**DRIVER DROWZINESS DETECTION USING RASPBERRY PI**

*ABSTRACT: This proposed system is used for Driver & Road safety system. Based on computer vision techniques, the driver’s face is located from a colour video captured in a car. Then, face detection is employed to locate the regions of the driver’s eyes, which are used as the templates for eye tracking in subsequent frames. The tracked eye’s images are used for drowsiness detection in order to generate warning alarms. The proposed approach has three phases: Face, Eye detection and drowsiness detection. The role of image processing is to recognize the face of the driver and then extracts the image of the eyes of the driver for detection of drowsiness. Hence, we conclude this approach is a low cost and effective solution to reduce the number of accidents due to driver's Drowsiness to increase the transportation safety.*

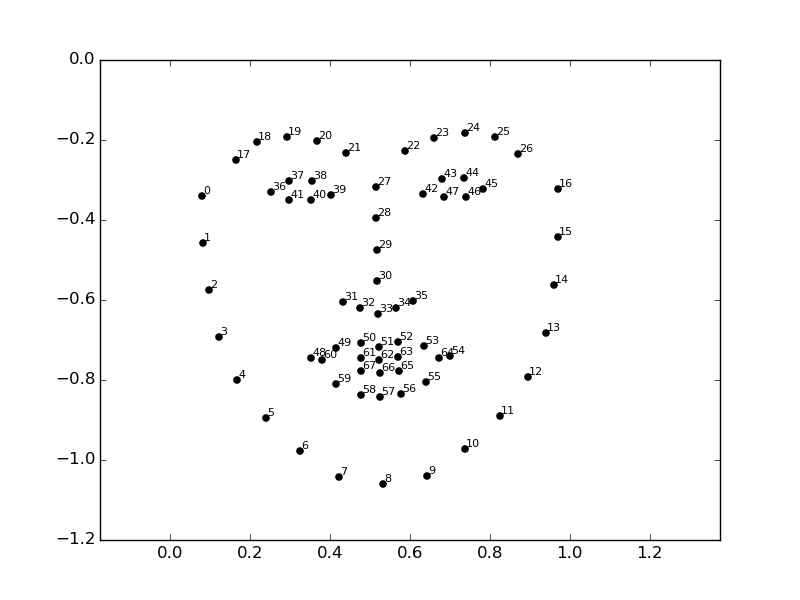
**INTRODUCTION:**

Drowsy driving is one of the major causes behind fatal road accidents. One of the recent studies shows that one out of five road accidents are caused by drowsy driving which is roughly around 21% of road accidents, and this percentage is increasing every year as per global status report on road safety 2015, based on the data from 180 different countries. This certainly highlights the fact that across the world the total numbers of road traffic deaths are very high due to driver’s drowsiness. Driver fatigue, drink-and-drive and carelessness are coming forward as major reasons behind such road accidents. Many lives and families are getting affected due to this across various countries. Real time drowsy driving detection is one of the best possible majors that can be implemented to assist drivers to make them aware of drowsy driving conditions. Such driver behavioural state detection system can help in catching the driver drowsy conditions early and can possibly avoid mishaps.

With this report, we are presenting technique to detect driver drowsiness using of Open CV, raspberry pi and image processing. Several studies have shown various possible techniques that can detect the driver drowsiness. Among these physiological measure and ocular measure can give more accurate results. But this leads to discomfortable driving conditions. But ocular measure can be done without physical connection. Ocular measure to detect driver eye condition and possible vision based on eye closure is well suited for real world driving conditions, since it can detect the eyes open/ closed state non intrusively using a camera.

In Real Time Driver Drowsiness System using Image Processing, capturing drivers eye state using computer vision-based drowsiness detection systems have been done by analysing the interval of eye closure and developing an algorithm to detect the driver’s drowsiness in advance and to warn the driver by in vehicles alarm. This section motivates how face is detected and how eye detection is performed for automotive application and their detection is necessary for assessing driver drowsiness.

**SYSTEM ARCHITECTURE:**

In the proposed system shown in Fig.1, the primary focus is given to the faster drowsiness detection and processing of data. The number of frames in which the eyes are kept closed is monitored and then counted. If the number of frames exceeds a threshold value, then a warning message is generated on the display showing that the drowsiness is detected. The system should be capable of detecting drowsiness despite the skin colour and complexion of the driver, spectacles used by the driver and the darkness level inside the vehicle. All these objectives have been well satisfied by choosing the system using appropriate classifiers in OpenCV for eye closure detection.

In this algorithm, first a driver’s image is acquired by the camera for processing. In OpenCV, the face detection of the driver’s image is carried out first followed by eye detection. The eye detection technique detects the open state of eyes only. Then the algorithm counts the number of open eyes in each frame and calculates the criteria for detection of drowsiness. If the criteria are satisfied, then the driver is said to be drowsy. The display and buzzer connected to the system perform actions to correct the driver abnormal behaviour.

**COMPONENTS USED:**

1. Raspberry Pi 3​
2. Pi Camera Module​
3. Micro USB Cable​
4. Buzzer​
5. LCD – 16x2 character lcd​
6. LED​

**SOFTWARES USED:**

1. **OpenCV and Dlib​:**

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.Dlib is used for face detection and facial landmark detection**.**

1. **Face recognition​:**

The face recognition algorithm is used in finding features that are uniquely described in the image. The facial image is already extracted, cropped, resized, and usually converted in the grayscale.

