A Novel Approach for Real Time Eye State Detection in Fatigue Awareness System

H Wang, L. B. Zhou, Y Ying
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
e-mail:zhou0145@ntu.edu.sg

***Abstract*—This paper proposes a novel eye state detection approach to construct an efficient real time driver fatigue awareness system with an ordinary webcam. Eye state detection has given big challenges to researchers as eye block takes only a small part of input image and can show at various appearances for its flexibility. Moreover, light illumination and viewpoint changes cause more confusions and difficulties for PC to robustly extract eye structure such as contours and iris circles. We transfer this tough problem to a classification problem by combining a discriminative feature, namely Color Correlogram, with machine learning method (Standard Adaboost in this paper). The novelty of this work is that we can efficiently and robustly detect eye states in real time with a single ordinary webcam, even in somewhat harsh conditions such as certain lighting changes, head rotation and different objects. Experimental evidence supports this method well and human fatigue conditions are simultaneously measured based on eye states.**

***Keywords*—eye state detection, color correlogram, adaboost, fatigue awareness**

I. INTRODUCTION

Eye information is a crucial factor in facial features. Researchers pay increasing attention on eye state (open or closed) detection with the growing needs of human computer interface, facial expression recognition and driver fatigue awareness systems. Eye state can be obtained using features like eye contour, iris, corners and eyelid. Many geometrical approaches based on eye structure have been developed during the past decade. For example, Hough Transform is applied to detect iris circle after extracting eye contour in binary image using chain code tracing in [1]. However, sometimes contours and iris circles may not be robust enough as there are much too sensitive to illumination conditions and Thresholding techniques. Recently, machine learning methods attract more interests of scholars. In [2], Gabor wavelets and Neural Network (NN) are used to determinate eye states. Eye corners features are extracted whereas their initial positions should be given manually in advance. In the work of [3], Support Vector Machines (SVM) and Naïve Bayes (NB) classifiers are employed for comparison by simply using intensity values of eye blocks. Local Binary Pattern (LBP) histogram features [4] and an Adaboost based cascaded classifier are combined to detect eye states in [5].

In this paper, eye state detection system consists of two modules: eye detection and eye state detection. Much work has been done on eye detection and our contribution focuses on eye state detection. Firstly, eyes are directly detected from frames captured by camera using Viola-Jones object detection method and post-processing. Then we present an excellent representation for eyes and train a binary classifier to detect eye states at runtime. Finally, human fatigue status is to be tracked by introducing a recognized parameter on eye closure level.

The outline for the rest of the paper is as follows: In section II, we simply describes the approaches we use to locate eye regions. Two vital components of eye state detection, eye representation and construction of the binary classifier are presented in Section III and followed Section IV gives the experimental results. Finally, we conclude the paper in SectionV.

II. EYE DETECTION

Various algorithms are available to extract eyes from an input image of visible camera. According to [6], eye detection approaches can be divided to two main classes. The first class, the holistic one, endeavors to search eyes with global representation, which is conceptually associated with template matching. The simplest way is correlation-based matching by scanning raw image from top to bottom. One representative of this class is Pentland et al.'s modular eigenspace [7]. Another class, so-called abstractive approach, extracts discrete local features followed by pattern recognition techniques. Deformable template-based algorithm presented in [8] provides a clear clue of this class.

In temporal sequence, eyes can be tracked from frame to frame given that eyes in initial frame are precisely detected. Eye tracking possesses outstanding low computation property. However, tracking tends to be lost with the occurrence of sudden head movement, large illumination or orientation changes. In this paper, we adopt generic object detection method, i.e. Adaboost learning algorithm together with boosted cascaded Haar-like feature [9]. It is proved to be fast and robust enough for our application, even with somewhat bad illuminations, certain head rotations, only if these conditions are considered during training procedure.

III. EYE STATE DETECTION

To judge the state of a given eye, we treat it as a binary classifier problem instead of direct determination using geometrical feature of the single eye image. In order to

978-1-4244-6506-4/10/$26.00 ©2010 IEEE

construct an excellent classifier, we should collect two sets of eye images, labeled open and closed respectively, as well as find a good representation for each eye. In this section, eye state detection is decomposed into eye feature representation and classifier construction.

