A new method for detecting fatigue driving with camera based on OpenCV

Xing Li
Department of Electrical and Computer Engineering
University of Illinois at Urbana Champaign
Urbana, IL, USA, 61801
xingli1@illinois.edu

Guang Han, Guangteng Ma,Yanshan Chen
Engineering Research Center of Wideband Wireless Communication and Display Technique
Nanjing University of Posts and Telecommunications
Nanjing, Jiangsu 210003, China

Abstract—**The new system, which will be discussed in this paper, is to make the car slower and gives warning if it finds driver's drowsy eyes. For instance, Toyota Motor Corporation announced that it has given its pre-crash safety system [1] the ability to determine whether a driver's eyes are properly open with an eye monitor.**

This paper is proposing a new method detecting the drowsy eyes in order to implement the fatigue driving detecting system. Eyes are dynamic objects and have highly variability; that is why detecting eyes is pretty difficult. Here, we introduce a new method for detecting the drowsy eyes which is very suitable for hardware implementation and some other processes.

Keywords- Fatigue detecting; image processing; eye detecting; binarization; OpenCV

I. INTRODUCTION

In the last 20 years, most of the automobile giants have been spent efforts on driving safety systems alone with their magnificent master pieces. Some of the safety systems which are basic equipped on all vehicles, such as Airbag System, Anti-lock Braking System (i.e. ABS), ultrasonic warning system and others are all invented few years ago. However, they are still searching for new technology to decrease the accident rate. Some of the automobile companies have introduced a new feature of drivers' safety system.

For the recent decades, automobile has become a major transportation method for people who traveling long distance. When people have a long distance driving, is it really hard for them to stay focus and keep heads up for hours. Even if the well-rested people still cannot keep their eyes open for 8 or 9 hours during the endless driving. And this might cause serious accidents. To prevent and cut such situation happen, engineers have been worked on detecting this situation in various ways. As soon as the detecting system detects the fatigue driving, the system will send an alarm to wake up the driver and give reminders and warnings [2].

For now, methods to detect fatigue driving can be classified as following three: monitoring drivers' body activity; measure drivers' biological signal and monitoring vehicles' coefficients. Among all the above methods, the second one is the most relied and easiest to approach. Recent years, analyzing real-time facial shapes is becoming a popular method which has also been proved to be an efficient one.

Image processing is becoming one of the major methods to analysis features of human body parts nowadays. In order to detect the fatigue driving, the mainly focused features are all on faces. To be more specific, eyes are the most important features that need to be discussed. In this paper, we used OpenCV to analysis the features we measured and build the system we need to detect fatigue driving. The reason we choose OpenCV is that this tool is user friendly and have huge library to process the features we want to analysis. And after the system works in the lab, it is easy to be planted in to DSP and use in practical. This paper will introduce this system from the following aspects: Detecting procedure, Analysis, Verification and Summary.

II. DETECTION

A. Eyes position

Based on OpenCV 1.0, it is very easy for us to detect the face through a camera. To find the face, just load the mode *haarcascade_frontalface_alt2.xml*, which already included in the library of OpenCV 1.0.

The big idea of this function is using Haar cascade classifier [3]. The core basis for Haar cascade classifier object detection is the Haar-like features. These features, rather than using the intensity values of a pixel, use the change in contrast values between adjacent rectangular groups of pixels. The classifier process an integrated image as sub-images. In order to eliminate as many sub-images as possible, only a few of the features that define an object are used when analyzing a sub-image. If the camera can capture the video stream, then process every frame coming into the system, otherwise repeat the capture step. Using *haarcascade_frontalface_alt2.xml*, we can draw a circle on faces on an RGB image. (shown as Fig.2_1) Then according to the circumscribed square of the circle on Fig.2_1, we get a small rectangle which fixed the position of eyes on normal face. If the upper left pinnacle point of the circumscribed square r is pt1, we defined the

Sponsor of NUPT and ITRI

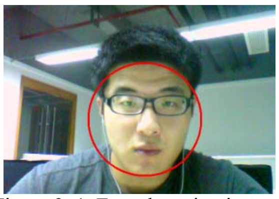
Figure 2_1 Face detecting image with a red circle showing the face of the user

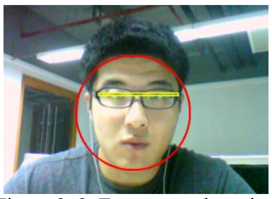
Figure 2_2 Eyes range detecting image with a yellow rectangle showing the minimized eye range

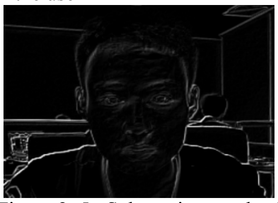
Figure 2_5 Subtraction result of Fig.2_3 and Fig.2_4

