Drowsiness Detection System

Driver fatigue is one of the major causes of accidents in the world. Detecting the drowsiness of the driver is one of the surest ways of measuring driver fatigue. In this project we aim to develop a prototype drowsiness detection system. This system works by monitoring the eyes of the driver and sounding an alarm when he/she is drowsy.

The system so designed is a non-intrusive real-time monitoring system. The priority is on improving the safety of the driver without being obtrusive. In this project the eye blink of the driver is detected. If the drivers 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 for this is done in OpenCV using the Haarcascade library for the detection of facial features.

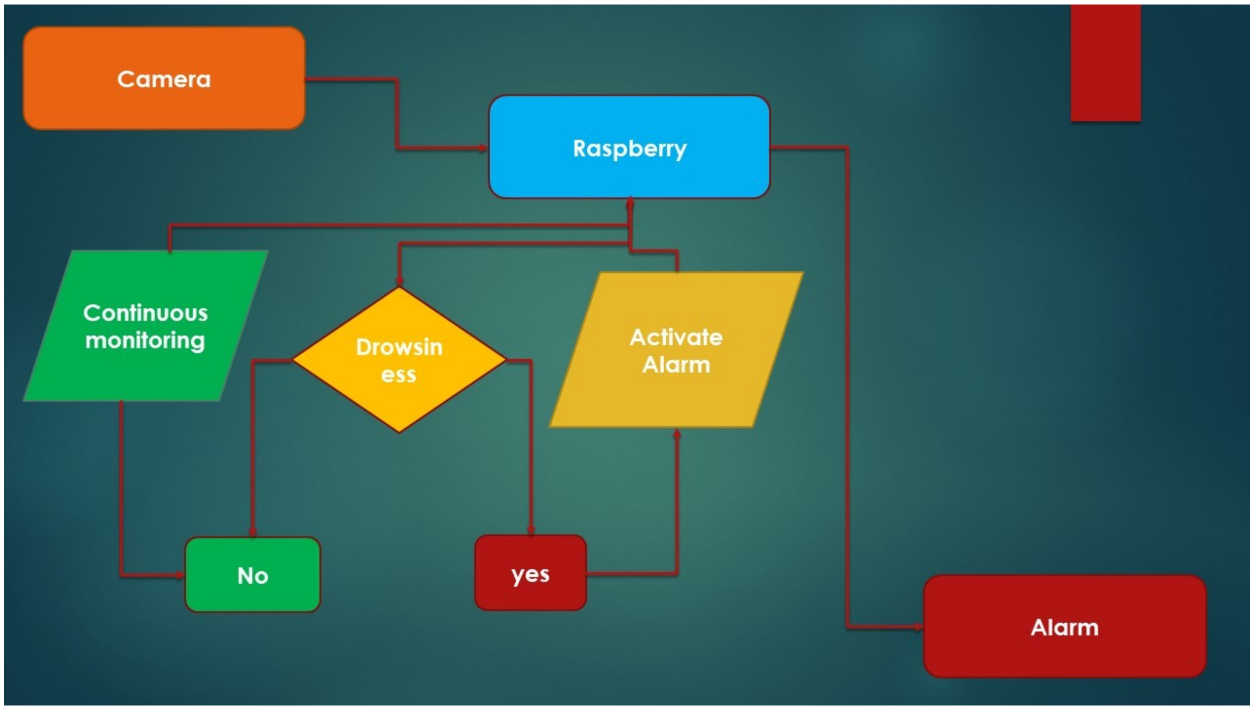
Libraries:

Libraries used in this project are mentioned below.

