International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 06 | June-2018 www.irjet.net p-ISSN: 2395-0072

Drowsiness Detection Using Image Processing Techniques

Sandhya Ramachandran[1], Sneha Rao GR[2] , Taruni Sunder[3]

PES University, Bangalore, Karnataka, India.
Guide: Dr. V.K. Agrawal,Department of Information Science and Engineering,Crucible of Research and Innovation (CORI),PES University, Bangalore.

Abstract - *The aim of this paper is to document the development and testing of various algorithms for drowsiness detection using image processing techniques. Two algorithms were formulated and their accuracies were compared with a third which is already available in literature. An account of their implementation and testing processes is given. In addition, alternate methods of drowsiness detection are discussed and the reasons for which they were discarded are enumerated. The paper concludes with a discussion touching on device instrumentation and possible areas of improvement.*

Keywords **: Drowsy driving, Drowsiness Detection, Image Processing, OpenCV, Dlib.**

1.INTRODUCTION

Drowsy driving is quickly becoming a leading cause of accidents all over the world. Identifying drowsiness as the cause of an accident is also extremely difficult, as there are no available tests that can be run on the driver. Therefore, mitigation is the best way to reduce such accidents. The with efficiency and accuracy. The most accurate way to gauge driver drowsiness is to monitor physiological signals such as heart rate, skin conductance and brain activity. However, such measurements require the attachment of electrodes to the body of the driver, which may cause discomfort and distraction.[1], [2], [3] Another technique that is commonly used is the monitoring of steering patterns and lane driving patterns, deviations from which could indicate that the driver is drowsy. The most popular approach is the use of image processing to detect physical changes that take place in a drowsy driver [4]. These changes include the drooping of the eyelids, the drooping of the head and yawning. We took up this approach and attempted to detect drooping eyelids in three different ways. Various simple operations can be applied on the isolated eye coordinates and their results can be used to determine whether the eyes are open or closed. To alert a drowsy driver, we decided to sound an alarm every time both eyes remained closed for a prolonged time period (We took this as 4 seconds). Before determining the duration of each blink, we had to address how to distinguish an open eye from a closed one. Research told us that using the Eye Aspect Ratio (EAR for short) put forward by Soukupova and Cêch in their 2016 paper "Real-Time Eye Blink Detection Using Facial Landmarks" [5] was an elegant way to go about the task. Dlib, a crossplatform software library equipped with machine learning toolkits was used for this purpose. Dlib recognizes eyes by identifying six defining coordinates as shown.

Fig. 1. The 6 eye coordinates as obtained from Dlib

$$EAR = \frac{|P2 - P6| + |P3 - P5|}{2 * |P4 - P1|}$$

The general approximation of EAR for a closed eye is 0.3. Thus, for a still frame, if the aspect ratio is less than 0.3, the eyes are considered to be closed. Correspondingly, a calculated EAR of above 0.3, could indicate that the eyes are open. Combining this principle with a suitable threshold value (time for which eyes are closed), blinks were detected.

2. METHODS

Before arriving at the methods implemented in this paper, the following techniques proved unsuccessful to detect drowsiness through the drooping of eyelids:

2.1 Calculation of Percentage of White in the Eyes

Our initial plan was to isolate the eyes, traverse through each pixel, and keep a count of how many pixels were white. Above a threshold percentage of white pixels, the eyes would be open, and below, they would be closed.
The pixel number as well as its value was stored in a NumPy array. Colors can be measured in different scales. We worked with the BGR (blue, red, green) value, as well as the intensity value (measured in grayscale). The following issues arose with this approach:

2.1.1 Definition of the Color White

Pure white on the BGR scale is [255,255,255], and on the intensity scale is 255. However, the whites of the eyes are not pure white. Rather, they come in wide ranging shades of off-white. Moreover, pixels whose colors look the same

to the human eye may actually have very different BGR and intensity values. Determining a threshold that will yield results became a difficult feat.

