P-2850

DROWSINESS TRACKING SYSTEM

A PROJECT REPORT

Submitted By

Nivedha.P	**71205104025**
Sudha.K	**71205104055**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600025

APRIL 2009

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**DROWSINESS TRACKING SYSTEM**" is a bonafide work of "**NIVEDHA.P**" and "**SUDHA.K**" who carried out the project work under my supervision.

SIGNATURE

Dr.S.Thangasamy

DEAN

Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore-641 006

27/4/'09

SIGNATURE

Dr.S.Thangasamy

GUIDE

Dean,

Computer Science and Engineering

Kumaraguru College of Technology

Coimabatore-641 006

The candidates with university Register Nos **71205104025** and **71205104055** were examined by us in the project viva-voce examination held on 29.04.09

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

This exhilaration achieved upon the successful completion of any task should be definitely shared with all the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken its shape.

At the onset, we take this opportunity to thank the management of our college for providing us excellent facilities to work with. We express our sincere thanks to **Prof.R.Annamalai M.E.,** Vice Principal, Kumaraguru College of Technology for ushering us to the path of the triumph.

With great veneration and sincere gratitude we express our profound thanks to our beloved Dean and guide, **Dr.S.THANGASAMY Ph.D**, for his immense encouragement, invaluable guidance, ideas, suggestions and support for being a source of inspiration all through the course of study.

We would like to place on record our heartfelt gratitude to our revered project coordinator **Mrs.P.DEVAKI,** Assistant Professor, Department of Computer Science and Engineering for her critical remarks and suggestions.

We are very much obliged to express our sincere thanks to our class advisor **Mrs.D.CHANDRAKALA,**Asst.Professor, Department of Computer Science and Engineering for lending her helping hand in caring out our project successfully.

We also feel elated in manifesting our deep sense of gratitude to all the **staff and lab technician**s in the department.

We express our deep thanks to our friend Miss **B.Shobana** and all our **family and friends** who were sharing their contribution in many subtle ways and indeed instrumental for lending us tips, support and co-operation throughout the project work.

ABSTRACT

ABSTRACT

A Drowsiness Tracking System for an automobile driver has been developed, using a non-intrusive machine vision based concepts. The system uses a small monochrome security camera that points directly towards the driver's face and monitors the driver's eyes in order to detect eye fatigue. If drowsiness is confirmed, a warning signal is issued to alert the driver. This report describes how to locate the eyes, and also how to determine if the eyes are open or closed. The system deals with using information obtained from the binary version of the image to find the edges of the face. Taking into account the knowledge that eye regions in the face represent 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 eye brow and upper eye lid determine whether the eyes are open or closed. If the distance is greater than the certain threshold value eye closure is confirmed. If the eyes are found closed for 5 consecutive frames(each frame taken 2/5 second), the system draws the conclusion that the driver is falling asleep and issues a warning signal. If there is no response from the driver for more than 5 seconds the vehicle breaks automatically. The system is also able to detect when the eyes cannot be found, and works under reasonable lighting conditions.

CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

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 all over [9].

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.

SYSTEM REQUIREMENTS

2. SYSTEM REQUIREMENTS

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

- A non-intrusive monitoring system that will not distract the driver
- A real-time monitoring system, to ensure accuracy in detecting drowsiness
- A system that will work in both daytime and nighttime conditions
- An alarm for alerting the drowsiness

2.1 SOFTWARE REQUIREMENTS

MATLAB 7.5

LITERATURE REVIEW

3. LITERATURE REVIEW

3.1.TECHNIQUES FOR DETECTING DROWSINESS

Possible techniques for detecting drowsiness in drivers can be generally divided into the following categories: sensing of driver's physiological characteristics, sensing of driver operational patterns and monitoring the response of driver or the vehicle for specified signals.

3.2.MONOTORING PHYSIOLOGICAL CHARACTERISTICS

Among these methods, the techniques that are best, based on accuracy are the ones based on human physiological phenomena [9]. This technique is implemented in two ways: measuring changes in physiological signals, such as brain waves, heart rate, and eye blinking; and measuring physical changes such as sagging posture, leaning of the driver's head and the open/closed states of the eyes[9].

The first technique, while most accurate, is not realistic, since sensing electrodes would have to be attached directly onto the driver's body, and hence be annoying and distracting to the driver. In addition, long time driving would result in perspiration on the sensors, diminishing their ability to monitor accurately. The second technique is well suited for real world driving conditions since it can be non-intrusive by using optical sensors of video cameras to detect changes. So we go for the second technique of finding the open/closed state of the eyes.

3.3 OTHER METHODS

Driver operation and vehicle behaviour can be implemented by monitoring the steering wheel movement, accelerator or brake patterns, vehicle speed, lateral acceleration, and lateral displacement. These too are non-intrusive ways of detecting drowsiness, but are limited to vehicle type and driver conditions. The final technique for detecting drowsiness is by monitoring the response of the driver. This involves periodically requesting the driver to send a response to the system to indicate alertness. The problem with this technique is that it will eventually become tiresome and annoying to the driver.

DESIGN ISSUES

4. DESIGN ISSUES

The most important aspect of implementing a machine vision system is the image acquisition. Any deficiencies in the acquired images can cause problems with image analysis and interpretation. Examples of such problems are a lack of detail due to insufficient contrast or poor positioning of the camera: in which case the object cannot be recognized and hence purpose of vision cannot be fulfilled.

4.1 ILLUMINATION

A correct illumination scheme is a crucial part in ensuring that the image has the correct amount of contrast to correctly process the image. In the drowsiness detection system, the light source is placed in such a way that the maximum light being reflected back is from the face. The driver's face will be illuminated using a 60W light source. To prevent the light source from distracting the driver, an 850nm filter is placed over the source. Since 850nm falls in the infrared region, the illumination cannot be detected by the human eye, and hence does not agitate the driver. Since the algorithm behind the eye monitoring system is highly dependent on light, the following important illumination factors are to be considered[1]

1. Different parts of objects are lit differently, because of variations in the angle of incidence, and hence have different brightness as seen by the camera.

2. Brightness values vary due to the degree of reflections of the object.

3. Parts of the background and surrounding objects are in shadow, and can also affect the brightness values in different regions of the object.

4. Surrounding light sources (such as daylight) can diminish the effect of the light source on the object.

4.2 CAMERA HARDWARE

The next item to be considered in image acquisition is the video camera. Review of several journal articles reveals that the face monitoring systems uses an infrared-sensitive camera to generate the eye images [3],[5],[6],[7]. This is because of the fact that infrared light source is used to illuminate the driver's face. CCD cameras have a spectral range of 400-1000nm, and peak at approximately 800nm. The camera used in this system is a Sony CCD black and white camera. CCD camera digitizes the image from the outset, although in one respect – that signal amplitude represents light intensity.

SYSTEM CONFIGURATION

5. SYSTEM CONFIGURATION

5.1. BACKGROUND AND AMBIENT LIGHT

Because the eye tracking system is based on intensity changes on the face, it is crucial that the background does not contain any object with strong intensity changes. Highly reflective object behind the driver, can be picked up by the camera, and be consequently mistaken as the eyes. Since this design is a prototype, a controlled lighting area was set up for testing. Low surrounding light (ambient light) is also important, since the only significant light illuminating the face should come from the drowsiness system. If there is a lot of ambient light, the effect of the light source diminishes. The testing area included a black background, and low ambient light (in this case, the ceiling light was physically high, and hence had low Illumination).This setup is somewhat realistic since inside a vehicle, there is no direct light and the background is fairly uniform.

5.2 CAMERA

The drowsiness detection system consists of a CCD camera that takes images of the driver's face. This type of drowsiness detection system is based on the use of image processing technology that will be able to accommodate individual driver differences. The camera is placed in front of the driver, approximately 30 cm away from the face. The camera must be positioned such that the following criteria are met

1. The driver's face takes up the majority of the image.

2. The driver's face is approximately in the centre of the image.

The facial image data is in 480x640 pixel format and is stored as an array.

5.3 LIGHT SOURCE

For conditions when ambient light is poor (night time), a light source must be present to compensate. Initially, the construction of an infrared light source using infrared LED was going to be implemented. It was later found that at least 50 LEDs would be needed so as create a source that would be able to illuminate the entire face. To cut down cost, a simple desk light was used. Using the desk light alone could not work, since the bright light is blinding if looked at directly, and could not be used to illuminate the face. However, light from light bulbs and even daylight all contain infrared light; using this fact, it was decided that if an Infrared filter was placed over the desk lamp; this would protect the eyes from a strong and distracting light and provide strong enough light to illuminate the face. A wideband infrared filter was placed over the desk lamp, and provides an excellent method of illuminating the face.

