Driver Drowsiness Detection System Based on Binary Eyes Image Data

Maninder Kahlon and Subramaniam Ganesan
Oakland University
mskahlon@oakland.edu
ganesan@oakland.edu

Abstract- **In this paper, driver drowsiness detection algorithm based on the state of eyes of the driver which is determined by his iris visibility has been implemented. If eyes remain in one state either open or closed longer than expected time as well as if the driver is not looking straight front, it is an indication that driver is drowsy and then the system warns the driver. System is capable of detecting the state of eyes with or without the regular glasses. Matlab with image processing tools has been used to process the image provided by a camera. Matlab creates System Object using Viola_Jones algorithm to detect the objects such as nose, mouth or upper body. After capturing an image, rectangular eyes area was adjusted to reduce the noise. RGB to Gray scale and finally to Binary image conversion is with a suitable threshold value. A median filter was used to reduce the noise and then the image was smoothened. The drowsiness detection is done based on the conditions like Black to White pixels ratio, number of pixels in the column greater than the threshold value and eye's shape. Light and position of the driver plays an important role. System can be set to self-learn at startup to setup threshold values.**

I. INTRODUCTION

Cars are becoming smarter and smarter nowadays as new technology is evolving. Cars can sense roads, environment and driver behaviors. Car manufactures are adding new safety features. It is found that the contributing factors to traffic accidents are driver's behavior which are mainly drowsiness, impaired driving and distraction [1]. Among these factors, driver drowsiness is the major cause of mortality traffic accidents worldwide [2]. Driver Drowsiness Detection is one of the car safety feature that helps prevent accidents caused by the drowsy driver.

Driver drowsiness cause many accidents every day. According to the National Highway Traffic Safety Administration (NHTSA), more than 1,550 people are killed and 71,000 are injured each year as a direct result of drowsy driving. According to the National Sleep Foundation's Sleep in America poll, 60% of adult drivers (168 million people) have driven a vehicle while feeling drowsy. 37% (103 million) people have actually fallen asleep at the wheel [3].

Thus, driver drowsiness detection feature is very important to prevent accidents and save lives.

This paper is organized as follows: Section 1 gives introduction, and Section 2 explains the current drowsiness techniques. The Section 3 explains the steps involved in processing the image. Results are presented and discussed in Section 4 and the last section concludes the paper.

II. CURRENT DROWSINESS DETECTION TECHNIQUES

There are several methods used to detect driver's drowsiness which can be grouped into three categories [4], [5], and [6]. The first category is based on vehicle data such as vehicle position in lane monitoring & Steering pattern monitoring [7], [8] and [12]. The advantage of this method is that data can be easily collected if it is available on Controller Area Network without need of any additional piece of hardware. The disadvantages of this approach is that road conditions and driver behavior can lead to false detection or may not detect micro sleep depending on calibration parameters. Micro sleeps are short bursts of sleep, often experienced without the person even being aware they took place. They can be experienced by anyone who is tired, but the individuals most at risk are those who work night shifts, have a sleep disorder like insomnia or sleep apnea, or are sleep deprived. The second category involves methods that are based on driver's physiological measurements [9]. Several researchers have considered the following physiological signals to detect drowsiness: electrocardiogram (ECG), electromyogram (EMG), electroencephalogram (EEG) and electro-oculogram (EoG) [10], [13]. To implement this method, an electrode needs to be attached to the driver's body. But this may not be acceptable to automobile companies as well as to the drivers. Thus, this has not been effectively used so far. The third category involves methods that are based on driver's eye/face monitoring [5], [9], [11], and [14].

This paper is based on third category which uses eyes's state to detect the drowsiness. The state of eyes is determined by using the iris visibility of the driver. A camera is utilized to capture driver's face. Matlab is used to extract eye's image as well to processes it in order to determine eye's state whether they are open or closed. This method is effective as compared to others as sleepiness of driver can be captured through their eyes.

III. IMAGE PROCESSING

First step in the process was to capture an image using a camera. An IR LED camera is needed for real application which is necessary for night time or low light conditions. Matlab with image processing tool is required to process the image. The flow chart in Fig. 1 shows the logic at high level. Each of the step mentioned in flowchart has been explained in detail in the following sub- sections.

