**DROWSY DRIVER DETECTION SYSTEM**

**A PROJECT REPORT**

**RUBIKA.R (16MIS0430)**



**School of Information Technology and Engineering**

# ABSTRACT:

The main idea behind this project is to develop a nonintrusive system which can detect fatigue of any human and can issue a timely warning. Drivers who do not take regular breaks when driving long distances run a high risk of becoming drowsy a state which they often fail to recognize early enough. According to the expert’s studies show that around one quarter of all serious motorway accidents is attributable to sleepy drivers in need of a rest, meaning that drowsiness causes more road accidents than drink-driving. This system will monitor the driver eyes using a camera and by developing an algorithm we can detect symptoms of driver fatigue early enough to avoid the person from sleeping. So, this project will be helpful in detecting driver fatigue in advance and will give warning output in form of alarm and pop-ups.

# INTRODUCTION:

Real Time Drowsiness behaviours which are related to fatigue are in the form of eye closing. Hence, we can either measure change in physiological signals, such as brain waves, heart rate and eye blinking to monitor drowsiness or consider physical changes such as sagging posture, leaning of driver’s head and open/clo sed state of eyes. The former technique, while more accurate, is not realistic since highly sensitive electrodes would have to be attached directly on the driver’ body and hence which can be annoying and distracting to the driver. In addition, long time working would result in perspiration on the sensors, diminishing their ability to monitor accurately. The second technique is to measure physical changes (i.e. open/closed eyes to detect fatigue) is well suited for real world conditions since it is non-intrusive by using a video camera to detect changes. In addition, micro sleeps that are short period of sleeps lasting

2 to 3 minutes are good indicators of fatigue. Thus, by continuously monitoring the eyes of the driver one can detect the sleepy state of driver and a timely warning is issued.

# LITERATURE REVIEW:

Manu B.N[1]Author presents a new method of drowsiness detection by using a camera acquires the video of the driver. Then this video is broken down into frames. Face detection done by Vialo Jones. Andaccurately locate the position of the eyes and the mouth, skin segmentation is performed by converting the image to YCbCr domain.So the influence of luminosity can be eliminated.The most important factor which helps detect driver fatigue is the state of eyes, i.e. open or closed.The position of the driver's eyes are determined by using Viola

Jones.The two eyes are separated using edge detection and in accordance with the symmetrical properties of the eye, the center of the eye is determined.Finally the pupil is identified. If eyes are open then it is treated as the normal state during which the alarm is not set off. If eyes are closed then it is treated as the fatigue state during which the alarm is set on. Sobel which is an edge detection method is used.The status of the eyes is determined in every frame using the correlation coefficient template matching method. Separated edge is stored as template.Each eye template, area, average height and width to height ratio are the crucial features to determine the eye’s status. Once the regions of mouth area are found using Viola Jones, the mouth region alone is segmented by K, means clustering and tracked using correlation coefficient template matching.

Jun-Juh Yan, et.al,[2] Author said that after an image is captured, the image pre- processing is performed to extract useful information. The first part of the system includes data collection and facial region detection.edge detection is to find edge pixels in gray-scale image; edge pixels are the positions where the gray level suddenly changes. By uses the Sobel Operator for edge detection.As a result, through the histogram of edge pixels, the face region can be located.each image needs only to retain two data sets, the center position of the eyes and the face width. A median filter can remove the very large or small data.PERCLOS is the abbreviation for the percentage of eyelid closure over the pupil over time.PERCLOS is measured over a fixed period of time.This can be regarded as a threshold, which separates the eyes state into two cases: closed more than 80% or not.Once the amount of black pixels in an image is less than Threshold , which means that in this image the eyes are under the P80 situation, it is defined as CLOSED ; otherwise, it means the eyes are open more than twenty per cent, and therefore defined as OPEN.

