DROWSY DRIVER DETECTION SYSTEM

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

Electronics & Communication Engineering

under the supervision of

Mr. Kaushlendra Kumar Pandey

By

Ashish Singh(111106)

Ashwin Singh(111101)

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled “Drowsy Driver Detection System.”, submitted by Ashish Singh and Ashwin Singh in partial fulfillment for the award of degree of Bachelor of Technology in Electronics & Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Supervisor’s Name : Mr. Kaushlendra Kumar Pandey

Designation : Assistant Professor

Date :

ACKNOWLEDGMENT

First and foremost, We would like to thank God for blessing us with the strength, intelligence, and patience to complete this project. We would also like to thank our Project Guide Mr. Kaushlendra Kumar Pandey for his continuing support, who has been an extraordinary mentor.

Date: Ashwin Singh

Ashish Singh

Abstract

Driver fatigue is one of the major causes of accidents in the world. Detecting the drowsiness of the driver is one of the surest ways of measuring driver fatigue. In this project we aim to develop a prototype drowsiness detection system. This system works by monitoring the eyes of the driver and sounding an alarm when he/she is drowsy.

The priority is on improving the safety of the driver without being obtrusive. In this project the eye blink of the driver is detected. If the drivers eyes remain closed for more than a certain period of time, the driver is said to be drowsy and an alarm is sounded.

Taking into account the knowledge that eye regions in the face present great intensity changes, the eyes are located by finding the significant intensity changes in the face. Once the eyes are located, measuring the distances between the intensity changes in the eye area determine whether the eyes are open or closed. A large distance corresponds to eye closure. If the eyes are found closed for 5 consecutive frames, the system draws the conclusion that the driver is falling asleep and issues a warning signal.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

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 a sequence of images of a face, and the observation of eye movements and blink patterns. The analysis of face images is a popular research area with applications such as face recognition, virtual tools, and human identification security systems. This project is focused on the localization of the eyes, which involves looking at the entire image of the face, and determining the position of the eyes by a self developed image-processing algorithm. Once the position of the eyes is located, the system is designed to determine whether the eyes are opened or closed, and detect fatigue.

1.1 Objective

To develop a prototype drowsiness detection system to monitor the open or closed state of the driver's eyes. The purpose of this study is to detect drowsiness in drivers to prevent accidents and to improve safety on the highways.

A method for detecting drowsiness/sleepiness in drivers is developed by using a camera that point directly towards the driver's face and capture for the video. Once the video is captured, monitoring the face region and eyes in order to detect drowsy/fatigue.

1.2 Technical Details

Software Used:

Matlab 2007:
Matlab is used because it is very helpful for image processing. It provides special function like RGB2GRAY to convert rgb image to Gray Scaled Image. It provides functions for image compression and image enhancement. It also provides a special kind of eye detection function.Licensed version of Matlab 2007 has been provoded in our project lab.

Hardware Used:

A non-intrusive monitoring system that will not distract the driver.

Integrated Laptop Webcam:
Dell Laptop integrated Webcam has been used, but Practically We have to use a day and night camera, which can capture the images at the night also.
HDD Memory 512 Mb (min):
We should have at least 512 mb memory to store continuously captures image
RAM 512 mb:
The processor should have at least 512mb Ram to process the images and at the end to detect the drowsiness and process the alarm.

Other requirements:

Good amount of light:

This is also the main requirement of our project, without having efficient light we will not able to get a good image, and further we'll not be able to get a threshold value for binarization.

