Project Report

on

# DRIVER DROWSINESS DETECTION

PYTHON FULL STACK DEVLOPEMENT

By

K. MANIDEEP REDDY (2010030309)

I. KAUSTUBH SASTRY (2010030064)

K.V. MANOHAR KARTHIK (2010030197)

M.ABHIRAM (2010030457)

Under the supervision of

# Ms. P. SREE LAKSHMI

# Assistant Professor



Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500 075, Telangana, India.

March, 2022

**01**

**DECLARATION**

The Project Report entitled “**DRIVER DROWSINESS DETECTION**” is a record of bonafide work of K. MANIDEEP REDDY (2010030309), I. KAUSTUBH SASTRY (2010030064), K.V. MANOHAR KARTHIK (2010030197), M. ABHIRAM (2010030457) submitted as a requirement for the completion of the course **PYTHON FULL STACK DEVLOPEMENT** in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/University/Institute.

Signature of the Students

K. MANIDEEP REDDY(2010030309)-

I. KAUSTUBH SASTRY(2010030064)-

M. ABHIRAM(2010030457)-

K.V. MANOHAR KARTHIK(2010030197)-

**02**

**CERTIFICATE**

This is to certify that the Project Report entitled “DRIVER DROWSINESS DETECTION” is a record of bonafide work of K. MANIDEEP REDDY (2010030309), I. KAUSTUBH SASTRY (2010030064), K.V. MANOHAR KARTHIK (2010030197), M. ABHIRAM (2010030457) submitted in partial fulfillment for the award of B. Tech in the Department of Computer Science and Engineering to the K L University, Hyderabad is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute

## Signature of the Supervisor

Ms. P. SREE LAKSHMI

Assistant Professor

## Signature of the HOD Signature of the External Examine

**03**

**ACKNOWLEDGEMENT**

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for  
   
completing this project successfully.

We take grateful opportunity to thank our beloved Founder and Chairman who has given constant encouragement during our course and motivated us to do this Social Internship. We are grateful to our Principal Dr. L. Koteswara Rao who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgement & sincere thanks to our Head of the Department Dr. Chiranjeevi Manike for his exemplary guidance, monitoring and constant encouragement throughout the course. We thank **Ms. P. Sree Lakshmi** of our department who has supported throughout this course holding a position of supervisor.

We whole heartedly thank all the teaching and non-teaching staff of our department without whom we won’t have made this project a reality. We would like to extend our sincere thanks especially to our parent, our family members and friends who have supported us to make this project a grand success.

**04**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE NO** |
| 1 | Abstract | 6 |
| 2 | Introduction | 7 |
| 3 | Literature Survey  &  System Requirements –  Hardware and Software | 8 - 10 |
| 4 | Flow Diagram | 11 |
| 5 | Proposed Algorithm | 12 - 14 |
| 6 | Implementation  🡪 Code  🡪 Results | 15 - 20 |
| 7 | Conclusion and Future Work | 21 |
| 8 | References | 22 |

**05**

**CHAPTER - 01**

**ABSTRACT**

Every year, traffic accidents due to human errors cause increasing amounts of deaths and injuries globally. To help reduce the number of fatalities, in the paper presented here, a new module for Driver Drowsiness detection (DDD) which deals with automatic driver drowsiness detection based on visual information and Artificial Intelligence is presented. The aim of this system is to locate, track, and analyze both the drivers face and eyes to compute a drowsiness index, where this real-time system works under varying light conditions (diurnal and nocturnal driving). Examples of different images of drivers taken in a real vehicle are shown to validate the algorithms used

Drowsy Driving is a deadly combination of driving and sleepiness. The number of road accidents due to Drowsy Driving is increasing at an alarming rate worldwide. Not having a proper sleep is the main reason behind drowsiness while driving. However, other reasons like sleep disorders, medication, alcohol consumption, or driving during night shifts can also cause drowsiness while driving.

