DROWSINESS DETECTION SYSTEM

A THESIS SUBMITTED IN PARALLEL FULFULMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor in Technology
In
Electronics and Instrumentation Engineering

Under the guidance of: - Prof. Manish Okade
Department of Electronics and Communication Engineering
National Institute of Technology, Rourkela

By:-
Manoj Rupanagudi(111EI0060)

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis titled - Drowsiness Detection System, submitted by Manoj Rupanagudi (111EI0060) in fulfilment for the requirements for the award of Bachelor of Technology Degree in Electronics and Instrumentation Engineering at National Institute of Technology, Rourkela, is an authentic work carried out by him under my supervision and guidance.

Date:

Prof. Manish Okade

Department of Electronics
and
Communication Engineering

Acknowledgement

I would like to express my deep gratitude to my project guide Prof. Manish Okade for his guidance and help in the project work. I am grateful to the lab attendant of Digital Communication lab for allowing me to use the resources. I would like to thank all the faculty members and staff of the Department of Electronics and communication Engineering, N.I.T. Rourkela for their generous help in various ways for this project.

Abstract

There have been a lot of approaches in the detection of drowsiness. The parameters taken into consideration were the eye opening window, the number of blinks during a time period, no. of yawns etc. there are about 12 facial features that can be determined by the camera mounted on the circuit board. The parameter considered here is only the window of eye opening. In addition to the 12 parameters, the head motion was also taken into consideration which, in turn, contributed to the improvement of the accuracy of the measurement. Driver Drowsiness is one of the real reasons for mishaps on the planet. In this undertaking I plan to build up a model of drowsiness recognition framework. This framework meets expectations by observing the eyes of the driver and sounding a caution when he/she is tired. The framework so outlined is a non-nosy continuous checking framework. The need is on enhancing the security of the driver without being prominent. In this venture the eye flicker of the driver is recognized. In the event that the drivers' eyes stay shut for more than a certain duration of time, the driver is said to be languid and an alert is sounded. The programming for this is done in matlab using image acquisition tool.

Key words: Eye Detection, Vehicular safety Management, Drowsiness Detection

Contents

1. Introduction:

To detect the state of drowsiness of the driver, I have designed a matlab based algorithm which catches the motion of the driver's eyes and gives a mechanism to implicate the result**.** In a case where the fatigue is detected, a warning signal is designed to alert the driver. The currently used algorithm, uses Viola Jones algorithm which is unique to the already published papers on the topic of drowsiness detection.

To detect the drowsiness of the driver, we opt for the facial features recognition algorithms concentration primarily on the eye pair. The identification process is divided into 3 parts. :

1. selection of features;
2. detection of the state of the eye pair;
3. decision making;

The main objective of the work is to develop a code which serves best for the detection of the state of the eye pair (closed or open) under the conditions such as low light, random movement of the face, tilts of the face, etc. For most people, the shape of eye is reasonably similar to elliptical. When using a circular eye template, its upper and lower parts may be not necessary for calculating the separeability, which will affect the accuracy of irises detection. In this paper, instead of circular deformable eye template, in one of the methods we utilize an elliptical eye template that is more similar to the shape of human eye for detecting the iris candidates more robustly.

The idea is it make the vehicle intelligent to avoid accidents by using add ons to the dash board. India boasts of a very high number of 142,485 traffic-related fatalities because the driver was not paying attention ergo was drowsy. So a lot of engineering modules based on video processing using a mounted web camera on a circuit which detects the state of the eye either in closed or open position. The state of the eye is detected under low lit conditions especially in dark backgrounds. This is where the difference in the modus operandi differs in the execution of the algorithms. This paper focuses on the different algorithms used in detection of the eye pair in closed position and open position.

Initially, I decided to detect eye blink using Matlab. The procedure used was the geometric manipulation of intensity levels. The details of the ways to implement the algorithm are mentioned which will be seen in this thesis.

We use a web cam to check and detect the state of the eye. We have to first pre-process that image by binarising it to the greyscale. The region of interest (RIO) is set to the eye pair region where we detect the state of the eyes. Using the sides of the face, the centre of the face was found which will be used as a reference when computing the left and right eyes. Moving down from the highest point of intensity of the face, level of moderate intensities of the face region were computed. Huge changes in the level of the intensities were utilized to characterize the eye zone. There was little change in the even normal when the eyes were shut which was utilized to recognize a blink.

The design of the drowsiness detection should comprise of the following characters and have the following requirements:

The requirements for an effective drowsy driver detection system are as follows:

• A non-intrusive method which will not the

• A real-time monitoring system, to insure accuracy in detecting drowsiness.

• A system that will work in both daytime and night time conditions.

2.Why image Acquisition Toolbox?

Image Acquisition Toolbox enables one to acquire images and video from cameras and frame grabbers directly into Matlab and Simulink. One can detect hardware automatically and configure hardware properties. Advanced workflows let one trigger acquisition while processing in-the-loop, perform background acquisition, and synchronize sampling across several multimodal devices. With support for multiple hardware vendors and industry standards, one can use imaging devices ranging from inexpensive Web cameras to high-end scientific and industrial devices that meet low-light, high-speed, and other challenging requirements.

2.1 Key Features

Together, Matlab and Image Acquisition Toolbox provide a complete environment for developing tailor made imaging solutions. With the image Acquisition tool, we can extract images from the video, visualize the data and change our mechanism to trigger the warning signal, develop processing algorithms and techniques to analyse and create a GUI (as we have done in the project). The image acquisition engine enables one to acquire frames as fast as the camera and PC can support for high speed imaging. In addition, we can use Image Acquisition Toolbox with Simulink and Computer Vision System Toolbox to develop a model and simulate real-time embedded imaging systems.