Pooneh.R et.al,[3]Author use by eye detection with high accurate rate, An image of a face which is isolated from its background.So,YCbCr used to reduce the redundancy present in RGB color.To detect eye region and make sure determines if the eye is open or closed.Pupil center localization is performed in the 2 upper quarters of the Eye Map which is obtained from eye region detection.Based on color gradient of eye image to estimate the iris boundary.Middle filter is applied to the eye image to eliminate the highlight of pupil.Distance between two eyelids.The upper eyelid should not cover the pupil

. Thus, if distance between two eyelids is less than iris radius, the eye is closed,if not eye is open so drowsiness is detected.

Huabiao Qin, et.al,[4]Author using the driver’s head movements are in 3D space. We map the tracking of head movements to the 2D plane.Adaptive algorithm used to track head movement.The features of dozing off movement may not exist in the horizontal direction. Only the Fourier coefficients in the

vertical direction are considered in the dozing off detection.Applying FFT to the head movements in the frequency domain, Author developed algorithms to highly correlate drowsiness to regular head movements, and distraction to stationary head positions.

Ping Wang, et.al,[5]Author used the method of AdaBoost algorithm to detection of face. The pre-processing includes graying and enhancing contrast by gray- scale histogram equalization method for every part of a face image. The width- Height Ratio of the circumscribed rectangle of the area should be 1-1.6. The pixels of the eye area are more than 10.The area of eye is more than half the one of the circumscribed rectangle. locate the mouth because the image is usually simple and more symmetric. Then the width-height ratio of the located mouth area is used to judge whether the mouth is opening fully. PATECP algorithm focuses on percentage and lifetime that eyelids cover the pupils. PATMIO algorithm concentrates the percentage and time that mouth is open. Based on PATECP and PATMIO algorithm drowsiness is detected.

Mohammad Amin Assari, et.al,[6] Author used facial expressions to detect drowsiness. The system consists of four steps: they are face detection, facial component extraction, facial components tracking, and drowsiness detection. The background subtraction method is used to detect face region in the image. Horizontal projection technique is used to determine eyebrow and eye region and template matching to determine region of mouth. status of each facial component separately and if any of the components that dealt with one of the facial expression caused by the fatigue and sleepiness will be created a warning message. Eyebrow is examined to increase the detection power of the system.

Usually, when the drowsiness started, the three components eyebrow, eye and mouth will have the most influence.

Mai Suzuki, et.al,[7]Author method for measuring driver's consciousness using driver's eye-blink information. To detect the eye area from the images, author system uses a three-layered neural network with back propagation method.First face area is extracted,then, after being transformed into images of low resolution, four-direction edge characteristic detection method is applied. The results are input to the neural network.First, this method slices the captured image into several vertical sections.When the eye gap is large, and when a blink starts, the eye gap decreases quickly. After the eyes are shut completely, the eye gap increases gradually. Let us call the period between the start and the end as blinkperiod.the upper and lower eyelids included in the eye area. Then,it detect drowsiness.

Hiroshi Ueno,et.al,[8]Author describes a system that the facial image data are converted to binary image to remove noise by processing median filter .The maximum width of the face is then detected so that the right and left edges of the face can be identified.Vertical position of each eye is detected independently within an area defined by the center line of the face width and lines running through the outermost points of the face.Eyeball tracking is to update the area of eye presence, in which an eye search is made in the following frame, according to the central coordinates of the eye in the previous frame.On the basis of the Feret's diameter of the eyes,eye open/close can be detected.

Tnkehiro Ito, et.al,[9] Author captures eye areas are extracted and tracked from the obtained facial images.In eyes the upper and lower eyelids are detected.

Then, blinlking is measured by processing the waveform of opening and closing of the upper and lower eyelid.To detect the eye area, from facial image by using separability filter. Then Euclidean distance between the eye area on smallest distance is discoverd.The captured two-dimensional image has heen sliced into vertical sections,and point ofthe upper and lower eyelids are detected.Then distance between upper and lower eyelid is measured and detect how much the two lids are closing and declare drowsiness.