- cv2 (OpenCV)
- Numpy
- dlib
- hypot(math)
- threading(Thread,Timer)
- time
- vlc (python-vlc)

OpenCV: OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source library that includes several hundreds of computer vision algorithms. We can install this pip through main terminal or anaconda terminal by the command " pip install python-opencv ".
Numpy: This library is useful to so manipulation with numbers.
Dlib: This is one of the most important library in this program. From this, we are using "frontal_face_detector" and "Shape_predictor_68_face_landmarks" shape predictor.
Hypot: Useful to calculate distance between two numpy points.
threading(Thread): Useful to create multi-parallel tasks.
Time: Useful to pause the program flow for few milliseconds.
Vlc: Used to start and stop audio.

Workflow With Raspberry Pi



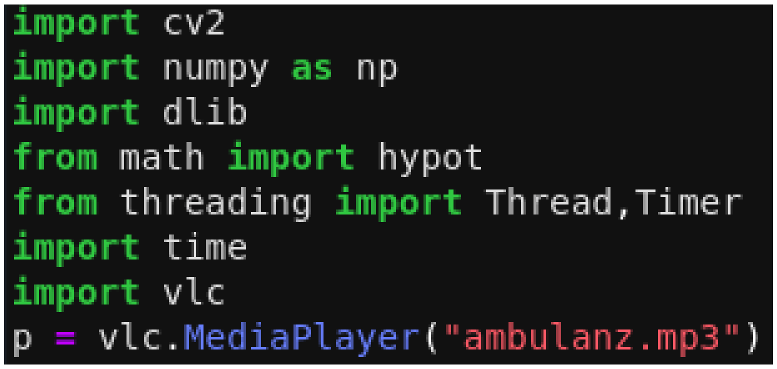
Algorithm and Implementation

Drowsiness of a person can be measured by the extended period of time for which his/her eyes are in closed state. In our system, primary attention is given to the faster detection and processing of data.

The number of frames for which eyes are closed is monitored. If the number of frames exceeds a certain value, then a warning message is generated on the display showing that the driver is feeling drowsy.

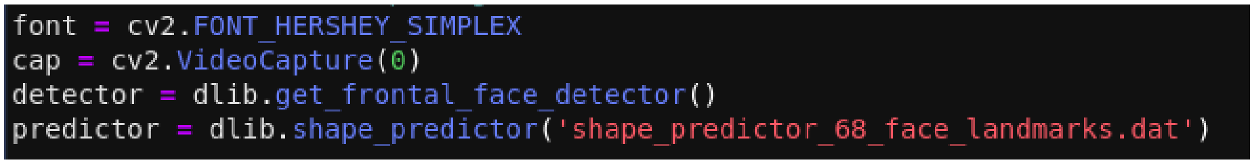
Step by step implementation of this project :-

As first and foremost step, we attached all the required python libraries. It includes:



```
import cv2
import numpy as np
import dlib
from math import hypot
from threading import Thread,Timer
import time
import vlc
p = vlc.MediaPlayer("ambulanz.mp3")
```

I selected *HERSHEY_SIMPLEX* **font** from cv2 to display information on the video screen below.