Image Acquisition Toolbox simplifies the acquisition process by providing a consistent interface across operating systems and hardware devices. The toolbox provides multiple ways to access hardware devices from Matlab and Simulink. The Image Acquisition tool provides a programmatic interface in Matlab and a block for Simulink modelling. Each workflow provides access to camera properties and controls while enabling one to solve different types of problems with the strengths of each environment.

- It supports the industry standards, including DCAM, Camera Link etc.
- Support for a range of industrial and scientific hardware vendors such as Aurdino, FPGA etc.
- It has multiple buffer options to reduce the size of the image and avoid the scope for a memory problem which is an alpha problem in using Matlab.
- Synchronization of multimodal acquisition devices with hardware triggering

- Image Acquisition app for rapid hardware configuration, image acquisition, and live video previewing
- Support for C code generation in Simulink

The good thing about using matlab in the image processing is that one can log data to disk, memory, or both simultaneously with the Image Acquisition GUI or code directly at the Matlab command line. The usual complain that all the designs on drowsiness detection is the memory usage in the execution but one can set a limit on memory usage to prevent overuse of resources in memory-intensive applications. Data acquired with the tool can also be shifted directly to the Image Viewer GUI in Image Processing Toolbox for greater control over visualization. In addition, one can:

- Log each image frame or log frames at specified intervals
- Log data to disk as compressed or uncompressed AVI streams and MAT-files
- Specify frame rate, compression techniques, and key frame rate for AVI streams
- Extract single images from a video stream and store them in standard formats, including JPEG 2000, BMP, JPEG, and TIFF
-

2.2 Hardware Compatibility

Image Acquisition Toolbox helps in accounting several industry standards including DCAM, Camera Link, and GigE Vision digital interfaces, as well as common interfaces including DirectShow, QuickTime, and video4linux2. In the process of making the prototype, there are a lot of specification to which one has to adhere. I have used Arduino board to check the signals if they are coming out to be right. The sound warning system has to be supported in the car. Cars have USB power supply for audio boxes and phone charging points. So this hardware has to be interfaced with appropriate software. Matlab coupled with Image Acquisition Tool supports Windows, Linux, and Macintosh systems, enabling code design flexibility and easy hardware implementation. In addition, the toolbox provides specific support for some manufacturers, enabling proprietary features and increasing performance in image acquisition.

2.3 Acquiring Image Data

Image Acquisition Toolbox supports several modes, including background acquisition and continuous acquisition, while processing the acquired data. The toolbox automatically buffers data into memory, handles memory and buffer management, and enables acquisition from an ROI. The image acquisition engine is designed to acquire imagery as fast as the camera and computer can support, enabling analysis and processing of high-speed imaging applications.

Data can be acquired in a wide range of data types, including signed or unsigned 8-, 16-, and 32-bit integers and single- or double-precision floating point. The toolbox supports any color space provided by the image acquisition device including RGB, YUV, or grayscale. Raw sensor data in a Bayer pattern can be automatically converted into RGB data.

2.4 Callbacks

One can create callback functions or customized code that automatically execute when specific events occur, such as when acquisition starts or stops, a trigger occurs, or a set number of frames is acquired. Callback events can be used to process oner data as it is acquired by the image acquisition engine or to automate configuration settings as acquisition starts and stops. For example, one can measure statistics of frames within a video stream and activate downstream processing when a threshold is surpassed. In addition, buffer and memory options let one control the acquisition process further and flush data when needed**.**

3.Algorithm and Implementation.

Drowsiness of a person can be recognised and measured by the extended period of time for which his/her eyes are in closed state. In my system, primary attention is given to the efficiency and the ease with which the algorithm is able to detect the state of the eye.

The number of frames i.e 4, for which eyes are closed is monitored. The idea is to identify the stae of the driver's eye. Once the number of the blinks of the

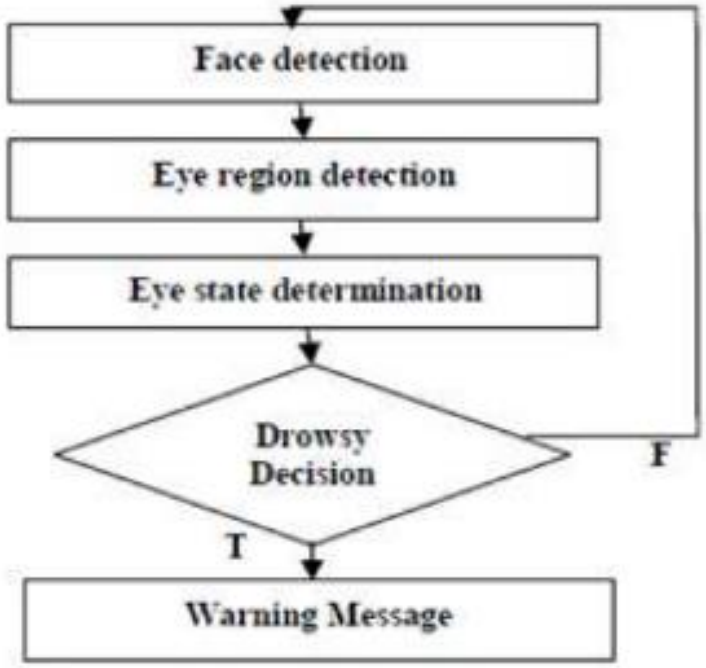
In my algorithm, first the image is acquired by the webcam (intex 1240X720_RGB) for processing. Once the face is detected, the region of interest, i.e. the eyes will be marked and the prime loop of the code will be executed to get the state of the eye. This region of interest contains the eyes. Defining a region of interest significantly reduces the computational requirements of the system.

