Drowsy Driver Detection System (DDDS)

Sudeep Deepak Ghate
Department of Information Technology
MCT's Rajiv Gandhi Institute of Technology
Mumbai,Maharashta,India.
ghatesudip@gmail.com

Vaibhav Jaiprakash Kheraria
Department of Information Technology
MCT's Rajiv Gandhi Institute of Technology
Mumbai,Maharashta,India.
kherariav@gmail.com

Mayur Prashant Vanmali
Department of Information Technology
MCT's Rajiv Gandhi Institute of Technology
Mumbai,Maharashta,India.
vanmalimayur@gmail.com

Diyog Sevalal Yadav
Department of Information Technology
MCT's Rajiv Gandhi Institute of Technology
Mumbai,Maharashta,India.
diyogyadav@gmail.com

Under the guidance of
Prof. Yogita Ganage

Abstract- Driver weariness is one of the key causes of road mishaps in the world. Detecting the drowsiness of the driver can be one of the surest ways of quantifying driver fatigue. In this project we have developed an archetype drowsiness detection system. This mechanism works by monitoring the eyes of the driver and sounding an alarm when he/she feels heavy eyed.
The system constructed is a non-intrusive real-time perceiving system. The priority is on improving the safety of the driver. In this mechanism the eye blink of the driver is detected. If the driver's eyes remain closed for greater than a certain period of time, the driver is deemed to be tired and an alarm is sounded. The programming for this is carried out in OpenCV using the Haar cascade library for the detection of facial features.

*****

I. 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 the observation of eye movements and blink patterns in a sequence of images of a face.
Initially, we decided to go about detecting eye blink patterns using Matlab. The procedure used was the geometric manipulation of intensity levels. The algorithm used was as follows.
First we input the facial image using a webcam. Pre-processing was first performed by binarizing the image. The top and sides of the face were detected to narrow down the area where the eyes exist. Using the sides of the face, the center of the face was found which will be used as a reference when computing the left and right eyes. Moving down from the top of the face, horizontal averages of the face area were calculated. Large changes in the averages were used to define the eye area. There was little change in the horizontal average when the eyes were closed which was used to detect a blink.
However Matlab had some serious limitations. The processing capacities required by Matlab were very high. Also there were some problems with speed in real time processing. Matlab was capable of processing only 4-5 frames per second. On a system with a low RAM this was even lower.
Today drowsy driving is a serious problem that leads to thousands of accidents each year. Motor vehicle collisions

 ISSN: 2321-8169
176 – 184

may lead to death and disability as well as significant financial cost to both security and individual due to the driver impairments. Drowsiness is one of the factors for collisions. In India, no monitoring device is used to measure the drowsiness of driver. Some kind of systems like driver fatigue monitor, real time vision based on driver state monitoring system, seeing driver assisting system, user center drowsiness driver detection and working system are implemented in foreign countries. All the systems focus either changes in eye movement, physiological measures or driver performance measure.

II. OVERVIEW OF THE DEVELOPED SYSTEM

The proposed system is a driver face monitoring system that can detect driver fatigue by processing of eye and face regions. After image acquisition, face detection is the first stage of processing. Then, symptoms of fatigue are extracted from face image. However, an explicit eye detection stage is not used to determine the eye in the face, but some of important symptoms related to eye region (top-half segment of the face) are extracted. Additionally, a template matching method is used for detecting the head rotation. Finally, we used a fuzzy expert system to estimate driver hypo-vigilance.

