Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Identifying drowsiness level using Opencv and Dlib

Course and Course Code: Image processing (CSE-4019)
Team Members:
Charit Gupta Paluri (19BCT0251)
Soham Adhikari (19BCE0816)
Kshitij Yogesh Kotnis (19BCE0979)
Manav Pajesh Parikh (19BCE2013)
Faculty: Prof. Rajakumar K
Slot: C2+TC2

Abstract:

Since the economic restructure of our nation, millions of vehicles have been bought by Indians and this number increases exponentially every year with our economic growth. The economic development unfortunately hasn't translated to better infrastructure in many parts of the country, due to which the state of our roads is pitiful. This coupled with the reckless nature of drivers in our country have made the roads very unsafe for pedestrians, animals and drivers alike. In a year thousands of accidents are caused by drowsy-driving. These crashes led to an estimated 50,000 people injured and nearly 800 deaths. To prevent these reckless accidents, we are designing an application which will detect the drowsiness level of the driver, using their eyes movement. The application is being created using OpenCV.

Problem statement

Aim: To create a program which helps late night driving much safer without causing any accidents.
Objective: This program helps to prevent late night accidents which are caused by drowsiness of a driver. This objective can be achieved by using image processing concepts segmentation, feature extraction, and pixel relationship to calculate the respective ratio, which helps to determine the drowsy level of the test driver.

Literature review

In every project literature reviews play a major role in the research process. In this phase the ideas are drawn together and they are arranged in a logical manner to give out expected output for any test samples. This tool is aimed to develop a prototype which helps drivers to identify their drowsiness level and react accordingly. According to the Survey on Driver Drowsiness Detection System, the detection system includes the processes of face image extraction, yawning tendency,and eye area extraction. In this project we have used OpenCV along with Dlib libraries in python. These libraries help with real time image processing which has free of cost implementations on latest computer vision algorithms. Using these algorithms we can check for timespan checking of a driver's cautiousness. These equipment and imaging calculations square measure created to at the same time remove various viewable signs that for the most part describe an individual's dimension of weariness.This system aims at using facial landmarks which helps to identify drowsiness of the driver. This is far better than previous methods, and this method reduces the errors and gives out accurate results.

Existing method

There are other methods to detect drowsiness of a person by using EEG signals in neuroscience and engineering which targets a range of practical safety-awareness applications and use. This method uses broadbands such as alpha and beta. Even though the broadband alpha, beta, delta, and theta waves are often used as features in the existing work,its lack of widely agreed precise definitions of such broadband signals and difficulty in accounting for interpersonal variability has led to poor classification performance as demonstrated in this study. While this is one of the major problems, there still exists one more major problem of using it while driving as a person won't like to be connected to a device while driving an automobile.
Over here we can also use SVM(support vector machine) to classify the components in the input video. While cropping the region of interest components in the video is not accurate. Sometimes it will show regions wrong. To sense the eyes first we have to create boundary boxes for that and a classification algorithm. The algorithm of SVM will not support it.

Proposed method

There are several different algorithms and methods for eye tracking, and monitoring. Most of them in some way relate to features of the eye within a video image of the driver. The aim of this project is to use the retinal reflection as a means to finding the eyes on the face, and then using the absence of this reflection as a way of detecting when the eyes are closed. Applying this algorithm on consecutive video frames may aid in the calculation of eye closure period.
So basically we use the method of detecting drowsiness by using facial landmarks with dlib libraries which just requires a person to install a small webcam in front of his seat and run a code which takes at most accurate values and is very efficient. This proposed method has some requirements such as Laptop, with camera and software requirements are OpenCV, dlib, and Python.

