IoT Enabled Driver Compatible Cost-Effective System for Drowsiness Detection with Optimized Response Time

Argha Sarkar[1], Mayuri Kundu[1], Prakash Pareek[2], Nishu Gupta[3(✉)], and Manuel J. Cabral S. Reis[4]

[1] School of Computer Science and Engineering, REVA University, Bangalore 560064, Karnataka, India

[2] Department of Electronics and Communication Engineering, Vishnu Institute of Technology, Bhimavaram 534202, Andhra Pradesh, India

[3] Department of Electronic Systems, Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, 2815 Gjøvik, Norway
nishu.gupta@ntnu.no

[4] Engineering Department, UTAD/IEETA, 5001-801 Vila Real, Portugal

Abstract. The present work researches driver drowsiness, which constitutes a huge problem, and can turn into fatal incidents potentially involving the losing of lives, being it while driving in a highway (car, bus, truck, etc.), in the railway, or any other transportation mean (ship, airplane, etc.). Recent technology has been involved in the study of the fatigue behaviour. The purpose of this work is to identify the in-driver's drowsiness, contributing to get rid of the accidents, mainly during the night, and to ensure better safety in train and on the highways. A camera is used to capture images from the driver, and face detection is executed in real-time to find out the exact position of driver's face. Here, the blinking of the driver's eye is under consideration. The fundamental concept lies in the closing of the driver's eye for a specific duration of time. If the closing time exceeds that certain duration, drowsiness is detected and an alarm is sounded for awareness. Python and OpenCV (using Haarcascade library) are the fundamental programming platforms for sensing the facial attributes. In this work, a cost-effective system for the driver's drowsiness detection is aimed. Further, the result also indicates the reduction of response and processing time, which is ideal for real time detection.

Keywords: IoT · Drowsiness · Detection · OpenCV

1 Introduction

The importance of vehicle safety has increased with the advancement of technology in all spheres of life, and the main focus is now on reducing the alarming time when an accident occurs so that the rescue team can treat the injured faster. When we try to go deep into the root causes of the road fatalities, drowsiness of the driver emerges as one of

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2023
Published by Springer Nature Switzerland AG 2023. All Rights Reserved
N. Gupta et al. (Eds.): IC4S 2022, LNICST 472, pp. 159–169, 2023.
https://doi.org/10.1007/978-3-031-28975-0_13

the prime causes. This drowsiness is caused by the intake of unwanted drugs or mental fatigue of driver. The number of accidents is expected to decrease drastically if the eye movement of the driver can be monitored. It involves the recognition of human face efficiently in a timely manner which is a complex problem.

The human face is lively and active with a high grade of changeability. In computer vision, it is a very difficult problem to perform face recognition in real time dynamic environments. Human eyes play a major role during face recognition as well as facial expression analysis. Salient features dynamically change while a person feels drowsiness. Eyes are the most relevant and fairly stable landmark on the face in contrast with other facial attributes. Therefore, it is more significant to detect eyes before recognizing other facial attributes. The eye position can estimate the location of other facial elements/features. Besides, the shape, size, position and the rotation of the image-plane of the face can be normalized by the location of both eyes [1–5].

The basic contributions of the present work are directed to:

- Be able to accurately detect a face from an image.
- Be able to detect the region of interest, in this case the eyes.
- Accurately classify the status of the eye either closed/ open.
- Provide a warning to the driver if drowsiness is detected.
- Provide cheaper systems that are affordable to the low-income people.

The rest of the paper is organized as follows. Section 2 provides stepwise methodology adopted in this work. It is followed by a brief explanation of implementation which is depicted in the section. Finally, the attained results are analysed and shown in Sect. 4. The conclusion of this work is provided in last section.

2 Methodology

As stated before, the main objective of the present work is to detect the drowsiness of a driver (driving a car, train, etc.). It can be estimated by the duration of the eye closing time [5, 6]. If the eyes are in a closed state for a certain period of time, an awareness notification sound will be ringed. In this work, faster response and detection, irrespective of specific drivers, are the fundamental objectives. The number of frames associated with eye closure are monitored. If the number of frames reach a specific value, then an awareness message is displayed on the screen showing the notification “driver is feeling drowsy”.