978-1-5386-5398-2/18/$31.00 ©2018 IEEE

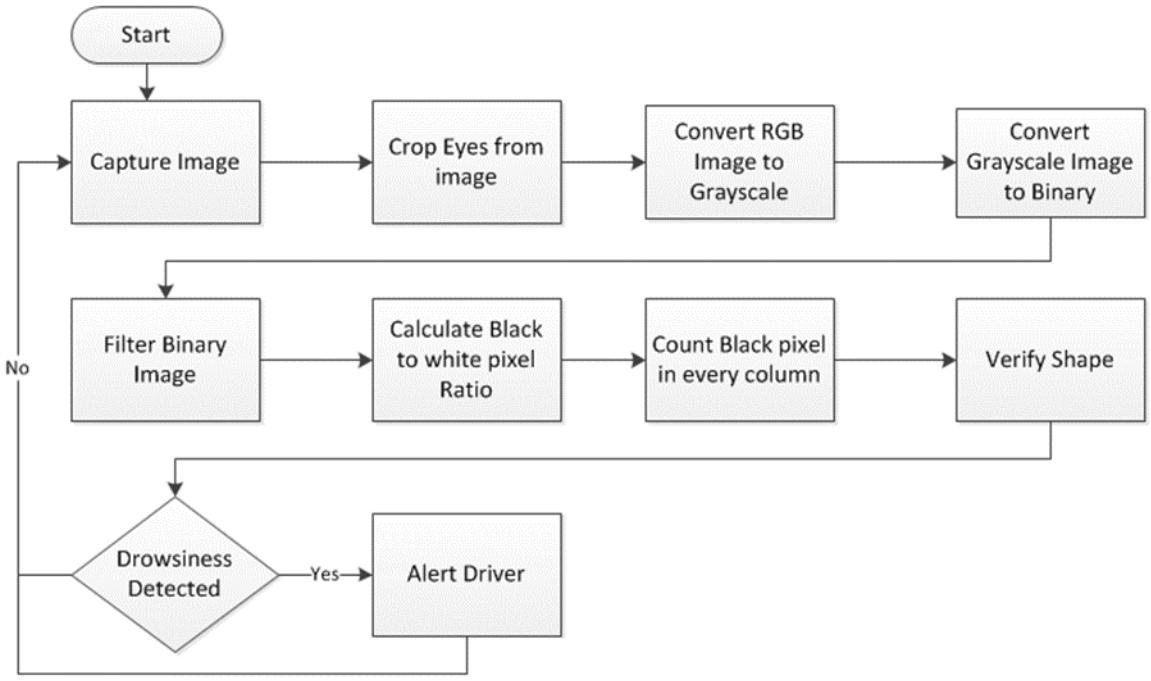

Fig. 1. Driver Drowsiness Detection Flow Chart

A. Capture Image

A web cam was used to capture an image of driver's face. Since this system is based on image processing techniques, camera must be positioned where it can get driver's face as shown in Fig. 2. The image size was fixed to 500 by 600 pixels.

Fig. 2. Sample Image Capture from the Camera

B. Eyes Detection

Matlab script obtains image from the camera. It takes one snap shot at a time and processes the image then it takes another one and so on. Once the image is captured, next step is to find eyes in the image and then crop the rectangular image from the original image.

To detect the rectangular portion of the eye as shown in picture below following Matlab method has been used:

$$Detector = Vision.CascadeObjectDetector \quad (1)$$

It creates System Object using Viola-Jones algorithm to detect the objects such as nose, mouth or upper body [15]. The Classification Model property can be set to the type of object to be detected. By default it is set to configure faces.

Then following step command is used which returns bounding [*x y height length*] values of rectangle.

$$Rectangle = step(EyeDetect, I) \quad (2)$$

In this Rectangle, values were updated as follows to avoid eyebrow and eye glasses in the rectangle

$$x = x + 10 \quad (3)$$

which moves starting point to right by 10 pixels

$$w = w - 10 \quad (4)$$

truncates 10 pixels from the end point.

Thus 10 pixels from the left sides and 10 pixels from the right side were removed.

$$y = y + 4 \quad (5)$$

starting point is pushed down by 4 pixels.

$$L = L - 20 \quad (6)$$

truncates 20 pixels from the length.

Then the rectangle around the eyes is found by using step method as shown in Fig. 3.