On the off chance that an eye is distinguished then there is no squint and the flicker counter is set to NULL. In the event that the eyes are shut in a specific casing, then the flicker counter is increased and a squint is distinguished. At the point when the eyes are shut for more than 4 times then it is deducible that the driver is feeling sleepy. Subsequently drowsiness is distinguished and a caution sounded. After that the entire algorithm is loopedto get the state of the drivers' eyes.

An algorithm for eye state analysis, which incorporates into a four step system for drowsiness detection: face detection, eye detection, eye state analysis, and drowsy decision. It requires no training data at any step or special cameras. The novel eye state analysis algorithm detects open, semi-open, and closed eye during two steps which is based on brightness and numeral features of the eye image.

The algorithms tested were:

1. Valley Detection method.
2. Local binary patter using spatial pyramids.
3. Viola Jones algorithm for facial features detection.



3.1 Image Acquisition

The challenge in the project was to get the images frame by frame in real time. Anyone can write a piece of code for one static image and do the corresponding thresholding and put values to get the desired output.

The issue in matlab without the image acquisition toolbox was the memory usage. With a laptop of 4GB ram, it was quite sufficient to make the image processing dynamic. The acquired the images, we brake the video processing intro frames of 4. Exceeding the no of frames will take up more memory. In the GUI, once we start the camera, the following code will be execute under the callback function.

The code to capture the face

```
vid= videoinput('winvideo',2,'MJPG_1240x480');
triggerconfig(vid,'maunal');
set(vid,'FramesPerTrigger'1);
set(vid,'TriggerRepeat',inf);
set(vid,'ReturnedColourSpce', 'rgb');
start(vid);
faceDetect = vision.CascadeObjectDetector();        // inbuilt function in the image
                                                    acquisition toolbox

while(1)
   try
   trigger(vid);
   temp=getdata(vid,1);
   axes(handles.axes1);
   imshow(temp);
   g=getsnapshot;
   data=g;
   videoframe=data;
   bb= step(faceDetect,videoframe);                  // bounded box to get the frame
                                                     of the ROI
   [Im1, Im2]=separate(data,bb);
   axes(handles.axes2);
   imshow(Im2);
   pause(0.2);
   catch
      break;
   end;
end;
```

3.2 Valley Detection Method

Drowsiness of a person can be measured by the extended period of time for which his/her eyes are in closed state. In my system, primary attention is given to the faster detection and processing of data. The number of frames for which eyes are closed is monitored. If the number of frames exceeds a certain value that is set to 4, then a warning message is generated oby giving an audio signal showing that the driver is feeling drowsy. In the algorithm, first the image is acquired by the webcam for processing. 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.
My strategy of eye detection is roughly divided into 3 steps. The summary is given as follows.

STEP 1**:** Localization of the face region. A histogram back-projection method is utilized to extract the rough face region.

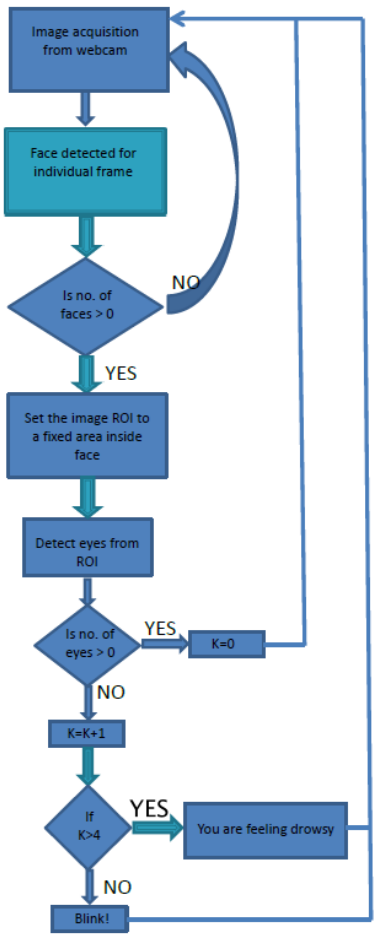
STEP 2**:** Detection of the iris candidates. Iris candidates are detected by using the elliptical separability filter (ESF).

STEP 3**:** Selection of the pair of irises. By calculating the total similarities of pairs of iris candidates, we determine the pair of iris, which has the largest similarity among others. The similarity of a pair of iris candidates consists of the separability of the pair of iris candidates, the VQ similarity and normalized correlation coefficient between the region including the pair of iris candidates and eye template.

First step by which we optimize the code is by setting the region of interest on which we specifically apply the algorithms. This reduces the computational requirements of the system to a large extent. The counter K is set to 0 initially. The counter is for the number of blinks the driver has over the 4 frames.

Once the number of blinks becomes less than 4, that is the eye is in the closed state, the sounding system is set off. So the process is very smooth in the transition from the point where the counter is 0, to the point where it detects the number of blinks and the counter is incremented, eventually to a point where the driver keeps his eyes shut and doesn't not blink resulting in the beeping sound of the system

The working of the algorithm includes the distance from the eyebrows to the upper curve of the eyelashes. The trigger to this mechanism is the number of blinks in a given frame of time. We are taking 1 frame per second. If in 4 successive frames, there is no change in the values of the state of the eyes which is detected closed from the code, the there is a warning signal generated to wake the driver up or sound an alarm.



