Driver Drowsiness Detection System

Aditya Jain[1] Dhruvit Jain[2] Swati Bairagi[3]
[1,2,3]NMIMS, MPSTME, Mumbai, India

Abstract— Driver Drowsiness Detection System is a paramount for road safety. There are about 16 people in India that die from road accidents every hour! [3] This is predominantly due to the fact that there are occupations of people in India that demand them to drive all through the night. Addressing the gravitas of the situation, there are technologies that are being developed in the field of drowsiness detection while driving. An important point here is drowsiness is not only limited to night driving but also day driving. We have implemented a model to detect and avoid drowsiness during driving by using Python, OpenCV and Keras technologies. [5].

Keywords: Driver Drowsiness Detection System

I. INTRODUCTION

Driver Drowsiness Detection System is a paramount for road safety. There are about 16 people in India that die from road accidents every hour! [1] This is predominantly due to the fact that there are occupations of people in India that demand them to drive all through the night. When the driver is suffering from drowsiness, the driver loses the control of the car, so the driver might be suddenly deviated from the road and hit an obstacle or a car to overturn.

Addressing the gravitas of the situation, there are technologies that are being developed in the field of drowsiness detection while driving. An important point here is drowsiness is not only limited to night driving but also day driving. In this paper, we have implemented a model to detect and avoid drowsiness during driving by using Python, OpenCV and Keras technologies. At this point we are using the camera of your laptop, if there is no camera built in your laptop, an external webcam maybe required.

II. METHODOLOGY

Drowsiness is a physiological state with a tendency to fall asleep. Technically, drowsiness is different from the fatigue that is the lack of willingness to continue performing the same activity. Fatigue occurs by performing tasks that are always performed in the same way using the same muscle groups, their repetition rate is high and are usually performed by adopting forced postures such as monitoring a screen. A person maybe fatigued without being drowsy, but conditions that produce fatigue such as driving cars over great distances unmask the presence of physiological drowsiness, but do not cause fatigue. [4]

The purpose of this paper is to implement a solution to the problem of driver drowsiness. Drowsiness can be detected in many forms. It can be analyzed by the PERCLOS method which is the percentage of eye closure, yawn detection, blink rate of the eye, eye closure duration, facial action method and more.

A. Algorithm

In this paper we have used the eye closure duration method. The basic working of this model can be depicted by a flowchart attached as follows:

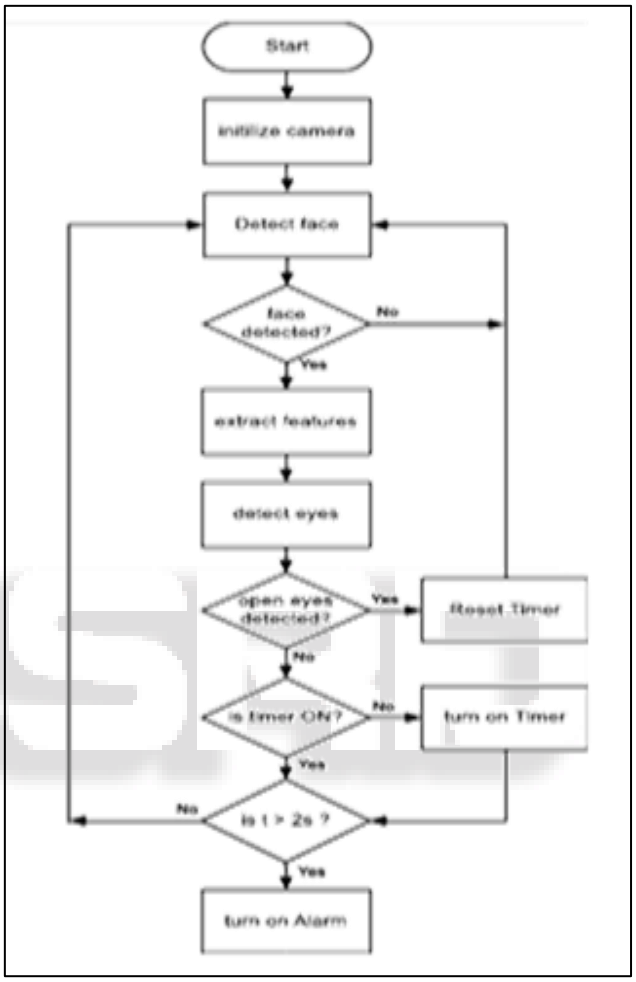


Fig. 1: Program Algorithm [1]

