Drowsiness and Yawn Detection

Krishno Gopal Das[1], Ankita Dutta[1], Bhagyadhar Jana[1], and Sourav Santra[1]

[1]NSHM Knowledge Campus, Durgapur

Abstract: The main objective of yawn detection is diver drowsiness. In today's date there are several cases of drivers diminished vigilance level. Statistics tell us that about 20% of road accidents happen because of drivers with a less vigilance level. Less vigilance in motorists can be observed easily by monitoring his/her body responses. These methods can be used in two ways: 1. Measuring some physiological signals like eye blinking, heart rate, brain waves. 2.Measuring his physical state like head pose and state of mouth and eyes. Motorist's behaviour can be detected by monitoring the –steering wheel movement, acceleration and brake patterns.

Keywords: OpenCV, Dlib, NumPy, SciPy

1. Introduction

A yawn is an involuntary reflex where the lungs inhale a lot of air in and the mouth opens widely. It is a quite similar process for everyone as there is not such emotion related to it. Yawn detection is detecting whether a person is yawning or not using openCV. Yawning is also a sign of tiredness, as it occurs only when we go to sleep or we wake up. It also happens when someone does a monotonous job for a long time.

The main reason behind yawning is the brain temperature regulation. It is observed that in winter yawning occurs less. Drowsiness of a person can be detected by observing their facial behaviour. The detection relies on the mouth's position, and the images can undergo processing using the cascade classifier developed by Viola-Jones for facial recognition. Yawning can be detected by comparing the images with a set of images data for mouth and yawning. There is also an obstacle if the person places his hand while yawning then it can't be known for how much time the mouth was open. To resolve this problem, the eyes can be used as an observation. The duration for which the eyes remain open will decide whether the person is yawning or not. Longer the eyes closed, drowsier the person.

2. Design and implementation

To put it differently, the assist system collects information from the driver's behaviour and compares it with their steering movements. Bosch's driver drowsiness detection system uses data from sensors that track driving speed, lane assist camera, and turn signal usage to make decisions.

The algorithm in this system focuses on analysing visual features like the eyes and mouth to identify signs of drowsiness, instead of monitoring physiological reactions. This approach is chosen for its practicality in creating a non-intrusive system. Our algorithm includes yawn detection as a preventative measure to alert the driver before drowsiness or sleep occurs. The system will raise two types of alarms - one for sleeping and another for extreme drowsiness. Sleeping will be detected by moments of shut eyes, while prolonged states of drowsiness will trigger the other alarm.

2.1. OpenCV:

OpenCV, short for Open Source Computer Vision Library, is a readily accessible software library dedicated to computer vision and artificial intelligence. Its primary objective is to offer a standardized platform for computer vision applications while expediting the integration of machine intelligence into commercial products. With its BSD licensing, companies can conveniently employ and adapt the code to suit their needs. This extensive library encompasses more than 2500 advanced algorithms catering to a wide range of both traditional and contemporary computer vision tasks.

2.2. Dlib:

Dlib stands as a versatile C++ toolkit, encompassing machine learning algorithms and utilities for addressing practical challenges. Its application extends across a spectrum of industries and academic

domains, spanning robotics, embedded systems, mobile devices, and high-performance computing. Thanks to its open-source licensing, Dlib can be freely utilized in any application.

2.3. NumPy:

NumPy serves as a vital Python library, enhancing the language's capabilities for managing extensive, multi-dimensional arrays and matrices. It also provides an extensive suite of high-level mathematical functions for manipulating these arrays. NumPy's precursor, Numeric, was initially created by Jim Hugunin and subsequently improved by various developers. In 2005, Travis Oliphant integrated features from another library called Numarray into Numeric, resulting in the creation of NumPy with significant enhancements. NumPy operates as an open- source software driven by the community and enjoys contributions from multiple individuals. It is also a sponsored paper by NumFOCUS. [5] NumPy is specifically tailored for the CPython implementation of Python, which is a bytecode interpreter not optimized for performance. Consequently, mathematical algorithms written in this version of Python may run slower compared to compiled counterparts. To address this issue, NumPy provides efficient multidimensional arrays and