This gives a flow chart of the algorithm used to implement the drowsiness detection system. We have taken a counter K to get the number of blinks per freame. The working of the algorithm has been explained in the steps of the algorithm above.

3.3 Local binary patter using spatial pyramids.

Every image acquired from the video can be split into a sub regions of 3X3 where the central element serves as the threshold value, around which we threshold the 8 bit binary code. For example the face images can be perceived as a set of micro patters which will correspond to a histogram. The spatial pyramid matching was to find an approximate correspondence between the two vectors in space obtained in a standard size image.

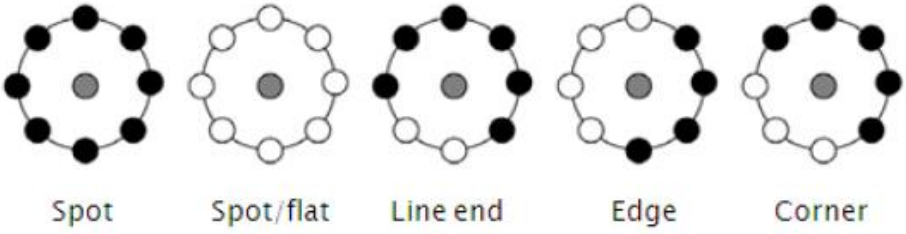
22	30	22
29	33	69
36	48	51

0	0	0
0		1
1	1	1

(a) (b)

The idea to understand is that the threshold values taken at the centre of the sub region is 33. Above 33, the value of the binary bit becomes 1 and below it becomes 0. The sub region (a) represents the values of the maximum intensity of that particular grid. 33 being the darkest region (containing the pupil which is black). We not take the value 33 as the base value. Above the value of 33, the other elements of the matrix become 1 and below that value of 33 become 0. Now the alignment of the 1s gives various shapes such as an edge, or a side or a circle.

To use the Local Binary Pattern in image pattern detection, we have a saved image to which we compare the binary values. We already have a saved image in the database to which we match the obtained image and overlap the values to check for difference using the following algorithm:

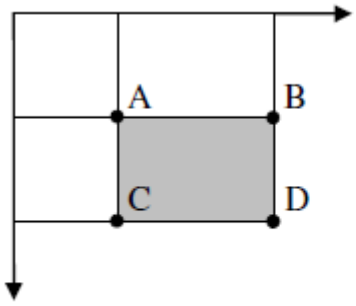


3.4 Viola Jones algorithm.

This algorithm was found to be the best fit to detect the features of the face, case in point the eye pair under low lighting conditions. The techniques such as image integration and attentional cascading act as an addendum to the working of the viola jones algorithm but here they were not used as it would increase he complexity of the code. For the algorithm to work with the standard PC, we use the standard 1280X720 resolution image's Harr like feature vector.

The face detector basically detects the face in a random sized image and gives the location of the face. Here in specific, it further gives the state of the eyes; closed or open. The image is broken down into binary values, i.e. a framework for 1s and 0s. The classifier will minimize the computational work. So in order to get the algorithm, working properly, I have to account for situations like the absence of faces, multiple faces and partial faces in the screen of capture. The algorithm must include false negative an false positive cases to achieve an acceptable performance. So all these exceptions and addendums on the code is achieved through an algorithm called Adaboost (details of which are given in the subsequent sections). This algorithm relies on classifiers which tell you which parameter is prominent and which are not. If the classifier is weak, then it cannot meet a classification in terms of the error generated by the system. Cascading makes this Viola Jones algorithm very accurate and it is the exact reason why we have implemented this algorithm and not Local Binary pattern or valley detection. This works well in PC with a standard RAM and a decent processor.

There major problem associated with any program to detect the face and eventually the yes is the conditions under which the image is being taken. The solution provided in the algorithm of Viola Jones is pretty simple. The position of the object (the face here) is unknown. Different drivers use a different length from the dashboard to drive a car. Generally it is up to a 100 cms but it cannot be generalised. For this very problem viola jones developed a simple solution.



Sum of grey rectangle = D - (B + C) + A

Given below are the 5 harr like features which give a skeleton to the code which we can write. So while taking the region of interest, the pixels that are taken are the white and the black one. For example, the ROI that I have chosen is the Haar like feature (b) which detects the eye pair region. So the pixel that contains only 2 colours that is converted into greyscale.

Five Haar-like patterns. The size and position of a pattern's support can vary provided its black and white rectangles have the same dimension, border each other and keep their relative positions. Thanks to this constraint, the number of features one can draw from an image is somewhat manageable.

The best performing feature is chosen based on the weighted error it produces. This weighted error is a function of the weights belonging to the training examples. As a result it is more 'expensive' for the second feature (in the final classifier) to misclassify an example also misclassified by the first feature, than an example classified correctly.

An alternative interpretation is that the second feature is forced to focus harder on the examples misclassified by the first. The point being that the weights are a vital part of the mechanics of the AdaBoost algorithm.

To improve the exiting conditions of the algorithm, there are two techniques applied to enhance the no of pixels we can use in a 24X24 image. The two techniques are

a) AdaBoost

AdaBoost (adaptive boosting) is an ensemble learning algorithm that can be used for classification or regression. Although AdaBoost is more resistant to overfitting than many machine learning algorithms, it is often sensitive to noisy data and outliers.