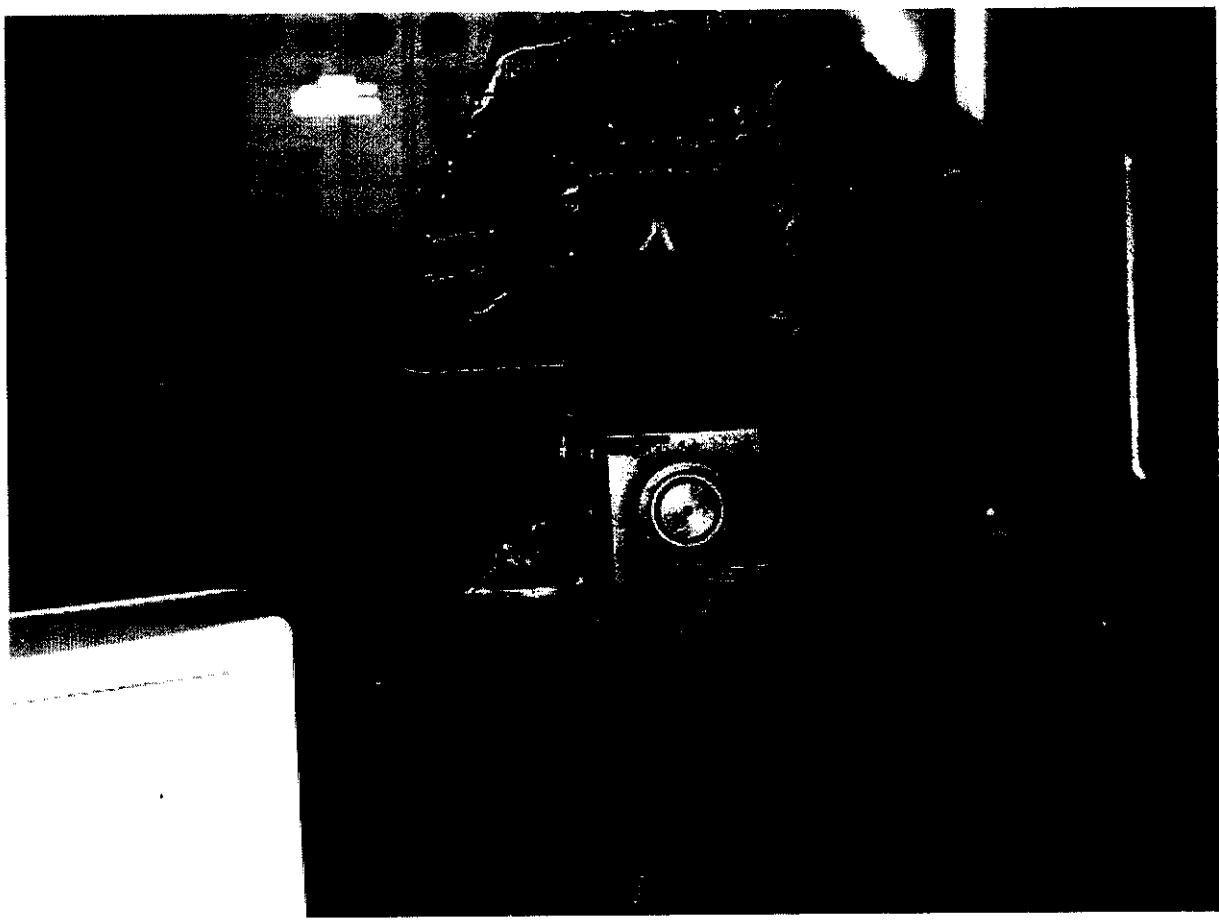


Figure 5.1 PROTOTYPE OF THE DROWSINESS DETECTION SYSTEM

ALGORITHM DEVELOPMENT

6. ALGORITHM DEVELOPMENT

6.1 EYE DETECTION FUNCTION

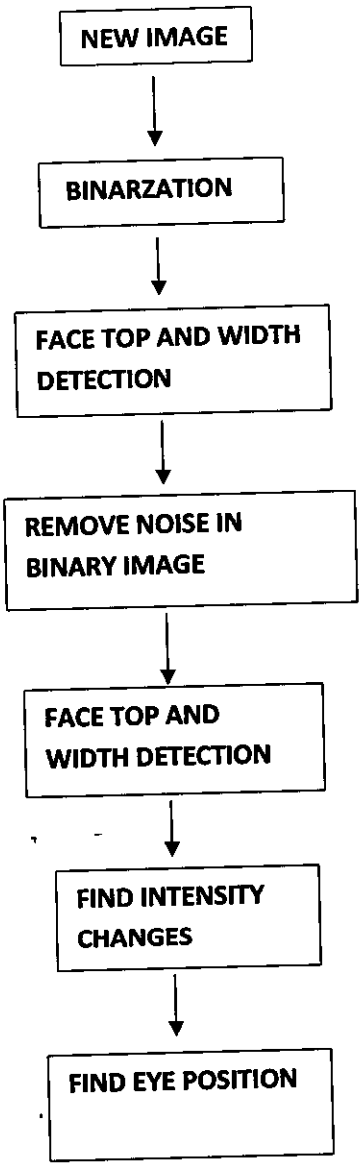
An explanation is given here of the eye detection procedure. After inputting a facial image, pre-processing is first performed by binarizing the image. The top and sides of the face are detected to narrow down the area of where the eyes exist. Using the sides of the face, the centre of the face is found, 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.

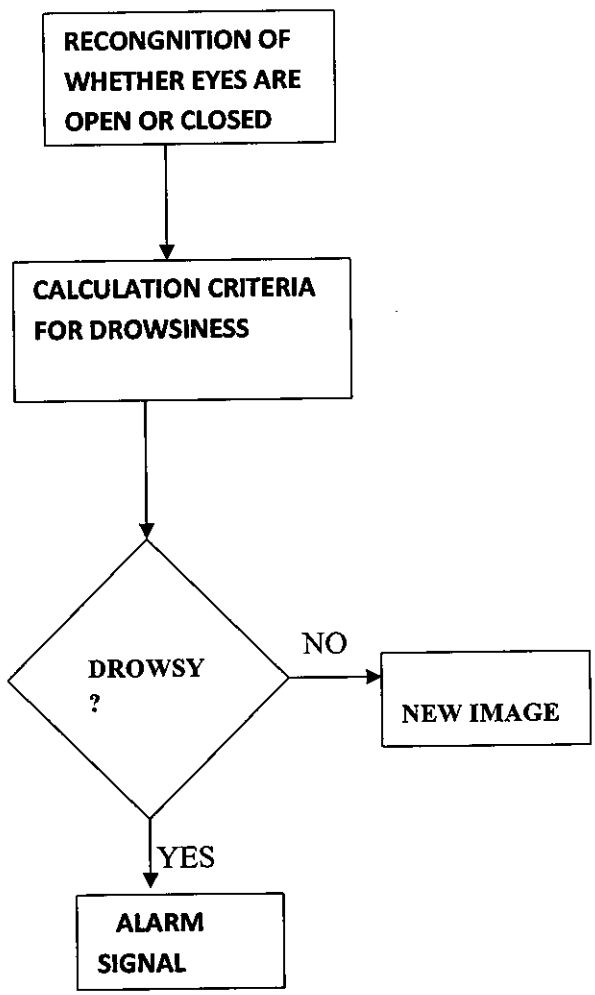
All images were generating in Matlab using the image processing toolbox.

6.2 SYSTEM FLOWCHART

6.2.1 EYE DETECTION FUNCTION



6.2.2 DROWSINESS DETECTION FUNCTION



BINARIZATION

7. BINARIZATION

The first step to localize the eyes is binarizing the picture. Binarization is converting the image to a binary image. The first step to localize the eyes is binarizing the picture. Binarization is converting the image to a binary image. 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 grayscale image is converting 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 150 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.

FACE AND WIDTH DETECTION

8. 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. The first step is to find the top of the face. 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 is (100,240). The starting x-coordinate of 100 was chosen, to ensure 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 (100,240), 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.

Figure 8.1 demonstrates the above algorithm.

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

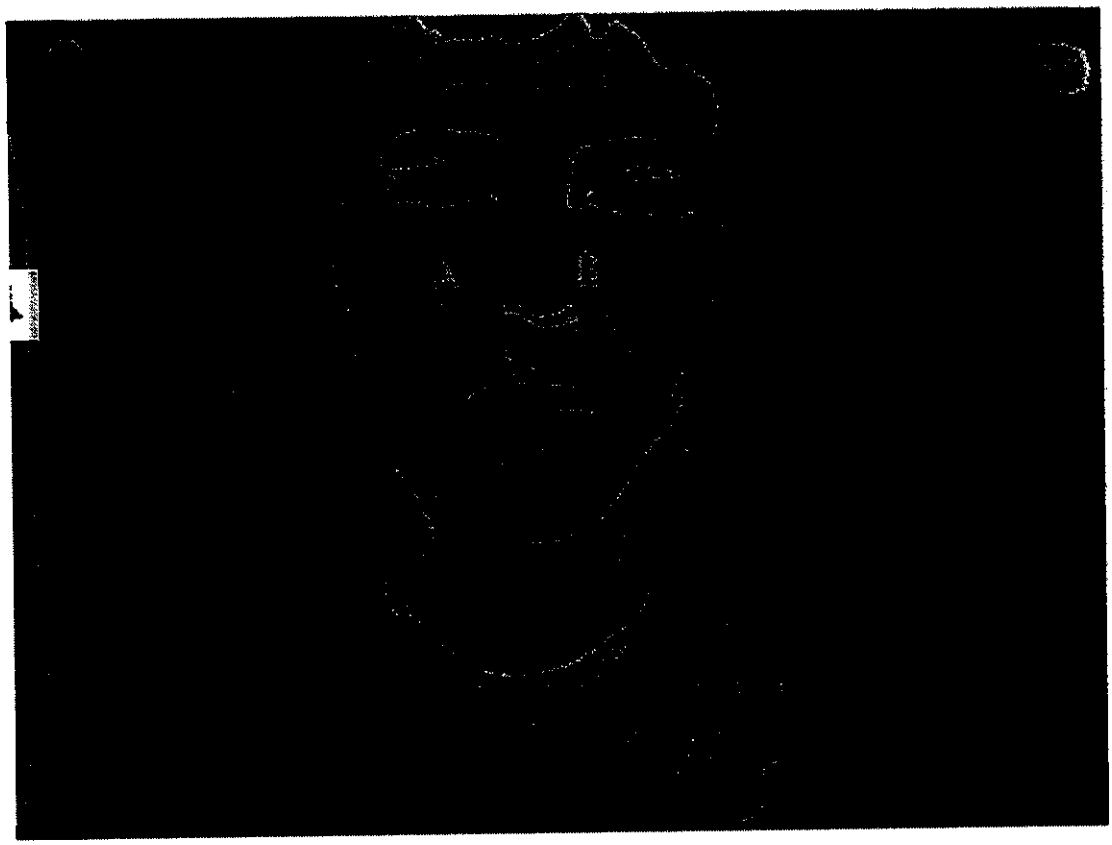


Figure 8.1 FACE TOP AND WIDTH DETECTION

☆ Starting point (320, 60)

A Count 25 white pixels from this point

B Count 25 black pixels from this point

x1 Middle point of A and B

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.

4. 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 labeled '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 labeled 'right_x'.

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

The result of the face top and width detection is shown in Figure 8.2; these were marked on the picture as part of the computer simulation

Figure 8.2 FACE EDGES FOUND AFTER FIRST TRIAL

As seen in Figure 8.2, the edges of the face are not accurate. Using the edges found in this initial step would never localize the eyes, since the eyes fall outside the determined boundary of the face. This is due to the blobs of black pixels on the face, primarily in the eye area, as seen in Figure 8.2. To fix this problem, an algorithm to remove the black blobs was developed.

REMOVAL OF NOISE

9. REMOVAL OF NOISE

The removal of noise in the binary image is very straightforward. Starting at the top, (x2,y2),move left on pixel by decrementing x2, and set each y value to white (for 200 y values). Repeat the same for the right side of the face. The key to this is to stop at left and right edge of the face; otherwise the information of where the edges of the face are will be lost. Figure 9.1, shows the binary image after this process.

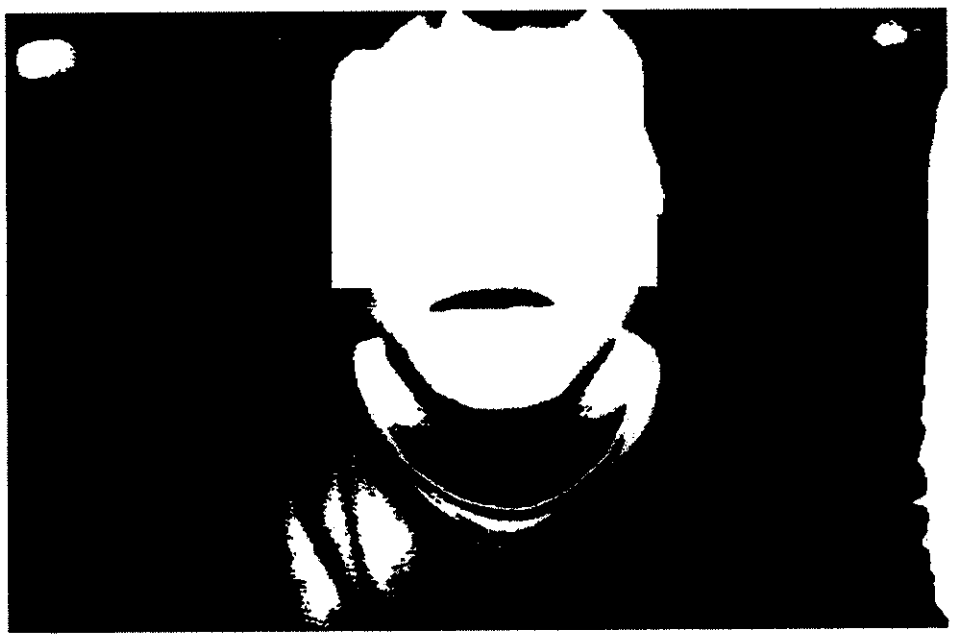


Figure 9.1 BINARY PICTURE AFTER NOISE REMOVAL

After removing the black blobs on the face, the edges of the face are found again. As seen below, the second time of doing this results in accurately finding the edges of the face.