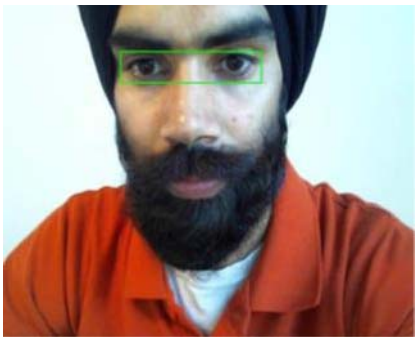

Fig. 3. Eyes Detection

In case the eye is not detected, for instance person is not looking straight, error is captured and thrown and the counter is incremented at the same time. If counter keeps on increasing, it is an indication that driver is not focused straight front on the road.

After capturing the image, rectangular eyes area was adjusted to reduce the noise. Original algorithm captured eyes that end up in the black pixels. Four pixels from each side were reduced which further helped in the reduction of noise from the image caused by user's eyes glasses. Then rectangular image was obtained from the above image as shown in the Fig. 4. The purpose to crop eyes from the original image was to reduce noise and the data size which speeds up the process.

Fig. 4. Eyes cropped from the image

C. Convert Image RGB to Grayscale Image

Matlab saves RGB image data into three *m* x *n* matrices. RGB to Grayscale Matlab Command is '*rgb2gray*'. This command converts RGB values to grayscale values by forming weighted sum of *R*, *G*, and *B* components. Grayscale image pixel intensity is set according to the calculated grayscale value. The grayscale image converted from RGB to grayscale as shown in the Fig. 5.

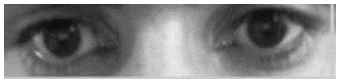

Fig. 5. Grayscale Image

D. Convert Grayscale image to Binary Image

Image conversion from grayscale to binary is very critical. The following Matlab command converts image from grayscale to binary.

- *im2bw*(*I*, level)
- where '*I*' is input image and level is desired threshold.

Matlab's default level is 0.5 but 0.15 value was used for this project. The output image replaces all pixels in input image with following luminance:

- Greater than threshold value to 1 (white)
- Less than threshold to 0 (black)

The following figures are converted to binary image with different threshold values. If the value is too small, data is lost but if the value is too big, noise is added to the image.

Fig. 6. B&W Image (Threshold Value = 0.08)

Fig. 7. B&W Image (Threshold Value = 0.16)

Fig. 8. B&W Image (Threshold Value = 0.32)

Fig. 9. B&W Image (Threshold Value = 0.40)

Fig. 10. B&W Image (Threshold Value = 0.50)

As it is clear from the above images [Fig. 6 to Fig. 10], as threshold value increases, noise level increases in the image. If the threshold value is too low, eyes are not found in the image and if the value is too large, then lot of noise is found around the eyes area.

The threshold value was determined based on the study of several images. If the threshold value is too high, it can contribute to noise to the binary image. On the other hand, if value is too low, system may not be able to detect whether eyes are closed or open. Since everyone has different skin and

eyes color, system should be able to use the same threshold value to convert the image from grayscale to binary.

In order to determine thresholds and detection algorithm, a lot of data was reviewed in different formats. The image data was from .mat file (Matlab generated file) and was imported to excel to study black and white (*BW*) pixels as shown in Fig. 11 and Fig. 12.

- Region with black pixel (0) is shaded in color other pixels as white (1)

An image data in the binary format is stored in a matrix. One of the eye's image data looks as in shown in the Fig. 11 and Fig. 12. The red circle in an image represents an Irish shape. Black pixels are zeros which are highlighted and all white pixel are ones. Open eye image has more zeros compared to closed eye image and also are in almost round shape except some pixels on the top and bottom are lost since Irish is hidden under eye lids and are not visible.