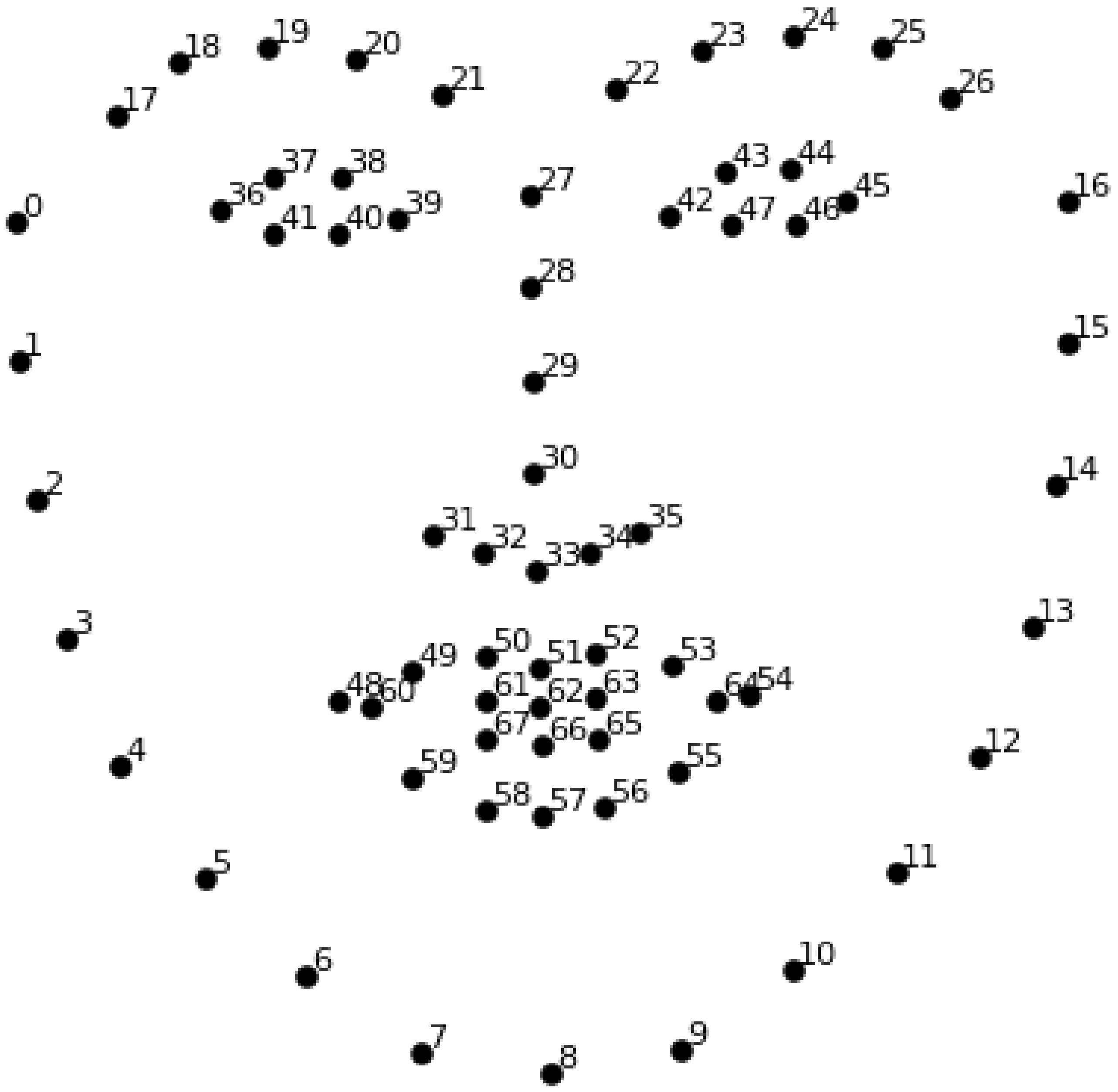
```
font = cv2.FONT_HERSHEY_SIMPLEX
cap = cv2.VideoCapture(0)
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
```

cap is the variable which will start video from my laptop webcam and hold all the frames which are capturing by the webcam.

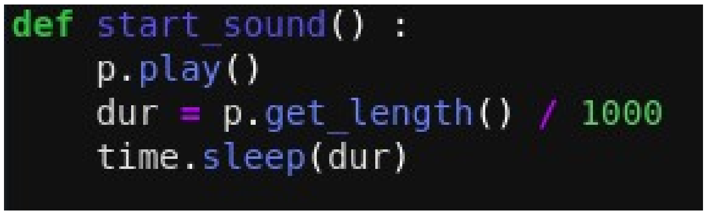
get_frontal_face_detector() is one of the function in dlib, which will detect faces from the images supplied to this function.

For more details http://dlib.net/python/index.html#dlib.get_frontal_face_detector

shape_predictor is another important library from dlib, to this function, we are passing **shape_predictor_68_face_landmarks.dat**, so that, this predictor variable will give all the facial points from the supplier image.