The following algorithm is implemented. Initially, the image is captured by a webcam and sent for further processing. The Haarcascade file face.xml is used to find and sense the faces in an individual frame. If the situation arises that no face is detected, then the next frame is acquired. On other hand, if a face is recognized, then a region of interest (RoI) is declared within the face. This RoI contains both eyes. The purpose of declaring a RoI is to reduce the computational effort of the system. Finally, by using Haarcascade_eye.xml, the eyes are detected from the RoI. Figure 1 depicts the overall representation of working of the proposed system.

In the next subsections a brief explanation of the main steps of the algorithm, presented in Fig. 1, is presented.

2.1 Image Acquisition

```
# Initiation of the video stream thread
print("[INFO] starting video stream thread...")
vs = VideoStream(src=0).start()
# vs = VideoStream(usePiCamera=True).start()
time.sleep(1.0)
```

The function "vs = VideoStream(src=0).start()" grabs the video streaming from a USB camera. There is a second sleep to warm up camera sensor.

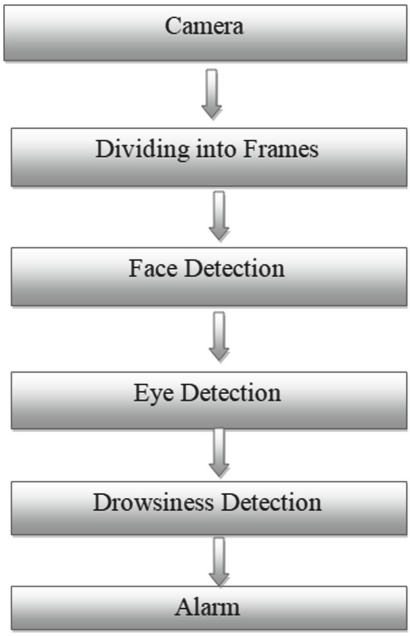


Fig. 1. Block diagram of the implemented algorithm

2.2 Dividing into Frames

Step 1. while condition is True:
frames from video stream are collected
Step 2. convert the collected frames into grayscale channels after resizing
Step 3: face detection takes place in the converted frames

In practice, in a real time scenario, the system deals with captured video which is in turn processed further. But here, the mentioned algorithm is compatible for image only. Therefore, the captured video is segmented into frames for the corresponding analysis.

2.3 Face Detection

In this specific stage, the aim is to build a specific logic for the recognition of the face. By face recognition, it is meant that marking the face in a frame, i.e., finding the position of facial attributes. The frame is supposed to be random in nature. The detection is very selective to the facial related features and does not include other structures or objects in the surroundings. A face can be considered as an object. In such cases, the location of the target object is intended to find out and the size is also approximated to known particular class. The face detection algorithms are mostly based on recognizing the front side of the face. Herein, a situation like a face in the tilted position, or positioned in any other way, and also multiple faces are also triggered. It implies that it may solve the purpose of the rotation axis with respect to the present observer from the reference of the face in a particular position, or even if there is a vertical rotation plane. In the present algorithm, all the real time situations are considered [6–9].

Step 1: Repeat for (x,y,w,h) in rects:
face in detection in converted frames.
Step 2: a rectangle object is constructed from the bounding box of Haarcascade
Step 3: facial landmark is detected in the facial region with (x,y) co-ordinate
Step 4: facial landmark (x,y) co-ordinates is converted to array

The above function basically draws a rectangle in the image to point out the corresponding corner points. The rest of the parameters are to illustrate the colour and thickness and type of lines in the rectangle and visualize the coordinates.

The Facial Landmarks Predictor

Step 1. Initialize dlib's face detector
Step 2. facial landmark predictor is created with a message "("[INFO] loading facial landmark predictor…")"
Step 3. Indexes for left and right eye is grabbed from facial landmarks

2.4 Computing the Euclidean Distances and Expression for Ear He Facial Landmarks Predictor

Step 1. Two sets of Euclidean distance between left and right eye is computed from vertical eye landmarks (x,y) – co-ordinates and store it to A and B

Step 2. Euclidean distance between left and right eye is computed from horizontal eye landmarks (x,y) – co-ordinates and store it to C

Step 3. Eye aspect ratio is calculated using A ,B and C value and store it to ear(Eye Aspect Ratio)

```
ear= (A+B)/(2.0*C)
```

Step 4. if(ear>x):

```
        x=ear
else if (ear<y):
        y=ear
return ear
```

2.5 Set Unique Threshold Value

EYE_AR_THRESH = x + ((y − x)/5)
EYE_AR_CONSEC_FRAMES = 48

2.6 Set Image Region of Interest

It extracts the left and right eye coordinates and compute the EAR (Eye Aspect Ratio), which is the average of Eye Aspect Ratios of left and right eyes.