EDGE DETECTION AFTER NOISE REMOVAL

10. EDGE DETECTION AFTER NOISE REMOVAI

Figure 10.1 FACE EDGES FOUND AFTER SECOND TRIAL.

FINDING INTENSITY CHANGES

11. FINDING INTENSITY CHANGES

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.

The valleys are found by finding the change in slope from negative to positive. And peaks are found by a change in slope from positive to negative. The size of the valley is determined by finding the distance between the peak and the valley. Once all the valleys are found, they are sorted by their size.

DETECTION OF VERTICAL EYE POSITION

12. 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. This is shown in Figures 12.1 and 2.

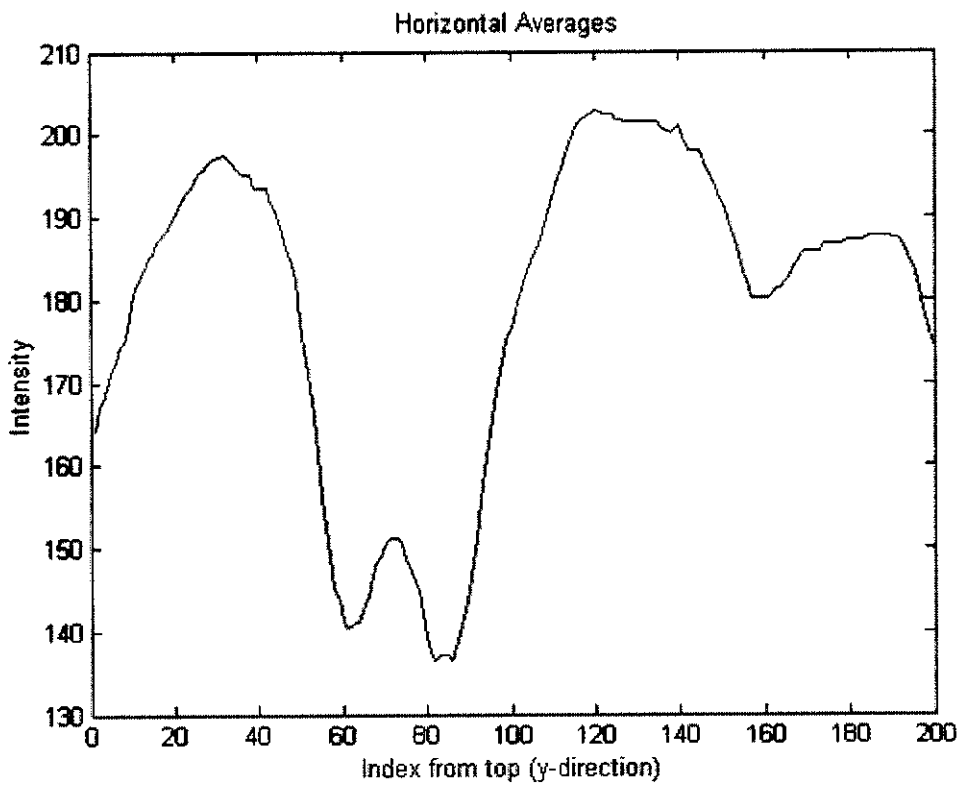


Figure 12.1 GRAPH OF HORIZANTAL AVERAGES FOUND FOR THE LEFT SIDE OF THE FACE

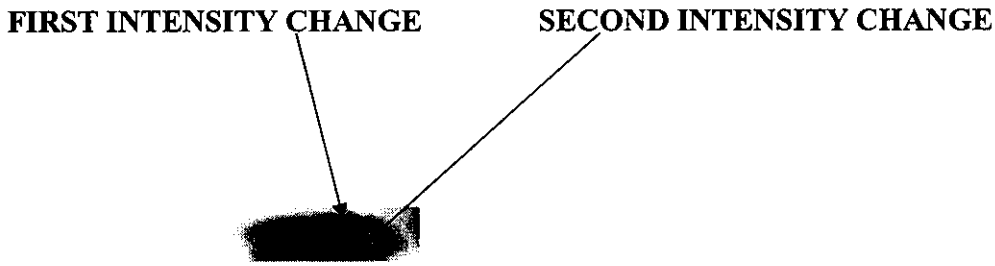


Figure 12.2 POSITION OF THE LEFT EYE FOUND FROM FINDING THE VALLEYS

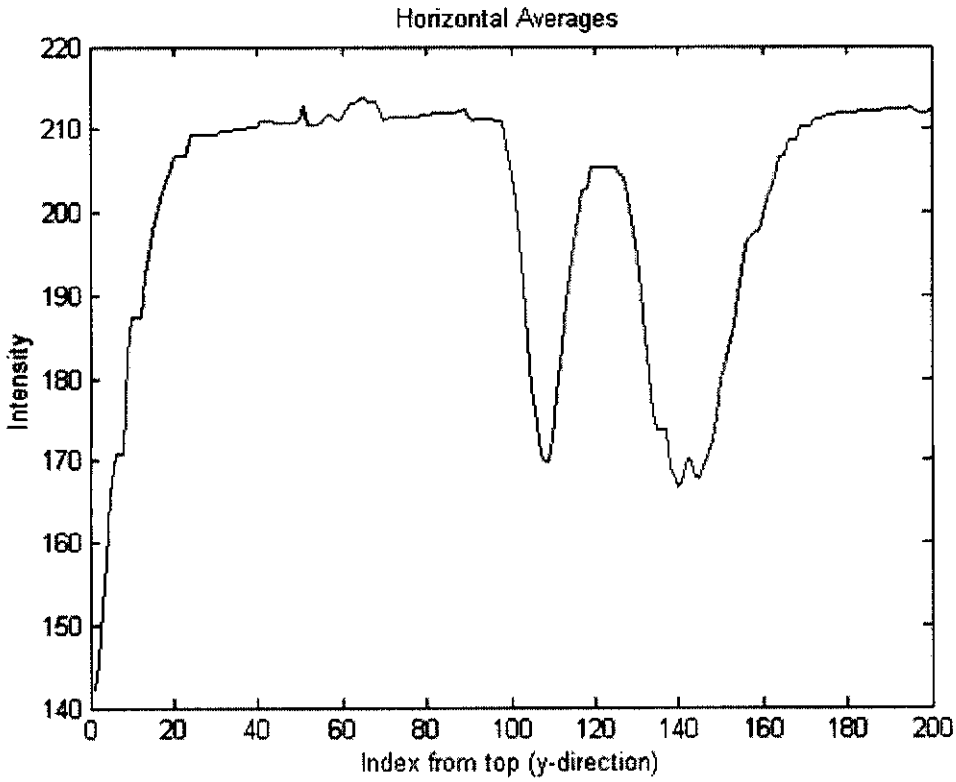
DROWSINESS DETECTION FUNCTION

13. DROWSINESS DETECTION FUNCTION

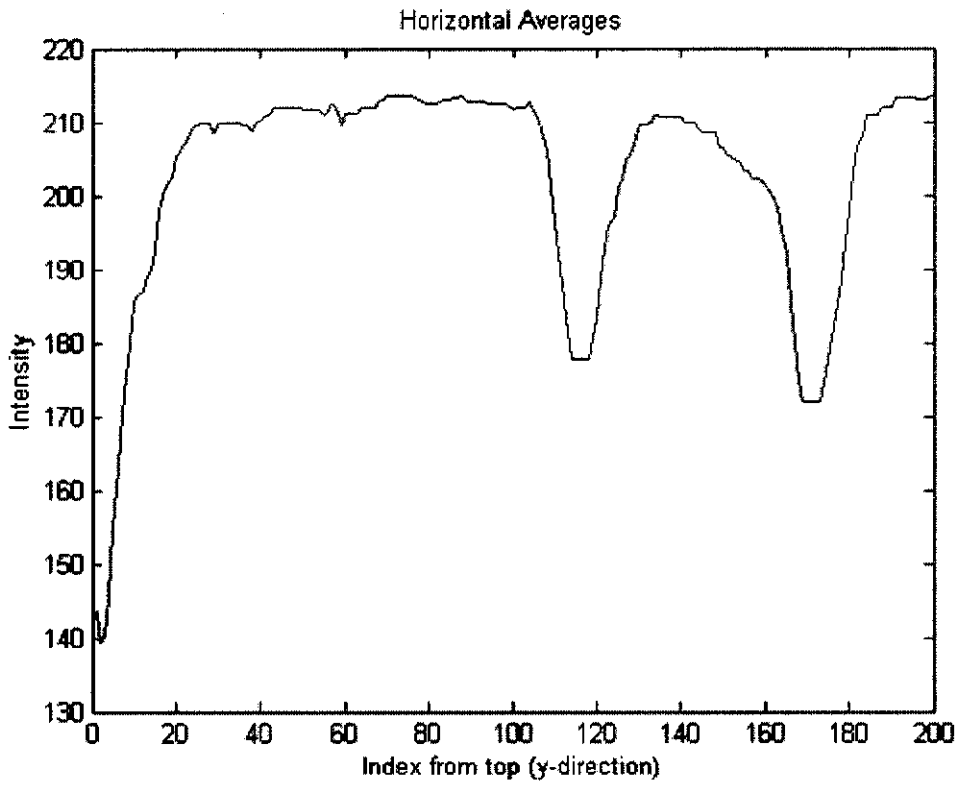
13.1 DETERMINING THE STATE OF THE EYES

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. This is shown in Figure 13.1

OPEN EYES



CLOSED EYES



GRAPH FOR CLOSED EYE

Figure 13.1 COMPARISON OF OPEN AND CLOSED STATE OF THE EYE

The limitation to this is if the driver moves their face closer to or further from the camera. If this occurs, the distances will vary, since the number of pixels the face takes up varies, as seen below. Because of this limitation, the system developed assumes that the driver's face stays almost the same distance from the camera at all times.

13.2 JUDGING DROWSINESS

When eyes are found closed for 5 consecutive frames, alarm is activated, and the driver is alerted. Consecutive number of closed frames is needed to avoid including instances of eye closure due to blinking. Criteria for judging the alertness level on the basis of eye closure count is based on the results found in the previous study.