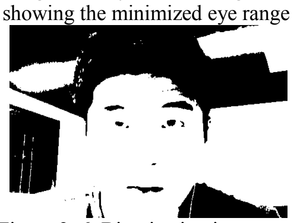
Figure 2_6 Binarization image of Fig.2_5 with a threshold from equation (7)

upper left pinnacle point of the small rectangle *pt1*, and the lower right point *pt2* as:

$$Pt1.x = (r.x+width\times0.2) \quad (1)$$
$$Pt2.x = (r.x+width\times0.9) \quad (2)$$
$$Pt1.y = (r.y+height\times0.33) \quad (3)$$
$$Pt2.y = (r.y+height\times0.35) \quad (4)$$

The width and height above is the size of the square. Thus, the small rectangle has width in a range [0.2, 0.9] of the square width and has height in a range [0.33, 0.35] of the square height [3]. The value was been obtained by experiments. The result is shown in Fig.2_2.

The above steps defined the position of the eyes, which made our following steps easier.

B. *Image Processing*

We need to convert the image to gray image first. And use the pinnacle points we got in the previous step to get the sub-image from the original gray image. In order to extract the eyes from the rectangular small image, we used an improved method based on [2, 4].

In [2], Yuille's team mentioned variable modeling, which can get the position and shape information simultaneously. However, it is pretty slow and has big effect on selection objects. In [4], Dan's team use infrared ray to detect human eyes to calculate the difference between images with light source on and off. The difference image can have the position of eyes.

The method is following:

1. we do morphologically closed operation on gray image with a 5×5 square structuring element in order to get the filtered image f□ :

$$f' = (f \oplus b)\Theta b \quad (5)$$

Note: ⊕ and Θ is morphological erode and dilate operation, in Fig.2_4 shows the result. Fig.2_3 is the original gray image. (Fig.2_4 is a blur image of Fig.2_3)

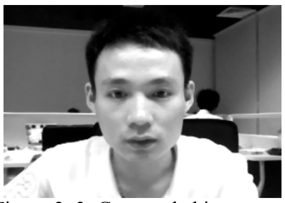
Figure 2_3 Gray scaled image directly from RGB image

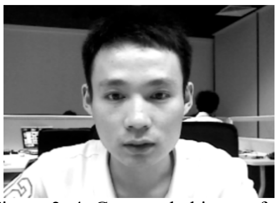
Figure 2_4 Gray scaled image from Fig.2_3 with the morphologically closed operation

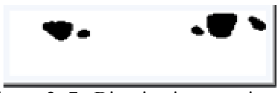
Figure 2_7 Binarization eyes image a patch from Fig.2_6 with a size of rectangle get from Fig.2_2

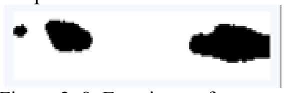
Figure 2_8 Eyes image from Fig.2_7 after morphologically closed operation

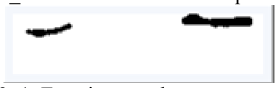
Figure 3_1 Eyes image when eyes are closed

2. Subtract the filtered image with the original gray image, darker the areas in the original image, brighter the areas in the difference image [5], as shown in Figure 2_5:

$$d_{diff} = f' - f \quad (6)$$

3. Get the binarization image from cutting the histogram of gray image by setting the threshold from the equation below:

$$T = \arg\max_{T'} \left[\frac{\sum_{i=T'}^{255} h[i]}{\sum_{i=0}^{255} h[i]} \right] > \epsilon \quad (7)$$

Note: h is the histogram of f_{diff} , and ϵ is a constant which normally choose $\epsilon = 0.04$, (i.e. human eyes are included in the 4% of the black pixels in the binarization image. We then switch black and white in the binarization image in order to analyze the image later, as shown in Fig.2_6.

As shown in Fig.2_6, from the above steps, we can get an image with only facial apertures.

Recall the small rectangle we obtain in the first part *eye position*. With its coordinate, we can also find the appropriate area for eyes on the binarization [6, 7] image as Fig.2_6. In Open CV, using *cvGetSubRect* function to copy the small rectangle area from original image to a small window fits the rectangle area automatically, shown as Fig.2_7.

As the object we detecting is dynamic, the rectangle area is changed through the process.

As shown in Fig.2_7, the black pixels of the eyes are not connected as a single area. Here we want to make as many black pixels connect together as possible. Thus, we did morphologically closed operation, exactly as (5), on the binarization image shown in Fig.2_7. The result is shown in Fig.2_8.