2.1.2 Inclusion of Skin Color

There was no threshold which was precise enough to distinguish fair skin from the whites of the eye.

2.1.3 Variation in Lighting Conditions

An ideal algorithm will have to be effective in a car under moving conditions. Naturally, light conditions are bound to vary. This causes the pixels to appear to be lighter or darker shade. Hence the white pixels will change even in the absence of closing eyes.

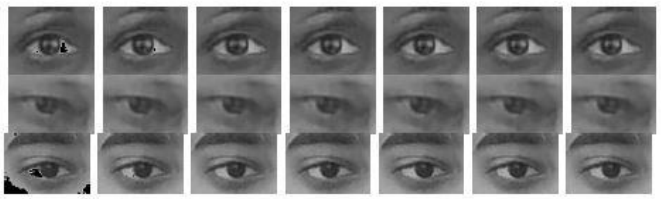

Fig. 2. Observed drawbacks of Approach 1. The following series of figures was generated from three different images. Each was converted to grayscale and a program was run to change the detected white pixels into black for better graphical understanding. Each image produced 7 images, corresponding to different detected intensities of white in grayscale- 160,175,190,205,220,235 and 250 respectively. As seen above, the results are unexpectedly poor, with either negligible amounts of white detected or the unnecessary detection of the skin surrounding the eyes.

2.2 Detection of Skin Surrounding the Eye

Visible skin percentage detection was considered as an option to recognize closed eyes (When the eyes are closed more skin is visible than when the eyes are open). This was to be done by setting the HSV (hue, saturation, value) range of skin color. However, as different people have different skin tones, standardization was not an option. Furthermore, calibration of skin tone will increase the algorithm's running time.

2.3 Hough Transform to detect the iris

cv2.HoughCircles is a built-in function in OpenCV used to detect circles. So far, only one method has been implemented to detect Circles - the Hough Gradient function (Yuen et al. [6]). The function is used as follows:

cv2.HoughCircles(*image,method, dp, minDist*)

where 'image' is the 8-bit single-channeled image in grayscale. MinDist is the minimum distance between the centres of circles. We tested this method, thinking that we could use Hough's Circle Transform to detect the irises of the eyes. However, as the entire iris is not visible (parts of it are covered by the eyelids), the transform is unable to detect it.

Fig.3. Hough Circles on a preloaded image. The program is unable to detect the iris since it is not a perfect circle.

2.4 Final Implementation

After further analysis, we finally arrived at the three methods used to realize our aims. The first step in blink detection is face recognition and the isolation of facial features. This was accomplished with the built-in shape predictor file in the Dlib library. The shape predictor identifies 68 landmarks on a face and assigns coordinates to each of them with respect to the bounding box of the face. These individual coordinates can be extracted from the created "shape" object as NumPy arrays. Using Dlib's coordinates for the left and right eyes, it is possible to highlight the eyes alone, so that a live stream will capture each frame and mark the contours of the eyes corresponding to the upper and lower lids. Using the general approximation of the EAR value, for a still frame, if the aspect ratio is less than 0.3, the eyes are considered to be closed. Correspondingly, a calculated EAR of above 0.3 could indicate that the eyes are open. For a live stream however, it must be appreciated that the length each frame is not significant enough discern whether the eyes are open or not. For a driver to be pronounced as drowsy, the eyes must remain closed for a good amount of time; roughly four to seven seconds. Therefore, it was essential to check whether the EAR remained below the threshold value for a fixed number of frames, depending on the frame rate of the camera used. After testing our code on a varied sample space, it became evident that the assumed value of 0.3 did not perform equally well on everyone. Reaching desired levels of accuracy cannot be achieved by standardizing eye shapes and sizes. The need for individual calibration arose. We modified our program to capture a still frame with the user's eyes closed, in order to calibrate individual EAR values. In our program, the EARs of the left and right eyes are calculated and an average is taken to generate the new threshold value.

