DROWSINESS DETECTING SYSTEM

##### 15IT322E - MINOR PROJECT REPORT

###### ***Submitted by***

##### ASHWANI KUMAR SINHA (RA1611003010685)

**ALAPAN KAR (RA1611003010633)**

***for the course***

**15IT322E – PYTHON PROGRAMMING**

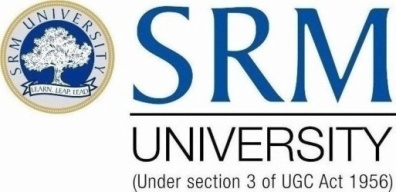
***in partial fulfillment for the award of the degree***

***of***

**Bachelor of Technology**

*in*

# COMPUTER SCIENCE AND ENGINEERING



SRM UNIVERSITY

KATTANKULATHUR

##### OCTOBER 2018

SRM UNIVERSITY

KATTANKULATHUR

**BONAFIDE CERTIFICATE**

Certified that this project report **“DROWSINESS DETECTING SYSTEM”** is the bonafide work of **“ASHWANI KUMAR SINHA (RA1611003010685), ALAPAN KAR (RA1611003010633)”** who carried out the project work as part of their course **15IT322E – Python Programming**.

**SIGNATURE SIGNATURE**

**Course Instructor** **Dr. B. Amutha**

**HEAD OF THE DEPARTMENT Computer Science & Engg**

**INTERNAL EXAMINER**

**ABSTRACT**

Today, number of accidents happen during drowsy driving on roads and are increasing day by day. It is a known fact that many accidents occur due to driver’s fatigue and sometimes due to inattention factor. This research mainly engages on maximizing the effort in identifying the drowsiness state of driver in real driving conditions. The goal of driver drowsiness detection systems is an attempt to contribute in reducing these road accidents. The secondary data collected focuses on past research on drowsiness detection systems and various methods have been used earlier for detection of drowsiness or inattention while driving. However, in this paper, a real time vision-based method is proposed to monitor driver fatigue. This research approach adopts the Viola-Jones classifier to detect the driver’s facial features. Firstly, the face is located by a Haar like feature based object detection algorithm. The face area is detected using the functions in the OpenCV library . Secondly, eye is detected. Also the eye areas are detected by using the functions in the OpenCV library and tracking by using a template matching method. Then, the open/close state of eyes is determined, and then fatigue is determined based on the series state of eyes. The correlation coefficient template matching method is applied to derive the state of each feature on a frame by frame basis. Vision- based driver fatigue detection method is a natural, non-intrusive and convenient technique to monitor driver’s vigilance.

**DROWSINESS DETECTING SYSTEM**

-Problem Statement

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.

Generally, there are many reasons behind highway traffic accidents. Driver drowsiness is one of the major causes of serious traffic accidents. According to the National Highway Traffic Safety Administration (NHTSA), there are about 56,000 crashes caused by drowsy drivers every year in US, which results in about 1,550 fatalities and 40,000 nonfatal injuries annually. The National Sleep Foundation also reported that 60% of adult drivers have driven while felling drowsy in the past year, and 37% have ever actually fallen asleep at the wheel. Many of the road accidents are occur due to driver fatigue/ driver drowsiness or driver sleepiness. Sleepiness reduces the concentration, activeness, alertness and vigilance of the driver and it makes the driver to take slow decisions and sometimes no decision. Drowsiness affects the mental alertness and decreasing the driver ability to operate a vehicle safely and increasing the risk of human error that could lead to fatalities and injuries. The reasons for the fatigue related crashes are long journeys on monotonous roads, driving after eating or taking an alcoholic drink, having less sleep than normal, after taking medicines that cause drowsiness, driving after long working hours and journeys after night shifts etc. Hence there is a need to address this problem to avoid accidents by alerting the driver so that road safety can be increased.

**Related Work**

In-vehicle camera is commonly installed to realize the possible reasons of car accidents. Such a camera can also be used to detect the fatigue of the driver. Several studies related to the fatigue detection are described as follows.