Block Diagram

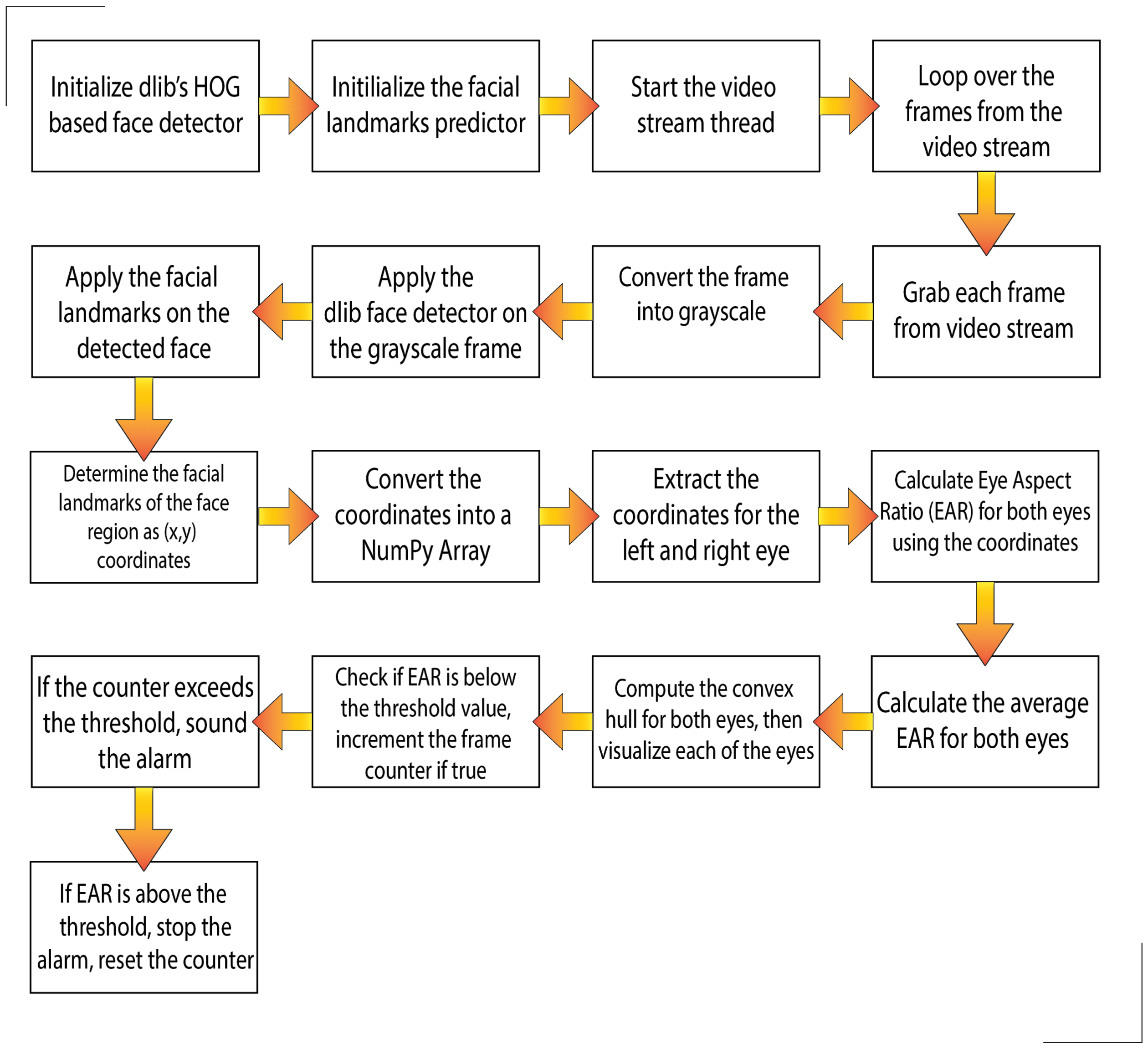

Initialize dlib's HOG based face detector
Initilialize the facial landmarks predictor
Start the video stream thread
Loop over the frames from the video stream
Grab each frame from video stream
Convert the frame into grayscale
Apply the dlib face detector on the grayscale frame
Apply the facial landmarks on the detected face
Determine the facial landmarks of the face region as (x,y) coordinates
Convert the coordinates into a NumPy Array
Extract the coordinates for the left and right eye
Calculate Eye Aspect Ratio (EAR) for both eyes using the coordinates
Calculate the average EAR for both eyes
Compute the convex hull for both eyes, then visualize each of the eyes
Check if EAR is below the threshold value, increment the frame counter if true
If the counter exceeds the threshold, sound the alarm
If EAR is above the threshold, stop the alarm, reset the counter

