EasyChair Preprint

№ 10097

A Driver Drowsiness Detection Using Machine Learning and OpenCV

Nimesh Sinha, Jasmine Minj and Pooja Patre

May 12, 2023

A Driver Drowsiness Detection using Machine Learning and OpenCV

Nimesh Kumar Sinha[1],Jasmine Minj[2],Pooja Patre[3]
[1] Research Scholar of Computer Science and Engineering VEC Ambikapur
[2,3] Assistant Professor in Computer Science and Engineering VEC Ambikapur
Nimeshs437@gmail.com[1], Jasmine5feb@gmail.com[2], Poojapatre89@gmail.com[3]

Abstract

The rate at which cars, buses, and other motor vehicle-related accidents have increased in the past few years is becoming mind-bulging and fearful and the majority of these mishaps are drowsy drivers related. This has resulted in the loss of lives, goods, properties, etc. The essence of this analysis is to study and review the previous works on drowsy driving-related accidents, their causes, and measures taken. The gaps in these studies were noted in order to propose and design a new or robust system. This was achieved using various techniques such as image acquisition, computer vision, face detection, feature extraction, training, and classification. The techniques were designed using a universal modeling diagram and mathematical modeling approach based on the requirements for the object-oriented analysis design methodology adopted for the study. The designs were implemented as a prototype system using Python and tested with real-time driving behaviors.

Keywords: Computer Vision, Data Science, Machine Learning, Python, OpenCV, Drowsiness detection, EAR.

1 Introduction

Road accidents are the nation's number one killer and are becoming more common as the number of vehicles on the road rises daily. We all know that the driver is in charge of the traffic system and the safety of the road. In addition to the other passengers in the car, the driver is also accountable for himself. Many people frequently disregard drowsiness when it comes to their own safety. However, if this trait is not taken into account and acted upon, it may result in an accident and be the cause of death, which might be problematic for both the driver and the passengers. In our daily lives, drowsiness is one of the main causes of actual car accidents. To increase the

safety of road traffic, it is imperative to address the challenging issue of driver drowsiness. According to the National Highway Traffic Safety Administration, over 100,000 police-reported crashes involving drowsy driving result in close to 800 fatalities and 50,000 injuries per year. The actual amount, however, might be substantially higher because it can be challenging to tell whether a motorist was drowsy at the time of a collision. There are many scientists and researchers who have given frameworks to detect the drowsiness of the driver all over the world. This paper proposed to detect the drowsiness of the driver using machine learning and OpenCV.

The organization of the paper is as follows section 1 describes the related work of the paper. In section 2, we have discussed the background and related terminology. Section 3, address the proposed framework. Section 4, discussed drowsiness detection. In section 5 we have concluded the paper. Background

In this section, we discuss the background and related terminology or tools, and platforms used for the research work as follows.

1.1 Python

Python is a high-level programming language that is intended to be simple to learn and use. Because it is open source, anyone can use it, even for commercial purposes. Python has been converted to Java and .NET virtual machines and can operate on Mac, Windows, and Unix operating systems [1]. Python is regarded as a scripting language, similar to Ruby or Perl, and is frequently employed for developing dynamic web content and Web applications. Many 2D and 3D imaging products also support it, allowing users to use Python to build unique plug-ins and extensions. GIMP, Inkscape, Blender, and Autodesk Maya are a few examples of programs that offer a Python API.

1.2 OpenCV

Open Source Computer Vision is what OpenCV stands for. It is an Open Source BSD licensed library with hundreds of cutting-edge Computer Vision algorithms that are designed to take advantage of hardware acceleration. Machine learning, image processing, image manipulation, and many more applications use OpenCV frequently. OpenCV is organized in modules. Along with a CV Namespace, there are static and shared libraries. Simply put, OpenCV is utilized in our application to quickly load bitmap files containing images of landscape and to combine two images so that one can be viewed in the background of the other[2]. Using OpenCV instead of alternative approaches, this picture alteration can be completed quickly in a few lines of code.