ALGORITHM IMPLEMENTATION

14. ALGORITHM IMPLEMENTATION

14.1 REAL TIME SYSTEM

The real-time system includes a few more functions when monitoring the driver, in order to make the system more robust. There is an initialization stage, in which the for the first 4 frames, the driver's eyes are assumed to be open, and the distance between the y – coordinates of where the intensity changes occur, is set as a reference. After the initialization stage, the distances calculated are compared with the one found in the initialization stage. If the lower distance is found (difference between 5-80 pixels), then the eye is determined as being closed. Another addition to the real-time system is a comparison of the left and right eyes found. The left and right eyes are found separately, and their positions are compared.

Assuming that the driver's head is not at an angle (tilted), the y – coordinates of the left and right eye should be approximately the same. If they are not, the system determines that the eyes have not been found, and outputs a message indicating that the system is not monitoring the eyes. The system then continues to try to find the eyes. This addition is also useful in cases where the driver is out of the camera's sight. In this case, the system should indicate that no drowsiness monitoring is taking place.

In the instance where there are 5 or more consecutive frames where the eye is not found, an alarm goes off. This takes account for the case when the driver's head has completely dropped down, and hence an alarm is needed to alert the driver.

14.2 CHALLENGES

14.2.1 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. Initially, the 'Dirt Cheap Frame Grabber' was constructed and tested. It was later found that this hardware (which uses the parallel port) would not be fast enough, and it would be difficult to write software to process the images in conjunction with the hardware.

14.2.2 CONSTRUCTING AN EFFECTIVE LIGHT SOURCE

Illuminating the face is an important aspect of the system. Initially, a light source consisting of 8 IR LEDs and a 9V battery was constructed and mounted onto the camera. It was soon realized that the source was not strong enough. After review other literature, the conclusion was that in order to build a strong enough light source, approximately 50 LEDs. To reduce the cost, a desk lamp and IR filter were used (as described previously).

14.2.3 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. Histogram equalization was attempted, but resulted in no progress. Finally it was decided that blobs on the face were acceptable, and an algorithm to remove such blobs was developed.

14.2.4 CASE WHEN THE DRIVER'S HEAD IS TILTED

As mentioned previously, if the driver's head is tilted, calculating the horizontal averages from the left side of the head to the right side is not accurate. It is not realistic to assume that the driver's head will be perfectly straight at all times. To compensate for this, the left and right horizontal averages are found separately.

14.2.5 FINDING THE TOP OF THE HEAD CORRECTLY

Successfully finding the top of the head was a big challenge. Depending on the binarization of the face, the top of the head could not be found correctly. The problem was with the black pixels on the face of the binary image. In some instances the nose, eye or lips were found to be the top of the head due to their black blobs in the binary image. Driver's with facial hair were also often a source of this error. After implementing the noise removal algorithm, this problem was eliminated.

14.2.6 FINDING BOUNDS OF THE FUNCTION

By observing the system algorithm and source code, many bounds are defined. For example, when finding the edge of the face, 200 points in the y-direction is used. This represents the approximate length of the face from the top of the head (for a given distance from the camera). Another example is counting 25 points to the left and right of the centre point, when finding the edge of the face. This represents the approximate width of the face. Both of these values, along with other bounds were a challenge to determine. After examining many images of different faces, and testing different values, these bounds were determined.

LIMITATIONS

15. LIMITATIONS

With 80% accuracy, it is obvious that there are limitations to the system. The most significant limitation is that it will not work with people who have very dark skin. This is apparent, since the core of the algorithm behind the system is based on binarization. For dark skinned people, binarization doesn't work. Another limitation is that there cannot be any reflective objects behind the driver. The more uniform the background is, the more robust the system becomes. For testing purposing, a black sheet was put up behind the subject to eliminate this problem. For testing, rapid head movement was not allowed. This may be acceptable, since it can be equivalent to simulating a tired driver. For small head movements, the system rarely loses track of the eyes. When the head is turned too much sideways there were some false alarms.

SYSTEM TESTING

16.SYSTEM TESTING

16.1 UNIT TESTING

Unit testing helps to eliminate uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

A heavily debated matter exists in assessing the need to perform manual integration testing. While an elaborate hierarchy of unit tests may seem to have achieved integration testing, this presents a false sense of confidence since integration testing evaluates many other objectives that can only be proven through the human factor. Some argue that given a sufficient variety of test automation systems, integration testing by a human test group is unnecessary. Realistically, the actual need will ultimately depend upon the characteristics of the product being developed and its intended uses. Additionally, the human or manual testing will greatly depend on the availability of resources in the organization.

16.2. INTEGRATION TESTING

Integration testing (sometimes called Integration and Testing, abbreviated I&T) is the phase of software testing in which individual software modules are combined and tested as a group. It follows unit testing and precedes system testing.

Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

16.3. REGRESSION TESTING

Regression testing is any type of software testing which seeks to uncover regression bugs. Regression bugs occur whenever software functionality that previously worked as desired, stops working or no longer works in the same way that was previously planned. Typically regression bugs occur as an unintended consequence of program changes.

16.4. VALIDATION TESTING

The goal for the regulators is to ensure that quality is built into the system at every step, and not just tested for at the end, as such validation activities will commonly include training on production material and operating procedures, training of people involved and monitoring of the system whilst in production. In general, an entire process is validated; a particular object within that process is verified. The regulations also set out an expectation that the different parts of the production process are well defined and controlled, such that the results of that production will not substantially change over time. This also extends to include the development and implementation as well as the use and maintenance of computer systems. The software validation guideline states: "The software development process should be sufficiently well planned, controlled, and documented to detect and correct unexpected results from software changes.

CONCLUSION

17. CONCLUSION

A non-invasive system to localize the eyes and to monitor drowsiness was developed. Information about the head and eyes position is obtained through various self-developed image processing algorithms. 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. In addition, during monitoring, the system is able to automatically detect any eye localizing error that might have occurred. In case of this type of error, the system is able to recover and properly localize the eyes. The following conclusions were made:

- Image processing achieves highly accurate and reliable detection of drowsiness.
- Image processing offers a non-invasive approach to detecting drowsiness without the annoyance and interference.
- A drowsiness detection system developed around the principle of image processing judges the driver's alertness level on the basis of continuous eye closures.

FUTURE WORK

18. 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. Using 3D images is another possibility in finding the eyes. The eyes are the deepest part of a 3D image, and this maybe a more robust way of localizing the eyes. Adaptive binarization is an addition that can help make the system more robust. This may also eliminate the need for the noise removal function, cutting down the computations needed to find the eyes. This will also allow adaptability to changes in ambient light. 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.

APPENDICES

19.1. CODINGS

1. MAIN FUNCTION

```
im=imread('D:\project\photos\n1.jpg');
[p]=binarization1(im);
imshow(p);
pixval;
first_cent=centre(p);
fprintf('centre is %g',first_cent);
[p]=binarization2(im);
top=topdetect(p,first_cent);
fprintf('top is %g',top);
acc_cent=accucentre(p,first_cent,top);
fprintf(' accurate centre is %g',acc_cent)
acc_top=accutop(p,acc_cent,top);
fprintf(' accurate top is %g',acc_top);
[left_edge]=leftedge(im,acc_cent,acc_top);
fprintf('left side edge is\n');
for i=1:230
   fprintf('%g\t',left_edge(i));
end
[q]=noiseremoval(p,acc_cent,acc_top,left_edge);
[p]=binarization3(im);
```

```
[left_edge2,vert_y]=edgedetect2(p,acc_cent,acc_top);
fprintf(' accurate left side edge is\n');
for i=1:230
   fprintf('%g, %g\t',left_edge2(i),vert_y(i));
end
[horz_values]=horzindalaverage(im,acc_cent,left_edge2,vert_y);
 [index,valley_val,valley_val1]=findvalley(horz_values);
[dist]=findindex(index,valley_val,valley_val1,acc_top);
fprintf('distance is %g\n',dist);
img_dist_1=0;
img_dist_2=0;
img_dist_3=0;
img_dist_4=dist;
threshold_val=dist;
for i=1:4
   im1=input('enter image\n','s');
   im=imread(im1);
   [p]=binarization1(im);
   first_cent=centre(p);
   fprintf('centre is %g\n',first_cent);
   [p]=binarization2(im);
   top=topdetect(p,first_cent);
   fprintf('top is %g\n',top);
   acc_cent=accucentre(p,first_cent,top);
   fprintf(' accurate centre is %g\n',acc_cent)
   acc_top=accutop(p,acc_cent,top);
   fprintf(' accurate top is %g\n',acc_top);
   [left_edge]=leftedge(im,acc_cent,acc_top);
```

```
  [q]=noiseremoval(p,acc_cent,acc_top,left_edge);
  [p]=binarization3(im);
  [left_edge2,vert_y]=edgedetect2(p,acc_cent,acc_top);
  [horz_values]=horzindalaverage(im,acc_cent,left_edge2,vert_y);
  [index,valley_val,valley_val1]=findvalley(horz_values);
  [dist]=findindex(index,valley_val,valley_val1,acc_top);
  fprintf('distance is %g\n',dist);
  img_dist_1=img_dist_2;
  img_dist_2=img_dist_3;
  img_dist_3=img_dist_4;
  img_dist_4=dist;
  if (img_dist_1>threshold_val & img_dist_2>threshold_val & img_dist_3>threshold_val &
img_dist_4>threshold_val)
        fprintf('\ndrowsiness detected\n');
        [y, Fs, nbits] = wavread('C:\WINDOWS\Media\ringin.wav')
        for(i=1:3)
            wavplay(y,Fs)
        end
   end
end
```

2.Purpose:

To convert the image into a binary picture.

Based on a predefined threshold, the image is binarized into two values, black and white.
If a pixel in the Pic array is less than or equal to the threshold it is set to black,
if greater, set to white. The binary image is stored in the bwpic array.

```
function bwpic=binarization1(z)

for i=1:640
  for j=1:480
    if z(i,j)<=150
      bwpic(i,j)=0;
    else
      bwpic(i,j)=1;
    end
  end
end
```

3.Purpose:

To find an accurate centre of the face (in the x- direction).

Starting at the point (100,240) of the binary image, the edges of the face are found along the x-direction. When the left and right edges are found, the centre of the two is assigned to x1. This value (x1) is returned, and is later used as the new centre of the face.

```
function [x1]=centre(bwpic)
y=320;
x=60;
while bwpic(x,y)==0
   x=x+1;
   end
for j=1:25
   if bwpic(x+j,y)==0
      x=x+j;
      while bwpic(x,y)==0
         x=x+1;
      end
   end
end
x1=x;
while bwpic(x,y)==1
   x=x+1;
end
```

```
for j=1:25
   if bwpic(x+j,y)==1
      x=x+j;
      while bwpic(x,y)==1
         x=x+1;
      end
   end
end
x1=(x1+(x-x1)/2);
x1=round(x1);
```

4. Purpose:

To convert the image into a binary picture.

```
function bwpic= binarization2(f)
for i=1:640
   for j=1:480
      if f(i,j)<=75
         bwpic(i,j)=0;

      else
         bwpic(i,j)=1;
      end
   end
end
```

5.Purpose:

To find the top of the face.