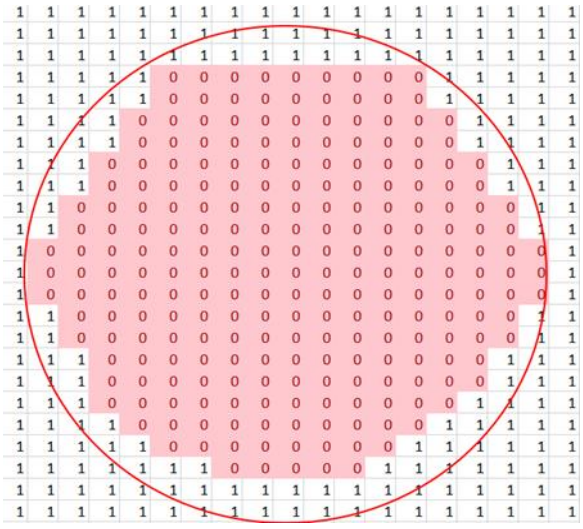

Fig. 11. Open Eye Image Binary Data

Fig. 12. Closed Eye Image Binary Data

Furthermore, it is clear that when eyes are open, each of the columns has more number of 0's compared to the closed eyes. Open eye's image has more than 30 columns where each column has 12 or more black pixels. On the other hand, closed eye has at the most 5 black pixels in each of the column. Considering the number of pixels in each of the columns, the open or closed eyes are determined. If the number of black pixels are more than 12 in at least 15 columns, the eyes are considered open.

E. Morphological and Filter Operation

There were holes in an eye's image due to reflection of light. They are filled up by using "*imfill*" command. *imfill*(*BW*, 'holes') fills holes in the input binary image. It helps to create eye structure back.

Image noise can lead to false eye state detection. It is very vital to eliminate maximum noise from the image. The noise in the image is in black pixels. A median filter was used to reduce the noise. Median filter does nonlinear operation to reduce noise from an image. It also preserves the image edge compared to the other filters. The Matlab command is '*medfilt2*(*A*, [*m n*])'. It performs median filtering of the matrix *A* in two dimensions [*m n*]. The Fig. 14 and Fig. 15 show the filtered image which are obtained from Fig. 13.

Fig. 13. Binary Image

Fig. 14. Filter using [2 2] matrix

Fig. 15. Filter twice using [2 2] matrix

Using *medfilt2* twice of the original image noise was removed as shown in Fig. 15. Then image was further smoothened using following method:

strel('disk', *r*, *n*) creates structure element of disk shape, where *r* is the radius, *n* specifies the number of line structuring elements used to approximate the disk space.

F. Detection Algorithm

Once the image is processed the final step was to make decision whether or not eyes are closed. Many images data was analyzed in order to understand difference between open and closed eyes.

Drowsiness Detection method is based on the following conditions:

- Black to White pixels Ratio
- Column Count Greater than threshold value
- Eye's Shape

First method counts black and white pixels in the image and then takes their ratio where black pixel has value 0 and white has value 1. If eyes are open, black to white pixels ratio

is higher than when eyes are closed. The ratio is used instead of just counting black pixels. If image size is reduced white and black are reduced with same ratio. So changing image size does not affect the result. The second detection algorithm is based on the length of column of black pixels of the image matrix which is illustrated in Fig. 16.

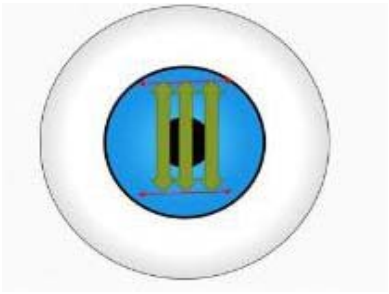

Fig. 16. Column Width Detection

- Sum the number of black pixels (0) of each column of the iris.
- Check if the length is greater than the threshold value.
- *If eyes are partially closed, column length will be reduced.*
- *When eyes are closed, eye lid hides the iris and only eye lashes are converted to black pixel*

The third method which is Eye's Shape (iris) Detection method was added to eliminate the false state detection (i.e. open or close). The binary image of the eyes is divided into two halves and then each of these eyes is diagnosed to check their shape by using the same method which is explained below.

- The method is based on the radius of the eye. As illustrated in Fig. 17, an eye has bigger radius in middle and it decreases while moving it to either left or right.

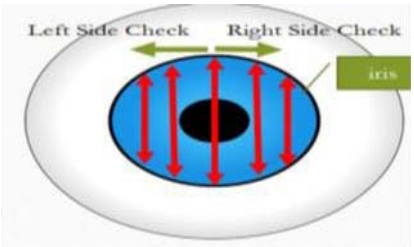

Fig. 17. Shape Detection

This method checks the eye's shape by taking the column which has the maximum number of black pixels. Then, move towards left to check if number of black pixel in next column is decreasing and similarly check the right side. If at least two of the above explained methods result into open eyes, then eyes are considered as open.