**06**

**CHAPTER - 02**

**INTRODUCTION**

“1 in 25 adult drivers report that they have fallen asleep at the wheel in the past 30 days”

If you have driven before, you’ve been drowsy at the wheel at some point. It’s not something we like to admit but it’s an important problem with serious consequences that needs to be addressed. 1 in 4 vehicle accidents are caused by drowsy driving and 1 in 25 adult drivers report that they have fallen asleep at the wheel in the past 30 days. The scariest part is that drowsy driving isn’t just falling asleep while driving. Drowsy driving can be as small as a brief state of unconsciousness when the driver is not paying full attention to the road. Drowsy driving results in over 71,000 injuries, 1,500 deaths, and $12.5 billion in monetary losses per year. Due to the relevance of this problem, we believe it is important to develop a solution for drowsiness detection, especially in the early stages to prevent accidents.

Additionally, we believe that drowsiness can negatively impact people in working and classroom environments as well. Although sleep deprivation and college go hand in hand, drowsiness in the workplace especially while working with heavy machinery may result in serious injuries similar to those that occur while driving drowsily.

**07**

**CHAPTER - 03**

**3.1. LITERATURE SURVEY**

We as a Team, have studied various resources to learn about the different methods and to learn about the uses and effectiveness of the concepts i.e., machine learning concepts We used the available means to avail the resources, we mainly used Internet as the primary source.

* The international journal of engineering research and technology’s article: A Machine-Learning Approach for Driver-Drowsiness Detection based on Eye-State, the article suggests, “Generally, the driving person feels drowsy due to continues driving for long hours or Physical illness or might be drunken and this leads to major road accidents. Our aim is to detect the drowsiness, make them alert to prevent accidents and generate a notification in the app and an alarm sound.”
* Detection and prediction of driver drowsiness using artificial intelligence, suggests, that the driver drowsiness, can be monitored through different parameters such as, eye monitoring, yawning, vehicle driving behavior.
* An Efficient Approach for Detecting Driver Drowsiness Based on Deep Learning, suggests different methods for the predictions, they mainly talked about 2 methods   
   Drowsiness Detection and Prediction Based on Facial Landmarks and prediction based on the vehicle drive pattern detection, they concluded by saying facial landmarks as the best methodology.

From the literature survey we conducted, we learnt that Simulators basically provides the

complete analysis of network topology by giving the information about its nodes and links between those nodes etc. some simulators, provide information based on graphics and some are programming based which help to develop a particular application to analyze a network.

**3.2. SYSTEM REQUIREMENTS**

**SOFTWARE REQUIREMENTS: -**

* Operating system - Windows 10
* Tools – PyCharm Community Edition

**HARDWARE REQUIREMENTS:** -

* RAM - 8.00 GB (7.87 GB usable)
* Processor - Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz 2.50 GHz
* System-type - 64-bit operating system, x64-based processor
* Version - 20H2
* Edition - Windows 10 Home Single Language

**10**

**CHAPTER – 04**

**FLOWCHART**

High vision cameras are embedded to monitor and capture to extract frames one by one and generate the alerts accordingly. Each extracted frame is analyzed to study the pattern of facial features; using Haar Cascade Classifier and determined Eye Aspect Ratio (EAR) and Mouth Aspect Ratio(MAR) for each frame. EAR and MAR values exceed their respective threshold values, a blink and a yawn is considered respectively. The system alerts the driver by playing an alarm if eye blinking rate and yawns are suspected for a certain number of consecutive frames. The alarm is activated to grab the driver’s attention and it keeps on ringing until driver wakes up. Diagram

Description automatically generated

**Fig 4.1: showing flowchart of the problem**

**11**

**CHAPTER – 05**

**5.1 PROPOSED ALGORITHM**

## A vision-based real time driver fatigue detection system based on the distance of the eyelid has built by Dong and Wu (2005). It located the face using the characteristics of skin colors and then it detected the eyes using projection and dynamic template matching.