This function passes two integers. The first, x, is the x- coordinate corresponding to the centre, obtained from either the output of Face_detect or Detect_centre depending on at which point in the algorithm it is being called. Similarly, y is the y- coordinate corresponding to the centre of the face. The top is detected by decrementing the y-coordinate, keeping track of the number of black pixels found. The function returns the top value (y-coordinate).

```
function y=topdetect(bwpic,x)
y=240;
while bwpic(x,y)==1
   y=y-1;
end
```

6. Purpose:

To find the centre of the face.

This function passes two integers. The first integer is the centre x-coordinate, the second is the y-coordinate representing the top of the head. Starting at the pixel value of the two passed coordinates, the left and right sides of the face are found by looking for a change in pixels value from a white to black. The centre is the mid-point of the left and right side. This value is returned back (x-coordinate).

```
function x=accucentre(bwpic,x,y)
y1=abs(y-50);
while bwpic(x,y1)==1
  x=x+1;
end
for j=1:25
  if bwpic(x+j,y1)==1
    x=x+j;
    while bwpic(x,y1)==1
      x=x+1;
    end
    j=1;
  end
end
rx=x;
%fprintf('rx is %g',rx);
x=x-1;
while bwpic(x,y1)==1
  x=x-1;
end
for j=1:25
  if bwpic(x-j,y1)==1
    x=x-j;
    while bwpic(x,y1)==1
      x=x-1;
    end
    j=0;
  end
end
```

```
lx=x;
x=((rx-lx)/2)+lx;
x1=round(x);
x=x1;
top_x=x;
```

7. Purpose:

To find the accurate top of the face

```
function top_y=accutop(bwpic,x,y)
while bwpic(x,y)==1
   y=y-1;
end
top_y=y;
```

8. Purpose:

To find the edges of the face.

This functions passes two integers. They are the x and y coordinates for the top of the head. The y coordinate is decremented and the left and right sides are found as described in Detect_center. The y coordinate is decremented 200 times which corresponds to an approximate length of the face. The left and right pixel values of the sides are stored in two separate arrays. A new centre value of the face is returned.

```
function [lx]=leftedge(f,x,y)
lx=[];
rx=[];
i=1;
while i<=230
   y=y+1;
   while f(x,y)>=75
      x=x-1;
   end
   for j=1:5
      if f(x-j,y)>=75
         x=x-j;
         while f(x,y)>=75
            x=x-1;
         end
         j=1;
      end
   end
   lx(i)=x;
   i=i+1;
end
```

9. Purpose:

To convert the image into a binary picture.

```
function bwpic=binarization3(f)
for i=1:640
  for j=1:480
    if f(i,j)<=20
      bwpic(i,j)=0;
    else
      bwpic(i,j)=1;
    end
  end
end
```

10. Purpose:

To remove the black blobs in the binary picture.

Starting at the pixel value of the top of the head,(and centre in the x-direction), the left side and then right side of the face are set to white. This is only done up to the left and right edges found in Detect_edge.

```
function [bwpic]=noiseremoval(bwpic,x,y,lx)

top_y=y;
top_x=x;
for k=1:100

  if lx(k+1)< lx(k)
    for i=lx(k+1):-1:lx(k)
      for y=top_y+k:top_y+400
        bwpic(i,y)=1;
      end
    end

  else
    for i=lx(k):lx(k+1)
      for y=top_y+k:top_y+400
        bwpic(i,y)=1;
      end
    end
  end
end

for i=top_x:-1:lx(1)
  for y=top_y:top_y+200
    bwpic(i,y)=1;
  end
end
```

11. Purpose:

To find the edges of the face.

```
function [lx1,vertical_y]=edgedetect2(bwpic,x,y)
vertical_y=[230];
 lx1=[600];
 rx1=[600];
 top_x=x;
 top_y=y;
 i=1;
 while i<=230
    y=y+1;
     while bwpic(x,y)==1
       x=x-1;
     end
     for j=1:5
       if bwpic(x-j,y)==1
          x=x-j;
          while bwpic(x,y)==1
             x=x-1;
          end
          j=1;
       end
     end
     lx1(i)=x;
     vertical_y(i)=y;
     i=i+1;
  end
```

12. Purpose:

To calculate the horizontal averages.

The function passes three parameters. The first two are the arrays containing the left and right pixel values of the sides found in Detect_edge. The third parameter is an integer representing which side of the face the calculations are being done on (left = 1; right = 2). A smoothing algorithm is implemented to remove any small changes in the horizontal values so they are not latter confused as being valleys.

```
function horz_avg_1=horzindalaverage(f,top_x,lx,vertical_y)
 sum=0;
 horz_avg=[230];
 horz_avg_1=[230];
 for j=1:230
    for i=lx(j):top_x
       sum=sum+f(i,vertical_y(j),1);
    end
    horz_avg_1(j)=sum/(top_x-lx(j));
    sum=0;
 end
 %smoothening algorithm
 for j=1:227
    if (horz_avg_1(j+1)-horz_avg_1(j) >=0) & (horz_avg_1(j+2)-horz_avg_1(j+1) < 0) &
(horz_avg_1(j+3)-horz_avg_1(j+2)>=0)
       horz_avg_1(j+2)=horz_avg_1(j);
       horz_avg_1(j+3)=horz_avg_1(j);
    end
```

```
    if (horz_avg_1(j+1)-horz_avg_1(j)<=0) & (horz_avg_1(j+2)-horz_avg_1(j+1) > 0) &
(horz_avg_1(j+3)-horz_avg_1(j+2)<=0)
        horz_avg_1(j+2)=horz_avg_1(j);
        horz_avg_1(j+3)=horz_avg_1(j);
    end
  end
```

13. Purpose:

To identify the valleys among the horizontal averages.

This function passes an integer to indicate the side of the face (see above explanation). The valleys are found based on slope changes, and the size of a valley is defined by taking the distance between a valley and peak (in the y-direction. This function also sorts the valley according to their size. The number of valleys found, ind, is returned.

```
%Valley
function [ind,valley,valley_sort]=findvalley(horz_avg)
valley=[230];
 valley_sort=[230];
 peak=[230];
 index=[230];
 index1=[230];
 index2=[230];
```

```
for j=1:30
   valley(j)=0;
   valley_sort(j)=0;
   peak(j)=0;
   index(j)=0;
   index1(j)=0;
   index2(j)=0;
end
peak1=0;
ind=1;
for i=1:230
    horz_avg_1(i)=horz_avg(i);
end
a=horz_avg(1);
peak1=a;
slope=1;
for j=1:227
   if (horz_avg(j+1)-horz_avg(j)>0) & (slope==0)
      valley(ind)=peak1-(horz_avg(j));
      index(ind)=j;
      b=horz_avg(j+1);
      peak1=b;
      peak(ind)=peak1;
      slope=1;
      ind=ind+1;
   end
   if (horz_avg(j+1)-horz_avg(j)>0)
      peak1=horz_avg(j+1);
      slope=1;
   end
```

```
   if (horz_avg(j+1)-horz_avg(j)<0)
      slope=0;
   end
end

for j=1:ind
   valley_sort(j)=valley(j);
end
for j=1:(ind-1)
   next_small=valley_sort(j);
   index_next_small=j;
   for k=j:ind
   if valley_sort(k)>next_small
      next_small=valley_sort(k);
      index_next_small=k;
   end
end
temp=valley_sort(j);
valley_sort(j)=valley_sort(index_next_small);
valley_sort(index_next_small)=temp;
end
```

14. Purpose:

To find the indexes (y-coordinates) of the vertical position of the eye.

This function using the valleys found in the previous function to find the vertical positions of the eyes. The parameters passed are the number of valleys and the integer indicating which side is being processed. Once the two vertical positions of the eyes are found (usually the eyebrow and upper eye), the starting and ending areas of the eye are defined. This is done using only the second index. The difference of the two indexes is a measurement of the eye, and is returned. This value is used to determine the state of the eye.

```
% Index
function distance=findindex(ind,valley,valley_sort,top_y)
index=[230];
 index1=[230];
 index2=[230];
  index11=0;
 index22=0;
  for j=1:30
    valley(j)=0;
    valley_sort(j)=0;
    peak(j)=0;
    index(j)=0;
    index1(j)=0;
    index2(j)=0;
 end
 count=1;
for j=1:ind+1
    for k=1:ind
```

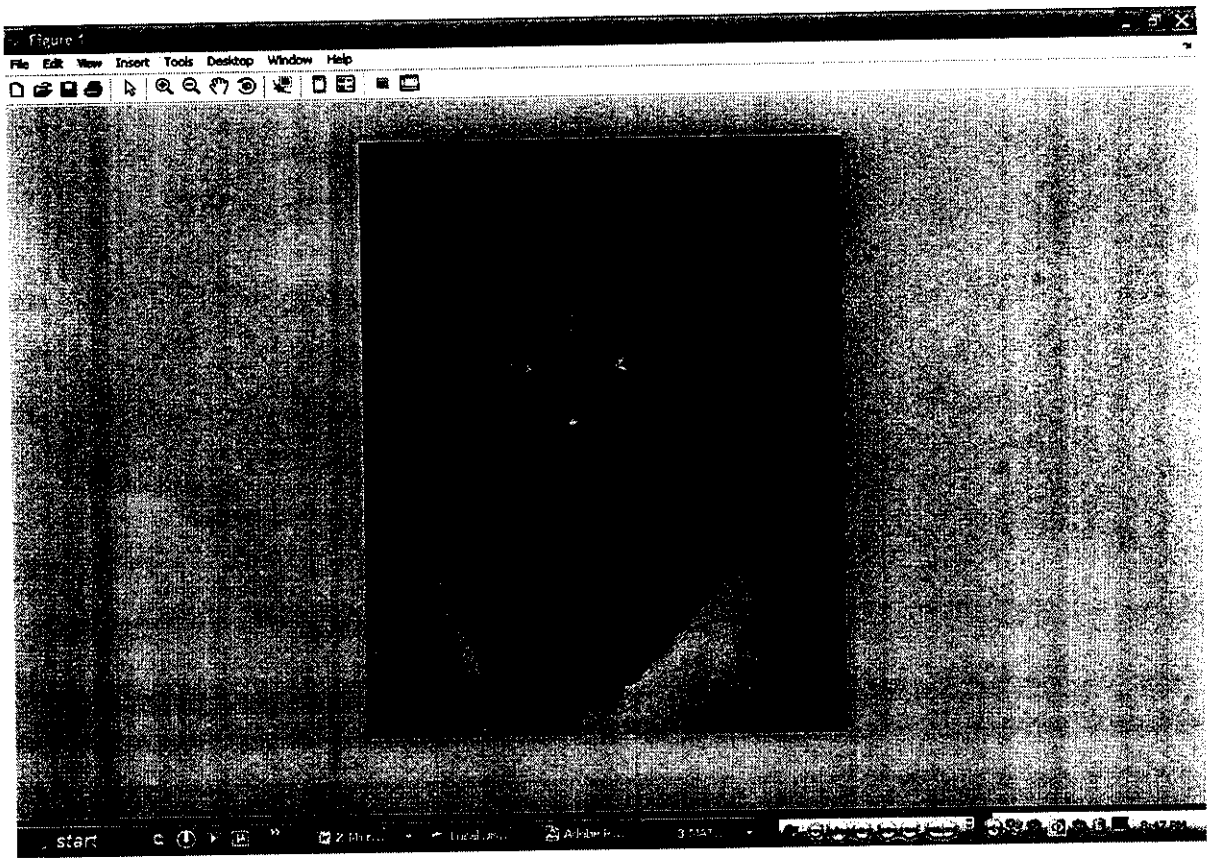
```
      if valley_sort(k)==valley(j)
        index1(k)=index(j);
      end
    end
end
index2(1)=0;
if index1(1)>index1(2)
   index11=index1(2)+top_y;
   temp=index1(1);
   index1(1)=index1(2);
   index1(2)=temp;
   for j=1:ind
      if (index1(j)-(index11-top_y)<70) & (index1(j)-(index11-top_y)>5)
        index2(count)=index1(j);
        count=count+1;
      end
   end
end
 if index1(1)<index1(2)
   index11=index1(1)+top_y;
   for j=1:ind
      if (index1(j)-(index11-top_y)<70) & (index1(j)-(index11-top_y)>5)
        index2(count)=index1(j);
        count=count+1;
      end
   end
 end
 index22=index2(1)+top_y;
 distance=index22-index11;
 index1_left=index11;
 index2_left=index22;
```