To begin with the camera is initialized once the program is executed. Then your face is detected by the camera. If the detection is successful then it goes on to extract facial features else it retries to detect the face. After extraction of facial features, it then goes on to detect eyes. If they are open then the timer is not affected at all and the program executes all the above steps without any further worries. Once the eyes are detected closed then the timer begins. If the timer value reaches to 2 seconds (2 seconds is just a threshold value selected by us, the value can be changed at the time of coding) then an alarm/buzzer is rung. This precisely means that your eyes are shut for too long. The longer the eyes are shut the longer will be the duration of the buzzer. Once the eyes are detected open, the timer value starts decreasing. The moment timer value falls below the threshold value, the buzzer will stop. Hence the only the way to stop this buzzer is to open your eyes.

B. Prerequisites and Dataset

For this project we have used a webcam (of our laptop) that can capture real time images. We have used python 3.7 for our project but the same project can be implemented using python 3.6 as well. Then using PIP which is a package manager for python we have downloaded the following packages

1) OpenCV-

It stands for Open Source Computer Vision. Used for real time computer vision. All image related operations are performed on OpenCV

2) TensorFlow-

TensorFlow is a Python library for fast numerical computing created and released by Google. It consists of nodes and edges that performs mathematical operations to build complicated Deep Learning models.

3) Keras-

It is used for building the classification model and it uses Tensor flow as backend. If the dataset obtained is not trained and tested, the same can be classified using Keras

4) Pygame-

It is used to play the alarm sound

The dataset which is used by us was already trained and tested. The data comprises around 7000 images of people's eyes under different lighting conditions and classified them as 'Open' or 'Closed'.

In all the dataset obtained by us, the model was built with Keras using Convolution Neural Network (CNN). CNN is a class of Deep Neural Network that is used for analysing the visual imagery. It consists of an input layer, an output layer and a hidden layer which can have multiple numbers of layers. As the name suggests, CNN uses convolution technique on the layers using a filter.

In all the layers, ReLU activation function is used except the output layer where Softmax is used.

ReLU, Rectifier Linear Unit activation is given by

$$\text{ReLU: } f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}, \; f'(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}.$$

In this function, output will be input if the function is positive, otherwise, the output will be zero.

5) Softmax

$$\text{Softmax: } f(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{d} e^{x_k}}, \; \frac{\partial f(x_j)}{\partial x_i} = f(x_j)(\delta_{ij} - f(x_i)).$$

Softmax in our project is used in the output. It converts input components that can be positive or negative numbers that may not even add up to 1 to range in between the interval [0,1] so that they can be interpreted as probabilities and always add up to 1. Larger the input, larger will be the corresponding probability.

C. Detection of Face and Eyes

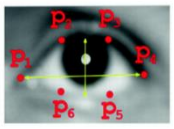
The images in our dataset have been trained using Haar Cascade classifier which is used to detect the object for which it has been trained for, from the source. The Haar Cascade is trained by superimposing the positive image over a set of negative images. The training is generally done on a server and on various stages. In our case, we are detecting the eyes and face.

The real time images in our project is taken from the camera. Hence for accessing each frame we need to have an infinite loop. We do this by OpenCV by using the cv2.VideoCapture (0) and then cap.read() will read each frame. Here cap is a capture object. After reading the frames are stored in the frame variable.

Once we have read the frames and stored them, we have to convert our images into grayscale as OpenCV needs grayscale images as Input. Using haar cascade classifier, we can detect the faces. It returns an array of detections with x,y coordinates, and height, the width of the boundary box of the object.

For detecting eyes we use the same procedure, for face detection. We have the height and width of the boundary box of the object, which in this case happens to be the eye. With the function

eye = frame [y : y+h, x : x+w] we can extract the eye.



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

EAR- Eye Aspect Ratio

The ratio of length of the eyes to the width of the eyes is called Eye Aspect Ratio. The length of the eyes is calculated by averaging over two distinct vertical lines across the eyes

D. Categorisation into open or close images

For categorising the image into open and closed we are using a CNN classifier. We resize the image as per the required pixels in our dataset. In our case we have 24*24 pixels. For finding the region of convergence more accurately, it is better to normalize the data. After normalization all values obtained will be between 0 and 1. Using our prediction function,

pred=model.predict_classes(eye) [2]

We get the value of pred as 0 or 1. If pred = 1 then your eye is open, else your eye is closed.

