Detecting Driver Fatigue, Over-Speeding, and Speeding up Post-Accident Response

Navan Chauhan[1]

[1]Student, Birla Vidya Niketan, New Delhi, India

Abstract - Sleeping while driving, over-speeding and untimely response after a road-accident are a major cause of loss of human life and loss of money. The implementation of real-time monitoring the driver, car parts, speed and monitoring of accidents will aid in minimising accidents. Except, the project is intended for four-wheelers but can also be modified for 2 wheelers. The project is coded in Python and can thus be ported for a number of micro-controllers. It mainly utilises OpenCV2, Overpass, Requests and SimpleJSON modules.

Key Words: Raspberry Pi Zero W, GSM, OpenCV, Python, driver fatigue detection, Eye Aspect Ratio

1. INTRODUCTION

A research conducted by the AAA Foundation for Safety has found that drivers who have slept for less than 7 hours in the past 24 hours, and drivers who have slept for 1 or more hours less than their usual amount of sleep in the past 24 hours have significantly elevated crash rates[1]. According to the Government of India, Over-speeding accounted for 55.9 per cent out of all road accidents registered in India in the year 2016[2]. Therefore it is important to create ways to detect them. If we are somehow able to prevent these accidents, more human lives will be able to fully contribute to humanity and will not be a victim just because they could not sleep properly the previous night. The programs were written in Python3 and were tested on MacOS and Raspbian (On a Raspberry Pi Zero W)

2. METHODOLOGY

A need arises to combat these problem. Calculating the Eye Aspect Ratio is appropriate for detecting if the driver is feeling drowsy or not[3][4]. A check for over-speeding is done by constantly matching the current speed, which is obtained via the OBD-II Port to the maximum speed limit of the road. As the connection to the OBD-II port is already made, a check for all the parts is also done. The speeding up of post-accident response is done by simultaneously tweeting and sending a message.

1.1 Driver Fatigue Detection

Unlike other solutions which check if the sclera of the eyes disappears for a period of time, this calculates the ratio between the different landmarks of the eyes Called the Eye Aspect Ratio (EAR).

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

Figure-1: The EAR Equation

The EAR remains constant when the eye is open, whereas it rapidly declines while the eye closes.

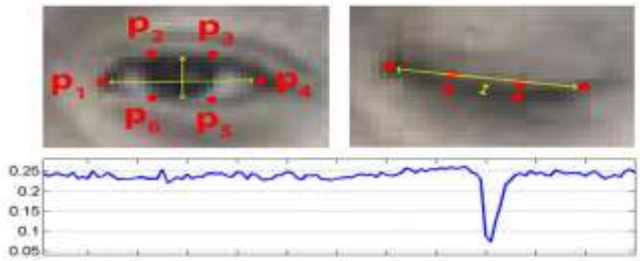

Figure-2: *Top: Eye landmarks when the eye is open or closed. Bottom:* The EAR over time. The dip in EAR indicates that the eye blinked (Figure 1 of Soukupová and Čech).

The dlib module for Python already contains the facial-landmarks function. This checks whether the EAR is below a particular threshold or not, if yes, then it triggers another function to wake the user.

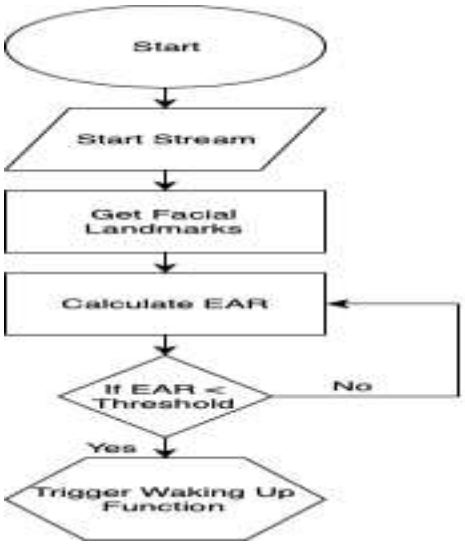

Figure-3: *Flowchart of the Driver Drowsiness Function*

The program was created with a sample function which used the location of the device and then used Zomato's API to find the nearest eatery, advising the user to have a cup of coffee.

