International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 12 | Dec 2020 www.irjet.net p-ISSN: 2395-0072

Driver Drowsiness Detection with Face Recognition System

Rushikesh Rajendra Kamble[1]

[1]*Student, Department of Computer Engineering in Degree, Sinhgad Institute of Technology and Science, Pune, Maharashtra, India.*

Abstract – *In modern days, we see how car accidents are increasing due to many reasons like drowsy driving or drunk driving or speeding and many more reasons. Hence we develop a modern solution, were my system will alert the driver if driver is sleeping.*

My system recognizes the face of driver, which will help us to maintain a report. The system will display a bar chart for how much time a driver has drove a car and line chart for at which time driver had drowsiness.

Report will help the drivers, how much time he/she has to drive a car and will also prevent accidents.

System is using EAR technique to detect if the driver is sleeping or not.

Key Words: Driver, Drowsiness, Face, Recognition, System, Python

1. INTRODUCTION

In modern days increase in car accidents is a major issue. The reason behind accidents is 40% - 50% drivers feels drowsiness or might be drunk. In that case system will help us to prevent the car accidents. Second major issue is for those parents who don't know how much their son/daughter has driven the car. Report will show on which date and for how much time the car is driven by their child.

The system detects whether the driver is sleeping or not. To detect the drowsiness, system plots 6 points on eyes and find the EAR (Eyes Aspect Ratio). If the EAR values is below the trash value then it will count the frames. After 15 frames it will play the alert to wake up the driver.

I have used python 3.8 to write this code. System uses OpenCV to work on images. I have used stranded face_recognition library to recognize the faces. With the help of facial landmarks, system detects where the eyes are. Matplotlib is used to plot the graphs.

In the report part, I have design two graphs. First one is bar chart and second one is line chart. In bar chart, system displays the driving hours of various drivers. In line chart, system displays at which time driver feels drowsiness. Reports are sorted in date manner.

1.1 Literature Survey

Universidad de las Fuerzas Armadas - ESPE, This College has done work on Real Time Driver Drowsiness Detection Based on Driver's Face Image Behavior Using a System of Human Computer Interaction Implemented in a Smartphone in year January 2018.This project was done by Eddie E. Galarza, Fabricio D. Egas, Franklin M. Silva, Paola M. Velasco, Eddie D. Galarza1. They develop the mobile application to detect the drowsiness. The main drawback of that system is if user did not set phone then reports are not generated. System cannot show how much time one driver has driven a car.

But all operations like detecting faces and checking drowsiness is working well. They did not implement the face recognition part.

2. DRIVER DROWSINESS DETECTION WITH FACE RECOGNITION SYSTEM

Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving. The objective of this intermediate Python project is to build a drowsiness detection system that will detect that a person's eyes are closed for a few seconds. This system will alert the driver when drowsiness is detected.

2.1 Project Description

In this Python project, we will be using OpenCV for gathering the images from webcam and feed them into a Deep Learning model which will classify whether the person's eyes are 'Open' or 'Closed'. The approach, we will be using for this Python project is as follows

Step 1 – Take image as input from a camera.
Step 2 – Detect the face in the image and recognize the face.
Step 3 – Detect the eyes from landmarks.
Step 4 – Find EAR of both eyes.
Step 5 – Check EAR to check whether the person is drowsy

- **Prerequisites**

The requirement for this Python project is a webcam through which we will capture images. You need to have Python (3.6 version recommended) installed on your system, then using pip, you can install the necessary packages.

OpenCV – pip install OpenCV-python (face and eye detection).
MatplotLib - pip install matplotlib (To plot a graph)
pyttsx3 - pip install pyttsx3 (for text to speach)
face_recognition - pip install face-recognition (To recognize a face)

3. TECHNICAL DOCUMENTATION

3.1 EAR

As mentioned, to a certain extent, the state of eyes indicates whether the driver is drowsy or not. Because there are significant differences about time of eyes closed between awake and drowsy. In a method of ellipse fitting was proposed to describe the shape of pupil. As shown in Fig 1, the method segments the pupil with traditional image process firstly. Then, an ellipse is fitted with the white pixels, which represent the shape of eyes. Lastly, the ratio of the major and minor axes of the ellipse was used to evaluate the eyes state.

We noticed its performance might be limited by the following facts:

(1) The pixel values are sensitive. Changeable environment is easy to make image segmentation to be worse.