Sharma et al. utilized the number of pixels in the eye image to determine the eye state, open or close. Hornget at. established an edge map to locate the eyes locations and the eye state is determined based on the HSL color space of the eye image. Its accuracy is dependent on the location of the eyes. Sharma and Banga converted the face image to YCbCr color space. The average and standard deviation of the pixel number in the binarization image is computed. Then, fuzzy rules are used to determine the eye state. Liu et al. and Tabrizi et al. proposed methods to detect the upper and lower eyelids based on the edge map. The distance between the upper and lower eyelids is then used to analyze the eye state. Besides, Dong et al. and Li et al. proposed methods by utilizing AAM (Active Appearance Model) to locate the eyes. Then, a PERCLO (percentage of eye closure) was computed to detect the fatigue. For the above methods, the locating of eye areas was easily influenced by the change of brightness. Circular Hough transform is popular method to overcome the influence of brightness. Several studies proposed methods to locate the pupil of eyes by using circular Hough transform. Then, the eye state was analyzed according to the locations of pupils. He calculated the ratio of eye closing during a period of time. The ratio can reflect driver’s vigilance level. He proposed a method to detect the distance of eyelid, and then judged the driver’s status by this kind of information. Nikolaos P used front view and side view images to precisely locate eyes. Edge detection and gray-level projection methods were also applied for the eyes location by Wen-Bing. X located the face and proposed an eye tracking method. Abdel Fattah Fawky presented a combination of algorithms, namely wavelets transform, edge detection.

**Model**

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.



**image**

Now, all possible sizes and locations of each kernel are used to calculate lots of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose.

But the same windows applied to cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by **Adaboost**.



**image**

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found).

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow.. Isn't it a little inefficient and time consuming? Yes, it is. The authors have a good solution for that.

In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions.

For this they introduced the concept of **Cascade of Classifiers**. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. (Normally the first few stages will contain very many fewer features). If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

Software Implementation :

We have used Open CV as a platform to develop a code for eye detection in real time. The code is then implemented on system installed with Open CV software. To detect human eyes, face has to be detected initially. This is done by OpenCV face haar cascade classifier. Once the face is detected, the location of the eyes is estimated and eye detection is done using eye Haar-cascade classifier. Hence using the open CV, face and eyes are detected accurately and displayed on the monitor. The larger yellow square indicates the face while smallerred squares indicate the eyes.

**Program**

videoSteam = cv2.VideoCapture(0)

ret, frame = videoSteam.read()

size = frame.shape

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor("shape\_predictor\_68\_face\_landmarks.dat")

(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"]

(rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"]

model\_points = np.array([(0.0, 0.0, 0.0),

(0.0, -330.0, -65.0),

(-225.0, 170.0, -135.0),

(225.0, 170.0, -135.0),

(-150.0, -150.0, -125.0),

(150.0, -150.0, -125.0)])

focal\_length = size[1]

center = (size[1]/2, size[0]/2)

camera\_matrix = np.array([[focal\_length, 0, center[0]],

[0, focal\_length, center[1]],

[0, 0, 1]], dtype = "double")

dist\_coeffs = np.zeros((4,1))

t\_end = time.time()

faceCascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier('haarcascade\_eye.xml')

while(True):

ret, frame = videoSteam.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

rects = detector(gray, 0)

faces = faceCascade.detectMultiScale(

frame,

scaleFactor=1.1,

minNeighbors=5,

minSize=(30, 30),

flags=cv2.cv.CV\_HAAR\_SCALE\_IMAGE

)

for (x, y, w, h) in faces:

cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

roi\_gray = gray[y:y+h, x:x+w]

roi\_color = frame[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(roi\_gray)

for (ex,ey,ew,eh) in eyes:

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

for rect in rects:

shape = predictor(gray, rect)

shape = face\_utils.shape\_to\_np(shape)

leftEye = shape[lStart:lEnd]

rightEye = shape[rStart:rEnd]

jaw = shape[48:61]

leftEAR = eye\_aspect\_ratio(leftEye)

rightEAR = eye\_aspect\_ratio(rightEye)

ear = (leftEAR + rightEAR) / 2.0

mar = mouth\_aspect\_ratio(jaw)

image\_points = np.array([

(shape[30][0], shape[30][1]),

(shape[8][0], shape[8][1]),

(shape[36][0], shape[36][1]),

(shape[45][0], shape[45][1]),

(shape[48][0], shape[48][1]),

(shape[54][0], shape[54][1])