operators capable of executing operations on these arrays. However, optimizing code, especially inner loops, may be necessary to fully exploit its capabilities. By incorporating NumPy into Python, users can attain similar performance to MATLAB, as both are interpreted languages. When most operations involve arrays or matrices rather than individual values, programmers can create speedy programs using NumPy. While MATLAB offers various additional toolboxes, including Simulink, NumPy seamlessly integrates with Python, a more extensive and contemporary programming language. Moreover, Python offers other valuable packages such as SciPy, which provides MATLAB- like functionalities, and Matplotlib, a plotting package with similar features. Both MATLAB and NumPy leverage BLAS and LAPACK for efficient linear algebra computations.

2.4. SciPy:

SciPy [3] stands as a freely accessible open-source Python library extensively employed for scientific and technical computing tasks. It encompasses a diverse array of modules catering to functions such as optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other fundamental functions commonly encountered in the domains of science and engineering. Furthermore, SciPy hosts a series of conferences for both users and developers of these tools, which include SciPy in the United States, EuroSciPy in Europe, and SciPy.in in India. The United States SciPy conference was initially launched by Enthought and continues to be a prominent sponsor of

international conferences, while also maintaining the SciPy website [5]. The SciPy library is presently distributed under the BSD licence and benefits from the strong backing of a collaborative community of developers. This support is further augmented by NumFOCUS, a community foundation dedicated to advancing accessible and reproducible scientific research.

For this paper, we will make use of OpenCV to grab images from a webcam. Here's the methodology we intend to follow:

Step 1: Capture an image from the camera input.

Step 2: Detect the face in the captured image and establish a region of interest (ROI).

Step 3: Inside the ROI, detect and extract the eyes, which will be subsequently fed into the classifier.

Step 4: The classifier will ascertain whether the eyes are open or closed. Step 5: Assess whether the person is exhibiting signs of drowsiness.

2.5. Proposed Algorithm:

The algorithm under consideration operates through seven stages as illustrated in the figure. For testing this technique, the video feed is sourced from the webcam of a laptop.

2.6. Pre-processing:

The camera is focused on the driver's face to capture the video, which is recorded at a rate of 25 frames per second. Afterward, these frames are horizontally mirrored and converted to grayscale.