III. ANALYSIS

Detecting drowsy eyes

Big Idea: when we watch the image processing result, we found that when we close our eyes, the number of black pixels in the image (shown as Fig.3_1) is much less than that when we keep our eyes open (shown as Fig.2_8). If the eyes are open, we consider it is not drowsy eyes. However, if the eyes are closed [8] for more than half of a certain time period, we consider it is a drowsy eye and give warning.

Because of our objects are dynamic, we cannot use the number of black pixels as a criterion to judge if the eyes are open. When the object moves further, the image is smaller than when the object is close to the camera. Therefore, we consider distance as a criterion to judge if it is a drowsy eye.

To detect the drowsy eyes, we did following steps to satisfy different people and different distance:

For different people, the sizes of eyes are different. Thus, we had self-adaptation program to solve the problem. Every time start the system, the system will collect the present driver's "eye size", which here considered as number of black pixels.

$$\text{totalcount} = \text{totalcount} + \text{count} \tag{8}$$

Note: *totalcount* is the total number of black pixels through n frames, *count* is the number of black pixels in each frame.

In the system, we define the length of the self adaptation process is 50 frames, which can calculate a fairly accurate value.

$$value_{adapt} = totalcount \div n \tag{9}$$

The above equation gives the average number of black pixels for present driver. With the adapted value, we can match the value with several thresholds we got from experiments. Count the total number of pixels on the small image by ergodic and define it to be *pixel*. In order to fit different people, we need to set appropriate thresholds which get from experiments.

Below are some representative statistics:

Case 1, very big eyes

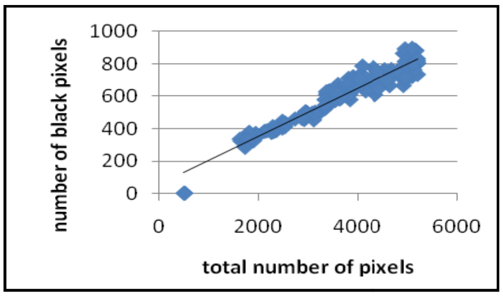

Figure 3_2 the ratio for very big eyes

$$y = 0.149x + 54.808 \tag{10}$$

Case 2, big eyes

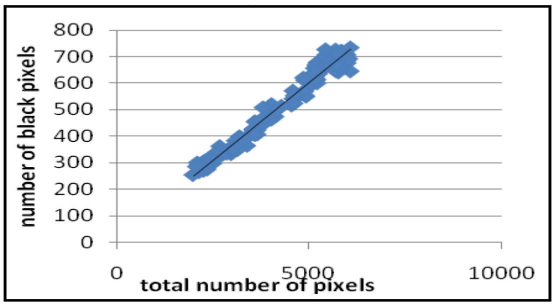

Figure 3_3 the ratio for bug eyes

$$y = 0.1205x + 108.41 \tag{11}$$

Case 3, normal eyes

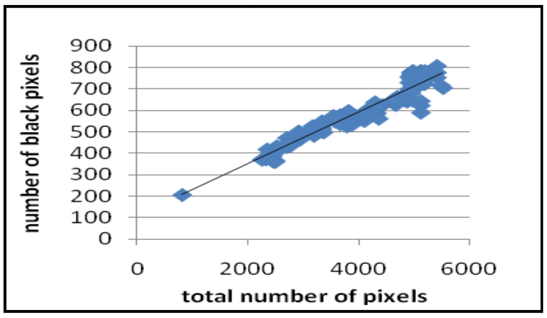

Figure 3_4 the ratio for normal eyes

$$y = 0.116x + 21.225 \tag{12}$$

Case 4, small eyes

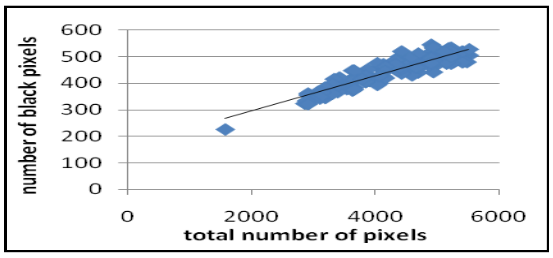

Figure 3_5 the ratio for small eyes

$$y = 0.0656x + 165.28 \tag{13}$$

Case 5, very small eyes

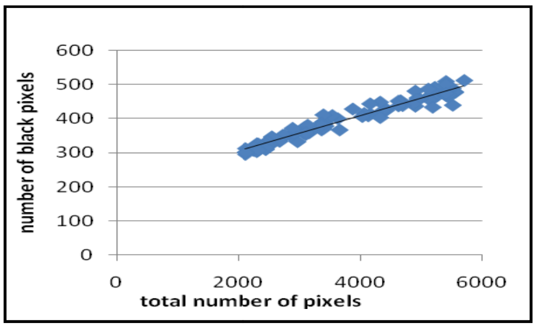

Figure 3_6 the ratio for very small eyes

$$y = 0.0517x + 200.85 \tag{14}$$

Case 6, we left an empty case for extremely big or small eyes.