1.3 Machine Learning

The type of programming known as "machine learning" enables computers to automatically learn from data without explicit programming. In other words, these systems adapt to new situations by learning from data. One of the greatest languages for machine learning is undoubtedly Python. Scipy, Pandas, and Numpy are three machine learning-specific libraries in Python that are excellent for learning linear algebra and the machine learning kernel methods. The language has relatively simple syntax and is excellent for usage when working with machine learning algorithms.

Machine learning is the ability of systems to learn from training data that is relevant to a given problem in order to automate the construction of analytical models and complete related activities [3]. A machine learning concept called "deep learning" is based on artificial neural networks. Deep learning models outperform shallow machine learning models and conventional data analysis techniques for many applications.

1.4 Dlib

DLib is a free and open source library that implements machine learning methods such as classification, regression, clustering, data transformation, and structured prediction. DLib, like DMTL, provides a generic high-performance machine learning toolbox with many different algorithms, however DLib has been updated more frequently and has more examples. DLib also has a lot more helper capabilities. Dlib and OpenCV are used for face recognition[4].

2 Related Work

J. Gwak et al. [5] have attempted to distinguish alert and slightly drowsy states with machine learning algorithms based on hybrid measurements of driving ability, behavioral traits, and physiological markers, with the goal of early identification of driver drowsiness. In this study, when using hybrid measurements and eliminating physiological indicators, 78.7% accuracy was obtained when identifying alert vs. mildly drowsy states.

R. M Sanchez et al. [4] have successfully developed a program for facial recognition for only nose and eyes area with 92% accuracy. The authors have achieved this with Dlib and OpenCV using SVM.

A. Zaki et al. [6] have proposed a framework for fatigue while performing repetitive tasks on a production line is the fundamental cause of many mishaps. To address this issue, a study was done to build a fatigue detecting algorithm. Authors have used machine learning techniques such as Euclidian distance to identify the fatigue. However, they have not mentioned the accuracy.

A. Altameem et al.[7] have used machine learning approaches to implement real-time image segmentation and sleepiness in this work. An emotion recognition system based on Support Vector Machines (SVM) has been constructed using facial expressions in the proposed work.

Drunkenness or tiredness is a major cause of car accidents, with serious consequences for road safety. More fatalities could be avoided if weary drivers were alerted in advance. Several drowsiness detection technologies can be used to monitor for indicators of inattention while driving and alert the driver.

F.You et al. [8] provide a real-time driving drowsiness detection system that takes into account the driver's individual variations. To detect the face region, a deep cascaded convolutional neural network was built, which addresses the problem of poor accuracy caused by artificial feature extraction. The frontal driver facial landmarks in a frame are discovered using the Dlib toolbox. A new parameter called the Eyes Aspect Ratio is created based on the eyes landmarks to measure the tiredness of the driver in the current frame. The authors have achieved 94.80 % accuracy in this research work.

Archita Mohanty et al.[9] constructed and developed a model for drowsiness or fatigue detection system using the OpenCV library and the KNN technique of Machine Learning and Shape predictor face landmarks in this suggested study. The devised technique was effectively tested, and dramatic changes in the EAR were also observed while opening and closing the eyes. The authors have shown different types of drowsiness alerts as a result.

The summary table for similar works may be seen in Table 1 based on the literature review.

Table 1. Summary of the literature survey.

Authors	Techniques used	Result
J. Gwak et al. [5]	Machine Learning algorithms	78.28% accuracy
R. M Sanchez et al. [4]	SVM using Dlib and OpenCV	92% accuracy
A. Zaki et al. [6]	Euclidian distance using machine learning.	Accuracy is not addressed
A. Altameem et al.[7]	Support Vector Machines (SVM)	Accuracy is not addressed
F.You et al. [8]	Dlib toolbox	94.80% accuracy
Archita Mohanty et al.[9]	KNN and OpenCV	Identified different Alerts

3 Proposed framework

The process of the drowsiness detection system involves following steps. The steps are shown in the Figure 1.