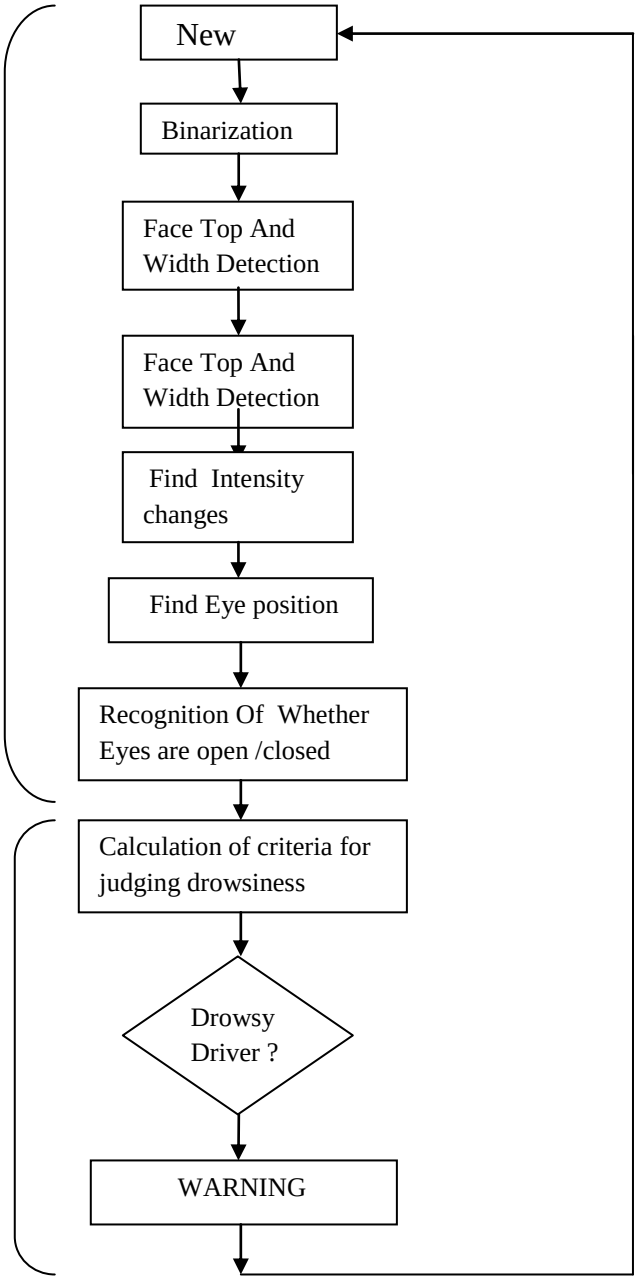
CHAPTER 2

METHODOLOGY OF FIRST ALGORITHM

2.1 System Flowchart:

EYE DETECTION FUNCTION

DROWSINESS DETECTION FUNCTION



(Fig.1)

2.2 System Process

2.2.1 Eye Detection Function

After inputting a facial image, pre-processing is first performed by binarizing the image.

The top and sides of the face are detected to narrow down the area of where the eyes exist. Using the sides of the face, the centre of the face is found, which will be used as a reference when comparing the left and right eyes.

Moving down from the top of the face, horizontal averages (average intensity value for each y coordinate) of the face area are calculated. Large changes in the averages are used to define the eye area. The following explains the eye detection procedure in the order of the processing operations. All images were generating in Matlab using the image processing toolbox.

2.2.2 Binarization

The first step to localize the eyes is binarizing the picture. Binarization is converting the image to a binary image.

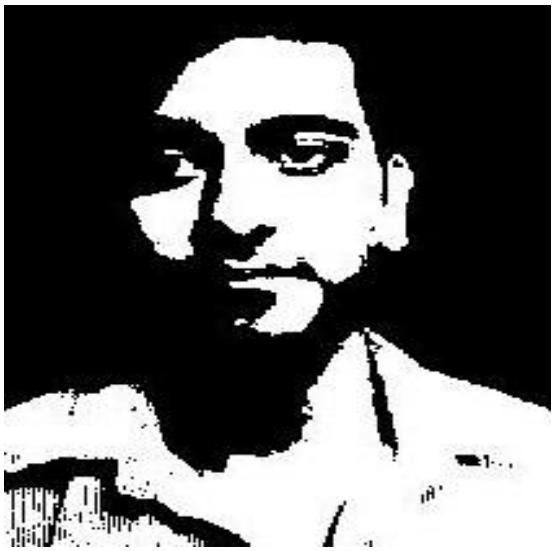
RGB Image (Fig.1.1)