(2) In practical application, the pixel values between pupils and glasses are very close, which lead to false ellipse fitting. In this paper, we design a new more stable parameter based on Dlib toolkit to evaluate the state of driver's eyes. It is more stable and precise than ellipse fitting method thanks to avoiding the traditional image process.

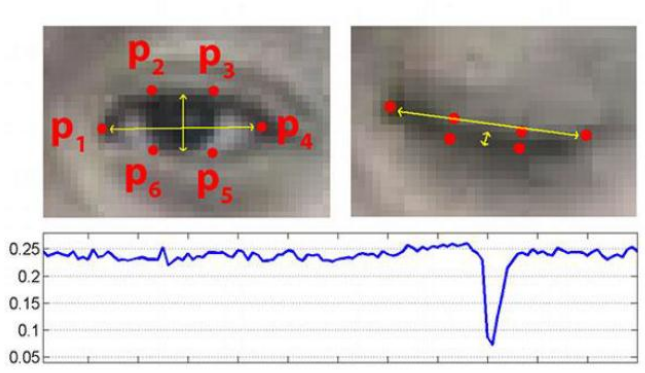

Fig. 1. Eyes landmarks. Upper: the distribution of eyes landmarks has significant differences. Bottom: the values of EAR at open and closed state.

3.2. OpenCV:

Deep Learning is a fast growing domain of Machine Learning and if you're working in the field of computer vision/image processing already (or getting up to speed), it's a crucial area to explore.

With OpenCV 3.3, we can utilize pre-trained networks with popular deep learning frameworks. The fact that they are pre-trained implies that we don't need to spend many hours training the network — rather we can complete a forward pass and utilize the output to make a decision within our application.

OpenCV does not (and does not intend to be) to be a tool for training networks — there are already great frameworks available for that purpose. Since a network (such as a CNN) can be used as a classifier, it makes logical sense that OpenCV has a Deep Learning module that we can leverage easily within the OpenCV ecosystem.

Popular network architectures compatible with OpenCV 3.3 include:

- GoogleLeNet (used in this blog post)
- AlexNet
- SqueezeNet
- VGGNet (and associated flavors)
- ResNet

4. IMPLEMENTATION

Let's now understand how our algorithm works step by step.

Step 1 – Take Image as Input from a Camera With a webcam.
We will take images as input. So, to access the webcam, we made an infinite loop that will capture each frame. We use the method provided by OpenCV, cv2.VideoCapture (0) to access the camera and set the capture object (cap). cap.read () will read each frame and we store the image in a frame variable.
Step 2 – Detect the face in the image and recognize the face.
To detect the face in the image, we need to first convert the image into grayscale as the OpenCV algorithm for object detection takes gray images in the input. We don't need color information to detect the objects. We will be using haar cascade classifier to detect faces. This line is used to set our classifier face = cv2.CascadeClassifier ('path to our haar cascade xml file'). Then we perform the detection using faces = face.detectMultiScale (gray). It returns an array of detections with x, y coordinates, and height, the width of the boundary box of the object. Now we can iterate over the faces and draw boundary boxes for each face.

```
Gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
rects=det.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, flags=cv2.CASCADE_SCALE_IMAGE)

Match = face_recognition.compare_faces(encode_list, face,
0.5)
dist = face_recognition.face_distance(encode_list, face)
```

matin = numpy.argmin(dist)

Step 3 – Detect the eyes from landmarks.
The facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 point mappings were obtained by training a shape predictor on the labeled iBUG 300-W dataset.

```
Def find_eye(shape):
 (lstart,
lend)=face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rstart,
rend)=face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
lefteye=shape[lstart: lend]
righteye=shape[rstart:rend]
```