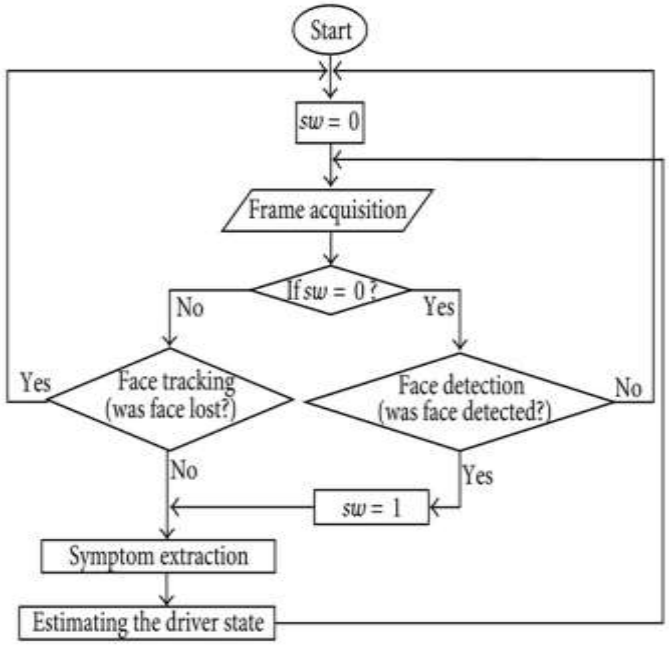


Fig.1 Flowchart of the developed System

Performing the face detection algorithm for all frames is computationally complex. Therefore, after face detection in the first frame, face tracking algorithms are used to track driver face in the next frames unless the face is lost. Therefore, we use an auxiliary variable denoted by sw for determination of face tracking status in Figure 1. If sw is 0, the face is lost, and face detection algorithm must be performed to localize the driver face. In contrast, if sw is 1, it shows that face is tracked successfully by face trackingmethod. For system initialization, sw is 0. It means that the system must perform face detection algorithm for first frame.

For face tracking, full search method is used to find the driver face image in the new frame. The search region is around the center of face image in the last frame which the size of search region is changed according to the size of face image (1.5 times bigger than the size of face image). Then, correlation coefficient between the face image and the sub-windows of search region is used as the matching criteria.[2]

III. DESIGN OF THE SYSTEM

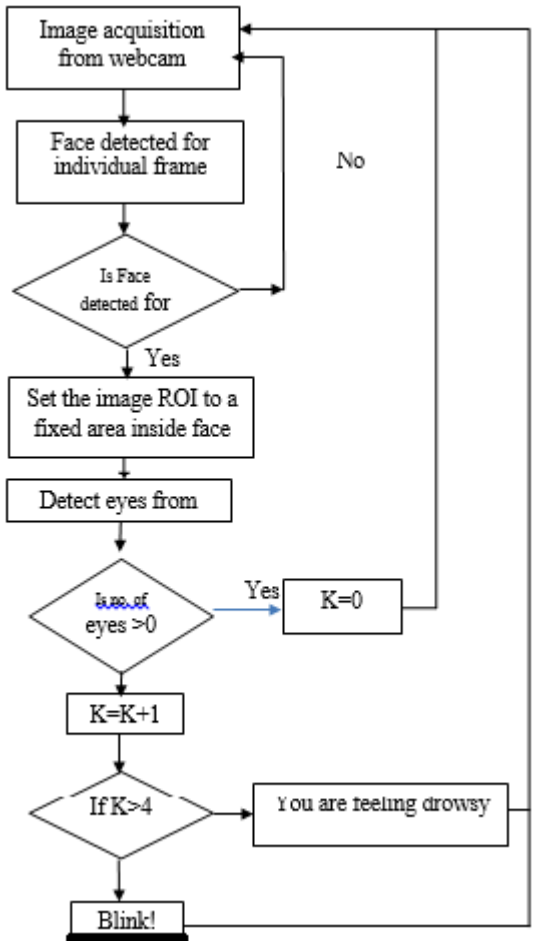


Fig.2 Work Flow of the DDDS

The block diagram represents the workflow of the project. The diagram depicts how the news or reviews are acquired through Hadoop tools and how reports are made after processing them. Apache Flume is used as a service for extracting live data and format conversion with text processing are the methods considered to mine the data for analytical use. Given below is the architecture of the system-
:

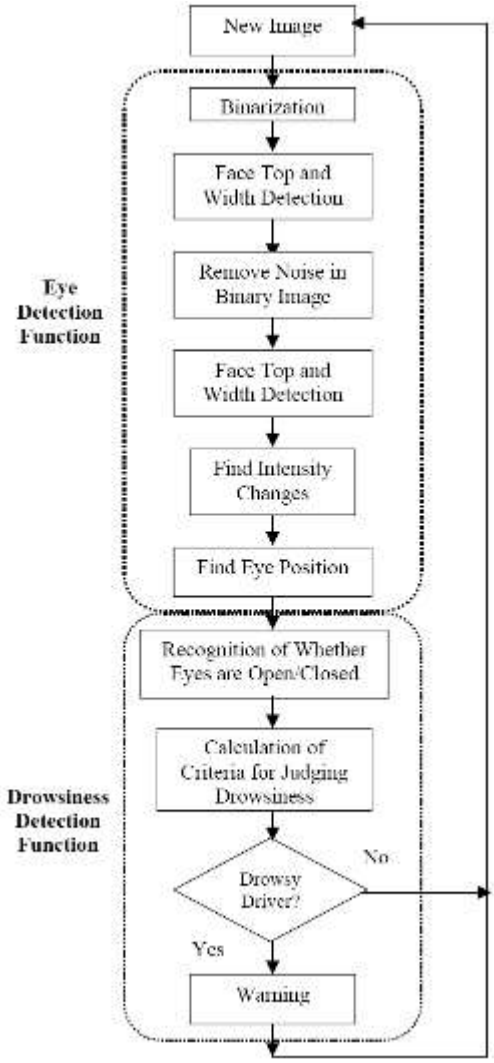


Fig.3 Architecture of the DDDS

IV. IMPLEMENTATION OF THE SYSTEM

a) SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

Processor	:	Intel Processor IV and more
RAM	:	1GB
Hard disk	:	160 GB
Web Cam		

SOFTWARE REQUIREMENTS

Operating System	:	Windows XP/2000
Technologies used	:	.NET
Database	:	MS SQL Server 2005
Software	:	Visual Studio 2008

A Drowsy Driver Detection System 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 fatigue. In such a case when fatigue is detected, a warning signal is issued to alert the driver. This report describes how to find the eyes, and also how to determine if the eyes are open or closed. The system deals with using information obtained for the binary version of the image find the edges of the face, which narrows the area of where the eyes may exist. Once the face area is found, the eyes are found by computing the horizontal averages in the area. 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. The system is also able to detect when the eyes cannot be found, and works under reasonable lighting conditions.

b) System Implementation

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.

Binarization

The first step to localize the eyes is binarizing the picture. Binarization is converting the image to a binary image. Examples of binarized images are shown in Figure 4(a,b,c)(only for illustration)

Fig.4(a) Threshold value 100

Fig.4 (b) Threshold value 150

Fig. 4(c) Threshold value 200

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 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, noseand/or lips. Figure 4 demonstrates the effectiveness of varying threshold values.

Figures 4(a), 4(b), and 4(c) use the threshold values 100, 150 and 200, respectively. Figure 4(b) is an example of an optimum binary image for the eye detection algorithm in that the backgroundis uniformly black, and the face is primary white. This will allow finding the edges of the face, as described in the next section.

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 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 (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 7.3 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.

 ISSN: 2321-8169
176 – 184

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 theinitial 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.

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 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).
The result of the face top and width detection is determined and these were marked on the picture as part of the computer simulation.

Fig.5 Face edges found after first trial

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 7.5, shows the binary image after this process.

Fig.6 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.[11]

Fig.7 Face edges found after second trial

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, as shown below.

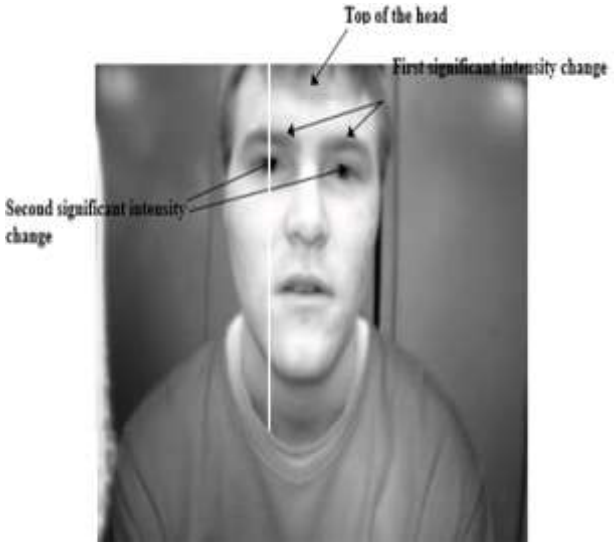


Fig.8 Labels of top head and first two intensity changes

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 7.8a and 7.8b.