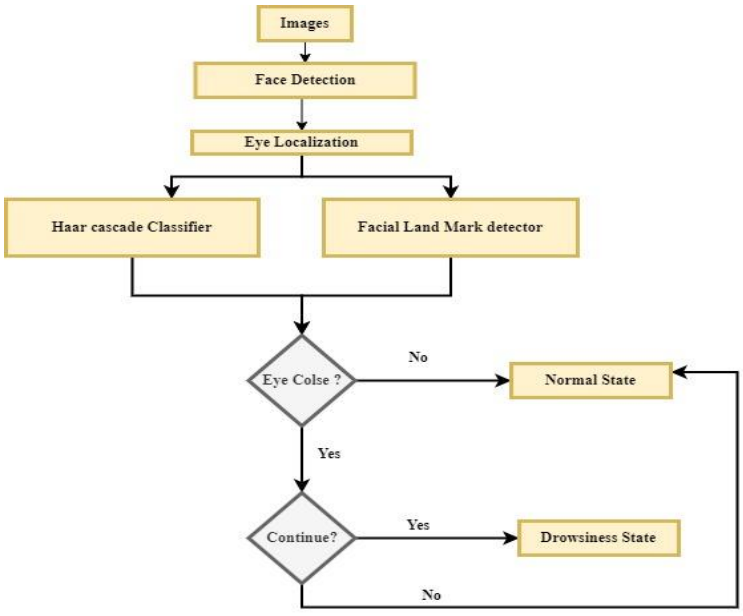

Fig.1 Steps involve in drowsiness detection

1.1 Face detection

Haar feature-based cascade classifiers are used for face detection. This is a successful object detection method presented by Paul Viola and Michael Jones in their paper "Rapid Object Detection Using a Boosted Cascade of Simple Features" in 2001. It is a machine learning approach in which a cascade function is taught using a large number of positive and negative images. It is then used to other photos to detect things[10]. Face detection using Haar cascading is machine learning approach training is provided for cascading functions form various positive and negative images.

To train the classifier, the method requires a large number of positive images (images of faces) and negative images (images without faces). Then we must extract characteristics from it.

Haar features such as those displayed in the image below are employed for this. They are identical to our convolutional kernel. Each feature is a single value calculated by subtracting the sum of the pixels under the white rectangle from the sum of the pixels under the black rectangle. Fig. 2 shown the examples of the haar cascading.

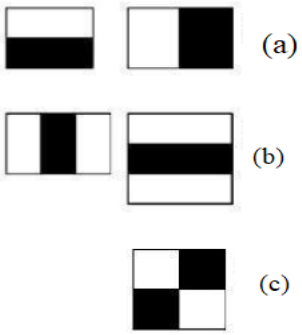

Fig.2 Examples of haar cascading (a) Edge features (b) Line Features (c) Four-rectangle features.

Fig.3 shows that five haar like features.

Fig 3. Five haar features

1.2 Face detection

In the system we have used facial landmark prediction for eye detection Facial landmarks are used to localize and represent salient regions of the face, such as: Eyes, Eyebrows, Nose, Mouth, and Jawline etc.

Facial landmarks have been used successfully in face alignment, head pose estimation, face swapping, blink detection, and many more applications. In the context of facial landmarks, we want to use shape prediction methods to discover essential facial structures on the face. Localize the face in the image and Detect the key facial structures on the face ROI are the two steps to detect the facial landmarks.

Localize the face in the image: As mentioned in the first step of our approach, Haar feature-based cascade classifiers are used to localize the face image**.**

Detect the key facial structures on the face ROI : There are various facial landmark detectors, but all methods attempt to localize and classify the following facial regions: Mouth, left eyebrow, right eyebrow, nose, left eye, and right eye. The dlib library's facial landmark detector is an implementation of the One Millisecond Face Alignment with an Ensemble of Regression Trees [11].

This step incudes:

1. On an image, a training set of labelled face landmarks. These images are labelled by manually, with particular (x, y)-coordinates of regions surrounding each facial structure specified.
2. Priors, or more precisely, the likelihood of distance between pairs of input pixels.

The 68 (x, y)-coordinates that correspond to facial structures on the face are estimated using the pre-trained facial landmark detector found inside the dlib library. The indexes of the 68 coordinates may be depicted on the image as shown in the Fig.4. The following facial landmark index can identify and access both the eye region; [36, 42] for the right eye and [42, 48] for the left eye.These annotations are from the 68-point iBUG 300-W dataset, which was used to train the dlib face landmark predictor. It's worth noting that there are various types of face landmark detectors, such as the 194-point model that can be trained on the HELEN dataset. The same dlib framework can be used to train a shape predictor on the input training data regardless of which dataset is used [12].