AdaBoost is called adaptive because it uses multiple iterations to generate a single composite strong learner. AdaBoost creates the strong learner (a classifier that is well-correlated to the true classifier) by iteratively adding weak learners (a classifier that is only slightly correlated to the true classifier). During each round of training, a new weak learner is added to the ensemble and a weighting vector is adjusted to focus on examples that were misclassified in previous rounds. The result is a classifier that has higher accuracy than the weak learners' classifiers.

Adaptive boosting includes the following algorithms:

- AdaBoost.M1 and AdaBoost.M2 – original algorithms for binary and multiclass classification
- LogitBoost – binary classification (for poorly separable classes)
- Gentle AdaBoost or GentleBoost – binary classification (for use with multilevel categorical predictors)
- RobustBoost – binary classification (robust against label noise)
- LSBoost – least squares boosting (for regression ensembles)
- LPBoost – multiclass classification using Local programming boosting
- RUSBoost – multiclass classification for skewed or imbalanced data
- TotalBoost – multiclass classification more robust than LPBoost

b) Cascaded classifier.

This is a method used in OpenCV. Since we have used Image Acquisition Tool in Computer Vision, information regarding this will not be much useful in this thesis.

Algorithm 1 Computing a 24 × 24 image's Haar-like feature vector

Input: a 24 × 24 image with zero mean and unit variance
Output: a $d \times 1$ scalar vector with its feature index f ranging from 1 to d
Set the feature index $f \leftarrow 0$
Compute feature type (a)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ do
for all (w, h) such that $i + h - 1 \le 24$ and $j + 2w - 1 \le 24$ do
compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
record this feature parametrized by (1, i, j,w, h): $S_1 - S_2$
$f \leftarrow f + 1$
end for
end for
Compute feature type (b)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ do
for all (w, h) such that $i + h - 1 \le 24$ and $j + 3w - 1 \le 24$ do
compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
compute the sum S_3 of the pixels in $[i, i + h - 1] \times [j + 2w, j + 3w - 1]$
record this feature parametrized by (2, i, j,w, h): $S_1 - S_2 + S_3$
$f \leftarrow f + 1$
end for
end for

Compute feature type (c)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ do
for all (w, h) such that $i + 2h - 1 \le 24$ and $j + w - 1 \le 24$ do
compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
record this feature parametrized by (3, i, j,w, h): $S_1 - S_2$
$f \leftarrow f + 1$
end for
end for

Compute feature type (d)
for all (i, j) such that $1 \le i \le 24$ and $1 \le j \le 24$ do
for all (w, h) such that $i + 3h - 1 \le 24$ and $j + w - 1 \le 24$ do
compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
compute the sum S_3 of the pixels in $[i + 2h, i + 3h - 1] \times [j, j + w - 1]$
record this feature parametrized by (4, i, j,w, h): $S_1 - S_2 + S_3$
$f \leftarrow f + 1$
end for
end for

42: Compute feature type (e)
43: for all (i, j) such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ do
44: for all (w, h) such that $i + 2h - 1 \leq 24$ and $j + 2w - 1 \leq 24$ do
45: compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
46: compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
47: compute the sum S_3 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
48: compute the sum S_4 of the pixels in $[i + h, i + 2h - 1] \times [j + w, j + 2w - 1]$
49: record this feature parametrized by (5, i, j,w, h): $S_1 - S_2 - S_3 + S_4$
50: $f \leftarrow f + 1$
51: end for
52: end

The challenge now it to process the live feed taken from the algorithm to get images snap. Processing a live feed the next step of the project was to perform the same on a live video feed obtained by either using an external USB operated camera or by using the built-in Webcam. The accuracy of this method of eye detection is based on the sensitivity of the camera. It is found to have a direct relationship with the accuracy. The greater the accuracy needed, the better quality of webcam has to be used. The first step towards implementing this, is to first identify the webcam drivers installed and then configure the webcam to obtain the necessary video feed. The associated webcams were identified by using the imaqhwinfo() function. The next step was to configure the webcam and assign the video properties. This involved setting the FramesPer Trigger and Returned Color Space properties of the video object. The live feed was then obtained using the start(videoobject) function. The vision. Cascade Object Detector statement for detecting the face was used to initialize an object FaceDetect. The next step was to crop the image such that only the face is retained static for further eye detection. This is achieved by visualizing the live video feed as individual frames and processing each frame distinctly. The vision. Cascade Object Detector for detecting the eye region was used to initialize an object EyeDetect. The video capturing was initially performed for the first 50 frames. The video was converted to individual frames using the getsnapshot() function which returns a matrix corresponding to an RGB image. The next step involved was similar to identifying the eye region in a static image, the difference being instead of the image being stored in the computer memory, it is stored virtually in a Matlab script. Since the getsnapshot() function works by contacting the webcam every time it is called, the processing time is increased. In order to minimize the time taken by the getsnapshot(), the triggerconfig() property of the video object was set to manual mode. The EyeDetect object is given as input to the step function along with the image and the values returned correspond to the X-Coordinate, Y-Coordinate, Width and Height of the eye region.

The image is then cropped using the imcrop() function with one input as the n cross 4 matrix and the other being the image itself. The RGB image thus obtained is first converted to its equivalent grayscale form using the rgb2gray() function. This is followed by converting the thus obtained gray scale image to its black and white form using the im2bw() function.

3.5 Hardware Specifications

The next stage of any image acquisition system must convert the video signal into a format, which can be processed by a computer. The common solution is a frame grabber board that attaches to a computer and provides the complete video signal to the computer. The resulting data is an array of greyscale values, and may then be analysed by a processor to extract the required features. Two options were investigated when choosing the system's capture hardware. The first option is designing a homemade frame grabber, and the second option is purchasing a commercial frame grabber.