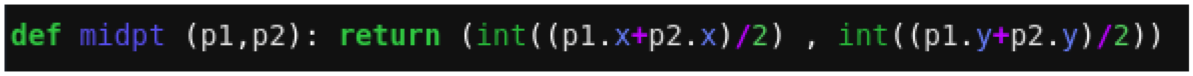

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

After defining the user defined variables, I started preparing User_Defined_Functions.



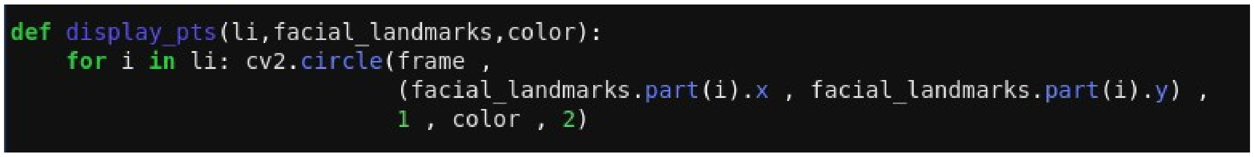
```python
def start_sound() :
    p.play()
    dur = p.get_length() / 1000
    time.sleep(dur)
```

A simple start_sound function, it will start the audio file which is in **p** variable (ambulanz.mp3). And hold the processing thread until the audio file playing is completed.



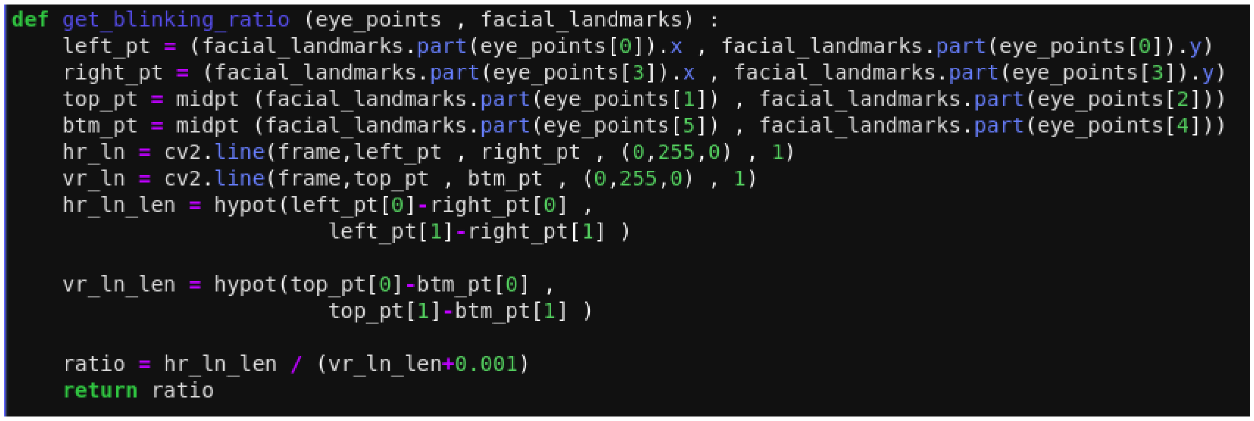
```python
def midpt (p1,p2): return (int((p1.x+p2.x)/2) , int((p1.y+p2.y)/2))
```

One more simple function midpt, which is taking two numpy co-ordinate points and returns the middle point co-ordinates of those two input points.



```python
def display_pts(li,facial_landmarks,color):
    for i in li: cv2.circle(frame ,
                            (facial_landmarks.part(i).x , facial_landmarks.part(i).y) ,
                            1 , color , 2)
```

display_pts function will take a list of facial points, output landmark from predictor and type of color. It will create circles at all locations of points in input list with a 1mm radius and 2mm thick with the input color.