Note: normally, human eyes have 200 to 1200 black pixels. In the above 6 cases, we separate the total range to 6 parts. During the self-adaptation, the system gives out $value_{adapt}$ (9) as the value for the present driver.

With the above coefficients, we are able to calculate the threshold for every single person. Plug *pixel* (the number of all pixels) into one of the equations above as x and y is the threshold we want. Through multiple times of experiments, the threshold we use is:

$$threshold_{final} = threshold - 150 \tag{15}$$

After the threshold is set, comparing the number of black pixels with the threshold for every frame, if it is no less than threshold, the eyes are open, otherwise, the eyes are closed.

During a time period of 50 frames, if there are more than half of the time (i.e. 25 frames) is less than threshold, we consider fatigue exist. The scale represents PERCLOS [9] calculated over a 3-minute period where PERCLOS is defined as the proportion of time the driver's eyes are closed over a specified period.

IV. VERIFICATION

In the previous step, although we can detect the drowsy eyes pretty accurate, there still exists fake waning. Therefore, we made projection charts for both binarization images of drowsy eyes and no-drowsy eyes cases, as shown in Fig.4_1, Fig.4_2:

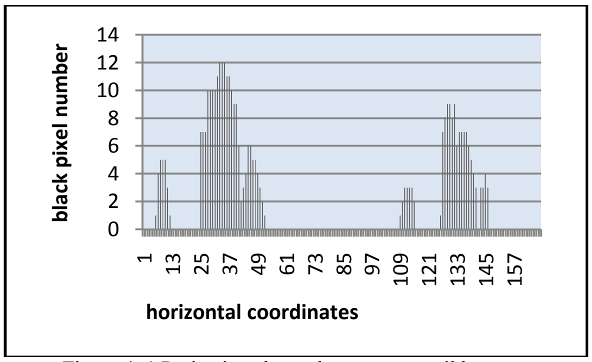

Figure 4_1 Projection chart when eyes are wild open

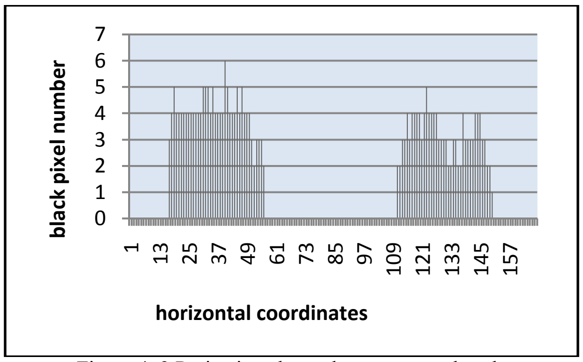

Figure 4_2 Projection chart when eyes are closed

As two charts shown above, Fig.4_1 is the situation that eyes are open, the two valued ranges can be considered as the positions of eyes and the two pinnacle points can be considered as the positions of papilla. Fig.4_2 is the chart for closed eyes.

Comparing the two charts, we can easily find that in eye positions, when eyes are open, black pixel number is almost twice of that of drowsy eyes.

We define the threshold=10, as the result of experiment, number of black pixels in the area of papilla is more than 10. If there are more than 15 pixels more than 10, we considered no drowsy eyes exist. Otherwise, verifying drowsy eyes found and gives warning.

V. COMPARISON

Compare to the previous paper on the same topic, few papers presents the method based on OpenCV.

Chun-Hai Fan's team, [2] discussed driver fatigue detection based on eye tracking and dynamic template matching. They had 4 video streams of eye tracking. And their eye tracking method has an accuracy of 99.1%. Comparing to our experiment, we had 100 video streams from 100 samples we used, and each of them last 2 minutes. By counting the frames which failed to capture the eyes and use the number to divide by the total frame number, we got an average accuracy of 99.7%. Thus, our eye detecting method is robust and irrelevant with different sizes and the mobility of eyes. Shown by their result of fatigue detection, their average Precision Rate is 88.9%, and 3 out of 4 of their videos have 100% precision rate. Compare to our system, we have an average precision rate of 91.6%. Overall, we have a more accurate system than the system discussed in [2].