## Steering pattern monitoring primarily uses steering input from electric power steering system. Monitoring a driver this way only works if a driver actually steers a vehicle actively instead of using an automatic lane-keeping

## Physiological measurement requires body sensors to measure parameters like brain activity, heart rate, skin conductance, muscle activity.

## Eye Aspect Ratio (EAR)

## EAR, as the name suggests, is the ratio of the length of the eyes to the width of the eyes. The length of the eyes is calculated by averaging over two distinct vertical lines across the eyes as illustrated in the figure below.

## Logo Description automatically generated with medium confidence

## Fig5.1.1 ear

## Our hypothesis was that when an individual is drowsy, their eyes are likely to get smaller and they are likely to blink more. Based on this hypothesis, we expected our model to predict the class as drowsy if the eye aspect ratio for an individual over successive frames started to decline i.e. their eyes started to be more closed or they were blinking faster.

## Mouth Aspect Ratio (MAR)

## Computationally similar to the EAR, the MAR, as you would expect, measures the ratio of the length of the mouth to the width of the mouth. Our hypothesis was that as an individual becomes drowsy, they are likely to yawn and lose control over their mouth, making their MAR to be higher than usual in this state.

## A picture containing text Description automatically generated

## Fig 5.1.2 mar

## 12

**Pupil Circularity (PUC)**

PUC is a measure complementary to EAR, but it places a greater emphasis on the pupil instead of the entire eye.

Text, letter

Description automatically generated

For example, someone who has their eyes half-open or almost closed will have a much lower pupil circularity value versus someone who has their eyes fully open due to the squared term in the denominator. Similar to the EAR, the expectation was that when an individual is drowsy, their pupil circularity is likely to decline.

**Mouth aspect ratio over Eye aspect ratio (MOE)**

Finally, we decided to add MOE as another feature. MOE is simply the ratio of the MAR to the EAR.

Text

Description automatically generated

The benefit of using this feature is that EAR and MAR are expected to move in opposite directions if the state of the individual changes. As opposed to both EAR and MAR, MOE as a measure will be more responsive to these changes as it will capture the subtle changes in both EAR and MAR and will exaggerate the changes as the denominator and numerator move in opposite directions. Because the MOE takes MAR as the numerator and EAR as the denominator, our theory was that as the individual gets drowsy, the MOE will increase.

The main aim is to detect drowsiness of driver, it can be done in different ways like detecting facial expression of the driver and measuring Eye Aspect Ratio (EAR). Blinking pattern is different for each and every individual. The pattern gets varied in terms of squeezing degree of eye, blink duration and speed of closing and opening the eye [16]. The proposed method involved with the following methodologies such as Haar Cascade Classifiers, Shape Predictor\_68\_facial landmark detection, Eye Aspect Ratio (EAR), Ubidots cloud service and Twilio API

