f}".format(ear), (300, 30),
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == 27:
            break
cv2.destroyAllWindows()
video.stop()
```

CHAPTER 4

4. RESULTS & DISCUSSIONS

Firstly, a camera is setup that monitors a stream for faces. If a face is found, we apply facial landmark detection using dlib and extract the landmark for the eye regions. Now that we have the eye regions, we can compute the eye aspect ratio to determine if the eyes are closed or not. If the eye aspect ratio indicates that the eyes have been closed for a sufficient period of time, the alarm functionality shall go off to alert the dozing driver. A screenshot of the working model is given below:

We also used blink detection using facial landmark detection and implemented the project using dlib, Python and OpenCV. User can use any camera mobile or laptop web camera. It is recommended to place the camera in a proper lighted place and also the distance between the face camera should not be too large.

Since our software uses lots of services like camera and different programs going on continuously in the laptop it will consume large amount of power, so for long distance phones or laptop must be kept plugged in. Performance of driver’s drowsiness detection will be highly diminished in complex dark place. So the camera must not be placed in complex dark place for this purpose.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The drowsiness detection and alert system is capable of detecting drowsiness, sleep deprivation and lack of focus of driver in a short time. The Drowsiness Detection System developed based on eye closure of the driver can differentiate normal eye blink and drowsiness and detect the drowsiness while driving. The proposed system can prevent 40% of the accidents that occur due to the sleeplessness while driving. The system works well even in case of drivers wearing spectacles and even under low light conditions if the camera delivers better output. Information about the head and eyes position is obtained through various self-developed image processing algorithms. During the monitoring, the system is able to decide if the eyes are opened or closed. When the eyes have been closed for too long a warning signal is issued which restores the driver's alertness level on the basis of continuous eye closures.

5.2 Future Work

The drivers drowsiness detection is based on an algorithm which begins with the determination of how long given persons eyes have been closed for. If their eyes have been closed for a certain amount of time, we'll assume that they are starting to doze off and play an alarm to wake them up and grab their attention. It warns the driver of drowsiness and the risk of microsleep; compliances with driver warnings helps to avoid crashes caused by fatigue. There are several different algorithms and methods for eye tracking, and monitoring. Most of them in some way relate to features of the eye (typically reflections from the eye) within a video image of the driver.

In order to prevent these devastating accidents, the state of drowsiness of the driver should be monitored. The following measures have been used widely for monitoring drowsiness:

Vehicle-based measures(A number of metrics, including deviations from lane position, movement of the steering wheel, pressure on the acceleration pedal, etc.,), Behavioral measures (The behavior of the driver, including yawning, eye closure, eye blinking, head pose, etc.,) Physiological measures (The correlation between physiological signals (electrocardiogram (ECG), electromyogram (EMG), electrooculogram (EoG) and electroencephalogram (EEG)) and driver drowsiness)

By mixing all the system's advantages we can develop an ultimate hybrid system to avoid the accidents from drowsiness.

REFERENCES

[1] Miaou, “Study of Vehicle Scrap pageRates,” Oak Ridge National Laboratory, Oak Ridge, TN, S.P., April 2012

[2] Wreggit, S. S., Kim, C. L., and Wierwille, W. W., FourthSemi-Annual Research Report”,Research on Vehicle-Based Driver Status Performance Monitoring”, Blacksburg, VA: Virginia Polytechnic Institute and State University, ISE Department, January 2013.

[3] Bill Fleming, “New Automotive Electronics Technologies”, International Conference on Pattern Recognition, pp. 484-488,December 2012.

[4] Ann Williamson and Tim Chamberlain,“Review of on-road driver fatigue monitoring devices”, NSW Injury Risk Management Research Centre, University of New South Wales,, July 2013.

[5] E. Rogado, J.L. García, R. Barea, L.M. Bergasa, Member IEEE and E. López, February, 2013, “Driver Fatigue Detection System”, Proceedings of the IEEE International Conference on Robotics and Biometics, Bangkok, Thailand.

[6] Boon-Giin Lee and Wan-Young Chung, Member IEEE, “Driver Alertness Monitoring Using Fusion of Facial Features and Bio-Signals”, IEEE Sensors Journal, VOL. 12, NO. 7, July 2012.[7]H. Singh, J. S. Bhatia, and J. Kaur, “Eye tracking based driver fatigue monitoring and warning system”, in Proc. IEEE IICPE, New Delhi, India, Jan. 2014.

[7] Eriksson, M and Papanikolopoulos, N.P, “Eye-tracking for Detection of Driver Fatigue”, IEEE Intelligent Transport System Proceedings, pp 314-319, September 2014

[8] Perez, Claudio A. et al,“Face and Eye Tracking Algorithm Based on Digital Image Processing”, IEEE System, Man and Cybernetics Conference, vol. 2 ,pp 1178-1188,, February 2012.

[9] Magdalena Lindman1, Jordanka Kovaceva, Daniel Levin, Bo Svanberg, Lotta Jakobsson, Henrik Wiberg,“A first glance at Driver Alert Control in FOT-data”, IRCOBI Conference,August 2012.

[10] Singh, Sarbjit and Papanikolopoulos, N.P, “Monitoring Driver Fatigue Using Facial Analysis Techniques”, IEEE Intelligent.