We studied the coordinates for several cases of closed and open eyes and explored our own original ways to detect drooping eyelids apart from the eye aspect ratio. We observed that there was a noticeable decrease in the original distance (seen in the open eyes) between the y coordinates of the two upper and two lower points in closed eyes. Furthermore, this was accompanied by a decrease in the absolute value of the slope measured between the corner and first upper coordinate in both eyes. Using this information, we developed two other ways to detect closed eyes. Once again, we set a fixed number of frames for which closed eyes must be detected above which the driver is assumed to be drowsy. Individual calibration is carried out through a simple and interactive

interface. For each eye, the average vertical distance between the top and bottom eyelid is obtained. To conclude that the user is drowsy, both eyes must have their measured distance values below their respective thresholds. The slope concept was utilized in a similar way. In order to maximize the efficiency of the programs and normalize the error caused by changes in physical conditions, appropriate buffer values were added to the respective thresholds. Absolute values are used for calibration and comparison. All distances are measured in pixels. Whenever drowsiness is detected, an alarm is rung to wake up the drowsy driver.

3. TESTING

The next stage of our project was to test out our three final techniques and perform a comparative study. We tried each method on twenty different subjects so that we could build our database and determine the respective accuracies. Our sample space consisted of ten subjects who wore glasses and ten who did not. We tested our three algorithms on each person ten times and noted down the calibrated EAR, slope and distance value. Each time the code detected drowsiness when the subject's eyes were partially or completely closed for 4 seconds, we recorded a success. If the eyes were closed and the program failed to detect it, we counted a failure. In case the subject's eyes were open and the program still indicated drowsiness, we recorded a false positive. Accuracies were calculated for each algorithm as the mean of all successes recorded a) overall, b) for subjects wearing glasses and c) for subjects not wearing glasses.

Fig. 4. Example of a successful run of drowsiness detection using reduction in vertical distance between upper and lower eyelids. The word "SLEEPY" flashed on the window of the video stream after both eyes remained closed for 4 seconds, accompanied by the sounding of an alarm.

	#	Subject	ALGORITHM: EAR Average Value	EAR Accuracy Score on 10	EAR False Positive Counts	Slope Open eye value (R/L)	Slope Closed eye value (R/L)	Slope Accuracy Score on 10	Slope False Positive Counts	Average Vertical Distance Open eye value pixels (R/L)	Average Vertical Distance Closed eye value pixels (R/L)	Average Vertical Distance Accuracy Score on 10	Average Vertical Distance False Positive Counts
Without Glasses	1	P1	0.274	10	-	1.0/1.0	0.143/0.143	10	-	7.5/7.5	2/2.5	9	-
	2	P2	0.288	10	1	0.8/0.667	0.6/0.5	9	-	6/5.5	2.5/3	8	-
	3	P3	0.284	10	-	1/0.857	0.143/0.143	6	-	8.0/8.0	2.5/3	8	-
	4	P4	0.281	8	-	0.667/0.714	0.333/0.333	9	-	5.0/5.0	4.0/4.0	10	-
	5	P5	0.232	10	1	0.833/0.667	0.167/0.167	5	-	4.0/4.0	3.0/3.0	10	-
	6	P6	0.286	9	-	0.667/0.667	0.333/0.333	3	-	5.0/5.0	3.0/3.0	10	-
	7	P7	0.238	8	2	1.144/0.5	0.571/0.1	8	-	7/7.5	4.5/5	9	-
	8	P8	0.317	10	-	0.8/0.667	0.333/0.333	6	-	6/6.5	3.0/3.0	9	-
	9	P9	0.236	10	-	0.5/0.625	0.25/0.222	10	-	4.0/4.0	2.0/2.0	10	-
	10	P10	0.3	10	1	0.667/0.833	0.428/0.667	10	-	8/8.5	4.5/5	10	-
With Glasses	11	P11	0.269	10	-	1.0/1.0	0.333/0.333	0	-	7/6.5	3.0/4.0	9	-
	12	P12	0.3	6	-	0.6/0.6	0.2/0.2	0	-	4.5/5	4.0/3.0	8	1
	13	P13	0.268	8	4	0.5/0.667	0/0	0	-	4.5/4.5	1.5/1.5	2	-
	14	P14	0.256	0	-	0.5/0.6	0.167/0.2	0	-	6.5/7	4.5/4	9	2
	15	P15	0.34	0	-	0.333/0.5	0/0.167	0	-	6.0/6.0	3.0/3.0	1	-
	16	P16	0.22	10	-	0.5/0.5	0/0.167	0	-	4.5/4.5	2.5/2	10	-
	17	P17	0.235	6	5	0.428/0.5	0.143/0.167	2	-	3/3.5	2.0/2.0	8	-
	18	P18	0.337	7	2	0.667/0.667	0/0.2	0	-	5.5/5.5	1.5/1	9	-
	19	P19	0.245	7	-	0.625/0.625	0.375/0.428	7	-	7.0/7.0	3/3.5	9	-
	20	P20	0.256	10	4	0.857/0.429	0.333/0.143	4	-	5.5/6	2.0/2.0	3	-