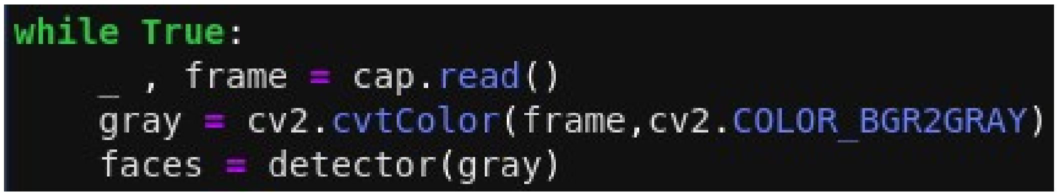


```python
def get_blinking_ratio (eye_points , facial_landmarks) :
    left_pt = (facial_landmarks.part(eye_points[0]).x , facial_landmarks.part(eye_points[0]).y)
    right_pt = (facial_landmarks.part(eye_points[3]).x , facial_landmarks.part(eye_points[3]).y)
    top_pt = midpt (facial_landmarks.part(eye_points[1]) , facial_landmarks.part(eye_points[2]))
    btm_pt = midpt (facial_landmarks.part(eye_points[5]) , facial_landmarks.part(eye_points[4]))
    hr_ln = cv2.line(frame,left_pt , right_pt , (0,255,0) , 1)
    vr_ln = cv2.line(frame,top_pt , btm_pt , (0,255,0) , 1)
    hr_ln_len = hypot(left_pt[0]-right_pt[0] ,
                      left_pt[1]-right_pt[1] )

    vr_ln_len = hypot(top_pt[0]-btm_pt[0] ,
                      top_pt[1]-btm_pt[1] )

    ratio = hr_ln_len / (vr_ln_len+0.001)
    return ratio
```

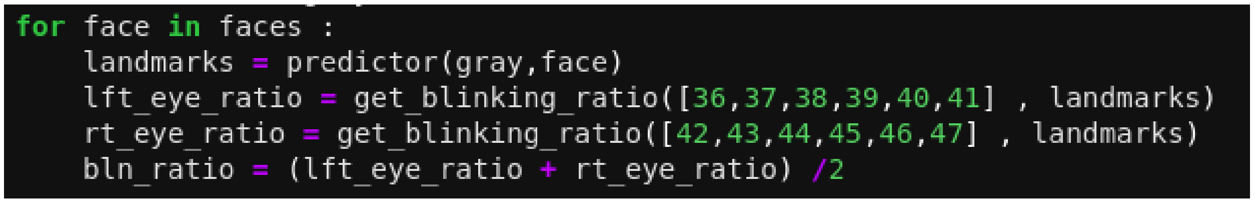
get_blinking_ratio function will take the current positions of one eye and create horizontal and vertical line for that eye position, and also it will calculate the length of both the lines and return the ratio of horizontal line length to vertical line length.



```
while True:
    _ , frame = cap.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
```

After defining all user defined functions and variables, I started reading the video. cap.read() will emit the current image of VideoCapture and it is saved in frame variable.