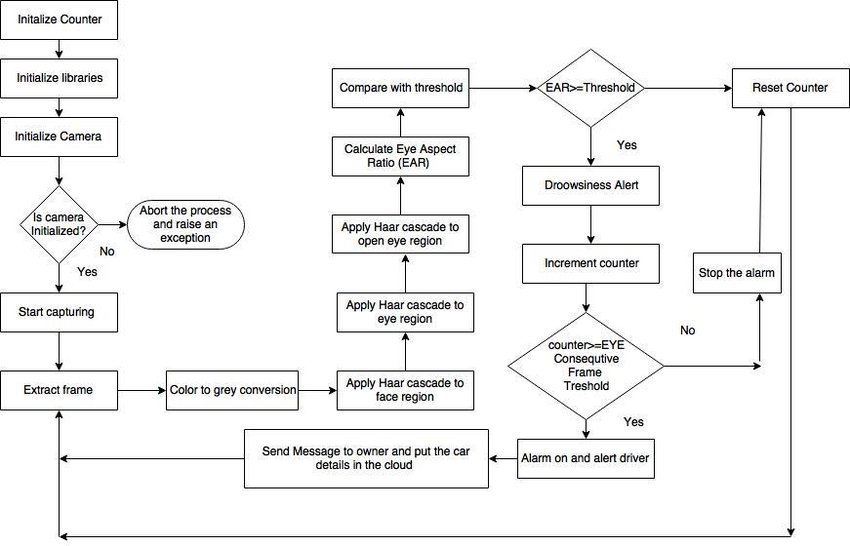
**Haar Cascade Classifiers**

In Haar Cascade Classifiers, a lot of similar and dissimilar images are trained in order to detect fatigue of the driver. OpenCV is a learning-based method, packed with a detector as well as a trainer. For training, a separate database is maintained for face and eye with several positive and negative images having eye closed and opened conditions and different set facial images [15]. In 2013, Patil et al suggested a vision-based drowsiness is carried out using Support Vector Machine and Haar Cascade Classifiers [14].

**13**

**Algorithm:**

In order to initiate the program execution, it will import the following libraries like numpy, OpenCV, play sound, argparse, dlib, distance, timer. The entire algorithm for drowsiness detection is shown with the help of a flowchart shown

.**Fig 5.1.3 showing the algorithm** .

**14**

**CHAPTER – 06**

**6.1. IMPLEMENTATION**

We developed a code in python language, using the Algorithm stated above

The code serves purpose of locating Facial landmark(eye) and monitoring it constantly(to check the frequency and time period between blinks)

Graphical user interface, text

Description automatically generated

**6.2 CONSOLE CODE**

import dlib

import sys

import cv2

import time

import numpy as np

from scipy.spatial import distance as dist

from threading import Thread

import playsound

import queue

# from light\_variability import adjust\_gamma

FACE\_DOWNSAMPLE\_RATIO = 1.5

RESIZE\_HEIGHT = 460

thresh = 0.27

modelPath = "models/shape\_predictor\_70\_face\_landmarks.dat"

sound\_path = "alarm.wav"

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

predictor = dlib.shape\_predictor(modelPath)

leftEyeIndex = [36, 37, 38, 39, 40, 41]

rightEyeIndex = [42, 43, 44, 45, 46, 47]

blinkCount = 0

drowsy = 0 **15**

state = 0

blinkTime = 0.15 #150ms

drowsyTime = 1.0 #1200ms

ALARM\_ON = False

GAMMA = 1.5

threadStatusQ = queue.Queue()

invGamma = 1.0/GAMMA

table = np.array([((i / 255.0) \*\* invGamma) \* 255 for i in range(0, 256)]).astype("uint8")

def gamma\_correction(image):

return cv2.LUT(image, table)

def histogram\_equalization(image):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

return cv2.equalizeHist(gray)

def soundAlert(path, threadStatusQ):

while True:

if not threadStatusQ.empty():

FINISHED = threadStatusQ.get()

if FINISHED:

break

playsound.playsound(path)

def eye\_aspect\_ratio(eye):

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

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

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

ear = (A + B) / (2.0 \* C)

return ear

def checkEyeStatus(landmarks):

mask = np.zeros(frame.shape[:2], dtype = np.float32)

hullLeftEye = []

for i in range(0, len(leftEyeIndex)):

hullLeftEye.append((landmarks[leftEyeIndex[i]][0], landmarks[leftEyeIndex[i]][1]))

cv2.fillConvexPoly(mask, np.int32(hullLeftEye), 255)

hullRightEye = []

for i in range(0, len(rightEyeIndex)):

hullRightEye.append((landmarks[rightEyeIndex[i]][0], landmarks[rightEyeIndex[i]][1]))

cv2.fillConvexPoly(mask, np.int32(hullRightEye), 255)