Step 1. Extract left eye co ordinates
Step 2. Extract right eye co ordinates
Step 3. Left Eye Aspect Ratio (leftEAR) is computed from Left eye co-ordinates
Step 4. Right Eye Aspect Ratio (rightEAR) is computed from Right eye co-ordinates
Step 5. Calculate the average eye aspect ratio (ear)

ear = (leftEA + rightEAR)/2.0

2.7 Visualization of Eye Regions on a Frame et Image Region of Interest

Step 1. The convex hull is computed for each eye
Step 2. Visualize each of the eyes to plot contours for both the eyes.

2.8 Drowsiness Alert Based on Eye Aspect Ratio

Here, the ear is compared with the EYE_AR_THRESH – if it goes below the threshold, eyes are closed. COUNTER is incremented and subsequently verified to know the eye closure time for enough consecutive frames, to blow the awareness alarm. If the alarm is in off state, it will be turned on to notify the drowsy driver.

```
 if eye aspect ratio (ear)<blink threshold
                        then increment the blink frame counter by 1.
                        if eyes are closed in maximum number of frames
                                then alarm= ON
if args["alarm"] > 0:
            th.buzzer.blink(0.1, 0.1, 10,background=True)
draw an alarm on the frame

else:
                        reset the counter as 0 and alarm
```

The dip in eye aspect ratio indicates driver's drowsiness.

3 Implementation

As stated above, to implement the principles and algorithm presented in the previous section, we have used Python and OpenCV (using Haarcascade library).

Here, the purpose of using OpenCV is mainly to detect the image. OpenCV [OpenCV] is an open-source computer vision library which is shown in schematic form in Fig. 2. Additionally, OpenCV is chosen basically because of its computational efficiency in real-time image recognition, and "Simple-to-use computer infra" is one key feature of OpenCV which results in the design of highly sophisticated and fast response vision applications.

In a more practical implementation perspective, the reasons behind choosing OpenCV are:

- Specificity: In OpenCV, all the functions and data structures are designed aiming to the image detection.
- Efficiency: One of the alternatives is MATLAB, but consumes huge system resources, whereas, OpenCV, requires only 10MBytes RAM for a real-time application.
- Speediness: By using MATLAB, a maximum of 4 to 5 frames can be processed in every second. In contrast, OpenCV can process around 30 frames per second. It makes OpenCV more applicable to real time situations.

Visualization of facial landmark coordinates is crucial here, and thus eye aspect ratio (EAR) is a priority for drowsiness detection to obtain the desired accuracy. The critical parameters of eye status are classified based on the EAR. The level of opening and its landmark positions are extracted from an image: (a) extracting exact landmark position of eyes (b) calculating distance between vertical horizontal coordinates.

From the given Fig. 3, A is the distance between 1 & 5. B is the distance between 2 & 4 and C is the distance between 0 & 3. Relating A, B, and C distances by an appropriate formula. The above process of calculation is performed for both eyes after getting the image from video or target face. Then the eyes are detected and the corresponding landmark of both eyes.