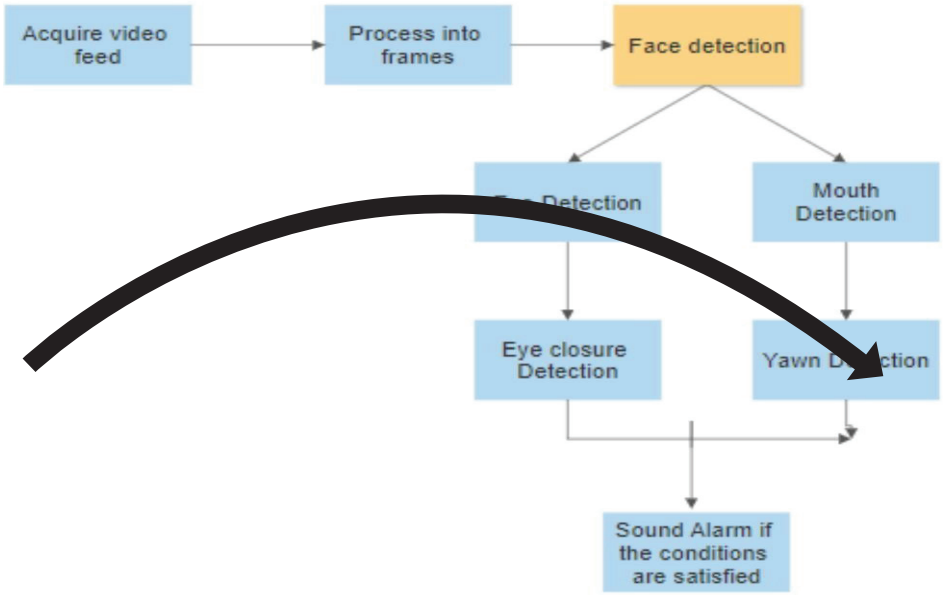


Figure 1: Pre-processing

2.7. Face detection

The Haar-based classifier integrates a diverse range of features, encompassing dimensions, weights, facial characteristics, and face color thresholds. It is constructed using a varied set of both positive and negative samples. From the positive instances, specific features are extracted. After Haar detection, edge detectors come into play, and the resulting output is stored in an array. The cascade includes both positive and negative samples. Features pertaining to the eyes and mouth are isolated, and parallel processing is employed upon successful detection of the driver's face. The focus is directed solely to the upper facial region, where the eyes are typically located, positioned just below the top edge of the face. Edge detection is applied to the region identified in the preceding step. Among the detected objects, the ones with the largest and second- largest surface areas are chosen, provided they are separated by at least twenty pixels and distinguishable from each other. These selected objects correspond to the driver's eyes.

Facial mapping using Dlib The algorithm is realised through the utilisation of the dlib Python library, which includes a pre-trained model for facial landmark detection. This model is capable of estimating and mapping the facial features of an individual, representing them as 68 Cartesian coordinates, as depicted in Figure 2. These 68-point landmarks were obtained through the training of the dlib facial landmark predictor, using the 300 dataset, which serves as the reference for these specific facial markings.

Figure-2: Facial mapping using Dlib

Eye closure detection The Eye Aspect Ratio (EAR) serves as a measure employed to evaluate the state of an eye, aiding in the determination of whether it is open or closed. This metric is derived from the 68 facial landmarks produced by the Python dlib library and is dependent on these landmarks within the specified region of interest for its calculation.

= 2-6 + 3-4 2 1-5 ……(1)

To calculate the EAR value, Equation (1) is employed, with l1, l2, l3, l4, l5, and l6 representing the landmark points that define the eye's boundaries within the marked region. A blink is identified when the computed value falls below 0.3. In this algorithm, a threshold value of 0.3 is utilized. If this condition persists for a consecutive span of 50 frames, it signifies eye closure.

Yawn detection: - Yawning is defined by the action of opening one's mouth widely. Similar to the approach used to detect eye closure, facial landmarks are employed to identify an open mouth. The process for detecting yawning events, depicted in Figure 3's flowchart, commences with the detection of a face. In this process, the parameter for evaluating whether the person's mouth is open is the lip distance. If the lip distance calculated from the frame surpasses the lip distance threshold, the individual is recognized as yawning. An alarm is activated if the person yawns consecutively more than the predefined threshold value, which is set at 10. Minor mouth openings, such as those that occur during conversation or eating, are not considered in this context.