Gray Scaled Image (Fig.1.2)

2.2.3 Thresholding:

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. With the binary image it is easy to distinguish objects from the background. The greyscale image can be converted to a binary image via thresholding. The output binary image has values of 0 (black) for all pixels in the original image with luminance less than level and 1 (white) for all other pixels. Thresholds are often determined based on surrounding lighting conditions, and the complexion of the driver. After observing many images of different faces under various lighting conditions a threshold value of 83 was found to be effective.

The criteria used in choosing the correct threshold was based on the idea that the binary image of the driver's face should be majority white, allowing a few black blobs from the eyes, nose and/or lips, demonstrates the effectiveness of varying threshold values.



At Threshold = 100 (Fig.1.3)

At threshold level= 83 (Fig .1.4)

2.2.4 Face Top and Width Detection

The next step in the eye detection function is determining the top and side of the driver's face. This is important since finding the outline of the face narrows down the region in which the eyes are, which makes it easier (computationally) to localize the position of the eyes. Requirement is to find the top of the face. So the first step is to find a starting point on the face, followed by decrementing the y-coordinates until the top of the face is detected. Assuming that the person's face is approximately in the centre of the image, the initial starting point used is (100,240). The starting x-coordinate of 100 was chosen, to insure that the starting point is a black pixel (no on the face). The following algorithm describes how to find the actual starting point on the face, which will be used to find the top of the face.

1. Starting at (x0, y0), increment the x-coordinate until a white pixel is found. This is considered the left side of the face.
2. If the initial white pixel is followed by 25 more white pixels, keep incrementing x until a black pixel is found.
3. Count the number of black pixels followed by the pixel found in step2, if a series of 25 black pixels are found, this is the right side.
4. The new starting x-coordinate value (x1) is the middle point of the left side and right side.

Using the new starting point (x1, 240), the top of the head can be found. The following is the algorithm to find the top of the head:

1. Beginning at the starting point, decrement the y-coordinate (i.e.; moving up the face).
2. Continue to decrement y until a black pixel is found. If y becomes 0 (reached the top of the image), set this to the top of the head.
3. Check to see if any white pixels follow the black pixel.

   If a significant number of white pixels are found, continue to decrement y.
   If no white pixels are found, the top of the head is found at the point of the initial black pixel.

Once the top of the driver's head is found, the sides of the face can also be found. Below are the steps used to find the left and right sides of the face.

1. Increment the y-coordinate of the top (found above) by 10. Label this y1 = y + top.

2. Find the centre of the face using the following steps:

   i. At point (x1, y1), move left until 25 consecutive black pixels are found, this is the left side (lx).
   ii. At point (x1, y1), move right until 25 consecutive white pixels are found, this is the right side (rx).
   iii. The centre of the face (in x-direction) is: (rx – lx)/2. Label this x2.

3. Starting at the point (x2, y1), find the top of the face again. This will result in a new y-coordinate, y2.

Finally, the edges of the face can be found using the point (x2, y2).

i. Increment y-coordinate.

ii. Move left by decrementing the x-coordinate, when 5 black consecutive pixels are found, this is the left side, add the x-coordinate to an array labelled ‘left_x’.