To take the values out from the code and implement it into the prototype, we have used the Arduino board for the testing of the Intex webcam 3.1with inbuilt LED lights. To implement the code into the board. One has to understand the working and the specifications of the Arduino Board. A content table is given to access and engineer the prototype

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

4. Results

4.1 Eye Detection using the following algorithms:

4.1.1 Valley Detection Method.

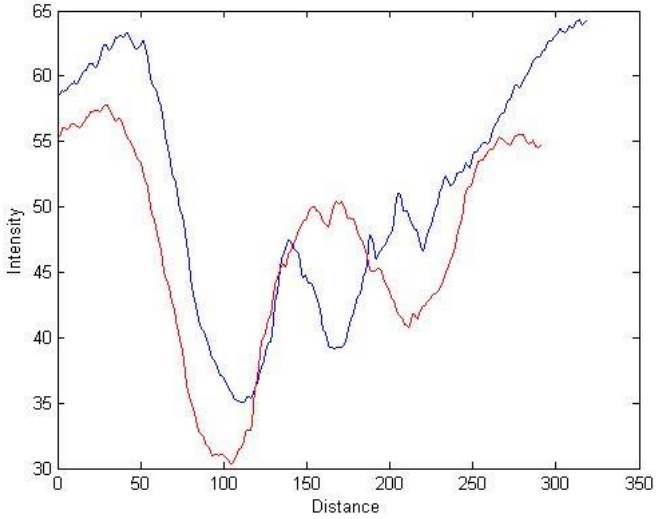
The first method approached included a complex mathematical interpretation where the space between the eyelashes and the eyebrows. The space between the eyebrows and the eyelashes is the giveaway for the detection for the position of the eye. As shown in the figure 1, the values of the valleys can be sued to see how the code will trigger the corresponding output. For example, the condition where the eye is closed, the red curve shows that the difference between the peak 1 and peak 2 is 112 and correspondingly, the difference between the peak 1 and peak 3 is 44. This is merely the measure of the intensity on the face where the presence of colour black, that is perceived as facial hair on the surface gives the dip in the intensity curve as clearly seen from the graph plotted.

(a)

(b)

The intensity curve for the image (a) corresponds to the blue line in the graph shown below and the intensity curve for the image (b) corresponds to the red line in the same graph.

The thing to be understood here is that the dips in the curve give two different values which are then used to get the state of the eye. The problem with coding singularly with the lone values is that these values are environment sensitive. Once any change in the background or in the facial surface is detected, the two values converge to a difference of 20 units and it becomes difficult to code with those values.



(c)

As we are plotting the intensity on a real time image extracted from the video feed, eye exhibits a very stable response when we use greyscale imaging. The intensity of the local region of the eye (the eyebrow, iris and the eyelashes). The iris exhibits a very dark circle (i.e. low intensity) which is used to detect the coordinates of the centre of the eye. Under bright lit conditions, this method serves best for the detection of the eye.

Since the driver uses the proposed module at night times when one feels drowsy, the calibration with regard to the intensity isn't the best fit. The method of detection of the state of the eye based on the intensity variation will give errors.

A binary image is an image in which each pixel assumes the value of only two discrete values. In this case the values are 0 and 1, 0 representing black and 1 representing white. This process is done for the left and right side of the face separately, and then the found eye areas of the left and right side are compared to check whether the eyes are found correctly. Calculating the left side means taking the averages from the left edge to the centre of the face, and similarly for the right side of the face. With the binary image it is easy to distinguish objects from the background. The grey scale image is converting to a binary image via thresholding.

(a) (b) (c)

(d) (e) (f)

There were 3 instances where the images were converted to greyscale to get the difference in intensities at the Specified region of interest. Image (a) was taken under brightly lit conditions using a CFL in the room of 100W power. This images a clear distinctive margin of the eyes open state. As we proceed to image (b) and image (c), the intensity decreases and the clarity of the grayscale image decreases at the specified region of interest. This is the major drawback of using this method to detect the difference in the states of the eye. Same is the situation with the set of images (d) (e) (f). As we proceed from (d) to (f), the intensity of the environment decreases and the ease with which we detect the state of the eys becomes difficult for a single piece of code.

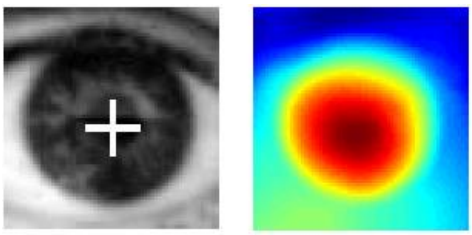
Criteria for judging the alertness level on the basis of eye closure count is based on the amount of intensity levels of the image. The binaryized values are thoroughly analysed and synthesized by the algorithm and detected if the driver eye caught in the image is an open eye or closed eye. Thus if the eye is open the hardware produces no effect but when the driver's closed eye is detected the corresponding signal goes high producing an alarm which wakes up the driver.

4.1.2 Local Binary Pattern using spatial pyramids

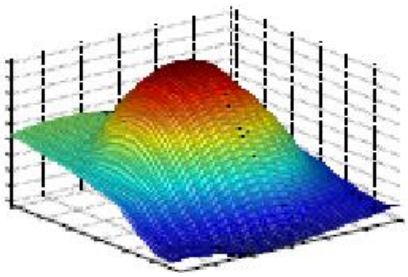
We start by summarizing the main common steps of the algorithms used in this work. Then we describe each step in detail. The proposed face recognition process consists of
Four main parts:

1. Pre-processing:
   We begin by applying the Tan and Triggs' illumination normalization algorithm to compensate for illumination variation in the face image. No further pre-processing, such as face alignment, is performed
2. LBP Operator Application :
   In the second stage LBP are computed for each pixel, creating a fine scale textural description of the image.
3. Local feature extraction:
   Local features are created by computing histograms of LBP over local image regions.
4. Classification:
   Each face image in test set is classified by comparing it against the face images in the training set. The comparison is performed using the local features obtained in the previous step.

