Driver Drowsiness Monitoring System using Visual Behaviour and Machine Learning

In this project by monitoring Visual Behaviour of a driver with webcam and machine learning SVM (support vector machine) algorithm we are detecting Drowsiness in a driver. This application will use inbuilt webcam to read pictures of a driver and then using OPENCV SVM algorithm extract facial features from the picture and then check whether driver in picture is blinking his eyes for consecutive 20 frames or yawning mouth then application will alert driver with Drowsiness messages. We are using SVM pre-trained drowsiness model and then using Euclidean distance function we are continuously checking or predicting EYES and MOUTH distance closer to drowsiness, if distance is closer to drowsiness then application will alert driver.

To implement above concept we are using following modules

Video Recording: Using this module we will connect application to webcam using OPENCV built-in function called VideoCapture.

Frame Extraction: Using this module we will grab frames from webcam and then extract each picture frame by frame and convert image into 2 dimensional array.

Face Detection & Facial Landmark Detection: Using SVM algorithm we will detect faces from images and then extract facial expression from the frames.

Detection: Using this module we will detect eyes and mouth from the face

Calculate: Using this module we will calculate distance with Euclidean Distance formula to check whether given face distance closer to eye blinks or yawning, if eyes blink for 20 frames continuously and mouth open as yawn then it will alert driver.

OpenCV is an artificial intelligence API available in python to perform various operation on images such as image recognition, face detection, and convert images to gray or coloured images etc. This API written in C++ languages and then make C++ functions available to call from python using native language programming. Steps involved in face detection using OpenCV.

Face Detection Using OpenCV

This seems complex at first but it is very easy. Let me walk you through the entire process and you will feel the same.

Step 1: Considering our prerequisites, we will require an image, to begin with. Later we need to create a cascade classifier which will eventually give us the features of the face.

Step 2: This step involves making use of OpenCV which will read the image and the features file. So at this point, there are NumPy arrays at the primary data points.

All we need to do is to search for the row and column values of the face NumPy N dimensional array. This is the array with the face rectangle coordinates.

Step 3: This final step involves displaying the image with the rectangular face box.

SVM Description

Machine learning involves predicting and classifying data and to do so we employ various machine learning algorithms according to the dataset. SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes. In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification. As a simple example, for a classification task with only two features (like the image above), you can think of a hyperplane as a line that linearly separates and classifies a set of data.

Intuitively, the further from the hyperplane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyperplane as possible, while still being on the correct side of it.

So when new testing data is added, whatever side of the hyperplane it lands will decide the class that we assign to it.

How do we find the right hyperplane?

Or, in other words, how do we best segregate the two classes within the data?

The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.

Project Description

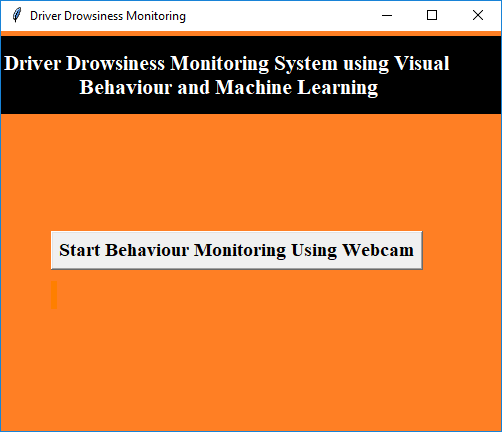
Drowsy driving is one of the major causes of road accidents and death. Hence, detection of driver’s fatigue and its indication is an active research area. Most of the conventional methods are either vehicle based, or behavioural based or physiological based. Few methods are intrusive and distract the driver, some require expensive sensors and data handling. Therefore, in this study, a low cost, real time driver’s drowsiness detection system is developed with acceptable accuracy. In the developed system, a webcam records the video and driver’s face is

detected in each frame employing image processing techniques. Facial landmarks on the detected face are pointed and subsequently the eye aspect ratio, mouth opening ratio and nose