A. Eye Feature Representation

1) Eye Image Preprocessing

After taking out eyes from background, preprocessing should be performed to normalize it before feature extraction to achieve better result. Eye images are transformed to a standard size (60×30). Then histogram equalization can be applied to drastically eliminate the effects of lighting variations which would lead to unreliable eye state detection. Afterwards, we use median filter to remove secular reflection spots in iris center. Intensities can be quantized into b_m bins ($b_1 \ldots b_m$, $b_m << 256$) to reduce computational complexity as we adopt intensity-pair based features below. Then the number of intensity pairs will be enormously decreased to b_m^2. Some main results of eye processing are shown in Figure 1.

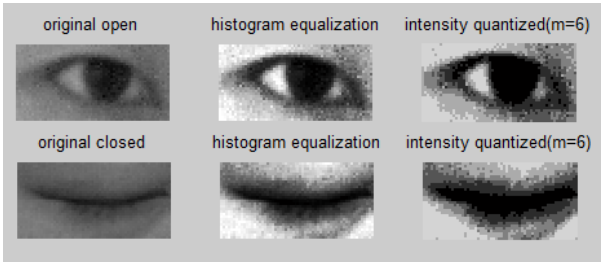

Figure 1. Main results of eye image processing

2) Color Correlogram Features

Before training an eye state classifier, we need to represent each eye image with an informative feature vector. In stead of directly using eye structure information such as contours and iris circles, here we adopt a global statistical feature representation, named Color Correlogram. Color Correlogram was first introduced in [10] for image indexing. It is similar to Gray Level Cooccurrence Matrix (GLCM). Color Correlogram explores the global distribution of local spatial relationship of intensity pairs.

Color Correlogram: Let I be an h × w image after preprocessing, I_{bj} be a pixel of gray level b_j. Spatial distance between two pixels $I_{bi}(x_1, y_1)$ and $I_{bj}(x_2, y_2)$ is measured by chessboard distance ($|I_{bi}-I_{bj}|=\max\{|x_1-x_2|, |y_1-y_2|\}$). Denote histogram of I by

$$hst_{bj}(I) = h \cdot w \cdot pb(b_j) \tag{1}$$

Where $b_j \in$ intensity set M $\{b_1 \ldots b_m\}$, $pb(b_j)$ is the occurance probability of gray level b_j.

Let I (b_i, b_j, d) be an intensity pair (b_i, b_j) with a spatial distance d (see Figure 2). Then a non-normalized elementary entity $N_I(b_i, b_j, d)$ of Color Correlogram is the number of element I (b_i, b_j, d) in image I. It can be described as

$$\begin{aligned} N_I(b_i,b_j,d) &= \sum I(b_i,b_j,d) \\ &= \{I_{bi}(x_1,y_1) \in I, I_{bj}(x_2,y_2) \in I, |I_{bi} - I_{bj}| = d\} \end{aligned} \tag{2}$$

Here d belongs to distance set $\{1 \ldots d_{max}\}$. Regardless of boundaries, equation (3) holds (see Figure.2). Both sides represent the number of pixels at a distance d to any pixel of gray level b_i. Then normalized Color Corelogram feature $\alpha_I(b_i, b_j, d)$ is given as (4). Obviously, equation (5) holds.

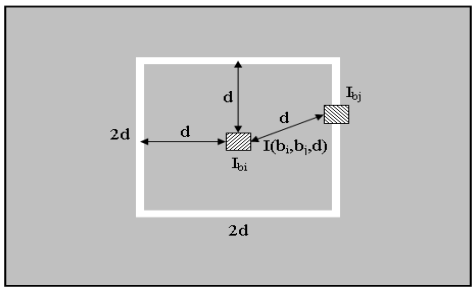

Figure 2. Illumination of elementary entity of color correlogram: an intensity pair with a distance

$$\sum_{j=1}^{m} N_I(b_i,b_j,d) = hst(b_i) \cdot 8 \cdot d \tag{3}$$

$$\begin{aligned} \alpha_I(b_i,b_j,d) &= N_I(b_i,b_j,d) \Big/ \sum_{j=1}^{m} N_I(b_i,b_j,d) \\ &= N_I(b_i,b_j,d) / hst(b_i) \cdot 8 \cdot d \end{aligned} \tag{4}$$

$$\sum_{j=1}^{m} \alpha_I(b_i,b_j,d) = 1 \tag{5}$$

3) *Characteristics* of Color Correlogram

Color Correlogram has many outstanding properties for image indexing and retrieval. It reveals the spatial correlations of intensities and has small feature set size. Moreover, it possesses the merits of classical histogram and outperforms it. Easy and efficient to compute is another strongpoint of Color Correlogram. Above all, Color Correlogram owns the characteristic of rotation invariant since it relates to global statistical features from local pixel relationships. Therefore, the trivial difference between left and right eyes could be ignored. Due to these advantages, we expand the application of this feature representation to real time eye state detection.