1.3 Eye state recognition

The optical flow, sparse tracking, and frame-to-frame intensity differencing and adaptive thresholding can all be used to estimate the eye area. Finally, whether or not the eyes are covered by eyelids is decided. Some other method is to infer the state of the eye opening from a single image, such as by correlation matching with open and closed eye templates, a heuristic horizontal or vertical image intensity projection across the eye region, a parametric model fitting to detect the eyelids, or active shape models.

One significant disadvantage of earlier techniques is that they frequently implicitly place excessively stringent criteria on the setup, such as a relative face-camera posture (head orientation), picture resolution, illumination, motion dynamics, and so on. Despite their real-time efficiency, heuristic approaches that use raw image intensity are

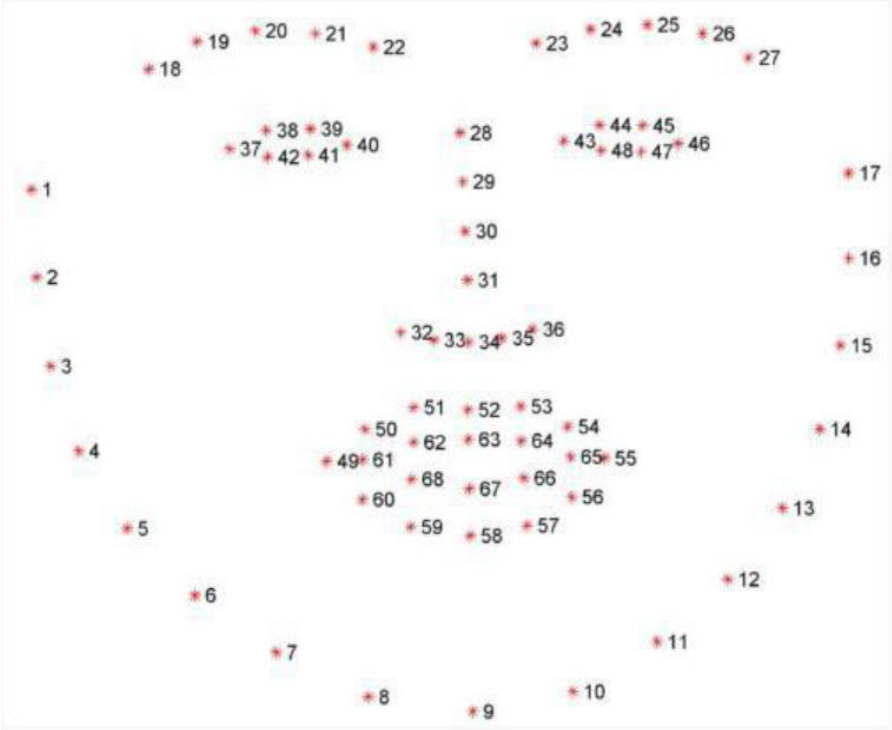

Fig.4 Generating a visual representation of the 68 facial landmark coordinates[12]

likely to be quite sensitive. Finally, using a per-frame sequence of the eye-opening estimations, an SVM classifier trained on instances of blinking and non-blinking patterns finds the eye blinks.

Eye Aspect Ratio Calculation

Blink detection is useful in a variety of situations, including face movement analysis and signal processing [13]. The eye landmarks are recognized for each video frame. The eye aspect ratio (EAR) is computed as the ratio of the eye's height to width. EAR can be computed as shown in equation (1).

$$EAR = \frac{||P_1 - P_2|| + ||P_3 - P_5||}{2||P_1 - P_4||} \quad (1)$$

Where $P_1 \ldots . P_6$ are the location of the 2D landmark as shown in Fig.5. When an eye is open, the EAR is essentially constant and approaches zero when it is closed. It is partially insensitive to person and head pose. The aspect ratio of the open eye varies little between individuals and is completely invariant to uniform image scaling and in-plane facial rotation. Because both eyes blink at the same time, the EAR of both eyes is averaged as shown in equation (2).