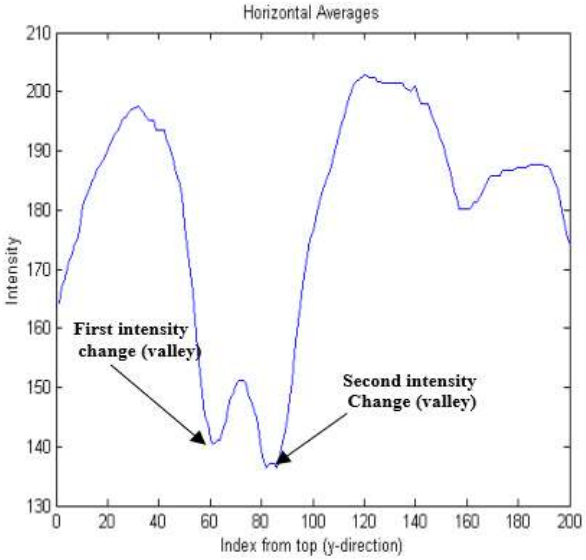


Fig.9 Graph of horizontal averages of the left side of the face

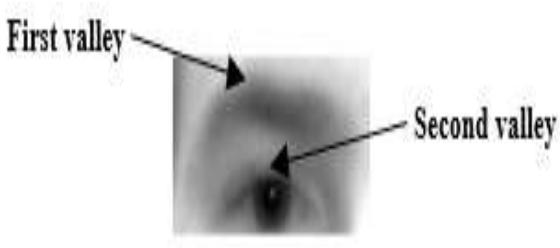
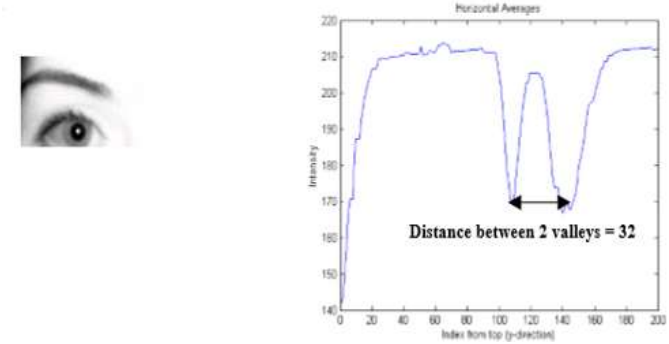


Fig.10(a)(b) position of the left eye found

The state of the eyes (whether it is open or closed) is determined by distance between the first two intensity

 ISSN: 2321-8169
176 – 184

changes found in the above step. When the eyes are closed, thedistancebetween the y – coordinates of the intensity changes is larger if compared to when the eyes are open. This is shown in fig 11



*Similar procedure is done for the right 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.**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. Criteria for judging the alertness level on the basis of eye closure count is based on the results found in a previous study .[11]

V. RESULTS

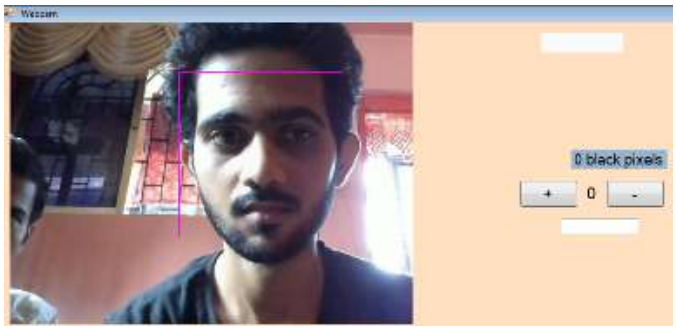


Fig. 11(a): Face detected; Alarm not sounded

Fig.11(b):Face detected; eye region detected; eyes are open , so alarm not sounded