IV. RESULTS

Many sample images were taken in order to test the developed algorithm. Some sample image results are shown in Fig. 18 – Fig. 24. Two sample images of each person one with open eyes and the other one with closed eyes are shown below. Then algorithm detect eyes area in each case and crop from the image as described in the above section. The algorithm was successfully able to find the correct eyes state. The following information was collected from each image to make the decision:

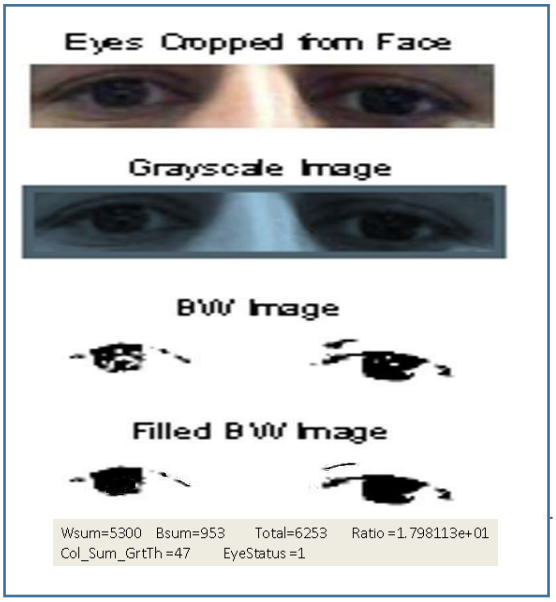

Fig. 18. Person 1 Image Open Eyes Detection

Fig. 19. Person 1 Image Closed Eye Detection

- *Wsum*: Sum of all the white pixels of the image.
- *Bsum*: Sum of all the black pixels of the image.
- *Total*: Sum of all the pixels of the image.
- *Ratio*: Black to white pixel ratio.
- *Col_Sum_GrtTh*: Sum of all columns whose value (sum of black pixels of each column) is greater than threshold value.

- *EyeStatus:*
- *1: Open Eyes*
- *0: Closed Eyes*

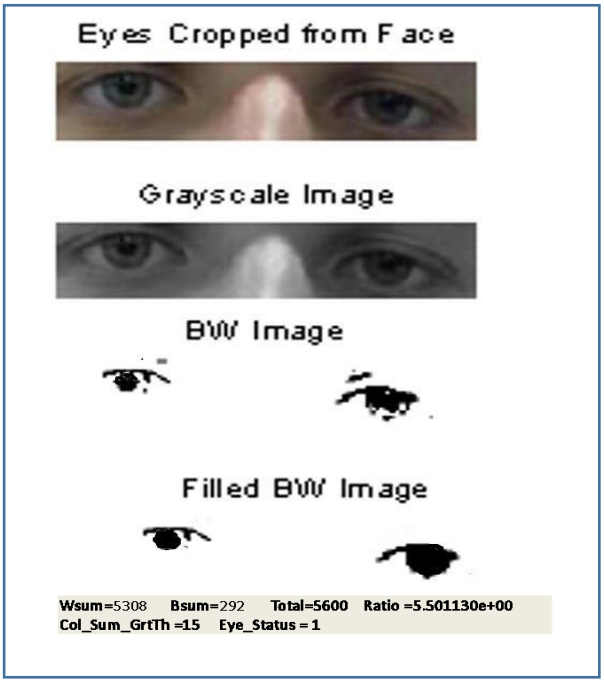

Fig. 20. Person 2 Image Open Eye Detection

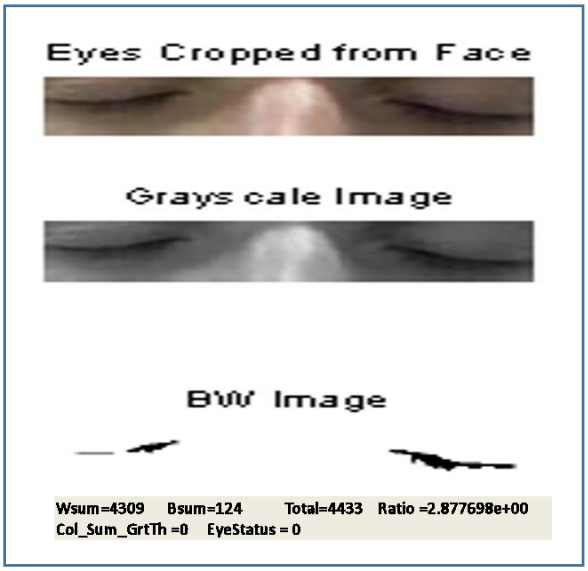

Fig. 21. Person 2 Image Closed Eye Detection

System can be set to learn itself to tune threshold at startup. For instance, if either black pixels in column are way more than the expected value or if the eye's shape is not found , in that case grayscale to BW conversion can be reduced and then black pixel count is verified and this process is repeated until reasonable black pixel count is found in column.