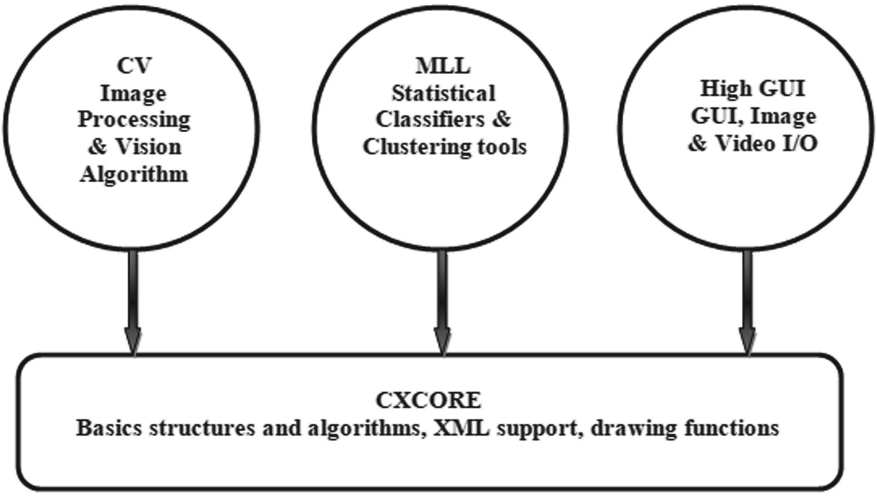


Fig. 2. Schematic representation of OpenCV structure

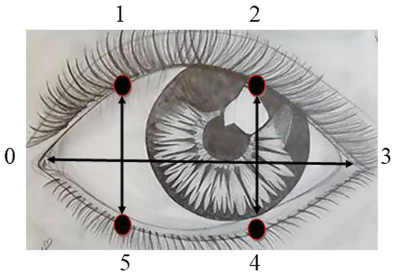


Fig. 3. Extraction and calculation of the exact landmark position of eyes

If this EAR value goes low compared to a unique threshold value, which automatically gets adjusted for every individual, an alarm is triggered. The novelty of the work lies on the automatic adjustment of the threshold value for every individual. One of the most important difficulties that we have had to ensure the effective drowsiness detection involving eye aspect ratio was the fixed threshold value, implying that the system was static, i.e., treating everybody equally. However, every individual can have unique threshold values based on their eye-opening area, and hence we have adopted a dynamic behaviour of our system, adapting the threshold value. This is one of the most important advancements in this work. Irrespective of a specific driver, the system calculates the threshold value for each and every driver dynamically.

The software-hardware interactions involved for each stage of the methodology are as follows:

Software components:

- Raspbian Stretch
- Python3
- Cmake Dlib
- OpenCV

Software implementation:

- Eye detection
- Detecting the eye center
- Determining pixel intensity of center
- Eye state determination
- Driver drowsiness detection with OpenCV
- Programming performed in python language and OpenCV using the Haarcascade library
- Drowsiness analysis

Hardware components:

- Raspberry Pi 3
- Logitech c270
- Buzzer

Hardware implementation:

- Implementation using LAPTOP and WEBCAM.
- Implementation using Raspberry Pi and Webcam.

Various steps and its related integration to the final drowsiness detection method and device are explained below:

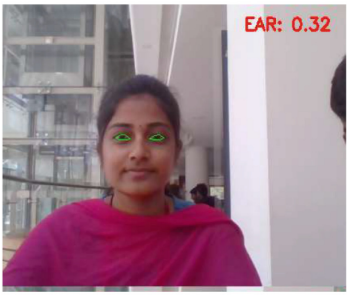
Step 1: Camera grabbing: video streaming
Step 2: Dividing into frames: grabbing of the frame resized and convert ed it to grayscale.
Step 3: Face detection: sent for face detection and a pause is initiated for a second and then loop over frame in video stream.
Step 4: Facial detection: detection of facial landmarks, conversion of facial landmark (x, y) coordinates to the numpy array.
Step 5: Eye Detection: extraction of left and right eye coordinates, and estimation of threshold value.
Step 6: Eye aspect ratio (EAR) calculation: The coordinates are used to compute eye aspect ratio.
Step 7: Comparison: EAR with unique threshold value.
Step 8: Triggering of Alarm: if EAR $<$ threshold value (signal is sent from raspberry pi to speaker)

4 Result and Analysis

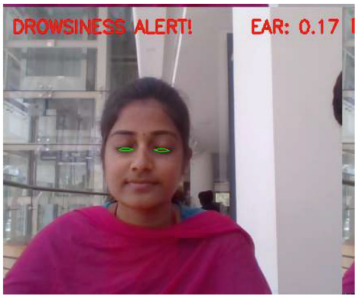
To confirm the accuracy of the system to determine the eye blinks and drowsiness, a huge set of samples are considered from live video.