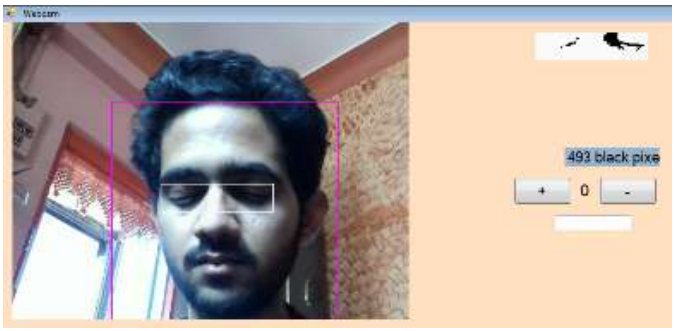


Fig.11(c): Eyes closed when captured, therefore alarm sounded

Fig.11(d): eyes detected and open; speed is 55, therefore alarm not sounded because speed is within the threshold

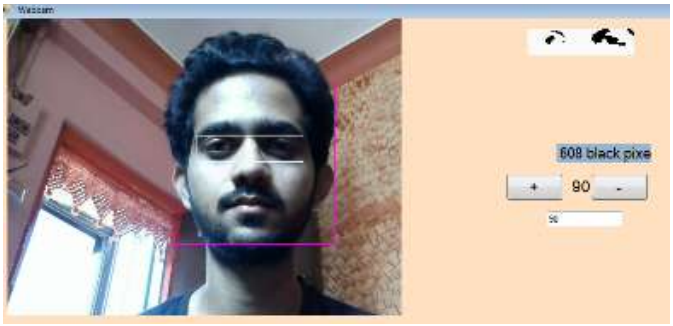


Fig.11(e): eye region captured; speed reaches limit but alarm will sound only after speed>90

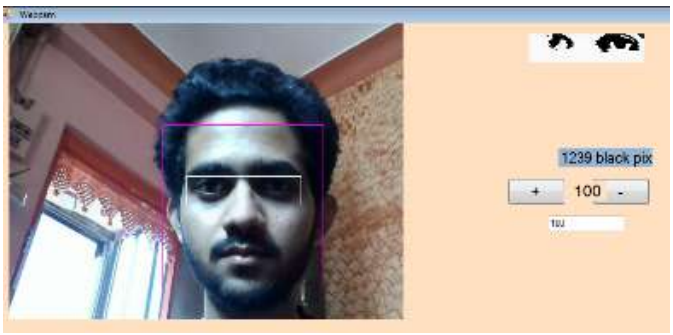


Fig.11(f): speed>100,therefore alarm sounded

Fig.11(g): face not captured as the person avoids the camera, but then also alarm is sounded.

Fig.11(h): Face captured; eyes detected and open; therefore alarm not sounded

Fig.11(i): eyes detected but they are closed, so alarm is sounded

VI. CONCLUSION

Thus we have successfully designed a prototype drowsiness detection system using OpenCV software and Haar Classifiers. The system so developed was successfully tested, its limitations identified and a future plan of action developed. Although only one of the testers was utilized in this introductory experimental phase, the outcomes achieved are fairly convincing. We are also considering to enhance the performance of the system in low light condition, by adding an array of infrared LEDs to the night vision camera. The proposed algorithm appears to be convincing in detecting the eye blinks of a driver. Owing to the combination of the template matching techniques and Viola-Jones. Our next goal will be directed towards determining clear correlations between eye blink data (like duration and speed) and the driver's vigilance state.

Though there are some limitations which are listed below:

1. **Dependence on ambient light:-**With poor lighting conditions even though face is easilydetected, sometimes the system is unable to detect the eyes. So it gives an erroneous result which must be taken care of. In real time scenario infrared backlights should be used to avoid poor lighting conditions.

2. **Optimum range required:-**when the distance between face and webcam is not atoptimum range then certain problems are arising.

   When face is too close to webcam(less than 30 cm) , then the system is unable to detect the face from the image. So it only shows the video as output as algorithm is designed so as to detect eyes from the face region.

   This can be resolved by detecting eyes directly using haardetector objects functions from the complete image instead of the face region. So eyes can be monitored even if faces are not detected.

   When face is away from the webcam(more than 70cm) then the backlight is insufficient to illuminate the face properly. So eyes are not detected with high accuracy which shows error in detection of drowsiness.