.



(a)



(b)

The problem with using LBP with spatial pyramid matching is that the iris is detected in rgb, owing to which manipulations regarding the state of the eye becomes difficult. The lighting becomes a problem again, intensity matched with the image in the database is very insignificant and the values to code become difficult.

Intensity matching is a problem encountered throughout the prototypes designed to detect the drowsiness. And apart from that, LBP gives a result out of histogram matching which cannot

be taken as a standard to detect the state of the eye. And for a driver whose uses spectacles, it is even worse.

Geometrically, the centre of a circular object can be detected by analysing the vector field of image gradients, which has been used for eye centre localisation previously. Kothari and Mitchell, for example, proposed a method that exploits the flow field character that arises due to the strong contrast between iris and sclera. They use the orientation of each gradient vector to draw a line through the whole image and they increase an accumulator bin each time one such line passes through it. The accumulator bin where most of the lines intersect thus represents the estimated eye centre. However, their approach is only defined in the discrete image space and a mathematical formulation is missing. Moreover, they don't consider problems that arise due to eyebrows, eyelids, or glasses.

Face images can be seen as a composition of micro-patterns which can be effectively described by the LBP histograms. A LBP histogram computed over the whole face image encodes only the occurrences of the micro-patterns without any indication about their locations. To also consider shape information of faces, face images were equally divided into small sub-regions to extract LBP histograms. The LBP features extracted from each sub-region are concatenated into a single, spatially enhanced feature histogram. The extracted feature histogram represents the local texture and global shape of face images. Since some regions of faces may take more important information then other regions, it is possible to assign weight to each region, depending on its importance for recognition.

4.1.3 Viola Jones Algorithm.

A face detector has to tell whether an image of arbitrary size contains a human face and if so, where it is. One natural framework for considering this problem is that of binary classification, in which a classifier is constructed to minimize the misclassification risk. Since no objective distribution can describe the actual prior probability for a given image to have a face, the algorithm must minimize both the false negative and false positive rates in order to achieve an acceptable performance. This task requires an accurate numerical description of what sets human faces apart from other objects. It turns out that these characteristics can be extracted with a remarkable committee learning algorithm called Adaboost, which relies on a committee of weak classifiers to form a strong one through a voting mechanism. A classifier is weak if, in general, it cannot meet a predefined classification target in error terms.

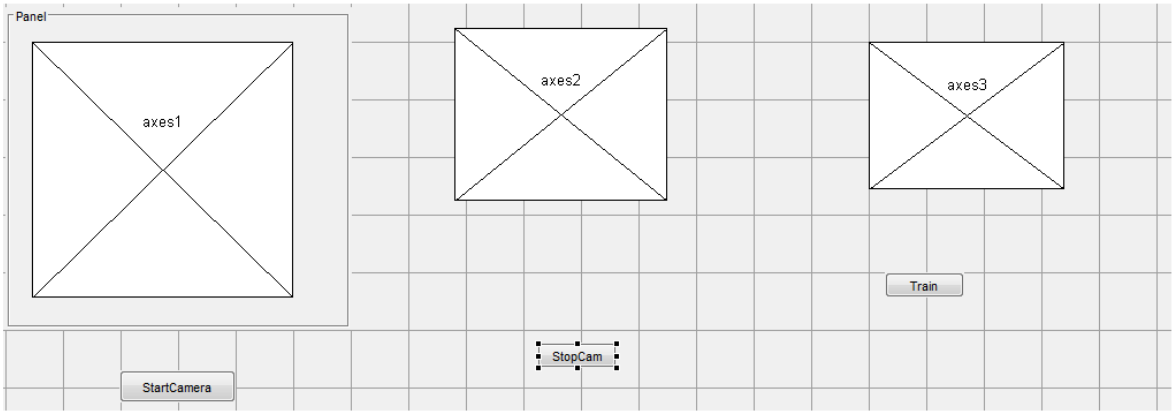
An operational algorithm must also work with a reasonable computational budget. Techniques such as integral image and attentional cascade make the Viola-Jones algorithm highly efficient. It is fed with a real time image sequence generated from a standard webcam, it performs well on a standard PC.

The first step is to convert the image into an integral image, this is similar to the Local Binary Pattern as we did earlier, but here we take a specific region of interest to get the grid.

As stated in the literature survey, Voila Jones algorithm works well even in low lit conditions. As evident from the picture below, the intensity is low on the face and yet the state of the eye is easily detected. The red bounded box tell when the eye is closed and the blue bounded box tell when the eyes are open. The decision of alarm sounding can be applied to the code in case the implementation. The colour coded boxes are shown just to give a proper understanding the working of the algorithm.

Haar-like features. Here as well as below, the background of a template like (b) is painted grey to highlight the pattern's support. Only those pixels marked in black or white are used when the corresponding feature is calculated.

Since I have used the image acquisition toolbox, its best fit once I made the Graphical User Interface and made corresponding push buttons trigger each of the code sections in the GUI code.