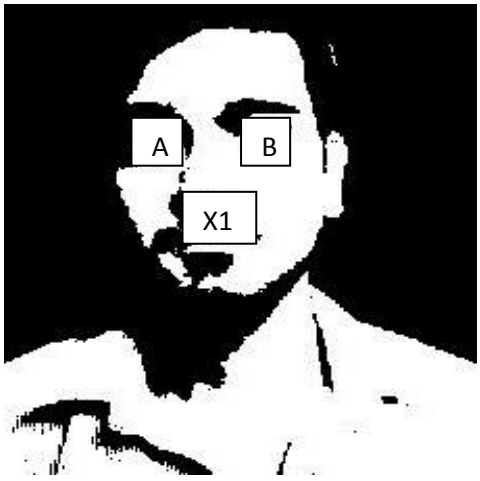
iii. Move right by incrementing the x-coordinate, when 5 black consecutive pixels are found, this is the right side, add the x-coordinate to an array labelled ‘right_x’.

iv. Repeat the above steps 200 times (200 different y-coordinates).

2.2.5 Finding Intensity Changes on the Face

The next step in locating the eyes is finding the intensity changes on the face. This is done using the original image, *not* the binary image. The first step is to calculate the average intensity for each y – coordinate. This is called the horizontal average, since the averages are taken among the horizontal values. The valleys (dips) in the plot of the horizontal values indicate intensity changes. When the horizontal values were initially plotted, it was found that there were many small valleys, which do not represent intensity changes, but result from small differences in the averages. To correct this, a smoothing algorithm was implemented. The smoothing algorithm eliminated and small changes, resulting in a more smooth, clean graph.

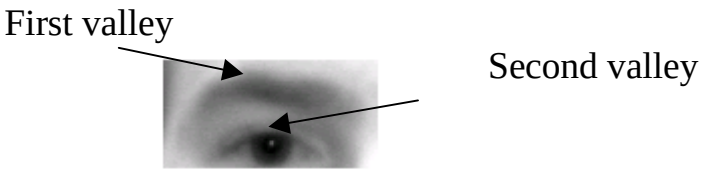
After obtaining the horizontal average data, the next step is to find the most significant valleys, which will indicate the eye area. Assuming that the person has a uniform forehead (i.e.; little hair covering the forehead), this is based on the notion that from the top of the face, moving down, the first intensity change is the eyebrow, and the next change is the upper edge of the eye.



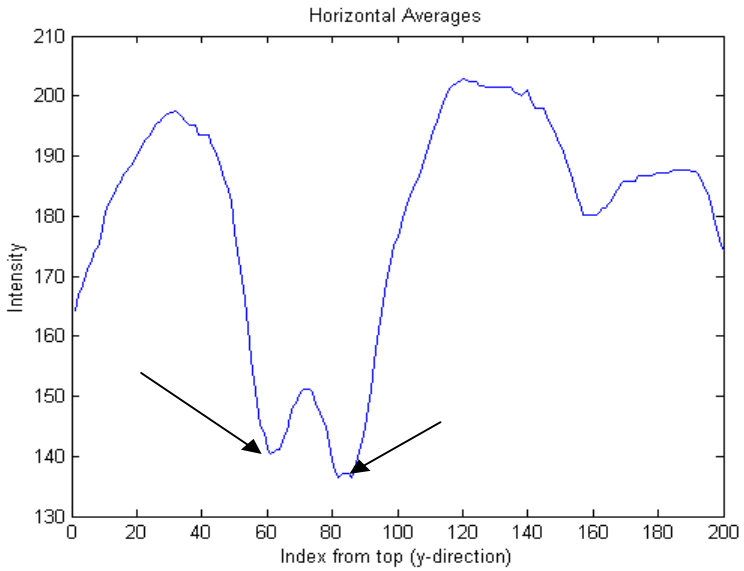
X1 is the middle point of A and B
(Fig.1.5)

2.2.6 Detection of Vertical Eye Position

The first largest valley with the lowest y – coordinate is the eyebrow, and the second largest valley with the next lowest y-coordinate is the eye.