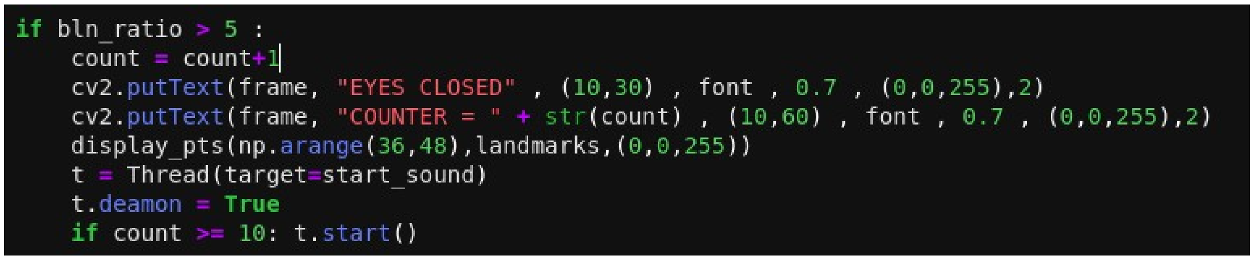
In cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY), I am converting the frame into gray (Black&White) and saved in "gray" variable. I submitted the gray image to detector to find out the frontal face features. I saved the face features in faces variable. As we know, the faces variable is having only the face feature.



```
for face in faces :
    landmarks = predictor(gray,face)
    lft_eye_ratio = get_blinking_ratio([36,37,38,39,40,41] , landmarks)
    rt_eye_ratio = get_blinking_ratio([42,43,44,45,46,47] , landmarks)
    bln_ratio = (lft_eye_ratio + rt_eye_ratio) /2
```

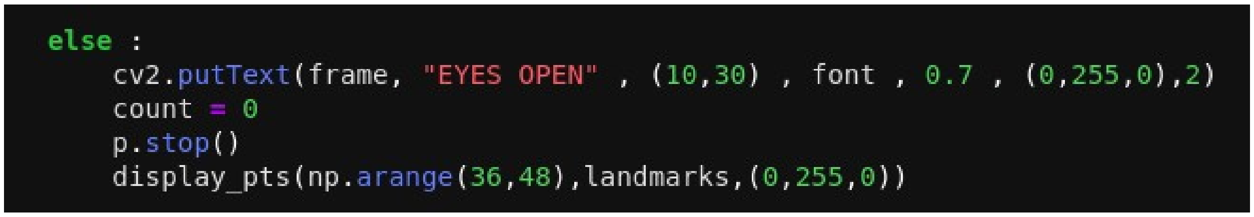
Now, out of faces I taken only first capture as face and sent it to predictor (**dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')**). It will give the position of all 68 facial points position as per Landmark Points — dlib. I saved all 68 points position in landmarks variable.

I calculated the left eye ratio and right eye ratio by using our predefined function (get_blinking_ratio). And further more the average ratio of both eyes.



```
if bln_ratio > 5 :
    count = count+1
    cv2.putText(frame, "EYES CLOSED" , (10,30) , font , 0.7 , (0,0,255),2)
    cv2.putText(frame, "COUNTER = " + str(count) , (10,60) , font , 0.7 , (0,0,255),2)
    display_pts(np.arange(36,48),landmarks,(0,0,255))
    t = Thread(target=start_sound)
    t.deamon = True
    if count >= 10: t.start()
```

If the average ratio of eyes is greater than 5 (means horizontal length is much more than vertical height), then I assumed that the eye is closed. I started a counter with closed eye frame. I placed a text as EYES CLOSED and COUNTER value on the output frame. I displayed the eye points in red color using our predefined user function display_pts . If the closed eye is continuing for 10 frames, then the sound starts.



```
else :
    cv2.putText(frame, "EYES OPEN" , (10,30) , font , 0.7 , (0,255,0),2)
    count = 0
    p.stop()
    display_pts(np.arange(36,48),landmarks,(0,255,0))
```

If the eye is opened (the ratio is less than 5), then the text will change to EYES OPEN , and the counter will become zero, the sound will stop and the display points of eye will become blue.