19.2 SCREEN SHOTS

BINARIZED IMAGE

IMAGE AFTER NOISE REMOVAL

19.3 SAMPLE IMAGES



ORIGINAL IMAGE

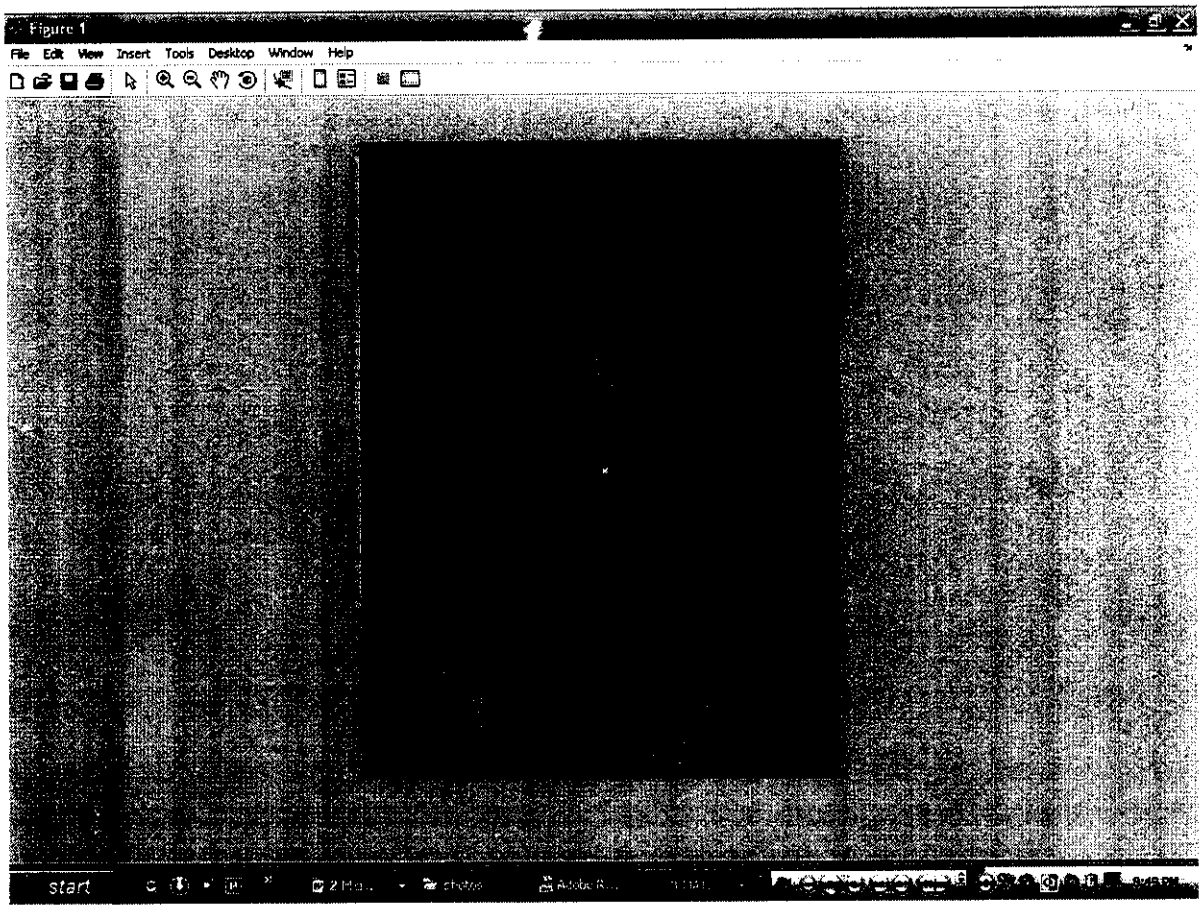


Figure 1
File Edit View Insert Tools Desktop Window Help
start

n10.JPG

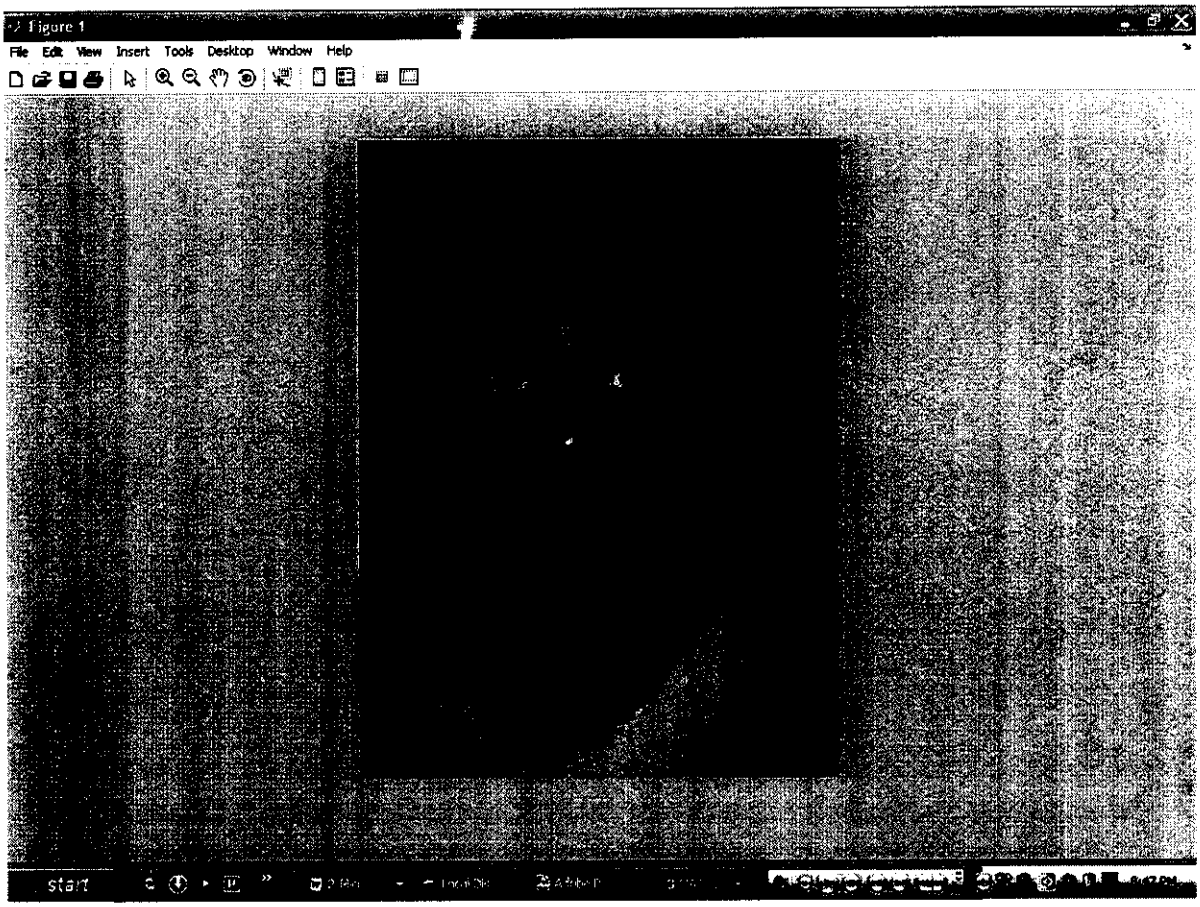


Figure 1
File Edit View Insert Tools Desktop Window Help
start

n2.JPG

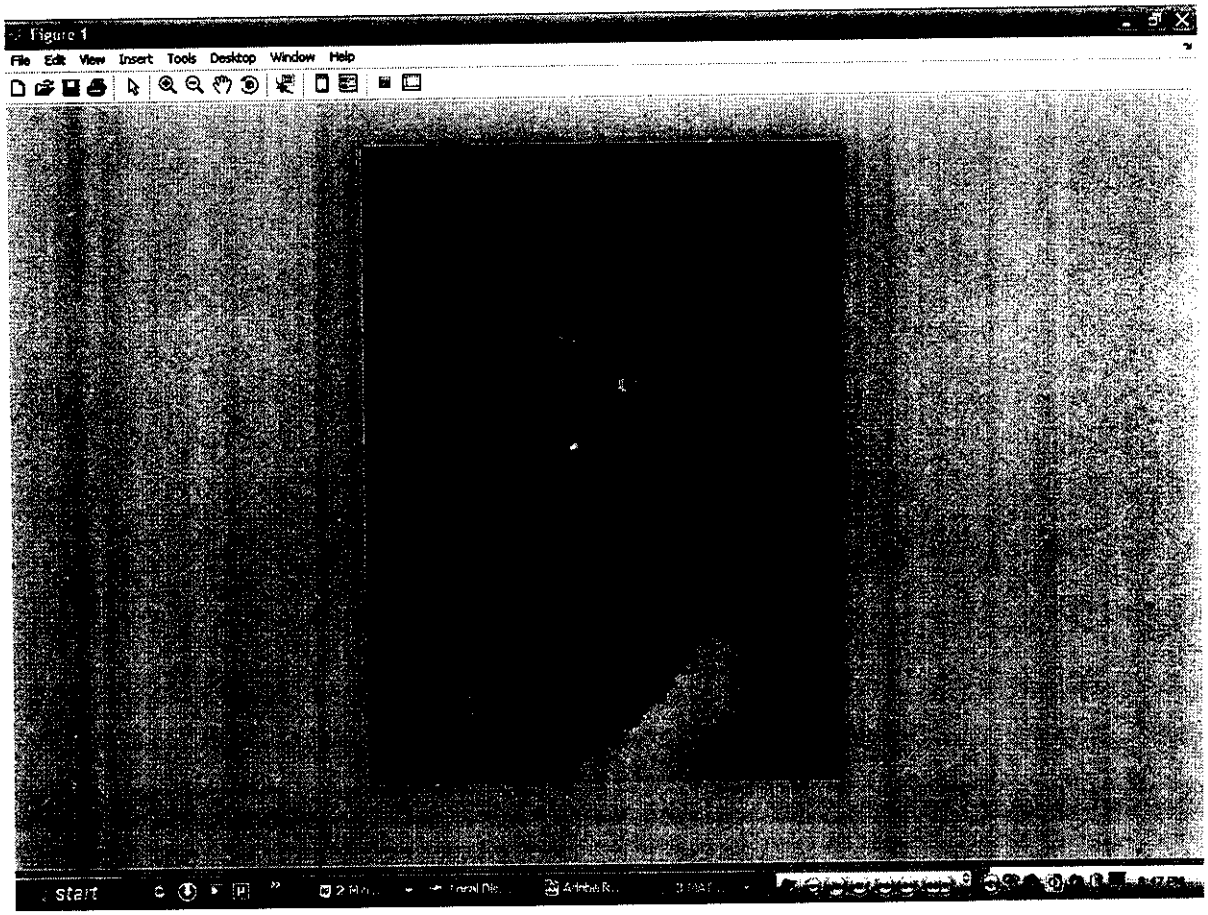


Figure 1
File Edit View Insert Tools Desktop Window Help
start

n3.JPG

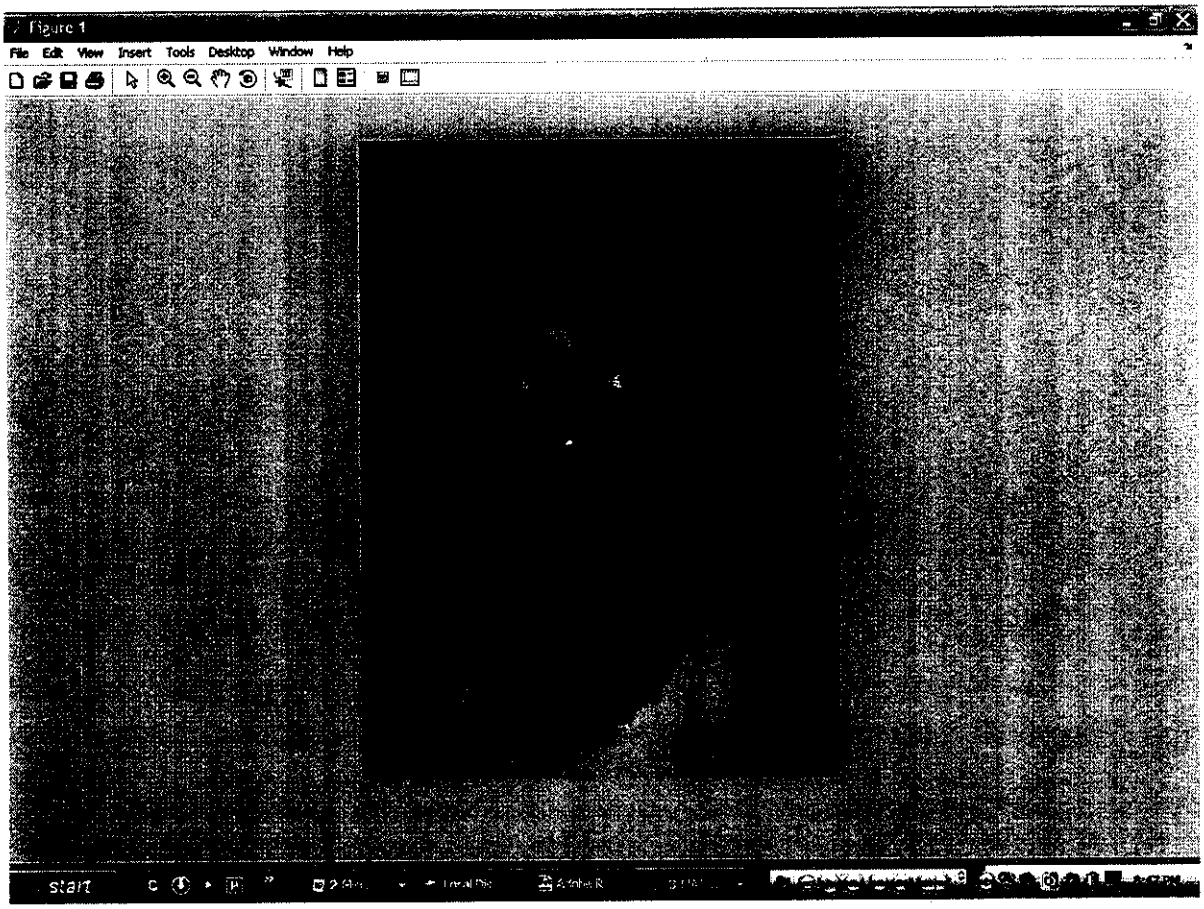


Figure 1
File Edit View Insert Tools Desktop Window Help
start

n4.JPG

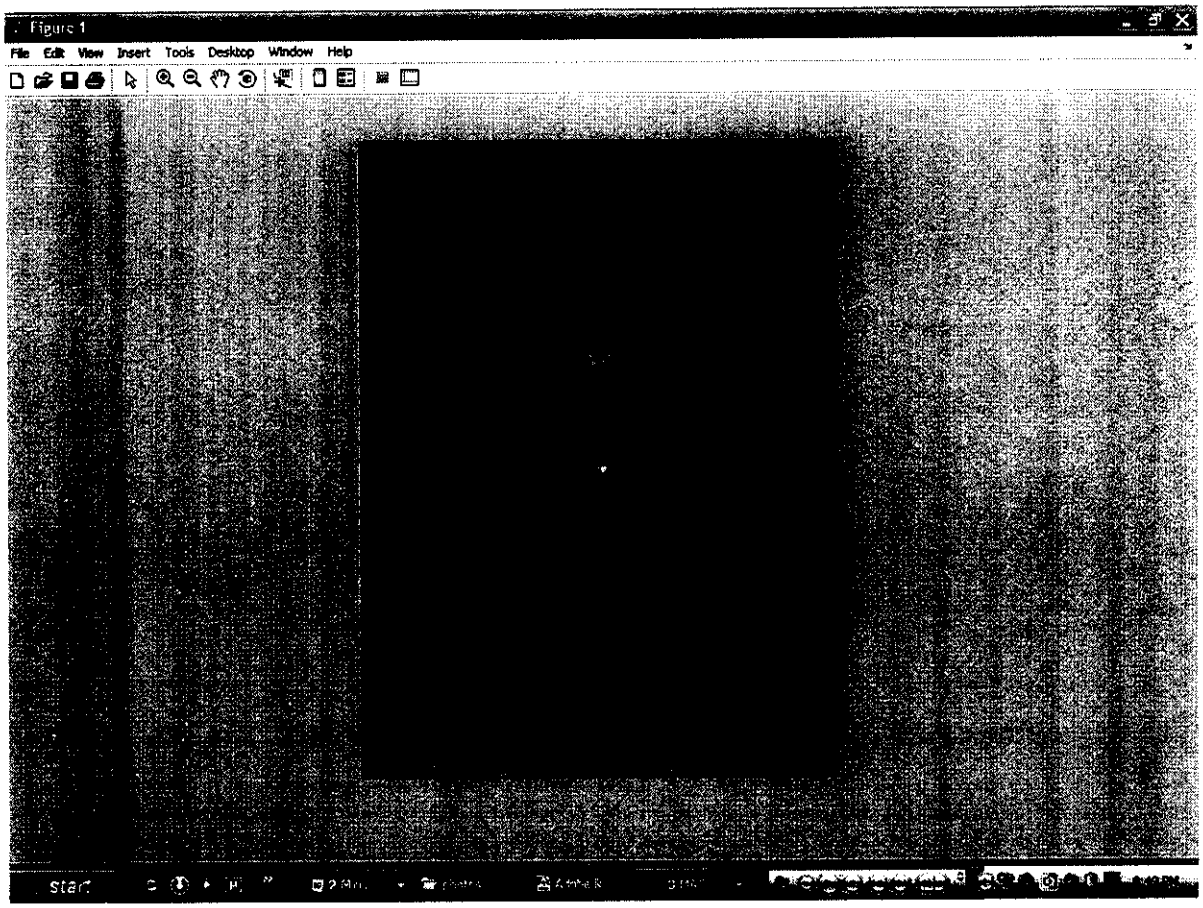


Figure 1
File Edit View Insert Tools Desktop Window Help
start

n11.JPG

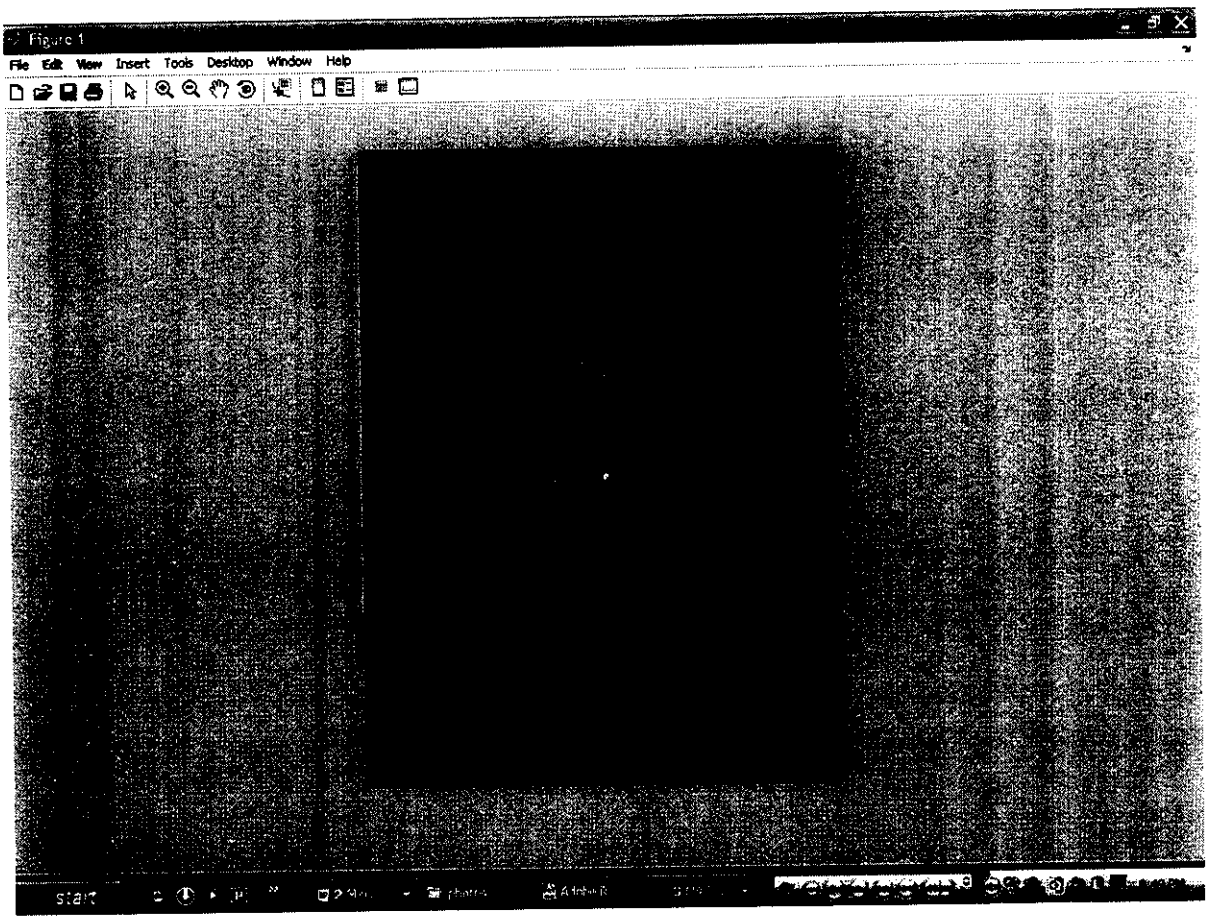


Figure 1
File Edit View Insert Tools Desktop Window Help
start

n12.JPG

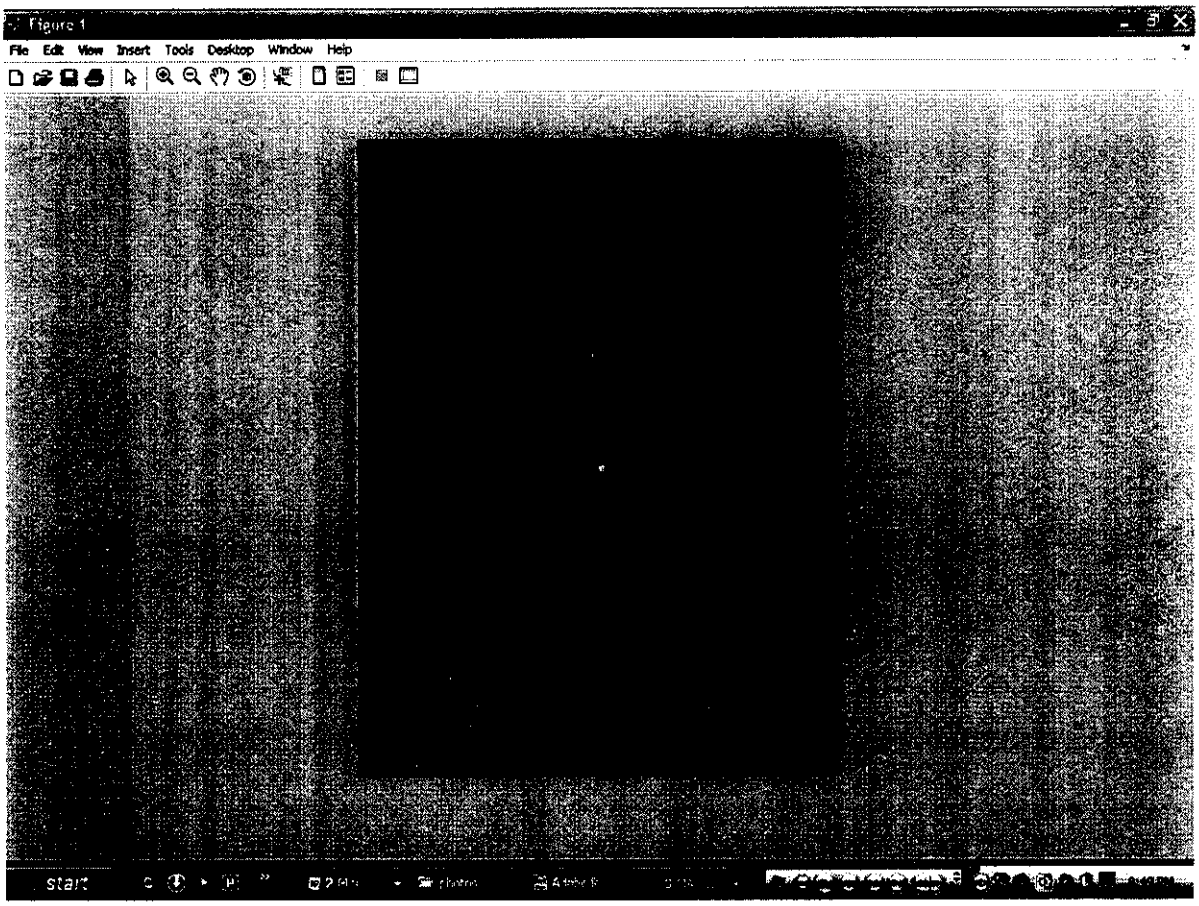


Figure 1
File Edit View Insert Tools Desktop Window Help
start

n13.JPG

OUTPUT

centre is 292

top is 82

accurate centre is 292

accurate top is 82

left side edge is

236	236	236	236	236	236
236	236	236	232	229	225
220	216	214	209	202	198
196	192	186	184	181	178
175	172	171	169	168	164
155	154	153	151	150	149
145	145	145	142	140	140
139	138	130	129	129	128
128	127	127	127	126	126
123	122	122	119	119	118
118	118	115	115	114	113
113	112	112	110	109	109
109	102	102	102	102	102
102	102	102	102	102	102

102	102	102	102	102	102
100	100	100	100	100	100
99	98	98	98	98	98
97	93	93	93	93	93
93	93	93	93	93	93
93	93	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92	92	92	92	92
92	92				

Accurate left side edge is

199, 83	199, 84	199, 85	199, 86	199, 87	199, 88
199, 89	199, 90	199, 91	199, 92	199, 93	195, 94
192, 95	191, 96	186, 97	184, 98	174, 99	172,100
168, 101	164, 102	163, 103	162, 104	160, 105	151, 106
148, 107	147, 108	145, 109	143, 110	142, 111	139, 112
129, 113	129, 114	129, 115	129, 116	128, 117	128, 118
127, 119	127, 120	123, 121	123, 122	122, 123	122, 124