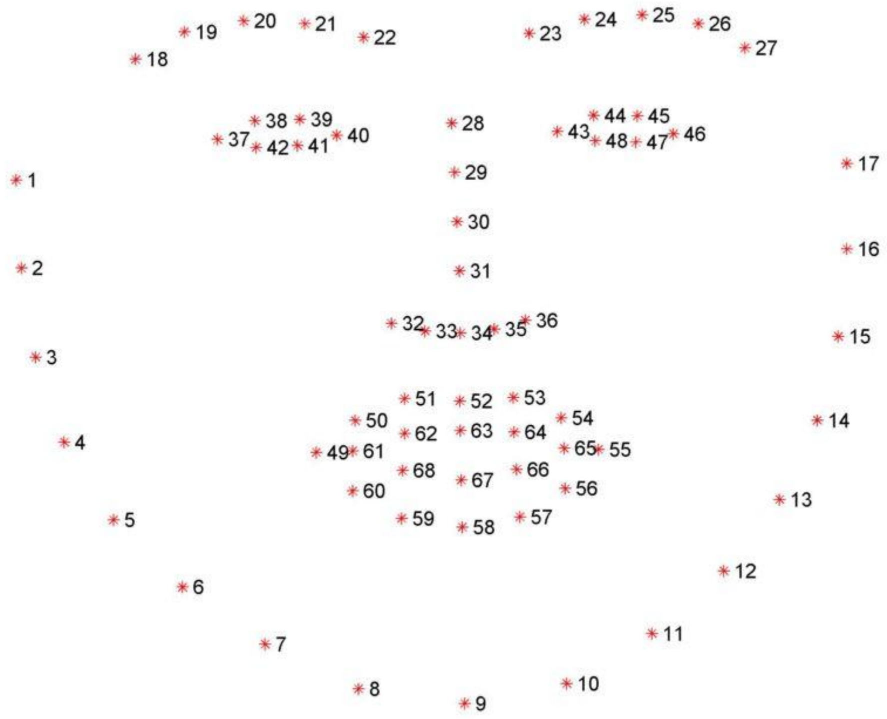
Image Processing Concepts Applied:

1. **Segmentation:** Image segmentation is the process of partitioning an image into multiple segments. Image segmentation is typically used to locate objects and boundaries in images.
2. **Feature Extraction:** Using symmetrical histogram and symmetrical value, outline of the face can be determined.

3. **Pixel relationship:** This was used during the calculations of EAR and MAR which used Euclidean distance.
4. **HOG(Histogram of Oriented Gradients)**: The idea behind HOG is to extract features into a vector, and feed it into a classification algorithm like a Support Vector Machine for example that will assess whether a face (or any object you train it to recognize actually) is present in a region or not.

Implementation:

This system works on two parts, one is detecting the face and predicting the landmarks of important regions in the detected face. These landmarks can be determined using eye aspect ratio(EAR) and Mouth aspect ratio(MAR), using these we can determine the drowsiness of a driver. Using the pre-trained facial landmarks, which is part of dlib library, this is used to estimate the location of 68(x,y)coordinates, which maps out the face structure. These 68 coordinates also exhibit the important aspects of the face such as mouth, eye, nose, jaw,and etc. Each eye uses 6(x,y) coordinates starting at the left corner of the eye, then goes in clockwise direction. If eyes are closed we get a slanted horizontal line, this shows that the driver is highly drowsy or probably sleeping.Using 6(x,y) coordinates on the eye we can calculate EAR. Here an illustration of how a user face is represented using 68(x,y) coordinates.



The facial landmark detector included in the dlib library is an implementation of the *One Millisecond Face Alignment with an Ensemble of Regression Trees* paper by Kazemi and Sullivan (2014).