E. Score Check

Considering a real life scenario this is the most essential part of our project. This is a counter that basically keeps a check of duration of closure of eyes. Once this value exceeds a certain threshold value, (in our case we have set this threshold as 2) an alarm starts ringing. Once the alarm is ringing it will only shut when both the eyes are open. The counter starts to decrease once eyes are detected open and the alarm will stop once the counter falls below that threshold value. The alarm is played using pygame.

III. EXPERIMENTAL RESULTS

After executing our code, we realised our model worked efficiently under following conditions,

1) A backlight source has to be present. With little light or no light our model becomes inefficient as it faces difficulty in extracting the image. A note to be kept in mind is that, we have been using a laptop camera. With a better quality externally connected camera, the result is sure to improve.

2) The naked eye of the person at all times should be in the field of view of the camera. Any obstruction to the naked eye will not help the camera detect the eye.

Following are two cases of detection of Region of Interest. (RoI).

The yellow box gives us a clear depiction of the region of interest chosen by the camera.

We have two cases. The first one where the eyes are detected closed and the second one where the eyes are detected open. The results of both these cases are described below for a better understanding.

1) Case 1: The eyes are closed.

Fig 3.1: Eyes are closed

Fig. 3.2: Eyes are closed. (ROI)

In this case, firstly the message is printed that state of is closed. For the score it increases every time the eye is detected closed. After the count value crosses the threshold value the picture above shows the window changing to orange and an alarm being rung simultaneously.

2) Case 2: The eyes are open.

Fig. 3.3: Eyes are open

Fig 3.4: Eyes are open (ROI)

Each time the eye is detected open the score decreases. To stop the alarm the score has to come below the threshold value. Once the score falls below the threshold, it is visible from the above picture that the boundary of the window restores to normal.

IV. CONCLUSION

This project proposes a drowsiness detection system based on driver's eye behaviour. At this point this has many limitations. The biggest being its failure to detect eyes in low light. Most cases of drowsiness happen in the night and its efficiency in the dark is low. The other limitation being time lag between score and alarm to ring.

By incorporating better quality hardware the overall efficiency of the system can be improved.

Although this system is now starting to find its application in cars, it is extremely costly. The major advantage of our project is, it is extremely portable and can be fixed on the dashboard of your car.

I future, many other physiological factors of drowsiness like, heart rate, yawn detection [6], PERCLOS - percentage of eye closure, frequency of blinking, fatigue and sleep pattern can be incorporated in our project [7]. This will improve the efficiency of our project and in turn help reduce accidents that are caused due to drowsiness.

REFERENCES

[1] Nikita G. Prajapati, PG Student, Department of Computer Engineering, IIET-Dharmaj and Ms. Pooja M. Bhatt, Assistant Professor "Driver Drowsiness Detection with Audio-Visual Warning", June,2016 IJIRST –International Journal for Innovative Research in Science & Technology| Volume 3 | Issue 01
[2] Belal Alshaqaqi et al.," Driver drowsiness detection system, Advances in Systems Science and Application (2016) Vol.16 No.2
[3] Global Road Safety Report 2015.
[4] Global Status Report on Road Safety 2009; World Health Organisation (WHO): Geneva, Switzerland, 2009.
[5] Bergasa, L.M.; Nuevo, J.; Sotelo, M.A.; Barea, R.; Lopez, M.E. Real-time system for monitoring driver vigilance. IEEE Trans. Intell. Transport. Syst. 2006, 7, 63–77.
[6] Shen, W.; Sun, H.; Cheng, E.; Zhu, Q.; Li, Q. Effective driver fatigue monitoring through pupil detection and yawing analysis in low light level environments. Int. J. Digit. Technol. Appl. 2012,6, 372–383.
[7] Lew, M.; Sebe, N.; Huang, T.; Bakker, E.; Vural, E.; Cetin, M.; Ercil, A.; Littlewort, G.; Bartlett, M.; Movellan, J. Drowsy driver detection through facial movement analysis. In Human-Computer Interaction; Springer: Berlin, Germany, 2007; Volume 4796, pp. 6–18.