Reference no. 3



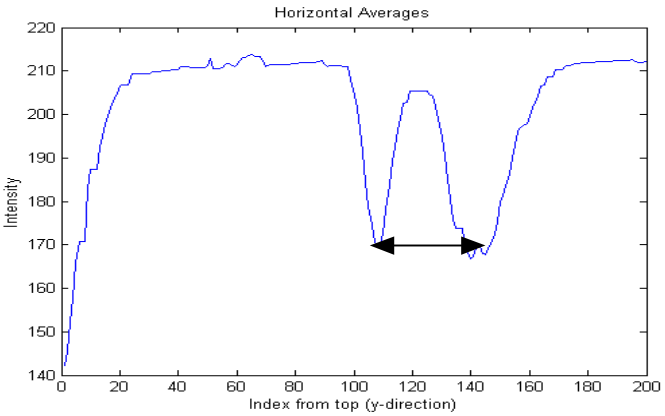
Reference no. 3

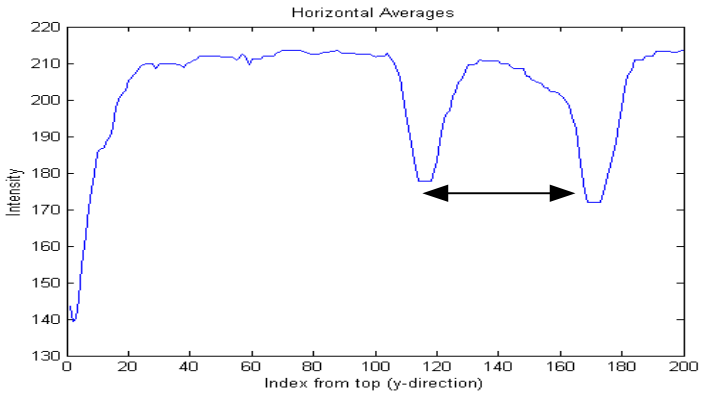
This process is done for the left and right side of the face separately, and then the found eye areas of the left and right side are compared to check whether the eyes are found correctly. Calculating the left side means taking the averages from the left edge to the centre of the face, and similarly for the right side of the face. The reason for doing the two sides separately is because when the driver's head is tilted the horizontal averages are not accurate.

2.2.7 Drowsiness Detection Function

Determining the State of the Eyes

The state of the eyes (whether it is open or closed) is determined by distance between the first two intensity changes found in the above step. When the eyes are closed, the distance between the y – coordinates of the intensity changes is larger if compared to when the eyes are open.





Images and Graphs reference 3

2.2.8 Judging Drowsiness

When there are 5 consecutive frames find the eye closed, then the alarm is activated, and a driver is alerted to wake up. Consecutive number of closed frames is needed to avoid including instances of eye closure due to blinking.