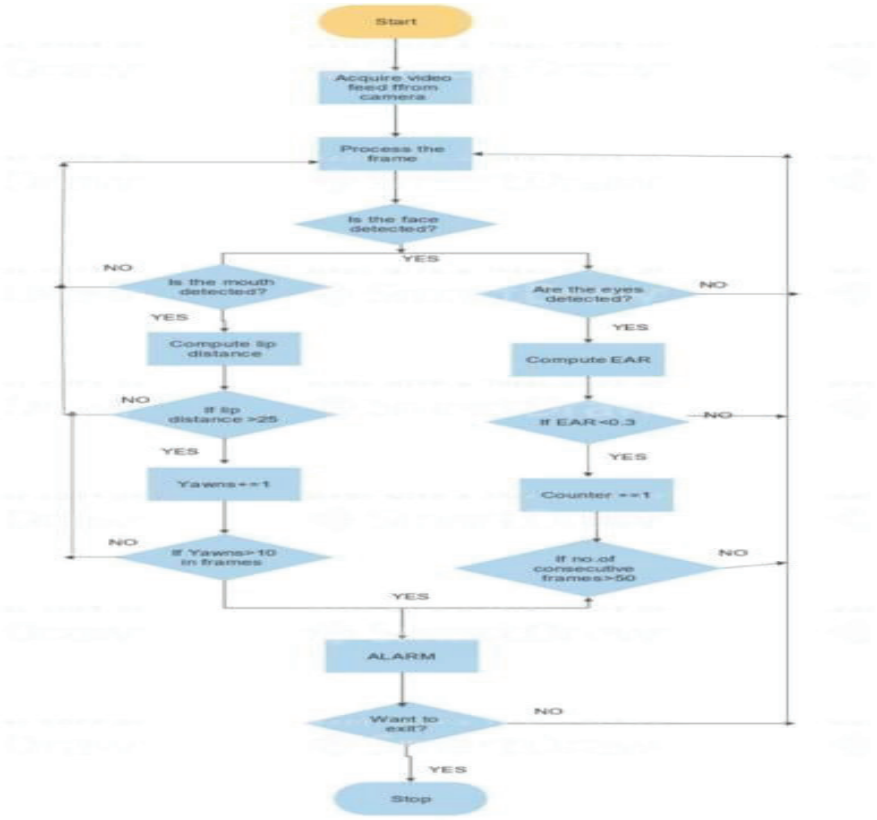


Figure-3: Yawn detection

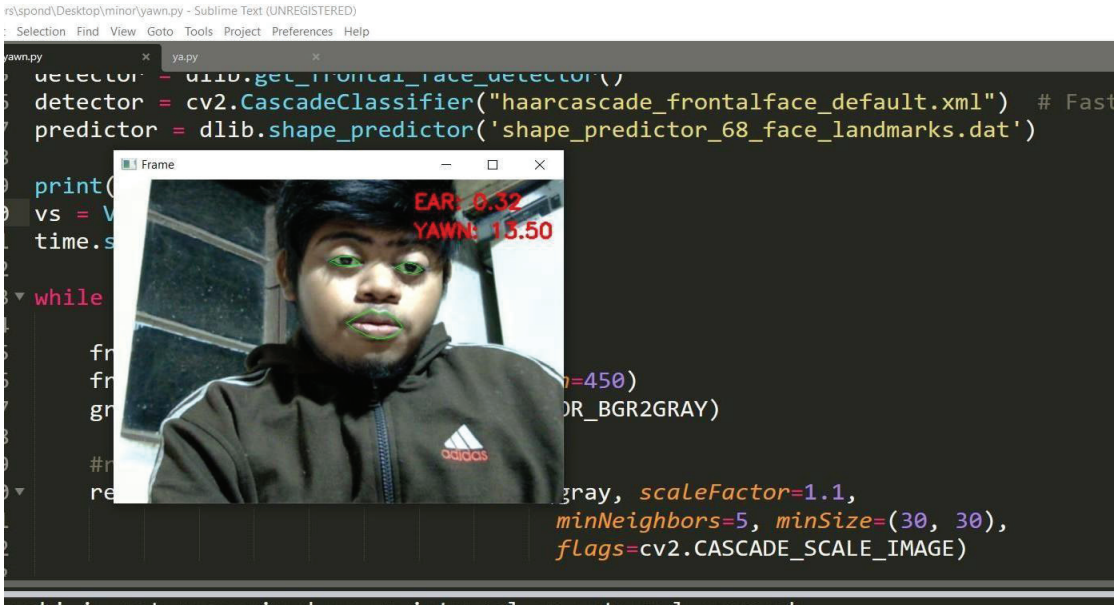
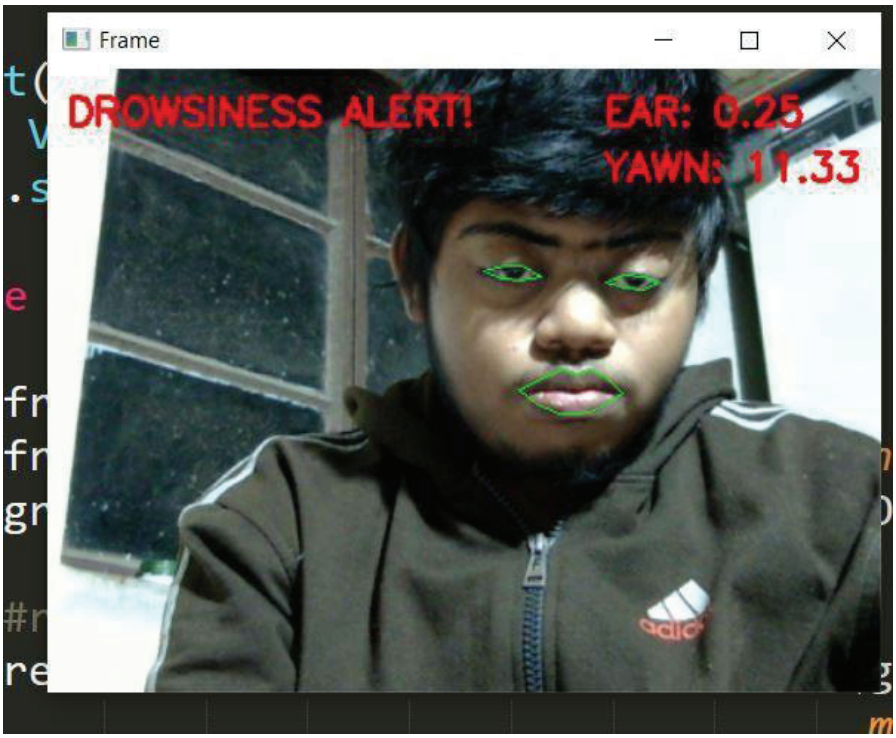