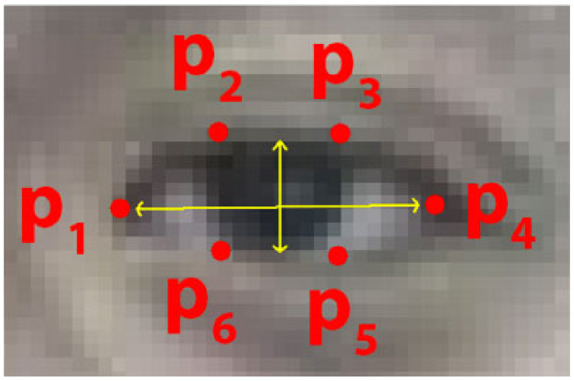
This method starts by using:

1. A training set of labeled facial landmarks on an image. These images are *manually labeled*, specifying specific *(x, y)*-coordinates of regions surrounding each facial structure.
2. *Priors*, or more specifically, the *probability of distance* between pairs of input pixels.

Given this training data, an ensemble of regression trees are trained to estimate the facial landmark positions directly from the *pixel intensities themselves*

Calculations:

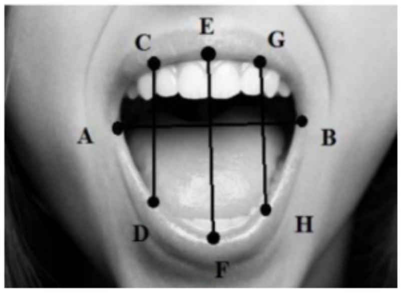
This project mainly works on two calculations which are Eye aspect ratio (EAR) and Mouth aspect ratio(MAR).EAR is an estimate of the eye opening state. It is a constant value but when a person's eye closes it rapidly falls to zero, this can be due to blinking or drowsiness. Using the below eye facial landmarks we can define an equation for EAR.



To calculate EAR we use a formula which is derived using facial landmarks. For this calculation we use euclidean distance between two coordinates.

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Mouth aspect ratio (MAR) is the ratio of length of the mouth to width of the mouth. This MAR comes in use because when an individual is feeling sleepy they are mostly likely to yawn and lose control over their mouth, this can be calculated using euclidean distance betweens horizontally and vertically. For Example MAR is absolute value of euclidean distance from EF and divided by euclidean distance of absolute value AB.



$$\mathrm{MAR} = \frac{|EF|}{|AB|}$$

Code:

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import playsound
import argparse
import imutils
import time
import dlib
import cv2

def sound_alarm(path):
  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

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
  help="path to facial landmark predictor")
ap.add_argument("-a", "--alarm", type=str, default="",
  help="path alarm .WAV file")
```

```
ap.add_argument("-w", "--webcam", type=int, default=0,
  help="index of webcam on system")
args = vars(ap.parse_args())


EYE_AR_THRESH = 0.2
EYE_AR_CONSEC_FRAMES = 48

COUNTER = 0
ALARM_ON = False

print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])


(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

print("[INFO] starting video stream thread...")
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)

while True:
  frame = vs.read()
  frame = imutils.resize(frame, width=450)
  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

  rects = detector(gray, 0)

  for rect in rects:
     shape = predictor(gray, rect)
     shape = face_utils.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 < EYE_AR_THRESH:
        COUNTER += 1
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
           if not ALARM_ON:
              ALARM_ON = True
              if args["alarm"] != "":
                 t = Thread(target=sound_alarm,
                    args=(args["alarm"],))
                 t.deamon = True
                 t.start()

           cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
              cv2.FONT_HERSHEY_COMPLEX, 0.7, (0, 120, 255), 2)

     else:
        COUNTER = 0
        ALARM_ON = False

     cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
        cv2.FONT_HERSHEY_COMPLEX, 0.7, (0, 120, 255), 2)

  cv2.imshow("Frame", frame)
  key = cv2.waitKey(1) & 0xFF

  if key == ord("q"):
     break

cv2.destroyAllWindows()
vs.stop()
```