# **DROWSINESS DETECTION AND ALERTING SYSTEM**

Brinda Venkataramana Pabba Vishnu Vaibhav Jaiswal Poojitha Lankapalle Computer Science Engineering Computer Engineering Electrical Engineering Computer Science Engineering [madhumathibrindha@gmail.com](mailto:madhumathibrindha@gmail.com) [pvishnusy@gmail.com](mailto:pvishnusy@gmail.com) [vaibhavjaiswal28@yahoo.com](mailto:vaibhavjaiswal28@yahoo.com) [poojithalankipalle@gmail.com](mailto:poojithalankipalle@gmail.com) Gitam University IIIT Bhubaneshwar Geethanjali College of Engineering And Technology Bapatla Engineering College Vishakapatnam , India Bhubaneshwar , India Hyderabad , India Bapatla, India

# **ABSTRACT**

Various investigations show that driver’s drowsiness is one of the main causes

of road accidents. The current technology in digital computer system allows

researchers around the world to study the fatigue behaviour. The purpose of

this study is to detect the drowsiness in drivers to prevent the accidents and

to improve the safety on the highways. Real time face detection is

implemented to locate driver’s face region. In this project the eye blink of the

driver is detected. If the driver’s eyes remain closed for more than a certain

period of time, the driver is said to be drowsy and an alarm is sounded. The

programming is done in python language and OpenCV for the detection of

facial features . In this project we aim to develop drowsiness system

# RASPBERRY PI(ALTERNATIVE METHOD)

The Raspberry Pi is a series of small single-board computers devolepment in U.K by Raspberry Pi Foundation.Raspberry pi 3 Model B is a tiny credit card size computer.Just add a keyboard,mouse,display,power supply,micro SD card with installed Linux Distribution and you’ll have a fully fledged computer that can run applications from word processor and spread sheets to games.

As Raspberry Pi 3 supports HD video,you can even create a media center with it. The Raspberry Pi 3 Model B is the first Raspberry Pi to be the first open source from the get-go,except it to be the defacto embedded Linux board in all the forums.The memory of Raspberry pi is 1GB

## `

## **PROBLEM STATEMENT:**

Driver fatigue sometimes results in road accidents every year. It is not easy to estimate the exact amount of sleep related accidents but research presents that driver fatigue may be a contributing reason in up to 20% in road accidents. These types of accidents are about 50% more expected to result in death or serious hurt. They happen mainly at higher speed impacts. And the driver who has fallen asleep cannot brake. Drowsiness reduces response time which is a serious element of secure driving. It also reduces alertness, vigilance, and concentration so that the capacity to perform attention-based activities i.e. driving is impaired. The speed at which information is processed is also reduced by drowsiness.

The quality of decision-making may also be affected. It is clear that drivers are aware when they are feeling sleepy, and so make a conscious decision about whether to continue driving or to stop for a rest. It may be that those who persist in driving underestimate the risk of actually falling asleep while driving.

Or it may be that some drivers choose to ignore the risks in the way drivers drink. Crashes caused by tired drivers are most likely to happen on long journeys on monotonous roads, such as motorways, between 2pm and 4pm especially after eating or taking an alcoholic drink, between 2am and 6am, after having less sleep than normal, after drinking alcohol, it driver takes medicines that cause drowsiness and after long working hours or on journeys home after long shifts, especially nightshifts.

Tiredness and fatigue can often affect a person’s driving ability long before he/she even notices that he/she is getting tired. Fatigue related crashes are often more severe than others because driver’s reaction times are delayed or the drivers have failed to make any manoeuvres to avoid a crash. The number of hours spent driving has a strong correlation to the number of fatigue related accidents.

# **PROJECT WORKING PROCESS:**

The main idea of project is to first detect if the driver of the vehicle is asleep or not. If the driver is asleep, then wake him/her with a buzzer. If the driver doesn’t wake up even after that it means that there is something wrong with the driver (there could have been an accident). Then this would further be conveyed to his/her family members through cloud.

To detect if the person is asleep, we use Euclidean distance using x and y coordinates of the eyes.

## **THE FINAL CODE:**

#Import necessary libraries  
from scipy.spatial import distance  
from imutils import face\_utils  
import numpy as np  
import pygame #For playing sound  
import time  
import dlib  
import cv2  
import requests  
  
#Initialize Pygame and load music  
pygame.mixer.init()  
pygame.mixer.music.load('audio/alert.wav')  
  
#Minimum threshold of eye aspect ratio below which alarm is triggerd  
EYE\_ASPECT\_RATIO\_THRESHOLD = 0.3  
  
#Minimum consecutive frames for which eye ratio is below threshold for alarm to be triggered  
EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES = 50  
  
#COunts no. of consecutuve frames below threshold value  
COUNTER = 0  
  
#Load face cascade which will be used to draw a rectangle around detected faces.  
face\_cascade = cv2.CascadeClassifier("haarcascades/haarcascade\_frontalface\_default.xml")  
  
#This function calculates and return eye aspect ratio  
def eye\_aspect\_ratio(eye):  
    A = distance.euclidean(eye[1], eye[5])  
    B = distance.euclidean(eye[2], eye[4])  
    C = distance.euclidean(eye[0], eye[3])  
  
    ear = (A+B) / (2\*C)  
    return ear  
  
#Load face detector and predictor, uses dlib shape predictor file  
detector = dlib.get\_frontal\_face\_detector()  
predictor = dlib.shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat')  
  
#Extract indexes of facial landmarks for the left and right eye  
(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS['left\_eye']  
(rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS['right\_eye']  
  
#Start webcam video capture  
video\_capture = cv2.VideoCapture(0)  
  