VI. SUMMARY

Result and conclusion

We tested this system on an Intel i7 workstation and had 100 samples to summaries up the normal people down to the 6 cases discussed in section III. The samples we choose have all sizes of eyes. In these 100 samples, we can capture the eye position accurately for all, and 7 sample fail to detect the drowsy eyes.

During the self-adaptation process, the system needs a very accurate measure value for the following detecting steps. In order to have an accurate result, we have some requirements for setup the camera.

This was tested carefully to determine how much freedom is available when setting up the camera, a potentially crucial point when considering a clinical environment, especially an Intensive Care Unit, which is a prime setting that would benefit from this system [10]. Aside from horizontal offset and orientation of the camera, another issue of concern is the vertical offset of the camera in relation to the user's eyes. The experiments showed that placing the camera below the user's head resulted in desirable functioning of the system. However, if the camera is placed too high above the user's head, in such a way that it is aiming down at the user at a significant angle, the drowsy eyes detection is no longer as accurate as before.

Considering the driving safety, we do not recommend to put the camera on window shield. Our suggestion is to place it on the odometer board. In this way, the camera can catch the face and eyes with a good angle. The system also need drive be static during the self-adaptation process. Here, static means keep the distance between drive's eyes and camera constant. In all of the experiments in which the subjects were seated between 1 and 2 feet from the camera.

Due to the blinks, the system applied PERCLOS method, in which although blinks may be counted by a frame of drowsy eyes but the criterion for fatigue driving is defined by half or more of the frames are fatigue frames, drowsy eyes exist.

The reliability of the system has been shown with the high accuracy results reported in the previous sections. However, there are still some problems can not solved by this system.

Some tests were conducted with users wearing glasses, which exposed somewhat of a limitation with the system. In some situations, glare from the computer monitor prevented the eyes from being located in the motion analysis phase. Users were sometimes able to maneuver their heads and position their eyes in such a way that the glare was minimized, resulting in successful location of the eyes, but this is not a reasonable expectation for severely disabled people that may be operating with the system.

ACKNOWLEDGMENT

Thank for the sponsor of Engineering Research Center of Wideband Wireless Communication and Display Technique, Nanjing University of Posts and Telecommunications and Industrial Technology Research Institute of Kunshan.

REFERENCES

[1] Gyuchoon Cho, “Real Time Driver Safety System”, TopSCHOLAR

[2] Wen-Bing Horng, Chih-Yuan Chen, Yi Chang, Chun-Hai Fan, “Driver Fatigue Detection Based on Eye Tracking and Dynamic Template Matching,” Proceedings of the 2004 IEEE, International Conference on Networking, Sensing & Control, Taipei, Taiwan, March 21-23, 2004

[3] Philip I W, Dr John F, “Facial feature detection using haar classifiers”, Journal of Computing Sciences in Colleges, Vol. 21, No. 4 (2006) 127–133

[4] Dan Witzner Hansen, Riad I, Hammoud, “An improved likelihood model for eye tracking”, Computer Vision and Image Understanding, 2007, 106 (2-3):220-230

[5] R. Grace, “Drowsy driver monitor and warning system,” presented at the Int. Driving Symp. Human Factors in Driver Assessment, Training and Vehicle Design, Aug. 2001

[6] J. Sauvola, M. Pietikäinen. “Adaptive document image binarization”, Pattern Recognition Volume 33, Issue 2, February 2000, Pages 225-236..

[7] Trier, O.D, Taxt, T. “Evaluation of binarization methods for document images”, Pattern Analysis and Machine Intelligence, IEEE Transactions on , Mar 1995, Page 312 – 315, 0162-8828

[8] D.O. Gorodnichy. “Second order change detection, and its application to blink-controlled perceptual interfaces”, Proceedings of the IASTED Conference on Visualization, Imaging and Image Processing (VIIP 2003), pages 140–145, Benalmadena, Spain, September 8-10 2003

[9] R. Grace, “Drowsy driver monitor and warning system,” presented at the Int. Driving Symp. Human Factors in Driver Assessment, Training and Vehicle Design, Aug. 2001

[10] . M.A. Miglietta, G. Bochicchio, and T.M. Scalea. “Computer-assisted communication for 5ritically ill patients: a pilot study”, The Journal of TRAUMA Injury, Infection, and Critical Care, Vol. 57, pages 488–493, September 2004

[11] Michael Chau, Margrit Betke, “Real Time Eye Tracking and Blink Detection with USB Cameras,” Boston Univetsity Computer Science Technical Report No. 2005-12

[12] Intel image processing library (ipl)
http://developer.intel.com/software/products/perflib/ijl

[13] OpenCV library, http://sourceforge.net/projects/opencvlibrary