], dtype="double")

(success, rotation\_vector, translation\_vector) = cv2.solvePnP(model\_points, image\_points, camera\_matrix, dist\_coeffs, flags=cv2.CV\_ITERATIVE)

(nose\_end\_point2D, jacobian) = cv2.projectPoints(np.array([(0.0, 0.0, 1000.0)]), rotation\_vector, translation\_vector, camera\_matrix, dist\_coeffs)

if SHOW\_POINTS\_FACE:

for p in image\_points:

cv2.circle(frame, (int(p[0]), int(p[1])), 3, (0,0,255), -1)

p1 = (int(image\_points[0][0]), int(image\_points[0][1]))

p2 = (int(nose\_end\_point2D[0][0][0]), int(nose\_end\_point2D[0][0][1]))

if SHOW\_CONVEX\_HULL\_FACE:

leftEyeHull = cv2.convexHull(leftEye)

rightEyeHull = cv2.convexHull(rightEye)

jawHull = cv2.convexHull(jaw)

cv2.drawContours(frame, [leftEyeHull], 0, (255, 255, 255), 1)

cv2.drawContours(frame, [rightEyeHull], 0, (255, 255, 255), 1)

cv2.drawContours(frame, [jawHull], 0, (255, 255, 255), 1)

cv2.line(frame, p1, p2, (255,255,255), 2)

if p2[1] > p1[1]\*1.5 or COUNTER\_BLINK > 25 or COUNTER\_MOUTH > 2:

cv2.putText(frame, "Send Alert!", (200, 60), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

sound()

if ear < EYE\_AR\_THRESH:

COUNTER\_FRAMES\_EYE += 1

if COUNTER\_FRAMES\_EYE >= EYE\_AR\_CONSEC\_FRAMES:

cv2.putText(frame, "Sleeping Driver!", (200, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

sound()

else:

if COUNTER\_FRAMES\_EYE > 2:

COUNTER\_BLINK += 1

COUNTER\_FRAMES\_EYE = 0

if mar >= MOUTH\_AR\_THRESH:

COUNTER\_FRAMES\_MOUTH += 1

else:

if COUNTER\_FRAMES\_MOUTH > 5:

COUNTER\_MOUTH += 1

COUNTER\_FRAMES\_MOUTH = 0

if (time.time() - t\_end) > 60:

t\_end = time.time()

COUNTER\_BLINK = 0

COUNTER\_MOUTH = 0

if SHOW\_INFO:

cv2.putText(frame, "EAR: {:.2f}".format(ear), (30, 450),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0), 2)

cv2.putText(frame, "MAR: {:.2f}".format(mar), (200, 450),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0), 2)

cv2.putText(frame, "Blinks: {}".format(COUNTER\_BLINK), (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0), 2)

cv2.putText(frame, "Mouths: {}".format(COUNTER\_MOUTH), (10, 60),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0), 2)

cv2.imshow("Output", frame)

key = cv2.waitKey(1) & 0xFF

if key == ord('q'):

break

if key == ord('p'):

SHOW\_POINTS\_FACE = not SHOW\_POINTS\_FACE

if key == ord('c'):

SHOW\_CONVEX\_HULL\_FACE = not SHOW\_CONVEX\_HULL\_FACE

if key == ord('i'):