Tiesheng Wang, et.al,[10]Author use face detection by using vialo jones algorithm. AdaBoost classifier ensures high detection rate and low false alarm rate.Kalman filter can be adopted for driver face region tracking to improve the speed of face detection in video. To detect mouth region, first one is a multi threshold binarization method in intensity space. Second one by the single Gaussian model in r-g-b space. Projection based method to detect mouth features. Integral Projection Function, Variance Projection Function, and Hybrid Projection Function used to detect eye region. Integral projection of vertical difference of mouth region along the direction of mouth rotation to detect the two lines running through upper and lower lip boundaries. Based on mouth openess detect drowsy state of individual.

# METHODOLOGY:

The different types of methodologies have been developed to find out drowsiness.

1. Physiological level approach:

This technique is an intrusive method where in electrodes are used to obtain pulse rate, heart rate and brain activity information. ECG is used to calculate the variations in heart rate and detect different conditions for

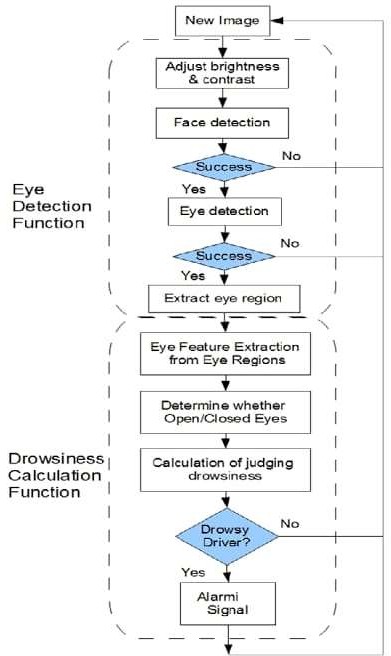
drowsiness. The correlation between different signals such as ecg (electrocardiogram), EEG(electroencephalogram), and EMG (electromyogram) are made and then the output is generated whether the person is drowsy or not.

1. Behavioural based approach:

In this technique eye blinking frequency, head pose, etc.of a person is monitored through a camera and the person is alerted if any of these drowsiness symptoms are detected.

In our project we are using behavioural based approach.

# FLOW-CHART AND ALGORITHM:



**Face Detection:**

For the face Detection it uses Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used.

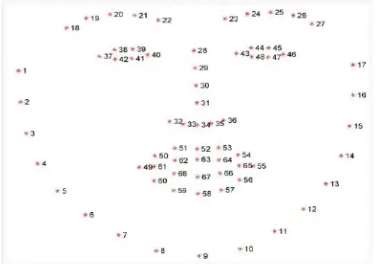
They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle. Due to the difference of facial feature, Haar- like feature is efficient for real-time face detection. These can be calculated according to the difference of sum of pixel values within rectangle areas. The features can be represented by the different composition of the black region and white region.

# Eye detection:

The first thing to do is to find eyes before we can move on to image processing and to find the eyes we need to find a face. The facial keypoint detector takes

a rectangular object of the dlib module as input which is simply the coordinates of a face. To find faces we can use the inbuilt frontal face detector of dlib. We can use any classifier for this task. In our project we are using SVM classifier for detect the drowsiness, as it will give much better accuracy especially for non-frontal facing faces and partially occluded faces.

The indexes of the 68 coordinates can be visualized on the image below:



We can detect and access both the eye region by the following facial landmark index show below

•The right eye using [36, 42].

•The left eye with [42, 48].

These annotations are part of the 68 point [iBUG 300-W dataset](https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/) which the dlib facial landmark predictor was trained on.

# Recognition of Eye's State:

The eye area can be estimated from optical flow, by sparse tracking or by frame-to-frame intensity differencing and adaptive thresholding. And finally, a decision is made whether the eyes are or are not covered by eyelids. A different approach is to infer the state of the eye opening from a single image, as e.g. by correlation matching with open and closed eye templates, a heuristic horizontal or vertical image intensity projection over the eye region, a parametric

model fitting to find the eyelids, or active shape models. A major drawback of the previous approaches is that they usually implicitly impose

too strong requirements on the setup, in the sense of a relative face-camera pose (head orientation), image resolution, illumination, motion dynamics, etc.