   This issue is not seriously taken into account as in real time scenario the distance between drivers face and webcam doesn't exceed 50cm. so the problem never arises.

   Considering the above difficulties, the optimum distance range for drowsiness detection is set to 40-70

3. **Hardware requirements:-**Our system was run in a PC with a configuration of 1.6GHzand 1GB RAM Pentium dual core processor. Though the system runs fine on higher configurations, when a system has an inferior configuration, the system may not be smooth and drowsiness detection will be slow. The problem was resolved by using dedicated hardware in real time applications, so thereare no issues of frame buffering or slower detection.

4. **Poor detection with spectacles:-**When the driver wears glasses the system fails todetect eyes which is the most significant drawback of our system. This issue has not yet been resolved and is a challenge for almost all eye detection systems designed so far.

5. **Problem with multiple faces:**- If more than one face is detected by the webcam, thenour system gives an erroneous result. This problem is not important as we want to detect the drowsiness of a single driver. [4]

VII. ACKNOWLEDGMENT

We would like to express our special thanks of gratitude to our Guide Prof. Yogita Ganage, as well as our Head of Department who gave us the golden opportunity to do this wonderful project on the topic Drowsy Driver Detection System, which alsohelped usin doing a lot of Research and we came to know about so many new concepts for which we are really thankful to them. Also, we are extremely grateful to the Authors of a research paper by the name "Driver Drowsiness Identification by Means of Passive Techniques for Eye Detection and Tracking" published on "www.academia.edu" from where we have cited some images, tables in Algorithm and Experiment section of our project

.

REFERENCES

[1] Q. Wang, J. Yang, M. Ren, and Y. Zheng, " Driver Fatigue Detection: A Survey," the 6th World Congress on Intelligent Control and Automation, pp. 8587- 8591, June 21-23, 2006.

[2] J. F. Xie, M. Xie, and W. Zhu, "Driver Fatigue Detection based on Head Gesture and PERCLOS," International Conference on Wavelet Active Media Technology and Information Processing, pp.128-131, Dec. 17-19, 2012.

[3] A. Singh and J. Kaur, "Driver Fatigue Detection Using Machine Vision Approach," IEEE 3rd International Advance Computing Conference, pp.645- 650, Feb 22-23, 2013.

[4] P. Viola and M. J. Jones, "Robust Real-Time Face Detection", International Journal of Computer Vision, vol.57(2), pp.137–154, May 2004.

[5] B. Jun, I. Choi, and D. Kim, "Local Transform Features and Hybridization for Accurate Face and Human Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.35, no.6, pp.1432-1436, June 2013.

[6] REFLECT project website. Available: http://reflect.pst.ifi.lmu.de/.

[7] F. Streff, and H. Spradlin, "Driver distraction, aggression, and fatigue: a synthesis of the literature and guidelines for Michigan planning," Ann Arbor, vol. 1001, p. 48109. [3] A. C. (IST 2000-28062), "Awake-system for effective assessment of driver vigilance and warning according to traffic risk estimation," Sept. 2001-2004.

[8] L. Bergasa, J. Nuevo, M. Sotelo, R. Barea, and M. Lopez, "Real-time system for monitoring driver vigilance," IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 1, pp. 63–77, 2006.

[9] J. Stutts, J. Feaganes, D. Reinfurt, E. Rodgman, C. Hamlett, K. Gish, and L. Staplin, "Driver's exposure to distractions in their natural driving environment," Accident Analysis and Prevention, vol. 37, no. 6, pp. 1093–1101, 2005. [10]T. Sheridan, "Driver distraction from a control theory perspective," Human factors, vol. 46, no. 4, p. 587, 2004.

[10] M. Ingre, T. ÅKerstedt, B. Peters, A. Anund, and G. Kecklund, "Subjective sleepiness, simulated driving performance and blink duration: examining individual differences," Journal of sleep research, vol. 15, no. 1, pp. 47–54,2006.

[11] **"https://www.academia.edu/26212998/Driver_Drowsiness_Identification_by_Means_of_Passive_Techniques_for_Eye_Detection_and_Tracking"**