$$avg\ EAR = \frac{1}{2}(EAR_{left} + EAR_{right}) \quad (2)$$

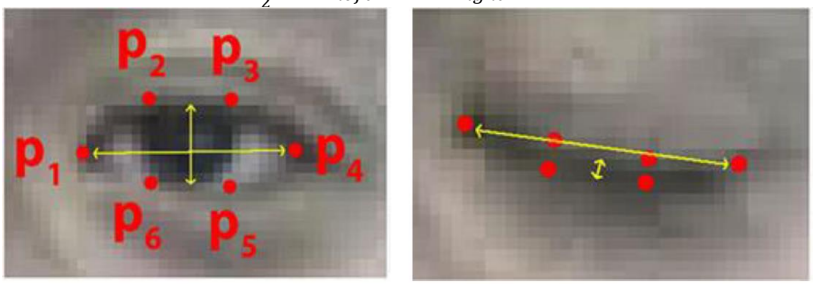

Fig.5 Open and closed eyes with landmarks p(i) automatically detected.

Eq. (1) plots the eye aspect ratio EAR for multiple frames of a video stream as shown in fig 6.

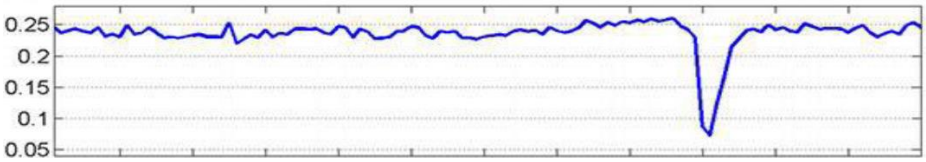

Fig.6 EAR for sight blink

4 Drowsiness detection system

In this step, includes Eye state determination and drowsiness detection.

1.4 Eye state determination

Finally, the eye state is determined using the EAR computed in the preceding phase. If the distance is 0 or is near zero, the eye state is categorized as "closed" otherwise the eye state is recognized as "open".

1.5 Drowsiness detection

The algorithm's final step is to determine the person's status based on a pre-set drowsiness condition. A person's typical blink duration is 100-400 milliseconds (i.e.0.1-0.4 of a second). As a result, if a person is drowsy, his eye closure must be longer than this period. We established a

time limit of 5 seconds. Drowsiness is detected if the eyelids are closed for five seconds or more, and a warning pop is triggered.

Implementation of drowsiness detection with Python and OpenCV was done which includes the following steps: Successful runtime capturing of video with the camera. The captured video was divided into frames and each frame was analyzed. Successful detection of the face followed by detection of an eye. If the closure of the eye for successive frames were detected, then it is classified as a drowsy condition else it is regarded as a normal blink, and the loop of capturing an image and analyzing the state of the driver is carried out again and again. In this implementation, during the drowsy state, the eye is not surrounded by a circle or it is not detected, and the corresponding message is shown.

The results/screenshots of different drowsiness states as shown in Fig. 7 and Fig.8.