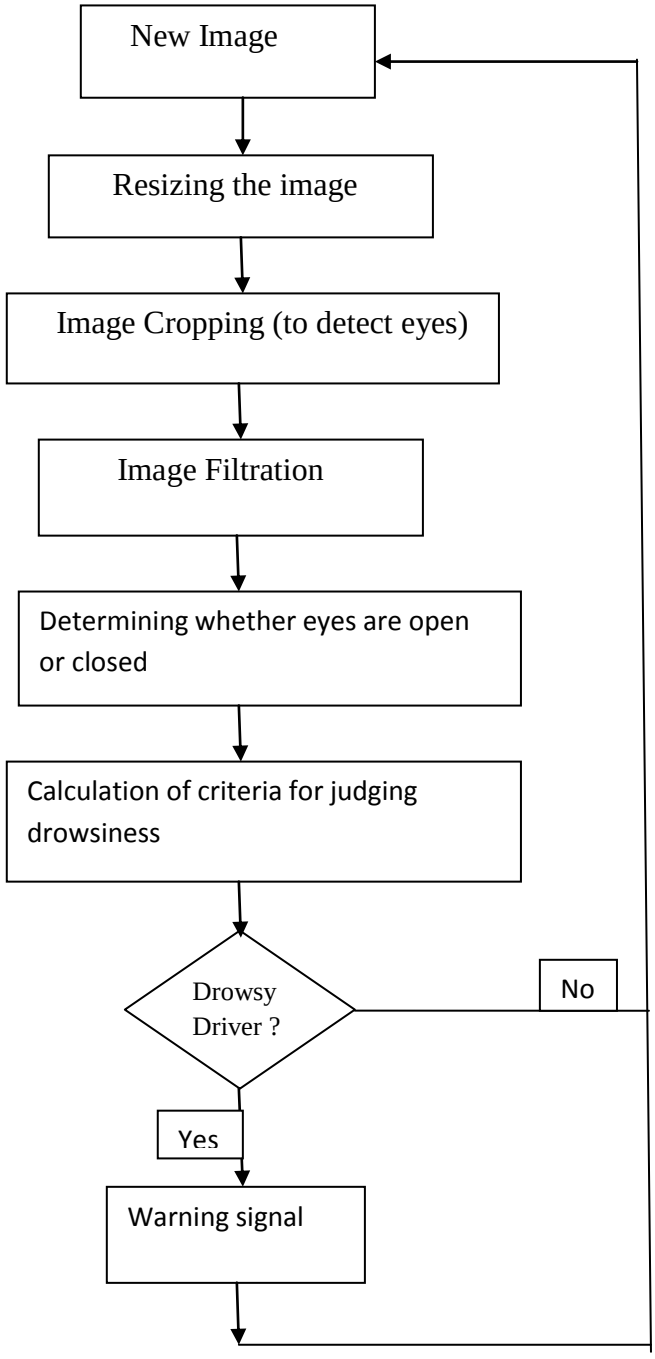
Work done using First algorithm:

- Algorithm has been developed.
- Image has been captured using webcam
- RGB Image is converted into binary image via thresholding.
- Face and edge detected.

CHAPTER 3

METHODOLOGY OF SECOND ALGORITHM

3.1 Flowchart:



(Fig . 2)

3.2 System Process

3.2.1 Obtaining the image:

Using integrated webcam image would be obtained. Most of the part of the image background should be black for the easy filtration process. Obtained image would be RBG image. Some inbuilt command of MATLAB would be used to obtain an image. Like ‘videoinput’ command is used to start the webcam and ‘getsnapshot’ is used to capture the image.
Face of the person should be in centre of the image. There should be some led facing towards face of the person to get a proper illuminated image.
Some facial recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face recognition.

Original Captured Image

(Fig. 2.1)

3.2.2 Resizing the image:

Obtained image would be resized into 200*200 size. Reducing images is a completely safe and rational operation. You're simply reducing precision and resolution by discarding information. Make the image as small as you want, and you have complete fidelity-- within the bounds of the number of pixels you've allowed. You'll get good results no matter which algorithm you pick.

Enlarging images is risky. Beyond a certain point, enlarging images is a fool's errand; you can't magically synthesize an infinite number of new pixels out of thin air. And interpolated pixels are never as good as real pixels.

Resized Image

(Fig. 2.2)

3.2.3 Image Cropping:

Cropping refers to the removal of the outer parts of an image to improve framing, accentuate subject matter or change aspect ratio. Depending on the application, this may be performed on a physical photograph, artwork or film footage, or achieved digitally using image editing software.

Some part of the image would be cropped where eyes would exist. According to a specific coordinates cropping would be done. And the coordinates would be [50 80 100 20].by doing this, we'll get the eyes from the obtained image. Cropping has done using MATLAB command "imresize". Where [50, 80] is the starting point from where cropping would take place and [100,20] is the width and height to be cropped.

Cropped Image

(Fig. 2.3)

3.2.4 Image filtration:

A filter is a device or process that removes from a signal some unwanted component or feature. Filtering is a class of signal processing, the defining feature of filters being the complete or partial suppression of some aspect of the signal

Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement.

Filtering is a neighbourhood operation, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighbourhood of the corresponding input pixel. A pixel's neighbourhood is some set of pixels, defined by their locations relative to that pixel.