Figure-4: Output frame when the subject is normal

Figure-5: Output frame when subject is yawning



3. Conclusion and Future Scope

The model has the capability to detect drowsiness by monitoring both the eyes and mouth, using shape prediction techniques to identify critical facial features. These methods rely on facial landmarks obtained through facial landmark detection. Specifically, the module involves the Eye Aspect Ratio (EAR) function, which calculates the ratio of distances between horizontal and vertical eye landmarks. Additionally, an eSpeak module, a text-to-speech synthesiser, is integrated to provide appropriate voice alerts when the driver shows signs of drowsiness or yawning. The overarching objective of this paper is to reduce the occurrence of accidents and contribute to technology aimed at preventing fatalities resulting from road accidents. In future research, there is potential to explore external factors for assessing fatigue and drowsiness. These factors could encompass weather conditions, vehicle status, sleep patterns, and mechanical data. Driver drowsiness poses a significant threat to road safety, especially for commercial vehicle operators, given factors like continuous service, unpredictable environmental conditions, high annual mileage, and demanding work schedules. A vital preventive measure is to continually monitor the driver's state of drowsiness and provide them with relevant information so that they can take necessary actions. Currently, no adjustments can be made regarding the camera's zoom or direction during system operation. However, in the future, further work can focus on automating the zoom on the eyes after their localization, enhancing the system's capabilities.

REFERENCES

1. G. Zhang, K. K. Yau, X. Zhang, and Y. Li, "Traffic accidents involving fatigue driving and their extent of casualties," Accident Analysis & Prevention, vol. 87, pp. 34–42, 2016.

2. Association for Safe International Road Travel (ASIRT). "RoadCrash Statistics." http://asirt.org/initiatives/informingroadusers/road-safetyfacts/road- crash-statistics, 2016.

3. International Classification of Sleep Disorders and Coding Manual, "Archived copy" (PDF). Archived from the original (PDF) on 2011-07-26. Retrieved 2011-07-26., page 343.

4. Paul, Amit, Linda Ng Boyle, Jon Tippin, Matthew Rizzo (2005). "Variability of driving performance during micro sleeps" (PDF). Proceedings of the Third International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design. Retrieved 2008-02-10.

5. Kaza Deepthi Sudha, Pooja Kilaru, Manna Sheela Rani Chetty, Currency Note Verification and Denomination Recognition on Indian Currency System, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7, Issue-6S4, April 2019.

6. Ramya keerthi P, Niharika B, Dinesh Kumar G, Sai Venakat K, Sheela Rani C M, Reorganization of License Plate Characteristics Using Image Processing Techniques, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7, Issue-6, March 2019.