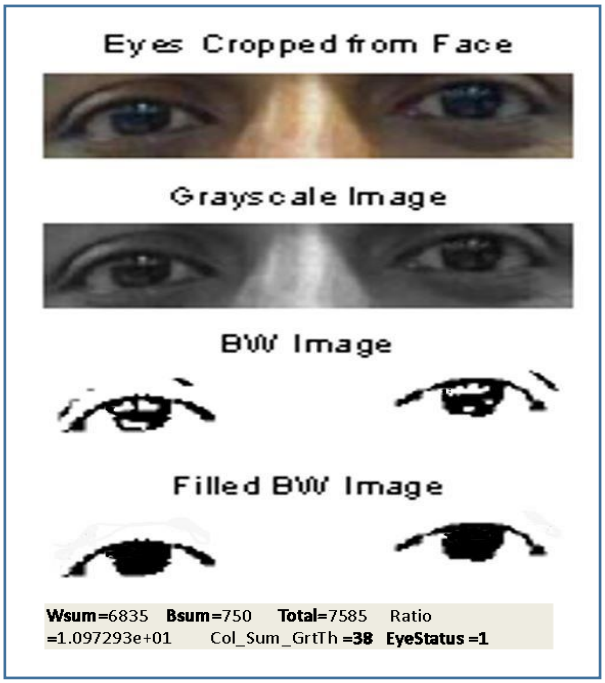

Fig. 22. Sample 3 Image Open Eye Detection

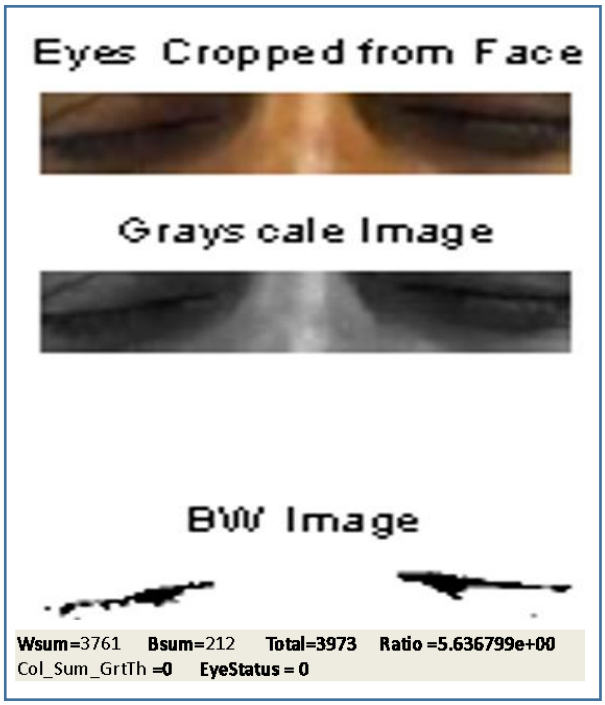

Fig. 23. Sample 3 Image Closed Eye Detection

The Fig. 24 was taken with eye glasses on. The cropped image eliminated noise from the image.

The status of all of the sample eye's images were correctly recognized by each of the above explained methods. They were either closed or open. Closed eyes have very small black to white pixel ratio and black pixel length in comparison to open eyes. In addition shape of closed eye is not round. If the image does meet either open or close eyes criteria, its result will be ignored and the next image will be processed.