Through masking filtration is used. Filtration is used to determine eyebrows and eyelashes, and the larger valued pixels in the cropped image.

Filtered Image

(Fig. 2.4)

3.2.5 Judging drowsiness:

Webcam would capture 10-12 images in 2 to 3 seconds and those images would be processed and if 5 to 6 image in a row shows that eyes are closed, a warning signal would get generated. If 5 to 6 images in row does not show that eyes are closed then, that data set would get discarded by the processor and new image would get captured.

CHAPTER 4

CHALLENGES

- **Obtaining the image**

  The first, and probably most significant challenge faced was transferring the images obtained from the camera to the computer. Two issues were involved with this challenge: 1) Capturing and transferring in real time; 2) Having access to where the image was being stored.

- **Determining the correct binarization threshold.**

  Because of varying facial complexions and ambient light, it was very hard to determine the correct threshold for binarization. The initial thought was to choose a value that would result in the least amount of black blobs on the face. After observing several binary images of different people, it was concluded that a single threshold that would result in similar results for all people is impossible.

- **An effective light source.**

  Illuminating the face is an important aspect of the system. This is why we are currently only working on properly illuminated face.

- **Position of driver's head.**

  This is the major problem for us that ,if driver 's face is tilted then how we'll find the driver's face position, And If we cannot detect the driver's face position we'll not be able to fund the driver's correct eye position.

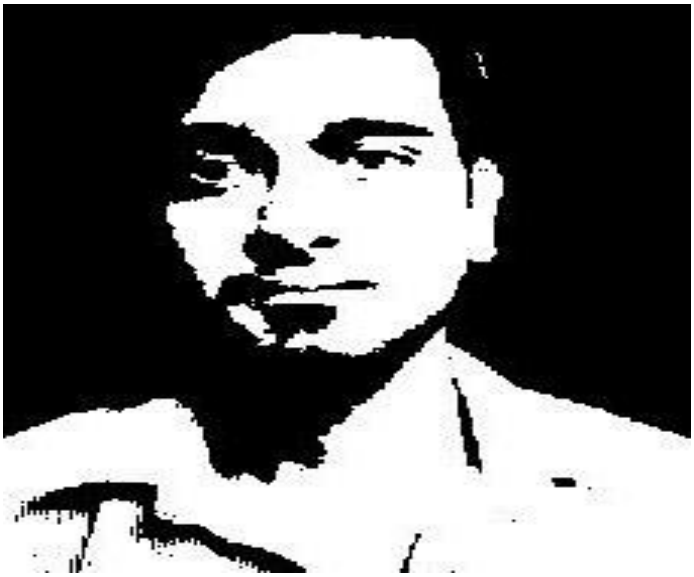
- **Person is wearing glasses.**

  This one also challenge for us. If driver is wearing sun glasses or spectacles the how we'll find whether eyes are closed or open.

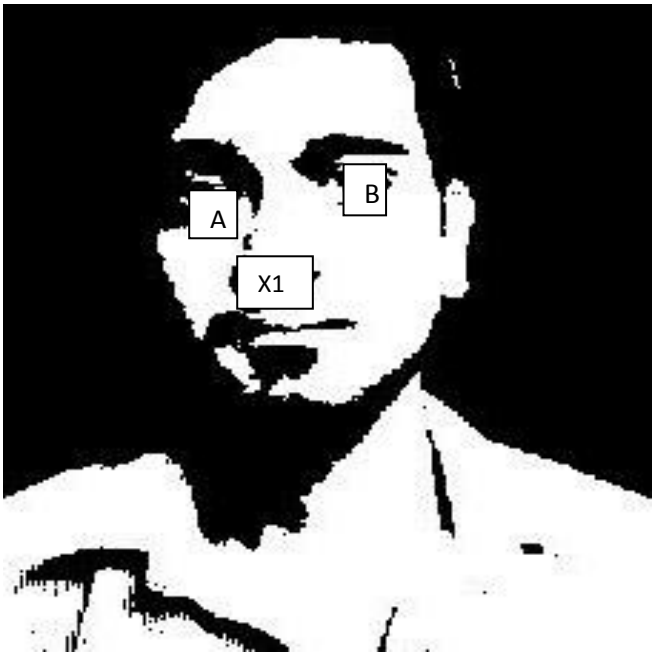
RESULTS

➢ From the First Algorithm:

Gray Scaled image



Binarised Image: At threshold level 83



Edge Detected Image

➢ From the Second Algorithm:

Original captured image

Resized Image

Cropped Image

Filtered Image

FUTURE WORK

Currently there is not adjustment in zoom or direction of the camera during operation. Future work may be to automatically zoom in on the eyes once they are localized. This would avoid the trade-off between having a wide field of view in order to locate the eyes, and a narrow view in order to detect fatigue.

This system only looks at the number of consecutive frames where the eyes are closed. At that point it may be too late to issue the warning. By studying eye movement patterns, it is possible to find a method to generate the warning sooner.

The system does not work for dark skinned individuals. This can be corrected by having an adaptive light source. The adaptive light source would measure the amount of light being reflected back. If little light is being reflected, the intensity of the light is increased. Darker skinned individual need much more light, so that when the binary image is constructed, the face is white, and the background is black.

APPENDIX

CODE:

```
% To start the video input
st=videoinput('winvideo',1);

set(st, 'ReturnedColorSpace', 'RGB');

%To capture the image
dx=getsnapshot(st);

figure,imshow(dx);

impixelinfo;

%To resize the captured image
i1=imresize(dx,[200 200]);

figure,imshow(i1);

impixelinfo;

%Coverting RGB image into gray scaled image
ig=rgb2gray(i1);

figure,imshow(ig);

impixelinfo;

% Lossy Filters

l3=[1,2,1];

e3=[-1,0,1];

s3=[-1,2,-1];

f1=l3'*l3;

f2=((e3'*l3)+(l3'*e3))/2;

f3=((e3'*s3)+(s3'*e3))/2;

f4=((s3'*l3)+(l3'*s3))/2;

f5=e3'*e3;

f6=s3'*s3;
```

```
17=[1,6,15,20,15,6,1];

e7=[-1,-4,-5,0,5,4,1];

s7=[-1,-2,1,4,1,-2,-1];

g1=l7'*l7;

g2=((e7'*l7)+(l7'*e7))/2;

g3=((e7'*s7)+(s7'*e7))/2;

g4=((s7'*l7)+(l7'*s7))/2;

g5=e7'*e7;

g6=s7'*s7;

%Image Filtration
i2=imfilter(ig,f5);

%Image Cropping
i3=imcrop(i2,[50 80 100 20]);

figure,imshow(i3);

impixelinfo;

sum=0; count=0;

% Counting Pixels values to determine Drowsiness

for k=1:200

  for j=1:200

    if (i3(k,j)>=5)

     sum=sum+i3(k,j);

       count=count+1;

      end

   end

end
```

```
disp('Number of pixels greater than or equal to 11');
disp(count);

disp('sum');
disp(sum);
disp('Average');
Avg=sum/count;
disp(Avg);
```

REFERENCES

1. A.K.Jain, Fundamentals of Digital Image Processing, Prentice Hall.

2. S.Sridhar, Digital Image Processing, Oxford University Press.

3. Neeta Parmar "Drowsy Driver Detection System"Ryerson University.

4. Danisman, T**.;** LIFL, Univ. Lille 1, Lille, France ; Bilasco, I.M.; Djeraba, C. ; Ihaddadene, N. **"**Drowsy driver detection system using eye blink patterns"

5. "Driver Drowsiness Detection System and Techniques " Vandna Saini, Rekha Saini ; Assistant Professor, CSE Department