# lenLeftEyeX = landmarks[leftEyeIndex[3]][0] - landmarks[leftEyeIndex[0]][0]

# lenLeftEyeY = landmarks[leftEyeIndex[3]][1] - landmarks[leftEyeIndex[0]][1]

# lenLeftEyeSquared = (lenLeftEyeX \*\* 2) + (lenLeftEyeY \*\* 2)

# eyeRegionCount = cv2.countNonZero(mask)

# normalizedCount = eyeRegionCount/np.float32(lenLeftEyeSquared)

#############################################################################

leftEAR = eye\_aspect\_ratio(hullLeftEye)

rightEAR = eye\_aspect\_ratio(hullRightEye)

ear = (leftEAR + rightEAR) / 2.0

#############################################################################

eyeStatus = 1 # 1 -> Open, 0 -> closed

if (ear < thresh):

eyeStatus = 0

return eyeStatus **16**

def checkBlinkStatus(eyeStatus):

global state, blinkCount, drowsy

if(state >= 0 and state <= falseBlinkLimit):

if(eyeStatus):

state = 0

else:

state += 1

elif(state >= falseBlinkLimit and state < drowsyLimit):

if(eyeStatus):

blinkCount += 1

state = 0

else:

state += 1

else:

if(eyeStatus):

state = 0

drowsy = 1

blinkCount += 1

else:

drowsy = 1

def getLandmarks(im):

imSmall = cv2.resize(im, None,

fx = 1.0/FACE\_DOWNSAMPLE\_RATIO,

fy = 1.0/FACE\_DOWNSAMPLE\_RATIO,

interpolation = cv2.INTER\_LINEAR)

rects = detector(imSmall, 0)

if len(rects) == 0:

return 0

newRect = dlib.rectangle(int(rects[0].left() \* FACE\_DOWNSAMPLE\_RATIO),

int(rects[0].top() \* FACE\_DOWNSAMPLE\_RATIO),

int(rects[0].right() \* FACE\_DOWNSAMPLE\_RATIO),

int(rects[0].bottom() \* FACE\_DOWNSAMPLE\_RATIO))

points = []

[points.append((p.x, p.y)) for p in predictor(im, newRect).parts()]

return points

capture = cv2.VideoCapture(0)

for i in range(10):

ret, frame = capture.read()

totalTime = 0.0

validFrames = 0

dummyFrames = 100

print("Caliberation in Progress!")

while(validFrames < dummyFrames):

validFrames += 1

t = time.time()

ret, frame = capture.read()

height, width = frame.shape[:2]

IMAGE\_RESIZE = np.float32(height)/RESIZE\_HEIGHT

**17**

frame = cv2.resize(frame, None,

fx = 1/IMAGE\_RESIZE,

fy = 1/IMAGE\_RESIZE,

interpolation = cv2.INTER\_LINEAR)

# adjusted = gamma\_correction(frame)

adjusted = histogram\_equalization(frame)

landmarks = getLandmarks(adjusted)

timeLandmarks = time.time() - t

if landmarks == 0:

validFrames -= 1

cv2.putText(frame, "Unable to detect face, Please check proper lighting", (10, 30), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE\_AA)

cv2.putText(frame, "or decrease FACE\_DOWNSAMPLE\_RATIO", (10, 50), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE\_AA)

cv2.imshow("Blink Detection Demo", frame)

if cv2.waitKey(1) & 0xFF == 27:

sys.exit()

else:

totalTime += timeLandmarks

# cv2.putText(frame, "Caliberation in Progress", (200, 30), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 255), 1,

# cv2.LINE\_AA) cv2.imshow("Blink Detection Demo", frame)

# if cv2.waitKey(1) & 0xFF == 27:

# sys.exit()

print("Caliberation Complete!")

spf = totalTime/dummyFrames

print("Current SPF (seconds per frame) is {:.2f} ms".format(spf \* 1000))

drowsyLimit = drowsyTime/spf

falseBlinkLimit = blinkTime/spf

print("drowsy limit: {}, false blink limit: {}".format(drowsyLimit, falseBlinkLimit))

if \_\_name\_\_ == "\_\_main\_\_":

vid\_writer = cv2.VideoWriter('output-low-light-2.avi',cv2.VideoWriter\_fourcc('M','J','P','G'), 15, (frame.shape[1],frame.shape[0]))

while(1):

try:

t = time.time()

ret, frame = capture.read()

height, width = frame.shape[:2]