Source Code:-

```
################### Importing all libraries ###########################
import cv2
import numpy as np
import dlib
from math import hypot
from threading import Thread,Timer
import time
import vlc
p = vlc.MediaPlayer("ambulanz.mp3")

################### Preparing initial level variables ###########################
font = cv2.FONT_HERSHEY_SIMPLEX
cap = cv2.VideoCapture(0)
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

################### User defined functions###########################
def start_sound() :
    p.play()
    dur = p.get_length() / 1000
    time.sleep(dur)

def midpt (p1,p2): return (int((p1.x+p2.x)/2) , int((p1.y+p2.y)/2))

def display_pts(li,facial_landmarks,color):
    for i in li: cv2.circle(frame ,
                            (facial_landmarks.part(i).x , facial_landmarks.part(i).y) ,
                            1 , color , 2)

def get_blinking_ratio (eye_points , facial_landmarks) :
    left_pt = (facial_landmarks.part(eye_points[0]).x , facial_landmarks.part(eye_points[0]).y)
    right_pt = (facial_landmarks.part(eye_points[3]).x , facial_landmarks.part(eye_points[3]).y)
    top_pt = midpt (facial_landmarks.part(eye_points[1]) , facial_landmarks.part(eye_points[2]))
    btm_pt = midpt (facial_landmarks.part(eye_points[5]) , facial_landmarks.part(eye_points[4]))
    hr_ln = cv2.line(frame,left_pt , right_pt , (0,255,0) , 1)
    vr_ln = cv2.line(frame,top_pt , btm_pt , (0,255,0) , 1)
    hr_ln_len = hypot(left_pt[0]-right_pt[0] ,
                      left_pt[1]-right_pt[1] )

    vr_ln_len = hypot(top_pt[0]-btm_pt[0] ,
                      top_pt[1]-btm_pt[1] )

    ratio = hr_ln_len / (vr_ln_len+0.001)
    return ratio

################### Main function ###########################

while True:
    _ , frame = cap.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    for face in faces :
        landmarks = predictor(gray,face)
        lft_eye_ratio = get_blinking_ratio([36,37,38,39,40,41] , landmarks)
        rt_eye_ratio = get_blinking_ratio([42,43,44,45,46,47] , landmarks)
        bln_ratio = (lft_eye_ratio + rt_eye_ratio) /2

        if bln_ratio > 5 :
            count = count+1
            cv2.putText(frame, "EYES CLOSED" , (10,30) , font , 0.7 , (0,0,255),2)
            cv2.putText(frame, "COUNTER = " + str(count) , (10,60) , font , 0.7 , (0,0,255),2)
            display_pts(np.arange(36,48),landmarks,(0,0,255))
            t = Thread(target=start_sound)
            t.deamon = True
            if count >= 10: t.start()

        else :
            cv2.putText(frame, "EYES OPEN" , (10,30) , font , 0.7 , (0,255,0),2)
            count = 0
            p.stop()
            display_pts(np.arange(36,48),landmarks,(0,255,0))


    cv2.imshow('Frame' , frame)
    key = cv2.waitKey(1)
    if key == ord('q') : break

cap.release()
cv2.destroyAllWindows()
```

Summary:-

To obtain the result a large number of videos were taken and their accuracy in determining eye blinks and drowsiness was tested. For this project we can used laptop webcam and mobile device connected to the computer.

The system was tested for different people in different ambient time. When the webcam back light was turned ON and the face is kept at an optimum distance, then the system is able to detect blinks as well as drowsiness with more than 90% accuracy. This is a good result and can be implemented in real-time systems as well.

Conclusion:

Thus we have successfully designed a prototype drowsiness detection system using OpenCV software. The system so developed was successfully tested, its limitations identified and a future plan of action developed.

References:

- get_frontal_face_detector: http://dlib.net/python/index.html#dlib.get_frontal_face_detector
- The-68-specific-human-face-landmark:http://researchgate.net/figure/The-68-specific-human-face-landmarks_fig4_331769278/download
- numpy: https://numpy.org/
- python-vlc: https://wiki.videolan.org/Python_bindings/
- threading: https://docs.python.org/3/library/threading.html
- dlib: http://dlib.net/python/index.html
- time: https://docs.python.org/3/library/time.html
- math(hypot): https://docs.python.org/3/library/math.html#math.hypot
- OpenCV: https://docs.opencv.org/master/
- github: https://github.com/ZayanShahid/Projects/tree/main/Driver_Drowsiness_Detection