Fig.7 When a driver closes the eye to sleep

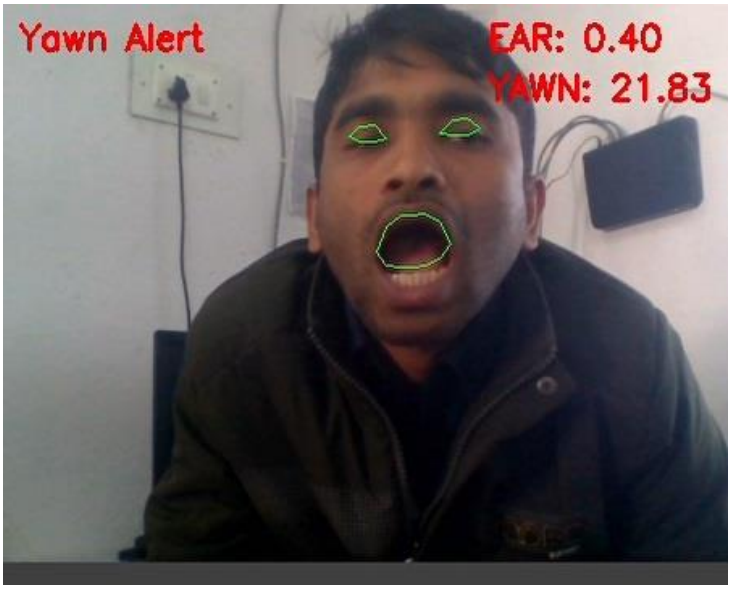

Fig.8. Yawn alert when mouth opens

5 Conclusion

Many researchers have worked on drowsiness detection over the last few decades with great results. However, research is continuously being conducted in this subject to improve the system's reliability. This application presents several difficulties. For example, if a motorist is wearing spectacles, a camera or sensor will only view a portion of his or her face. Our model is intended to detect drowsiness in the eye and provides an alert signal or warning in the form of an audio alarm. However, the driver's response after being warned may not be sufficient to prevent the accident, implying that if the motorist is sluggish to reply to the warning signal, an accident may occur. Hence to avoid this we can design and fit a motor-driven system and synchronize it with the warning signal so that the vehicle will slow down after getting the warning signal automatically.

6 References

[1] K. C. Bourne, "Education," *Appl. Adm. Handb.*, pp. 571–582, 2014, doi: 10.1016/b978-0-12-398545-3.00033-9.

[2] P. Goldsborough, "A Tour of TensorFlow," 2016, [Online]. Available: http://arxiv.org/abs/1610.01178.

[3] J. Díaz-Ramírez, "Machine Learning and Deep Learning," *Ingeniare*, vol. 29, no. 2, pp. 182–183, 2021, doi: 10.4067/S0718-33052021000200180.

[4] R. M. Sanchez, Earle Kit, Linsangam, NoelB, E.Angelia, "Three Triangle Method for Face Recognition using Dlib and OpenCV," vol. 9, pp. 5–9.

[5] J. Gwak, A. Hirao, and M. Shino, "An investigation of early detection of driver drowsiness using ensemble machine learning based on hybrid sensing," *Appl. Sci.*, vol. 10, no. 8, 2020, doi: 10.3390/APP10082890.

[6] A. Zaki, M. Noor, F. A. Jafar, M. R. Ibrahim, and S. N. Mohamed, "Fatigue Detection among Operators in Industry Based on Euclidean Distance Computation Using Python Software," *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 9, pp. 6375–6379, 2020, doi: 10.30534/ijeter/2020/236892020.

[7] A. Altameem, A. Kumar, R. C. Poonia, S. Kumar, and A. K. J. Saudagar, "Early Identification and Detection of Driver Drowsiness by Hybrid Machine Learning," *IEEE Access*, vol. 9, pp. 162805–162819, 2021, doi: 10.1109/ACCESS.2021.3131601.

[8] F. You, X. Li, Y. Gong, H. Wang, and H. Li, "A Real-time Driving Drowsiness Detection Algorithm with Individual Differences Consideration," *IEEE Access*, vol. 7, pp. 179396–179408, 2019, doi: 10.1109/ACCESS.2019.2958667.

[9] A. M. and S. Bilgaiya, *Drowsiness Detection System Using KNN and OpenCV*. 2020.

[10] M. L. R. Chandra, B. V. Kumar, and B. Sureshbabu, "IoT enabled home with smart security," *2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput. ICECDS 2017*, pp. 1193–1197, 2018, doi: 10.1109/ICECDS.2017.8389630.

[11] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1867–1874, 2014, doi: 10.1109/CVPR.2014.241.

[12] X. Huang, S. J. Lee, C. Z. Kim, and S. H. Choi, "An automatic screening method for strabismus detection based on image processing," *PLoS One*, vol. 16, no. 8 August, pp. 1–14, 2021, doi: 10.1371/journal.pone.0255643.

[13] C. Dewi, R. C. Chen, X. Jiang, and H. Yu, "Adjusting eye aspect ratio for strong eye blink detection based on facial landmarks," *PeerJ Comput. Sci.*, vol. 8, no. 2020, pp. 1–21, 2022, doi: 10.7717/peerj-cs.943.