In this work, a USB webcam of 5 megapixels is connected to Raspberry Pi 3 model B+. A piezoelectric buzzer is incorporated in the system to produce the notification sound to make the driver conscious. It happens only when the duration of the closing of eyes reaches a certain threshold. The whole setup is tested in different real time situations for different drivers. High accuracy is achieved for the driver who is wearing spectacles also. The different samples with different conditions are shown in Fig. 4.

This system is appropriate to use in vehicles like bus, taxi, train, etc., to avoid accidents due to the driver's drowsiness. Sample outputs are shown below where such conditions are considered.

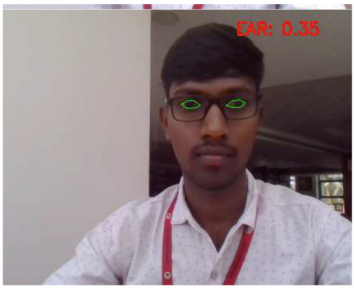


(a): Sample output (eyes open)



(b): Sample output (eyes closed)

(c): Sample output (eyes closed)



(d): Sample output (eyes open)

(e): Sample output (eyes closed)

(f): Sample output (eyes open)

(g): Drowsiness detection-Device prototype.

Fig. 4. (a): Sample output (eyes open). (b): Sample output (eyes closed). (c): Sample output (eyes closed). (d): Sample output (eyes open). (e): Sample output (eyes closed). (f): Sample output (eyes open). (g): Drowsiness detection – device prototype.

5 Conclusion

We have presented a simple methodology to detect drowsiness in a driver. We have also presented a system to implement this methodology. This system includes a process able to detect the region of interest, in this case the eyes aspect ratio. A unique technique is used to classify the state of the eye, either closed or open, based on the programming performed using the Python language and OpenCV using the Haarcascade library. The proposed driver drowsiness detection system automatically adapts to the different drivers' characteristics, and is not limited to a specific driver. The system also provides high accuracy for the driver who is wearing spectacles. The system has minimized response and processing time, which is highly desirable for real time detection, and it can be considered a low-cost of cost-effective system, affordable to the low-income people.

Acknowledgments. The author would like to acknowledge Sree Vidyanikethan Engineering College Tirupati, India for providing the infrastructural support to pursue the work.

References

1. Sanjay, S., et al.: Drowsiness detection with OpenCV. In: Proceedings of 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 1421–1425. IEEE, Coimbatore (2021)
2. Khunpisuth, O., et al.: Driver drowsiness detection using eye-closeness detection. In: Proceedings of 2016 12th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), pp. 661–668. IEEE, Naples (2016)
3. Thulasimani, L., Prithashasni, S.P.: Real time driver drowsiness detection using opencv and facial landmarks. Int. J. of Aquat. Sci. **12**(2), 4297–4314 (2021)
4. Chirra, V.R.R., Uyyala, S.R., Kolli, V.K.K.: Deep CNN: a machine learning approach for driver drowsiness detection based on eye state. Rev. d'Intelligence Artif. **33**(6), 461–466 (2019)
5. Satish, K., et al.: Driver drowsiness detection. In: Proceedings of 2020 International Conference on Communication and Signal Processing (ICCSP), pp. 380–384. IEEE, Chennai (2020)
6. Cech, J., Tereza, S.: Real-time eye blink detection using facial landmarks. In: Proceedings of 21st Computer Vision Winter Workshop. Rimske Toplice (2016)
7. Mohanty, A., Bilgaiyan, S.: Drowsiness detection system using KNN and OpenCV. In: Swain, D., Pattnaik, P.K., Athawale, T. (eds.) Machine Learning and Information Processing. AISC, vol. 1311, pp. 383–390. Springer, Singapore (2021). https://doi.org/10.1007/978-981-33-4859-2_38
8. Bergasa, L.M., et al.: Real-time system for monitoring driver vigilance. IEEE Trans. Intell. Transport. Syst. **7**(1), 63–77 (2006)
9. Cech, J., Franc, V., Matas, J.: A 3D approach to facial landmarks: Detection, refinement, and tracking. In: Proceedings of 2014 22nd International Conference on Pattern Recognition, pp. 2173–2178. IEEE, Stockholm (2014)