Table 1: *RESULTS OF ALGORITHM TEST RUNS (a. EAR , b.Slope from dlib coordinates , c. Y-axis distance from dlib coordinates)*

1. *All values are calibrated separately for each subject. The values in the table shown do not have the buffer values added to them.*
2. *EAR has no units, it being a ratio of distances. Similarly, slope has no units.*
3. *The distance values for both eyes are measured in pixels.*
4. *False positives are the number of times the program detected drowsiness when the subject's eyes were open. The accuracy values shown in the table have not taken these into consideration.*
5. *In the cases with zero accuracy, the program was unable to detect drowsiness in spite of successful calibration.*
6. *The buffers are fixed values added to calibrated values to enhance accuracy in detection of drooping eyelids.*

4. RESULTS AND DISCUSSION

ALGORITHM	ACCURACY WITHOUT GLASSES	ACCURACY WITH GLASSES	TOTAL ACCURACY
Slope using dlib coordinates	76%	13%	44.5%
EAR	90%	49%	69.5%
Vertical distance using dlib coordinates	93%	65%	79%

Table 2: *Cumulative Results of Test Runs*

The average accuracy values for both cases (subjects with and without glasses) along with the total average across the entire dataset for each algorithm.

The algorithm involving calculation of average vertical distances yielded the best results, with an overall accuracy of 79%. The algorithm involving the calculation of slopes fared the worst, with an overall accuracy of 44.5%. A dip in accuracy is observed in all three algorithms when the subject is wearing glasses.

A large number of high-end cars already have advanced drowsiness detection systems. As of now, we have implemented the algorithms as simple apps that can be

run on a laptop. However, as this holds little significance in real world situations, the next steps to be taken involve the design of a cost-effective device that can be easily fitted in cars. On the hardware front, we tested our code on an Arduino Uno microcontroller. We made the LED blink when drowsiness was detected. Since an Arduino poses a few problems where image processing is concerned, such as low processing speed and low memory, implementation using a Raspberry Pi looks more promising. In lieu of a webcam on a laptop, a stand-alone device will require a camera to capture the frames. Since a visible-spectrum camera will be unable to capture clear frames in conditions of dim lighting and darkness and the radiation from an infrared camera is harmful to the human eye (it has been shown to cause damage to the cornea, iris and retina), a near-infrared camera (NIR camera) can be used on the device. The Raspberry Pi is compatible with the PiCam NoIR (v2), an 8 megapixel near-infrared camera. As seen in Table 1, some readings were not recorded even in spite of accurate initial calibration. It was also observed that the algorithms (EAR in particular) rendered false positives that compromised the overall accuracy of the results. A further improvement would be to ensure proper working of the algorithms under all conditions and bring false positives to a minimum. For most of the subjects who wore glasses, negligible accuracy values were observed. This can be attributed to the fact that the Dlib shape predictor library often mistakes the upper boundaries of the lenses as the upper eyelids. This phenomenon was observed multiple times during testing. For example, in the case of one subject, we could easily perceive the upper contour settling over the upper boundary of the lens just a few frames into the video stream.