#Give some time for camera to initialize(not required)  
time.sleep(2)  
  
while(True):  
    #Read each frame and flip it, and convert to grayscale  
    ret, frame = video\_capture.read()  
    frame = cv2.flip(frame,1)  
    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
  
    #Detect facial points through detector function  
    faces = detector(gray, 0)  
  
    #Detect faces through haarcascade\_frontalface\_default.xml  
    face\_rectangle = face\_cascade.detectMultiScale(gray, 1.3, 5)  
  
    #Draw rectangle around each face detected  
    for (x,y,w,h) in face\_rectangle:  
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)  
  
    #Detect facial points  
    for face in faces:  
  
        shape = predictor(gray, face)  
        shape = face\_utils.shape\_to\_np(shape)  
  
        #Get array of coordinates of leftEye and rightEye  
        leftEye = shape[lStart:lEnd]  
        rightEye = shape[rStart:rEnd]  
  
        #Calculate aspect ratio of both eyes  
        leftEyeAspectRatio = eye\_aspect\_ratio(leftEye)  
        rightEyeAspectRatio = eye\_aspect\_ratio(rightEye)  
  
        eyeAspectRatio = (leftEyeAspectRatio + rightEyeAspectRatio) / 2  
  
        #Use hull to remove convex contour discrepencies and draw eye shape around eyes  
        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)  
  
        #Detect if eye aspect ratio is less than threshold  
        if(eyeAspectRatio < EYE\_ASPECT\_RATIO\_THRESHOLD):  
            COUNTER += 1  
            #If no. of frames is greater than threshold frames,  
            if COUNTER >= EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES:  
                pygame.mixer.music.play(-1)  
                cv2.putText(frame, "You are Drowsy", (150,200), cv2.FONT\_HERSHEY\_SIMPLEX, 1.5, (0,0,255), 2)  
                r=requests.get('<http://api.msg91.com/api/sendhttp.php?route=4&sender=TESTIN&mobiles=9652032914&authkey=275989AUXsTabE5cd50501&message=Hello>! This is a test message&country=91')  
                print (r.text)  
        else:  
            pygame.mixer.music.stop()  
            COUNTER = 0  
  
    #Show video feed  
    cv2.imshow('Video', frame)  
    if(cv2.waitKey(1) & 0xFF == ord('q')):  
        break  
  
#Finally when video capture is over, release the video capture and destroyAllWindows  
video\_capture.release()  
cv2.destroyAllWindows()

## **THE INSTALLATION, INSTALLATION PROCESS:**

We’ll need the [SciPy](https://www.scipy.org/) package so we can compute the Euclidean distance between facial landmarks points in the eye aspect ratio calculation (not strictly a requirement, but you should have SciPy installed if you intend on doing any work in the computer vision, image processing, or machine learning space).We’ll also need the [imutils package](https://github.com/jrosebr1/imutils), my series of computer vision and image processing functions to make working with OpenCV easier.If you don’t already have imutils  installed on your system, you can install/upgrade imutils  via:

|  |  |
| --- | --- |
| 1 | $ pip install --upgrade imutils |

We’ll also import the Thread  class so we can play our alarm in a separate thread from the main thread to ensure our script doesn’t pause execution while the alarm sounds. In order to actually play our WAV/MP3 alarm, we need the [playsound library](https://pypi.python.org/pypi/playsound/1.2.1), a pure Python, cross-platform implementation for playing simple sounds.The playsound  library is conveniently installable via pip :

|  |  |
| --- | --- |
| 1 | $ pip install playsound |

However, if you are using *macOS* (like I did for this project), [you’ll also want to install pyobjc](http://stackoverflow.com/questions/12767669/import-error-no-module-named-appkit), otherwise you’ll get an error related to App Kit  when you actually try to play the sound:

|  |  |
| --- | --- |
| 1 | $ pip install pyobjc |

I *only* tested play sound on *macOS*, but according to both the documentation and Taylor Marks (the developer and maintainer of playsound ), the library should work on Linux and Windows as well.

***Note:*** *If you are having problems with playsound ,* [*please consult their documentation*](https://github.com/TaylorSMarks/playsound)*as I am****not*** *an expert on audio libraries.* To detect and localize facial landmarks we’ll need the [dlib library](http://dlib.net/) which is imported on **Line 11**. If you need help installing dlib on your system, [please refer to this tutorial](https://www.pyimagesearch.com/2017/03/27/how-to-install-dlib/). Next, we need to define our sound\_alarm  function which accepts a path  to an audio file residing on disk and then plays the file:

|  |  |
| --- | --- |
|  | def sound\_alarm(path):  # play an alarm sound  playsound.playsound(path) |

We also need to define the eye\_aspect\_ratio  function which is used to compute the ratio of distances between the vertical eye landmarks and the distances between the horizontal eye landmarks:

|  |  |
| --- | --- |
|  | def eye\_aspect\_ratio(eye):  # compute the euclidean distances between the two sets of  # vertical eye landmarks (x, y)-coordinates  A = dist.euclidean(eye[1], eye[5])  B = dist.euclidean(eye[2], eye[4])    # compute the euclidean distance between the horizontal  # eye landmark (x, y)-coordinates  C = dist.euclidean(eye[0], eye[3])    # compute the eye aspect ratio  ear = (A + B) / (2.0 \* C)    # return the eye aspect ratio  return ear |

The return value of the eye aspect ratio will be approximately constant when the eye is open. The value will then rapid decrease towards zero during a blink. If the eye is closed, the eye aspect ratio will again remain approximately constant, but will be *much smaller* than the ratio when the eye is open.