The starting of the camera will trigger the image acquisition, the push button “train” is used to set the detection of the eyes code into execution and capture the frames as mentioned in the code. The output of the code was taken with the low intensity of the lights to ensure the Viola Jones algorithm based on ROI methods was best served in case of the low intensity conditions; as is the case in the real life scenarios.

As stated in the literature survey, Voila Jones algorithm works well even in low lit conditions. As evident from the picture below, the intensity is low on the face and yet the state of the eye is easily detected. The red bounded box tell when the eye is closed and the blue bounded box tell when the eyes are open. The decision of alarm sounding can be applied to the code in case the implementation. The colour coded boxes are shown just to give a proper understanding the working of the algorithm.

As seen in the pictures of the output, the red rectangle (bbox function) is used to tell the subject that the eye is in a closed state. Mechanism to trigger the weaning signal was generated. Simple code to generate the warning signal can be coded in matlab using Image Acquisition Tool. The kind of sound that is produced as a warning signal can be changed

The working of the drowsiness system where the closed state gives off a red Bounded Box and the open eye gives a blue Bounded Box as shown below:

4.2 Limitations

1. Dependence on ambient light: - In the prototype, we have a situation where the face is detected with a minimum threshold but the eyes become a bit difficult to detect. The result obtained is an erroneous one, so there has to be a mechanism to fix it. The best thing we can do is output LED lights to light up the face for detection or use an infrared camera.

2. Optimum range required: we assume the distance from the camera to the face to be 100 cms. So the prototype will vary in results for different cars.
The basic problem with the distance from the prototype is that different cars have different dash board lengths from the seat. Generally it is taken to be 100cms. When the face is to close to the dashboard say about 30cms, the system will not detect or rather is unable to detect the region of interest. So I had to calibrate the image distance from the webcam to the face to 70-90cms.

This problem was resolved using the function: vision.CascadeObjectDetect to get the complete image of the face instead of taking just the eyes. So in case there face is not recognised, the eyes are detected. When the face is away from the calibrated distance that is 70cms, the detection system will give error. This problem will be a matter of concern when we use the same values for different cars because the distance from the camera and driver will differ. As checked in Ritz, the distance did not exceed 50cms and this issue is just a tiny fragment of the real problems faced.

3. Hardware requirements: - The code was run on a system with 2.3 GHz processor with 4GB ram Pentium dual core processor.
The prototype can be designed with systems with configurations as minimum as 2GB ram with a dual core processor. The only issue will be that the detection of the region of interest will be slow. The detection will include some sort of error such as displacement of the region selected.

4. Poor detection with spectacles: - This problem has been there for almost all the prototypes that have been designed to detect the drowsiness of the driver. The glare coming off from the spectacles I an issue that is yet to be resolved. The closest one can guess to make this right is to use infrared cameras.

5. Problem with multiple faces: - we have to make the window of input very specific to the driver. In case there are multiple faces then the detection system gives an error with the

accuracy. The speed with it executes the algorithm, slows down and hence affecting the performance.

4.3 Future work:

We have developed a system where we can detect the state of the eye and give warning signal. Now this is an adjustable mechanism which drives through the state of the driver. Meaning, the driver has to listen to the detection system and the warning signal and then slow down the vehicle or get back to proper driving. Instead, one can design a prototype which slows down the vehicle once the drowsy state of the driver is identified. Parameter that can be put will not include just the drowsiness but the parameter considered will be fatigue. This fatigue will include the stress level, mental state and the cardio vascular variables. Once these results are combined to get a binary yes or no to the question of weather or no, the driver is in a state to drive, the car will automatically respond in the hydraulic systems.

Another thing that can be achieved without the headache of the hardware implementation is to make a mobile app which can be mounted on the dashboard of the car and it will do the same thing that we designed by taking images from the front camera of the mobile and pretty much do the same thing what this extended combination of code and hardware will do. The good thing about perusing this is that the app will be accountable to power consumption and efficiency. Its rather to make an intelligent software than make an intelligent hardware.

4.4 Conclusion

Thus I have designed a system for the detection of the state of the eye of the driver using Image Acquisition Tool in Matlab. Algorithm used to implement the Drowsiness detection system was Viola Jones, for reasons stated through the course of this thesis. There are a lot of algorithms used to detect the facial features for the drowsiness system, each of which has its own set of limitations. I have used 3, accounting to the limitations and overcoming them with a hybrid model to detect the eye pair. Viola Jones algorithm serves best for the system to detect the eyes under low light conditions and can also detect the facial features when the head is a little bit tilted. The system so developed was successfully tested, its limitations identified and a future plan of action developed.

5. References:

[1] http://in.mathworks.com/

[2] Patent reference: US 7864990 B2**.**

[3] Eriksson, M and Papanikolopoulos, N.P. "Eye-tracking for Detection of Driver Fatigue", IEEE Intelligent Transport System Proceedings.

[4] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[5] An Analysis of Viola Jones algorithm for face detection by Yi-Quin Wang, University of Malaysia Phang, 2014, pp: 15-20.

[6] Implementation of Voila Jones Algorithm by Ole Helvig Jensen, university of Denmark, 2008, pp: 20-36

[7] Face Recognition with local binary patterns. Daniel Maturana, Domingo Mery, 2010.

[8] Accurate Eye Detection, Qui Chen, Koji Kotani, Feifei Lee and Tadahiro Ohmi, 2009, University of Tohoku, japan

[9] Application of Local Binary Patterns to Facial Recognition Problems, Vladimir Petruk, Bauman Moscow State Technical University.

[10] Drowsiness detection for car assisted driver system using image processing analysis. Huong Nice Quan, 2010.