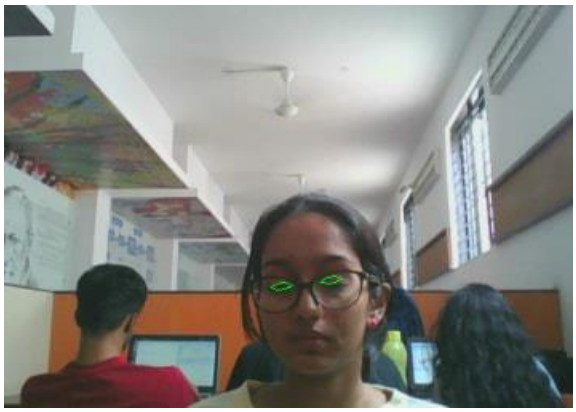

Fig. 5. The above image shows the displacement of the upper contours of both eyes from the actual upper eyelids to the frames of the subject's glasses. This was observed frequently and was shown to heavily impact the readings, often resulting in zero accuracy/ no viable readings.

We also noticed that whenever two or more faces are present in the frame, multiple pairs of eyes are recognized. This obfuscates the program and the results are erratic. A future challenge would be to produce an outcome for solely one pair of eyes in the frame. The quality of calibration can be improved. In our current programs, it is assumed that the user keeps their eyes open or closed as per the instructions displayed. In case they do not follow the instructions, a way to verify the same should be provided. Therefore, it is essential to find whether the eyes are open or closed as per our requirements. Although some Haar Cascade files are designed to detect only open eyes and others, both open and closed eyes, we were unable to achieve our objective by utilizing them as the results we obtained were inaccurate and erratic. One approach that could be undertaken is to train our own cascade file using the Haar training commands available on OpenCV to detect exclusively open or closed eyes. Its accuracy will ultimately depend on the sample size and number of training levels it has undergone.

5. CONCLUSION

In this paper, we compared the results of three algorithms for drowsiness detection, one of which has already been tried and tested. The algorithm involving the calculation of distance between the eyelids yielded the best results. We weighed in on the advantages and disadvantages of alternative solutions and analyzed various methods that could be used to improve on our work. The domain of drowsiness detection is ever-evolving, with the development
of better and faster image processing techniques and the possible use of non-contact electrodes to detect meaningful patterns in physiological rhythms that indicate the state of the driver. As the world around us becomes more fast-paced, stress levels and fatigue will be on the rise and drowsiness detection for accident mitigation will become a necessity to ensure the safety of drivers all around the world.

ACKNOWLEDGEMENTS

We would like to thank the CORI lab, PES University for providing us the opportunity to work on this topic for our summer internship during the months of June and July 2017. We acknowledge the contributions made by our peers Sai Shruthi Nakkina and Kavali Sai Tanya to the documentation and consolidation of the data recorded in this paper. We also thank Dr. K.N.B Murthy, Vice-Chancellor,PES University , for his encouragement during the course of our work.

REFERENCES

[1] The Royal Society For The Prevention Of Accidents. Driver Fatigue And Road Accidents: A Literature Review And Position Paper. February, 2001.

[2] Vandna Saini,Rekha Saini. Driver Drowsiness Detection System And Techniques: A Review.

[3] Samiksha Ravindra Suryawanshi, Deepali Vijay Gawade. A Review On Driver Drowsy Detection System.

[4] Mitharwal Surendra Singh L, Ajgar Bhavana G., Shinde Pooja S., Maske Ashish M. Eye Tracking Based Driver Drowsiness Monitoring And Warning System.

[5] Tereza Soukupova And Jan Cech. Real-Time Eye Blink Detection Using Facial Landmarks.

[6] H.K. Yuen, J. Princen, J. Illingworth And J. Kittler. A Comparative Study Of Hough Transform Methods For Circle Finding.