

Real Time Drowsiness Detection

**B.Tech Computer Science & Engineering
Maharaja Agrasen Institute of Technology
(GGSIPU)**

Team Name - Project Hunters

Member 1:Pratham Verma-07114802716

Member 2:Rohit Shastri-08114802716

Member 3:Nischal Tyagi-06614802716

ABSTRACT

Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects. The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time. By monitoring the eyes, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident. Detection of fatigue involves the observation of eye movements and blink patterns in a sequence of images of a face. First we input the facial image using a webcam. Preprocessing was first performed by binarizing the image. The top and sides of the face were detected to narrow down the area where the eyes exist. Using the sides of the face, the center of the face was found which will be used as a reference when computing the left and right eyes. Moving down from the top of the face, horizontal averages of the face area were calculated. Large changes in the averages were used to define the eye area. There was little change in the horizontal average when the eyes were

closed which was used to detect a blink. However Matlab had some serious limitations. The processing capacities required by Matlab were very high. Also there were some problems with speed in real time processing. Matlab was capable of processing only 4-5 frames per second. On a system with a low RAM this was even lower. As we all know an eye blink is a matter of milliseconds. Also a drivers head movements can be pretty fast. Though the Matlab program designed by us detected an eye blink, the performance was found severely wanting. This is where OpenCV came in. OpenCV is an open source computer vision library. It is designed for computational efficiency and with a strong focus on real time applications. It helps to build sophisticated vision applications quickly and easily. OpenCV satisfied the low processing power and high speed requirements of our application.

We have used the Haartraining applications in OpenCV to detect the face and eyes. This creates a classifier given a set of positive and negative samples. The steps were as follows:-

Gather a data set of face and eye. These should be stored in one or more directories indexed by a text file. A lot of high quality data is required for the classifier to work well. The utility application `createsamples()` is used to build a vector output file. Using this file we can repeat the training procedure. It extracts the positive samples from images before normalizing and resizing to specified width and height. The Viola Jones cascade decides whether or not the object in an image is similar to the training set. Any image that doesn't contain the object of interest can be turned into negative sample. So in order to learn any object it is required to take a sample of negative background image. All these negative images are put in one file and then it's indexed. Training of the image is done using boosting. In training we learn the group of classifiers one at a time. Each classifier in the group is a weak classifier. These weak classifiers are typically composed of a single variable decision tree called stumps. In training the decision stump learns its classification decisions from its data and also learns a weight for its vote from its accuracy on the data. Between training each classifier one by one, the data points are reweighted so that more attention is paid to the data points where errors were made. This process continues until the total error over the dataset arising from the combined weighted vote of the decision trees falls below a certain threshold.

Data Collection

This algorithm is effective when a large number of training data are available.

For our project face and eye classifiers are required. So we used the learning objects method to create our own haarclassifier .xml files.

Around 2000 positive and 3000 negative samples are taken. Training them is a time intensive process. Finally face.xml and haarcascade-eye.xml files are created.

These xml files are directly used for object detection. It detects a sequence of objects (in our case face and eyes). Haarcascade-eye.xml is designed only for open eyes. So when eyes are closed the system doesn't detect anything. This is a blink. When a blink lasts for more than 5 frames, the driver is judged to be drowsy and an alarm is sounded.