1. **Face\_utils:**

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV .

**PYTHON CODE:**

#Importing OpenCV Library for basic image processing functions

import cv2

# Numpy for array related functions

import numpy as np

# Dlib for deep learning based Modules and face landmark detection

import dlib

#face\_utils for basic operations of conversion

from imutils import face\_utils

import time

import board

import digitalio

import adafruit\_character\_lcd.character\_lcd as characterlcd

import RPi.GPIO as GPIO     # Import Library to access GPIO PIN

GPIO.setmode(GPIO.BCM)    # Consider complete raspberry-pi board

GPIO.setwarnings(False)

buzzer\_pin = 16                # Define PIN for LED

LED\_PIN = 5

lcd\_rs = digitalio.DigitalInOut(board.D4)

lcd\_en = digitalio.DigitalInOut(board.D17)

lcd\_d7 = digitalio.DigitalInOut(board.D26)

lcd\_d6 = digitalio.DigitalInOut(board.D22)

lcd\_d5 = digitalio.DigitalInOut(board.D27)

lcd\_d4 = digitalio.DigitalInOut(board.D18)

GPIO.setup(buzzer\_pin,GPIO.OUT)   # Set pin function as output

GPIO.setup(LED\_PIN,GPIO.OUT)   # Set pin function as output

# Define some device constants

GPIO.output(buzzer\_pin,GPIO.LOW)

lcd\_columns = 16

lcd\_rows = 2

lcd = characterlcd.Character\_LCD\_Mono(lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7, lcd\_columns, lcd\_rows)

#Initializing the camera and taking the instance

cap = cv2.VideoCapture(0)

#Initializing the face detector and landmark detector

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

predictor = dlib.shape\_predictor("/home/pi/Documents/group 13 project/shape\_predictor\_68\_face\_landmarks.dat")

#status marking for current state

sleep = 0

drowsy = 0

active = 0

status=""

color=(0,0,0)

def compute(ptA,ptB):

    dist = np.linalg.norm(ptA - ptB)

    return dist

def blinked(a,b,c,d,e,f):

    up = compute(b,d) + compute(c,e)

    down = compute(a,f)

    ratio = up/(2.0\*down)

    #Checking if it is blinked

    if(ratio>0.25):

        return 2

    elif(ratio>0.21 and ratio<=0.25):

        return 1

    else:

        return 0

lcd.cursor = True

lcd.message="welcome "#send msg to my LCD

time.sleep(2) #delay5 seconds

lcd.message="Driver Sleep\nDetection System"#send msg to my LCD

time.sleep(2) #delay

while True:

        \_, frame = cap.read()

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

        faces = hog\_face\_detector(gray)

        #detected face in faces array

        for face in faces:

                x1 = face.left()

                y1 = face.top()

                x2 = face.right()

                y2 = face.bottom()

                face\_frame = frame.copy()

                cv2.rectangle(face\_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

                landmarks = predictor(gray, face)

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

                #The numbers are actually the landmarks which will show eye

                left\_blink = blinked(landmarks[36],landmarks[37],

                landmarks[38], landmarks[41], landmarks[40], landmarks[39])

                right\_blink = blinked(landmarks[42],landmarks[43],

                landmarks[44], landmarks[47], landmarks[46], landmarks[45])

                #Now judge what to do for the eye blinks

                if(left\_blink==0 or right\_blink==0):

                        sleep+=1

                        drowsy=0

                        active=0

                        if(sleep>1):

                                status="SLEEPING !!"

                                print("SLEEPING !!")

                                GPIO.output(buzzer\_pin,GPIO.HIGH)

                                GPIO.output(LED\_PIN,GPIO.HIGH)

                                lcd.message='Please Wake up!'

                                time.sleep(1)

                                color = (255,0,0)

                                lcd.clear()

                elif(left\_blink==1 or right\_blink==1):

                        sleep=0

                        active=0

                        drowsy+=1

                        if(drowsy>1):

                                status="Drowsy :("

                                print("DROWSY :(")

                                GPIO.output(buzzer\_pin,GPIO.HIGH)

                                GPIO.output(LED\_PIN,GPIO.HIGH)

                                lcd.message='Dont be drowsy!'

                                time.sleep(1)

                                color = (0,0,255)

                                lcd.clear()

                else:

                        drowsy=0

                        sleep=0

                        active+=1

                        if(active>1):

                                status="Active :)"

                                print("Active :)")

                                lcd.message='All okay!\nDriver safe!'

                                time.sleep(1)

                                GPIO.output(buzzer\_pin,GPIO.LOW)

                                GPIO.output(LED\_PIN,GPIO.LOW)

                                color = (0,255,0)

                                lcd.clear()

                cv2.putText(frame, status, (100,100), cv2.FONT\_HERSHEY\_SIMPLEX, 1.2, color,3)

                for n in range(0, 68):

                        (x,y) = landmarks[n]

                        cv2.circle(face\_frame, (x, y), 1, (255, 255, 255), -1)

        cv2.imshow("Frame", frame)

        #cv2.imshow("Result of detector", face\_frame)

        key = cv2.waitKey(1)

        if key == 27:

                break

**RESULTS:**

1. **WHEN DRIVER IS ACTIVE:**

****

1. **WHEN DRIVER IS DROWSY:**

****

1. **WHEN DRIVER IS SLEEPY:**

****