120, 125	120, 126	120, 127	115, 128	114, 129	114, 130
114, 131	112, 132	112, 133	109, 134	108, 135	108, 136
108, 137	106, 138	106, 139	105, 140	105, 141	99, 142
98, 143	97, 144	97, 145	94, 146	92, 147	91, 148
91, 149	91, 150	91, 151	91, 152	91, 153	89, 154
88, 155	88, 156	85, 157	85, 158	85, 159	85, 160
85, 161	85, 162	85, 163	85, 164	85, 165	82, 166
79, 167	73, 168	72, 169	72, 170	72, 171	72, 172
72, 173	72, 174	72, 175	72, 176	72, 177	72, 178
71, 179	71, 180	71, 181	71, 182	71, 183	71, 184
70, 185	68, 186	68, 187	68, 188	68, 189	68, 190
67, 191	67, 192	67, 193	67, 194	67, 195	67, 196
67, 197	67, 198	67, 199	65, 200	65, 201	65, 202
65, 203	64, 204	64, 205	64, 206	64, 207	64, 208
64, 209	54, 210	53, 211	46, 212	46, 213	46, 214
44, 215	43, 216	43, 217	43, 218	43, 219	43, 220
43, 221	43, 222	43, 223	43, 224	28, 225	24, 226
24, 227	24, 228	24, 229	24, 230	24, 231	24, 232
24, 233	24, 234	24, 235	24, 236	24, 237	24, 238
24, 239	24, 240	24, 241	24, 242	24, 243	24, 244

24, 245	24, 246	24, 247	22, 248	22, 249	22, 250
22, 251	22, 252	22, 253	22, 254	22, 255	22, 256
22, 257	22, 258	22, 259	22, 260	22, 261	22, 262
21, 263	21, 264	21, 265	21, 266	21, 267	21, 268
21, 269	21, 270	21, 271	21, 272	21, 273	21, 274
21, 275	21, 276	21, 277	21, 278	21, 279	21, 280
21, 281	21, 282	21, 283	21, 284	21, 285	21, 286
21, 287	21, 288	21, 289	21, 290	21, 291	21, 292
21, 293	21, 294	21, 295	21, 296	21, 297	21, 298
21, 299	21, 300	21, 301	21, 302	21, 303	21, 304
21, 305	21, 306	21, 307	21, 308	21, 309	21, 310
21, 311	21, 312				

distance is 82

enter image

n2.jpg

centre is 292

top is 82

accurate centre is 292

accurate top is 82

distance is 82

enter image

n10.jpg

centre is 314

top is 103

accurate centre is 314

accurate top is 103

distance is 103

enter image

n3.jpg

centre is 292

top is 82

accurate centre is 292

accurate top is 82

distance is 82

enter image

n10.jpg

centre is 314

top is 103

accurate centre is 314

accurate top is 103