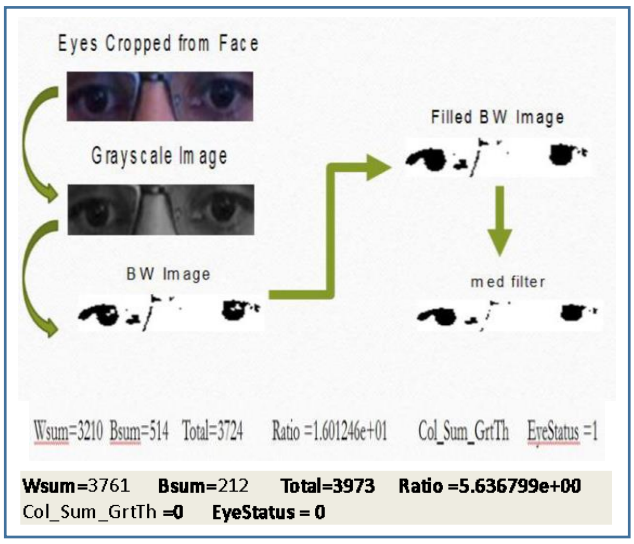

Fig. 24. Sample 4 Image with Eye Glass on

IV. CONCLUSION

The drowsiness detection algorithm that takes live image from the camera to detect the difference between the open and closed eyes of the driver has been successfully implemented. To detect the drowsiness, three or more consecutive open or closed eyes states are considered. If this occurs, it is an indication that driver is drowsy and then system gives warning signal to the driver. System is able to detect state of the eyes with or without the regular glasses on. The system may give inaccurate results in certain cases due to the effect of light, position of driver.

For future enhancements, system can be made more robust so that these three factors should not be obstacle in capturing the accurate result. System can be set to self-learn at startup to set threshold values for further enhancement. In addition to this, it can be made capable to read data from one or more other components such as steering wheel, lane watch, and gas Pedal.

REFERENCES

[1] X. Li, E. Seignez, W. Lu, and P. Loonis, "Vehicle Safety Evaluation based on Driver Drowsiness and Distracted and Impaired Driving Performance Using Evidence Theory," *IEEE Intelligent Vehicles Symposium*, vol. IV, Dearborn, Michigan, USA, June 2014.

[2] G.G. Li, B.-L. Lee, and W.-Y. Chung, "Smartwatch-Based Wearable EEG System for Driver Drowsiness Detection," *IEEE SENSORS JOURNAL*, vol. 15, no. 12, December 2015.

[3] http://drowsydriving.org/about/facts-and-stats/

[4] S. Lawoyin, X. Liu, D.-Y.Fei, and O. Bai, "Detection Methods for a Low-Cost Accelerometer-Based Approach for Driver Drowsiness Detection," *IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, CA, USA, October 2014.

[5] P. Wang, L. Shen "A Method of Detecting Driver Drowsiness Stateased on Multi-features of Face," *5th International Congress on Image and Signal Processing (CISP)*, 2012.

[6] C. Papadelis, Z. Chen, C.K. Papadeli, "Monitoring Sleepiness with On-board Electrophysiological Recordings for Preventing Sleepdeprived Traffic Accidents," *Clinical Neurophysiology*, 118 (9):1906-1922, 2007.

[7] Y. Takei, and Y. Furukawa, "Estimate of driver's fatigue through steering motion," *IEEE International Conference on Systems, Man and Cybernetics*., vol. 2, pp. 1765–1770, 2005.

[8] J.C. McCall, M.M. Trivedi, D. Wipf, and B. Rao, "Lane change intent analysis using robust operators and sparse bayesian learning," in *CVPR: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) - Workshops*, p.59, Washington, DC, USA: IEEE Computer Society, 2005.

[9] A. Azman, Q. Meng, and E. Edirisinghe, "Non intrusive physiological measurement for driver cognitive distraction detection: Eye and mouth movements," *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, 2010.

[10] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3571819/#b32-sensors-12-16937

[11] W. Han, Y. Yang, G.-B. Huang, O. Sourina, F. Klanner, and C. Denk, "Driver drowsiness detection based on novel eye openness recognition method and unsupervised feature learning," *IEEE International Conference on Systems, Man, and Cybernetics*, 2015.

[12] Z. Li, S.E. Li, R. Li, B. Cheng, and J. Shi, "Online Detection of Driver Fatigue Using Steering Wheel Angles for Real Driving Conditions," sensors.

[13] A.G. Correa, L. Orosco, and E. Laciar, "Automatic detection of drowsiness in EEG records based on multimodal analysis," *Med. Eng. Phys*., 36, 244–249, 2014.

[14] R. Pooneh. And R. Tabrizi, and A. Zoroofi, "Drowsiness Detection Based on Brightness and Numeral Features of Eye Image," *Fifth International Conference on Intelligent information Hiding and Multimedia Signal Processing*, 2009.

[15] https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html