IMAGE\_RESIZE = np.float32(height)/RESIZE\_HEIGHT

frame = cv2.resize(frame, None,

fx = 1/IMAGE\_RESIZE,

fy = 1/IMAGE\_RESIZE,

interpolation = cv2.INTER\_LINEAR)

# adjusted = gamma\_correction(frame)

adjusted = histogram\_equalization(frame)

landmarks = getLandmarks(adjusted)

if landmarks == 0:

validFrames -= 1

cv2.putText(frame, "Unable to detect face, Please check proper lighting", (10, 30),

cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE\_AA)

cv2.putText(frame, "or decrease FACE\_DOWNSAMPLE\_RATIO", (10, 50), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE\_AA)

cv2.imshow("Blink Detection Demo", frame)

if cv2.waitKey(1) & 0xFF == 27:

break

continue

eyeStatus = checkEyeStatus(landmarks)

checkBlinkStatus(eyeStatus)

for i in range(0, len(leftEyeIndex)):

cv2.circle(frame, (landmarks[leftEyeIndex[i]][0], landmarks[leftEyeIndex[i]][1]), 1, (0, 0, 255), -1, lineType=cv2.LINE\_AA)

for i in range(0, len(rightEyeIndex)):

cv2.circle(frame, (landmarks[rightEyeIndex[i]][0], landmarks[rightEyeIndex[i]][1]), 1, (0, 0, 255), -1, lineType=cv2.LINE\_AA)

if drowsy:

cv2.putText(frame, "! ! ! DROWSINESS ALERT ! ! !", (70, 50), cv2.FONT\_HERSHEY\_COMPLEX, 1, (0, 0, 255), 2, cv2.LINE\_AA)

if not ALARM\_ON:

ALARM\_ON = True

threadStatusQ.put(not ALARM\_ON)

thread = Thread(target=soundAlert, args=(sound\_path, threadStatusQ,))

thread.setDaemon(True)

thread.start()

else:

cv2.putText(frame, "Blinks : {}".format(blinkCount), (460, 80), cv2.FONT\_HERSHEY\_COMPLEX, 0.8, (0,0,255), 2, cv2.LINE\_AA)

# (0, 400)

ALARM\_ON = False

cv2.imshow("Blink Detection Demo", frame)

vid\_writer.write(frame)

k = cv2.waitKey(1)

if k == ord('r'):

state = 0

drowsy = 0

ALARM\_ON = False

threadStatusQ.put(not ALARM\_ON)

elif k == 27:

break

# print("Time taken", time.time() - t)

except Exception as e:

print(e)

capture.release()

vid\_writer.release()

cv2.destroyAllWindows()

**19**

**6.3. RESULTS**

The driver drowsiness can be measured using Eye Aspect Ratio(EAR). The ratio of the eye can vary for each and every person. Fig.4. The following case is tested for ten different set of people with two conditions. One is calculated for eye-opening condition and another one for eye closing condition. Fig.3. Eye closing rate is measured after every 0.5 seconds and if the value crosses already existed threshold value, then the raspberry pi 3 receives the alert signal from alarm connected to the GPIO pins of Pi 3 board. Fig . When the person closing his eyes for more than fixed threshold range then the alert signal is generated to wake up the driver from sleepy state and also through the cloud service the alert message is sent to the owner of the car along with the car plate number.