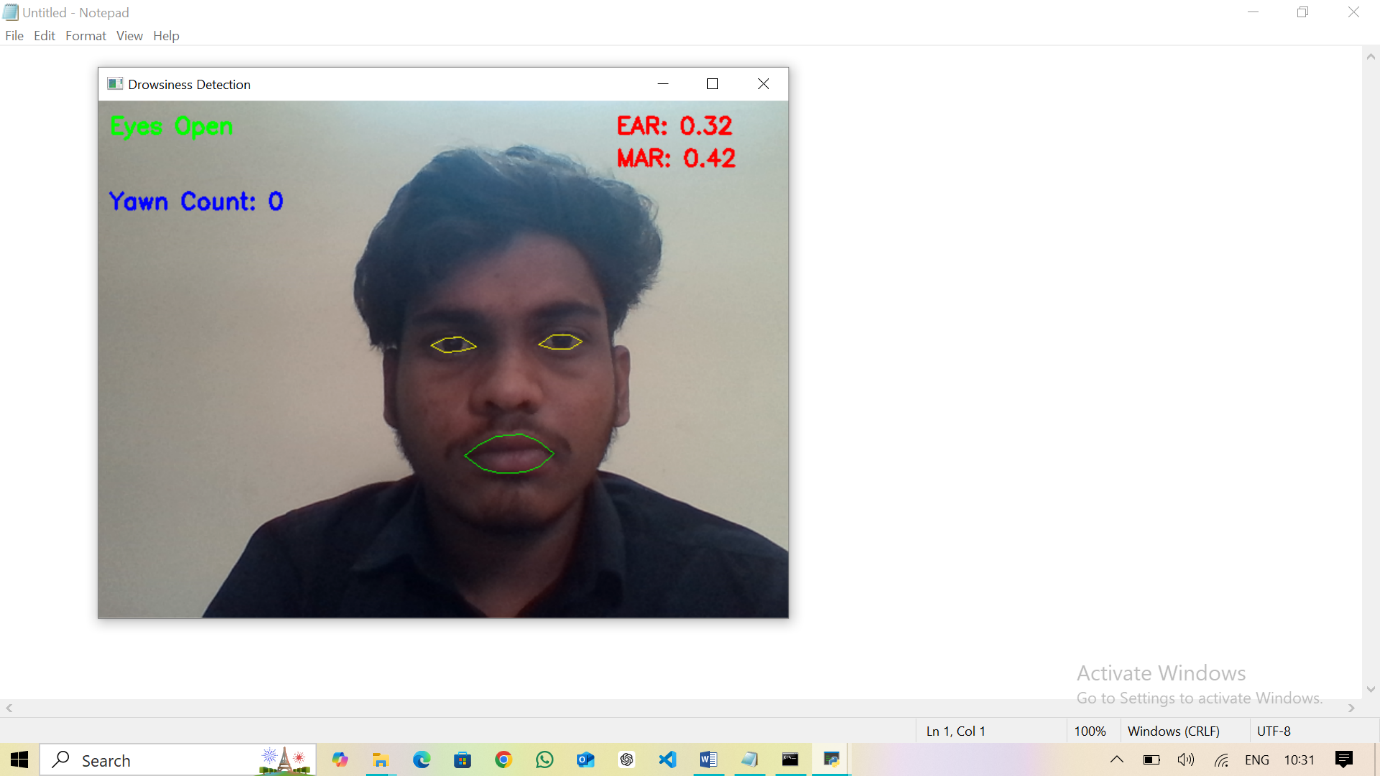
length ratio are computed and depending on their values, drowsiness is detected based on developed adaptive thresholding.

Screen shots

To run this project double click on ‘run.bat’ file to get below screen



In above screen click on ‘Start Behaviour Monitoring Using Webcam’ button to connect application with webcam, after clicking button will get below screen with webcam streaming



In above screen we can see web cam stream then application monitor all frames to see person eyes are open or not, if closed then will get below message



Similarly if mouth starts yawn then also will get alert message