Especially the heuristic methods that use raw image intensity are likely to be very sensitive despite their real-time performance. Therefore, we propose a simple but efficient algorithm to detect eye blinks by using a recent facial landmark detector. A single scalar quantity that reflects a level of the eye opening is derived from the landmarks. Finally, having a per-frame sequence of the eye-opening estimates, the eye blinks are found by an SVM classifier that is trained on examples of blinking and non- blinking patterns.

# Drowsiness Detection:

The last step of the algorithm is to determine the person’s condition based on a pre-set condition for drowsiness. The average blink duration of a person is 100- 400 milliseconds (i.e. 0.1-0.4 of a second). Hence if a person is drowsy his eye closure must be beyond this interval. We set a time frame of 5 seconds. If the eyes remain closed for five or more seconds, drowsiness is detected and alert pop regarding this is triggered.

# CODE: TO DETECT EYE BLINK:

import numpy

from pygame import mixer import time

import cv2

from tkinter import \* import tkinter.messagebox root=Tk() root.geometry('500x570')

frame = Frame(root, relief=RIDGE, borderwidth=2) frame.pack(fill=BOTH,expand=1)

root.title('Driver Cam') frame.config(background='light blue')

label = Label(frame, text="Driver Cam",bg='light blue',font=('Times 35 bold')) label.pack(side=TOP)

def hel():

help(cv2) def Contri():

tkinter.messagebox.showinfo("Contributors","\n1.Mayur Kadam\n2. Abhishek Ezhava \n3. Rajendra Patil \n")

def anotherWin():

tkinter.messagebox.showinfo("About",'Driver Cam version v1.0\n Made Using\n-OpenCV\n-Numpy\n-Tkinter\n In Python 3')

menu = Menu(root) root.config(menu=menu) subm1 = Menu(menu)

menu.add\_cascade(label="Tools",menu=subm1) subm1.add\_command(label="Open CV Docs",command=hel) subm2 = Menu(menu) menu.add\_cascade(label="About",menu=subm2) subm2.add\_command(label="Driver Cam",command=anotherWin) subm2.add\_command(label="Contributors",command=Contri)

def exitt():

exit()

def web():

capture =cv2.VideoCapture(0) while True: ret,frame=capture.read()

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

if cv2.waitKey(1) & 0xFF ==ord('q'): break

capture.release() cv2.destroyAllWindows() def webrec():

capture =cv2.VideoCapture(0) fourcc=cv2.VideoWriter\_fourcc(\*'XVID') op=cv2.VideoWriter('Sample1.avi',fourcc,11.0,(640,480)) while True:

ret,frame=capture.read()

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

op.write(frame)

if cv2.waitKey(1) & 0xFF ==ord('q'):

break op.release() capture.release()

cv2.destroyAllWindows() def webdet():

capture =cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier('lbpcascade\_frontalface.xml') eye\_glass = cv2.CascadeClassifier('haarcascade\_eye\_tree\_eyeglasses.xml') while True:

ret, frame = capture.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) faces = face\_cascade.detectMultiScale(gray)

for (x,y,w,h) in faces:

font = cv2.FONT\_HERSHEY\_COMPLEX

cv2.putText(frame,'Face',(x+w,y+h),font,1,(250,250,250),2,cv2.LINE\_AA) cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

roi\_gray = gray[y:y+h, x:x+w] roi\_color = frame[y:y+h, x:x+w]

eye\_g = eye\_glass.detectMultiScale(roi\_gray) for (ex,ey,ew,eh) in eye\_g:

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2) cv2.imshow('frame',frame)

if cv2.waitKey(1) & 0xff == ord('q'):

break capture.release()

cv2.destroyAllWindows() def webdetRec():

capture =cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier('lbpcascade\_frontalface.xml')

eye\_glass = cv2.CascadeClassifier('haarcascade\_eye\_tree\_eyeglasses.xml') fourcc=cv2.VideoWriter\_fourcc(\*'XVID') op=cv2.VideoWriter('Sample2.avi',fourcc,9.0,(640,480))

while True:

ret, frame = capture.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) faces = face\_cascade.detectMultiScale(gray)

for (x,y,w,h) in faces:

font = cv2.FONT\_HERSHEY\_COMPLEX

cv2.putText(frame,'Face',(x+w,y+h),font,1,(250,250,250),2,cv2.LINE\_AA) cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

roi\_gray = gray[y:y+h, x:x+w] roi\_color = frame[y:y+h, x:x+w]

eye\_g = eye\_glass.detectMultiScale(roi\_gray) for (ex,ey,ew,eh) in eye\_g:

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2) op.write(frame)

cv2.imshow('frame',frame)

if cv2.waitKey(1) & 0xff == ord('q'):

break op.release() capture.release()

cv2.destroyAllWindows() def alert():

mixer.init() alert=mixer.Sound('beep-07.wav') alert.play()

time.sleep(0.1) alert.play()

def blink():

capture =cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier('lbpcascade\_frontalface.xml') eye\_cascade = cv2.CascadeClassifier('haarcascade\_eye.xml') blink\_cascade = cv2.CascadeClassifier('CustomBlinkCascade.xml') while True:

ret, frame = capture.read()

gray = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY) faces = face\_cascade.detectMultiScale(gray)

for (x,y,w,h) in faces:

font = cv2.FONT\_HERSHEY\_COMPLEX

cv2.putText(frame,'Face',(x+w,y+h),font,1,(250,250,250),2,cv2.LINE\_AA) cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

roi\_gray = gray[y:y+h, x:x+w] roi\_color = frame[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(roi\_gray) for(ex,ey,ew,eh) in eyes: cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2) blink = blink\_cascade.detectMultiScale(roi\_gray) for(eyx,eyy,eyw,eyh) in blink:

cv2.rectangle(roi\_color,(eyx,eyy),(eyx+eyw,eyy+eyh),(255,255,0),2) alert()

cv2.imshow('frame',frame)

if cv2.waitKey(1) & 0xFF ==ord('q'):

break capture.release()

cv2.destroyAllWindows() but1=Button(frame,padx=5,pady=5,width=39,bg='white',fg='black',relief=GRO OVE,command=web,text='Open Cam',font=('helvetica 15 bold')) but1.place(x=5,y=104) but2=Button(frame,padx=5,pady=5,width=39,bg='white',fg='black',relief=GRO OVE,command=webrec,text='Open Cam & Record',font=('helvetica 15 bold')) but2.place(x=5,y=176) but3=Button(frame,padx=5,pady=5,width=39,bg='white',fg='black',relief=GRO OVE,command=webdet,text='Open Cam & Detect',font=('helvetica 15 bold')) but3.place(x=5,y=250) but4=Button(frame,padx=5,pady=5,width=39,bg='white',fg='black',relief=GRO OVE,command=webdetRec,text='Detect & Record',font=('helvetica 15 bold')) but4.place(x=5,y=322) but5=Button(frame,padx=5,pady=5,width=39,bg='white',fg='black',relief=GRO OVE,command=blink,text='Detect Eye Blink & Record With Sound',font=('helvetica 15 bold'))

but5.place(x=5,y=400) but5=Button(frame,padx=5,pady=5,width=5,bg='white',fg='black',relief=GROO VE,text='EXIT',command=exitt,font=('helvetica 15 bold')) but5.place(x=210,y=478)

root.mainloop()

# EXPERIMENTAL ANALYSIS:

EYE BLINK DETECTION:



**CODE: TO DETECT DROWSY**

#!/usr/bin/python

from future import division import dlib

from imutils import face\_utils import cv2

import numpy as np

from scipy.spatial import distance as dist import threading

import pygame def start\_sound():

pygame.mixer.init() pygame.mixer.music.load("z.ogg") pygame.mixer.music.play()

def resize(img, width=None, height=None, interpolation=cv2.INTER\_AREA): global ratio

w, h = img.shape

if width is None and height is None:

return img

elif width is None:

ratio = height / h width = int(w \* ratio)

resized = cv2.resize(img, (height, width), interpolation) return resized

else:

ratio = width / w height = int(h \* ratio)

resized = cv2.resize(img, (height, width), interpolation) return resized

def shape\_to\_np(shape, dtype="int"):

coords = np.zeros((68, 2), dtype=dtype) for i in range(36,48):

coords[i] = (shape.part(i).x, shape.part(i).y) return coords

def eye\_aspect\_ratio(eye):

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

camera = cv2.VideoCapture(1)

predictor\_path = 'shape\_predictor\_68\_face\_landmarks.dat\_2' detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor(predictor\_path)

(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"] (rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"] total=0

alarm=False while True:

ret, frame = camera.read() if ret == False:

print('Failed to capture frame from camera. Check camera index in cv2.VideoCapture(0) \n')

break

frame\_grey = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) frame\_resized = resize(frame\_grey, width=120)

# Ask the detector to find the bounding boxes of each face. The 1 in the

# second argument indicates that we should upsample the image 1 time. This # will make everything bigger and allow us to detect more faces.

dets = detector(frame\_resized, 1) if len(dets) > 0:

for k, d in enumerate(dets):

shape = predictor(frame\_resized, d) shape = shape\_to\_np(shape) leftEye= shape[lStart:lEnd] rightEye= shape[rStart:rEnd] leftEAR= eye\_aspect\_ratio(leftEye)

rightEAR = eye\_aspect\_ratio(rightEye) ear = (leftEAR + rightEAR) / 2.0 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)

if ear>.25:

print (ear) total=0 alarm=False

cv2.putText(frame, "Eyes Open ", (10, 30),cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

else:

total+=1

if total>20:

if not alarm:

alarm=True d=threading.Thread(target=start\_sound) d.setDaemon(True)

d.start()

print ("so jaaaaaaaaaa")

cv2.putText(frame, "drowsiness detect" ,(250, 30),cv2.FONT\_HERSHEY\_SIMPLEX, 1.7, (0, 0, 0), 4)

cv2.putText(frame, "Eyes close".format(total), (10, 30),cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

for (x, y) in shape:

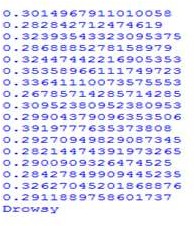
cv2.circle(frame, (int(x/ratio), int(y/ratio)), 3, (255, 255, 255), -1) cv2.imshow("image", frame)

if cv2.waitKey(1) & 0xFF == ord('q'): cv2.destroyAllWindows() camera.release()

break

DROWSY DETECTION**:**

After certain threshold the eye remain close, then it display message as drowsy (Eyes close) with alarm sound.



# RESULT:

Implementation of drowsiness detection with Python and OpenCV was done which includes the following steps: Successful runtime capturing of video with camera. Captured video was divided into frames and each frame were analysed. Successful detection of face followed by detection of eye. If closure of eye for successive frames were detected, then it is classified as drowsy condition else it is regarded as normal blink and the loop of capturing image and analysing the state of driver is carried out again and again.

# REFERENCES:

1. Facial Features Monitoring for Real Time Drowsiness Detection by Manu B.N, 2016 12th International Conference on Innovations in Information Technology.
2. Real Time Drowsiness Detection using Eye Blink Monitoring by Amna Rahman Department of Software Engineering Fatima Jinnah Women University 2015 National Software Engineering Conference.
3. Implementation of the Driver Drowsiness Detection System by K. Srijayathi International Journal of Science, Engineering and Technology Research (IJSETR) Volume 2, Issue 9, September 2013.