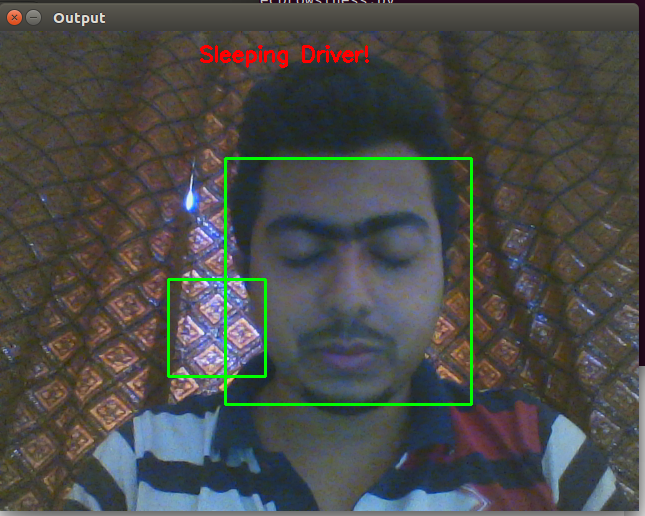
SHOW\_INFO = not SHOW\_INFO

time.sleep(0.02)

videoSteam.release()

cv2.destroyAllWindows()

**Results**





**Discussion/Conclusion**

The proposed system in this analysis provides detection of driver fatigue. The analysis and design of driver drowsiness detection system is presented. The proposed system can used to avoid various road accidents caused by drowsy driving and it can also help drivers to stay awake when driving by giving a warning when the driver is sleepy. And also this system used for security purpose of a driver. During the monitoring, the system is able to decide if the eyes are opened or closed. When the eyes have been closed for too long, a warning signal is issued. Image processing achieves highly accurate and reliable detection of drowsiness. This was achieved by interfacing a webcam to a PC and recording test videos and frame database under different lighting condition. The calculation speed, accuracy and robustness will be influenced by using combined algorithm.

**REFERENCES**

* NCSDR/NHTSA Expert Panel on Driver Fatigue and Sleepiness, “Drowsy driving and automobile crashes,” National Highway Traffic Safety Administration (NHTSA) report.
* M. R. Rosekind, “Underestimating the societal costs of impaired alertness: safety, health and productivity risks,” Sleep Medicine, vol. 6, pp. S21-S25, 2005.
* N. Sharma, V. K. Banga, "Development of a Drowsiness Warning System based on the Fuzzy Logic ," International Journal of Computer Applications, vol. 8, no. 9, pp.1-6, 2010.
* W. B. Horng, C. Y. Chen, Y. Chang and C. H. Fan, "Driver Fatigue Detection Based on Eye Tracking and Dynamic Template Matching," Proceedings of International Conference on Networking, Sensing & Control, Taipei, Taiwan, pp. 7-12, 2004.

**Individual Contribution**

Alapan Kar (Team-mate 1):

I have used Open CV as a platform to develop a code for eye detection in real time. The code is then implemented on system installed with Open CV software. To detect human eyes, face has to be detected initially. This is done by OpenCV face haar cascade classifier. Once the face is detected, the location of the eyes is estimated and eye detection is done using eye Haar-cascade classifier. Hence using the open CV, face and eyes are detected accurately and displayed on the monitor. The larger yellow square indicates the face while smallerred squares indicate the eyes.

Due to the complex background, it is not a good choice to locate or detect both the eyes in the original image, for this we will take much more time on searching the whole window with poor results. So firstly, we will locate the face, and reduce the range in which we will detect both the eyes.

Ashwani Kr. Sinha (Team-mate 2):

Actually, first the image is acquired by the webcam for processing. The images of the driver are captured from the camera which is installed in front of the driver on the car dashboard. It will be passed to preprocessing which prepares the image for further processing by the system. Its main operations are to eliminate noises caused by the image acquisition subsystem and image enhancement using Histogram Equalization. Then we search and detect the faces in each individual frame. If no face is detected then another frame is acquired. If a face is detected, then a region of interest in marked within the face. This region of interest contains the eyes. Defining a region of interest significantly reduces the computational requirements of the system. After that the eyes are detected from the region of interest. If an eye is detected then there is no blink. If the eyes are closed in a particular frame, then the blink counter is decremented and a blink is detected. When the eyes are closed for more than 4 frames then it is deducible that the driver is feeling drowsy. Hence drowsiness is detected and an alarm sounded. After that the whole process is repeated as long as the driver is driving the car.