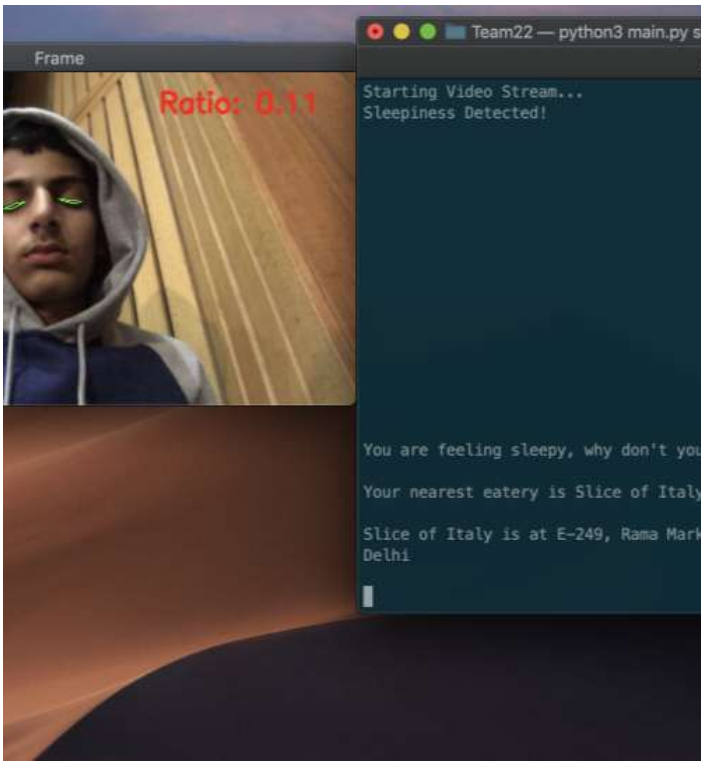

Figure-4: *Left: Live Stream with EAR being displayed. Right: Function to show the address of the nearest eatery*

2.2 Over-Speeding Detection

It utilises the data from the OBD-II port constantly check the speed of the car. This speed is compared with the speed limit of the road which is fetched simultaneously using the Overpass API. Upon Over-Speeding, the driver is constantly notified by beeps, If violation does not stop, the vehicle details are sent to the authority via Short Messaging Service (SMS) or using the Twilio API. If the program is unable to detect the speed limit of the road, the national speed limit is used.

2.3 Speeding up Post-Accident Response

As soon as an accident happens, the speed of the car quickly decreases. When this happens, the program waits for two minutes, if there is no user input (the input can be in the form of pressing a button or anything which is convenient) within two minutes of the sudden braking, the program automatically sends an SMS to the quick-response teams and posts the coordinates on Twitter using the Tweepy module which can thus ensure that a pileup does not happen and citizens can help. The tweet also triggers an IFTTT Applet on the phone of people close to the user.

Figure-3: *Screenshot of the triggered Applet*

3. CONCLUSIONS

As this program mainly uses the OBD-II port for collecting data, this can be installed in any car manufactured post-2010. Also, due to the modular nature of Python, more components can be added as Over The Air (OTA) Updates and/or by using Universal Serial Bus (USB) based modules, if additional hardware is required.

This program does not use any proprietary hardware and the code has been open sourced and is available on the GitHub repository navanchauhan/AutoSafe under the GNU GPLv3 License.

In the future, the code can be converted into a library so that it can be directly imported into programs

ACKNOWLEDGEMENT

The author would like to thank the entire open-source community for creating libraries and modules and private companies which create APIs to access their data.

REFERENCES

1. Tefft, B.C. (2016). Acute Sleep Deprivation and Risk of Motor Vehicle Crash Involvement. *AAA Foundation for Traffic Safety.*R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
2. GOVERNMENT OF INDIA, MINISTRY OF ROAD TRANSPORT & HIGHWAYS TRANSPORT, RESEARCH WING, NEW DELHI "ROAD ACCIDENTS IN INDIA – 2016"

3. Akalya Chellappa, Mandi Sushmanth Reddy, R.Ezhilarasie, S.Kanimozhi Suguna, A.Umamakeswari "Fatigue Detection Using Raspberry Pi 3" International Journal of Engineering & Technology, 7 (2.24) (2018) 29-32.
4. G. J. AL-Anizy, M. J. Nordin, and M. M. Razooq, "Automatic Driver Drowsiness Detection Using Haar Algorithm and Support Vector Machine Techniques", Asian Journal of Applied , vol. 8, no. 2. pp. 149–157, 2015.
5. Tereza Soukupova´and Jan C˘ech "Real-Time Eye Blink Detection using Facial Landmarks" (2016)