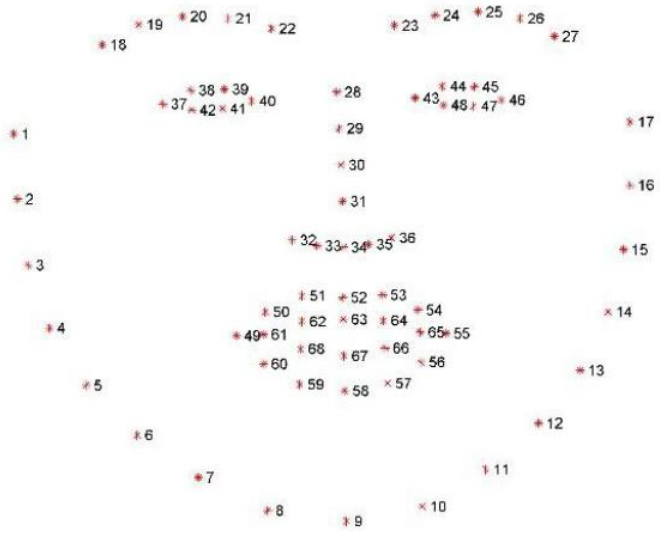

Fig 2:-facial points marked using facial landmark

Step 4 – Find EAR of both eyes.

```
leftEAR=eye_aspect_ratio(lefteye)
right_EAR=eye_aspect_ratio(righteye)
ear=(leftEAR+right_EAR)/2
```

Step 5 – Check EAR to check whether the person is drowsy

We have obtained the facial landmarks based on Dlib toolkit. As shown in Figure 6, for each eye, there are six points distributed around to locate the position of eye. The distribution of eyes landmarks has significant differences between open and closed state. In Eye Aspect Ratio was application to record the blink frequency. EAR can be computed according to the position of eyes landmarks by: EAR = kP2 − P6k + kP3 − P5k 2 kP1 − P4k where Pi, i = 1, 2, . . . , 6 is the coordinate of eyes landmarks. When the eyes of driver are open, the EAR is over 0.2. In contrast, the EAR is less than 0.2.

```
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)/(2.0*C)
return ear
```

5. Results

Let's start our project and see the working of our project. To start the project, you need to open a command prompt, go to the directory where our main file "GUI_DD.py" exists. Run the script with this command.

Fig 3:-Main View (find the user is drowsing and if drowsing then play the alarm)

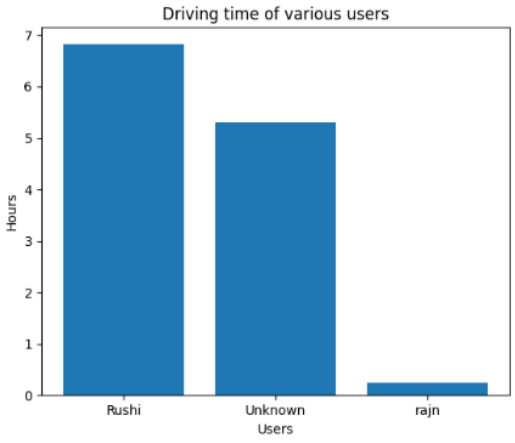

Fig:-4 Display a bar chart

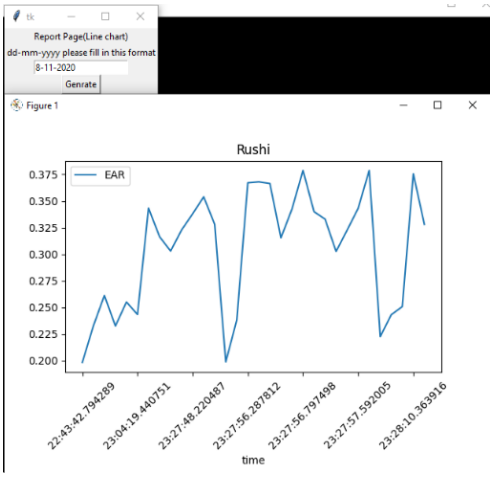

Fig 5 :- line chart for EAR for each user

6. CONCLUSIONS

In this Python project, we have built a drowsy driver alert system that you can implement in numerous ways. I have used OpenCV to detect faces and eyes using a haarcascade classifier and then we used an EAR technique to predict the status.

It is so accurate, it detect the drowsiness on run time and report are also very accurate. It will very helpful to prevent the car accidents.

REFERENCES

[1] Facial landmarks with dlib, OpenCV, and Python by Adrian Rosebrock on https://www.pyimagesearch.com/ in 2017

[2] Real Time Driver Drowsiness Detection Based on Driver's Face Image Behavior Using a System of Human Computer Interaction Implemented in a Smartphone on https://www.researchgate.net/ in 2018

BIOGRAPHIES

NAME: - Rushikesh R. Kamble
CLASS: - TE CSE
Occupation: - Student
COLLEGE: - Sinhgad Institute of Technology and Science (SITS).