A person wearing glasses

Description automatically generated with medium confidence

**6.3.1 result**

**20**

**CHAPTER – 07**

**7.1. CONCLUSION**

Driver drowsiness detection is designed mainly to keep the driver awake while driving to avoid the accident due to sleepiness. The alert signal is generated from embedded device to awake driver from sleepy state. The Pi along with Raspbian camera is used to calculate the drowsiness of the driver in real time. Fatigue is measured by detecting Eye and face using Haar Cascade Classifier, especially facial landmarks is detected using shape-predictor and Eye Aspect Ratio (EAR) by calculating the Euclidean distance between the eyes. Accurate eye detection and faces in every frame will help to calculate drowsiness level. Frequent detection of eye blinking and head tilting is measured properly and it helps to indicates drowsiness. When he/she reaches maximum threshold the driver will be alarmed by a loud warning that will wake up the driver from the sleep state. In future, the implementation can be carried out in a bright room with consistent light, for different lighting conditions and also for the persons with dark skin can be considered.

**7.2. FUTURE SCOPE**

A further developed application of the driver drowsiness detection system may involve In integration of the system directly with the vehicle.

Then vehicle, when it detects the driver is in a path of collision and is in drowsy and unreactive state will undergo emergency braking, thereby saving the life of the driver and passengers and the lives of other vehicle passengers.

The future developments which we want to incorporate with the project are:

1.Devlopment of an advanced driving monitoring system, which will bring the vehicle to halt if the driver is found drowsy or lack of attention is registered in the system

2.We want to improve the accuracy of the system by, incorporating distance between the facial landmarks to to account for any movement by the subject in the video. Realistically the participants will not be static on the screen and we believe sudden movements by the participant may signal drowsiness or waking up from micro-sleep.

3. we would like to collect our own training data from a larger sample of participants ,which includes new distinct signals of drowsiness like sudden head movement, hand movement, or even tracking eye movements.

**21**

**CHAPTER – 08**

**REFERENCES**

[1] L. Jia, D. Zhao, K. Zheng, Z. Li, G. Sun, and F. Zhang, “Smartphone-based fatigue detection system using the progressive locating method”, IET Intell. Transp. Syst, vol. 10, no. 3, pp. 148–156, 2016.

[2] J. S. Jayasenan and P. S. Smitha, “Driver Drowsiness Detection System”, vol. 4, no. 1, pp. 34–37, 2014.

[3] V. Triyanti and H. Iridiastadi, “Challenges in detecting drowsiness based on driver’s behavior”, IOP Conf. Ser. Mater. Sci. Eng, vol. 277, 2017.

[4] F. Omidi and G. N. Saraji, “Non-intrusive Methods used to Determine the Driver Drowsiness : Narrative Review Articles”, pp. 186–191, 2016.

[5] P. P. Bhatt GHPatel PG and J. A. Trivedi Patel PG, “Various Methods for Driver Drowsiness Detection : An Overview”, Int. J. Comput. Sci. Eng, vol. 9, no. 3, pp. 70–74, 2017.

[6] D. Sarkar and A. Chowdhury, “A Real-Time Embedded System Application for Driver Drowsiness and Alcoholic Intoxication Detection”, Int. J. Eng. Trends Technol, vol. 10, no. 9, pp. 461–465, 2014.

[7] R. Ahmed, Kazi Emrul Kayes Emon, and M. F. Hossain, “Robust driver fatigue recognition using image processing”, Int. Conf. Informatics, Electron. Vis, pp. 1–6, 2014.

[8] O. Khunpisuth, T. Chotchinasri, V. Koschakosai, and N. Hnoohom, “Driver Drowsiness Detection Using Eye-Closeness Detection”, 12th Int. Conf. Signal-Image Technol. Internet-Based Syst, pp. 661–668, 2016.

[9] L. F. Ibrahim et al., “Using Haar classifiers to detect driver fatigue and provide alerts”, Multimedia. Tools Appl, vol. 71, no. 3, pp. 1857–1877, 2014.

[10] T. Soukupová and J. Cech, “Real-Time Eye Blink Detection using Facial Landmarks”, 21st Comput. Vis. Winter Work, 2016.

**22**