distance is 103

enter image

n11.jpg

centre is 314

top is 103

accurate centre is 314

accurate top is 103

distance is 103

enter image

n12.jpg

centre is 314

top is 103

accurate centre is 314

accurate top is 103

distance is 103

enter image

n13.jpg

centre is 314

top is 103

accurate centre is 314

accurate top is 103

distance is 103

enter image

n11.jpg

centre is 314

top is 103

accurate centre is 314

accurate top is 103

distance is 103

DROWSINESS CONFIRMED

ALARM ACTIVATED

REFERENCES

REFERENCES

1. Davies, E.R. "Machine Vision: theory, algorithms, and practicalities", *Academic Press:* San Diego, 1997.

2. Dirt Cheap Frame Grabber (DCFG) documentation, file dcfg.tar.z available from http://cis.nmclites.edu/ftp/electronics/cookbook/video/, Jan 2009.

3. Eriksson, M and Papanikolopoulos, N.P. "Eye-tracking for Detection of Driver Fatigue", *Proceedings in the IEEE Intelligent Transport System* (1997).

4. Gonzalez, Rafel C. and Woods, Richard E. "Digital Image Processing", *Prentice Hall:* Upper Saddle River, N.J., 2002.

5. Grace R., et al. "A Drowsy Driver Detection System for Heavy Vehicles", *Proceedings in the Digital Avionic Systems Conference,* 17th DASC (1998).

6. Perez, Claudio A. et al. "Face and Eye Tracking Algorithm Based on Digital Image Processing", *Proceedings in the IEEE System, Man and Cybernetics 2001 Conference* (2001).

7. Singh, Sarbjit and Papanikolopoulos, N.P. "Monitoring Driver Fatigue Using Facial Analysis Techniques", *Proceedings in the IEEE Intelligent Transport System*(1999).

8. Ueno H., Kanda, M. and Tsukino, M. "Development of Drowsiness Detection System", *Proceedings in the IEEE Vehicle Navigation and Information Systems Conference,* (1994).

9. Weirwille, W.W. (1994). "Overview of Research on Driver Drowsiness Definition and Driver Drowsiness Detection," *Proceedings of 14th International Technical Conference on Enhanced Safety of Vehicles.*