4) Eye Color Correlogram Extraction

Before extracting Color Correlogram of each standardized eye image, we should optimize two parameters, number quantized color bins b_m and maximum pixel-to-pixel distance d_{max}, so as to achieve optimal balance between time efficiency and detection accuracy. To address this problem, computational complexity of Color Correlogram can be coarsely modeled as

$b_m^{\,2} \cdot d_{\max}$ while successful detection rate could be examined by experiments. Finally, all features can be stacked into the form of a vector, i.e., each eye is represented by a feature vector of dimension $b_m^{\,2} \cdot d_{\max}$. In our experiment, we adopt $b_m = d_{max} = 10$.

B. Classifier Construction

In previous subsection, an overcomplete feature vector is extracted from hundreds of local regions to describe an eye. However, practically detection of eye states only needs a small portion of these features. Therefore, to obtain all the features seems to be time costly and unnecessary. On the other hand, given a feature representation and a large set of positive and negative samples, various machine learning approaches are available to train a classification function. Among them, Neural Network [13], Support Vector Machine [14], AdaBoost Algorithm [11] have been repeatedly proved to be robust and efficient for classification problems. In this paper, Adaboost Algorithm is used to both select a small set of features and train the classifier. For trial, we simply apply standard Discrete Adaboost to test the robustness and powerful of Color Correlogram features. Adaboost Algorithm was first introduced by Freund and Schapire in 1995. It wisely combines a set of weak classifiers to form a strong classifer (see Table I).

IV. Experiment and Results

In this section, we practically implement and valuate the fatigue awareness system. More emphases are placed on the eye state classifier construction. For more details on eye detection, interested readers could refer to the classic object detection work in [9].

A. Experimental setup

For hardware, we use a very cheap visible webcam, modeled LzyCam PCC3220. The original size of frame captured from the webcam is 640 × 480. To acquire informative eyes, distance from human to camera should be limited to less than one meter, which is sufficient for the requirement of driver awareness systems. Our algorithm is implemented in Microsoft Visual Studio 2005 and OpenCV environment.

B. Eye State Classifier Training And Testing

In the experiment, we collect 1500 positive (open eyes) as well as 700 negative (closed eyes) samples which are horizontally reversed to double the number for training. Afterwards, 1618 positive and 812 negative samples are used to test performance of the classifier. These sets of eyes are partially cropped from database FERET [12] and others captured by us. We include in tens of persons' eyes with different illumination conditions and eye closure degrees to gain a better detection rate. Based on experiment results, we quantize each eye into $b_m = 10$ intensity bins and utilize at most $d_{max} = 10$ pixel-to-pixel distance to extract the feature vector. Therefore, totally 1000 features are used. We only choose the best 50 of them to construct a strong classifier, which means that it iterates T=50 times to get final classifier, each feature chosen one time. Table II demonstrates detection results of the constructed classifier when the input is a set of manually cropped testing eye samples. Detection accuracies for open and closed eyes are 99.26% and 97.54% respectively. These successful detection rates are far better than the results reported in [13], which are 93% and 81%. For comparison, we implement the algorithm based on Local Binary Pattern features proposed in [5] using same training and testing samples and the result is depicted in Table II. Detection accuracies for our algorithm are higher than theirs, with 98.02% and 96.92% successful detection rate for open and closed eyes; Meanwhile the number of features we used is drastically less, with smaller feature set size of 1000 dimensions compared to 68853 dimensions in [5].

TABLE I. AdaBoost Algorithm for learning classifier

- Given p positive and n negative samples $(x_1, y_1)\ldots, (x_{p+n}, y_{p+n})$ where $y_i = \pm 1$.
- Initialize i-th sample's weight $w_1(i) = 1/(p+n)$, where i = 1, p+ n.
- Weak classifiers are simply defined by finding a threshold with minimal weighted classification error to each single feature of X.
- For k= 1, … , N iterations
  - For each feature, train a weak classifier with minimal weighted error.
  - Choose the weak classifier $h_k(x)$ with lowest minimal weighted error $\varepsilon_k = W_k(i) \cdot \sum_{i=1}^{p+n} \left| h_k(x_i) - y_i \right|$.
  - Update $W_{k+1}(i) = \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k}, \text{ if } h_k(x_i) = y_i \\ e^{\alpha_k}, h_k(x_i) \neq y_i \end{cases}$ where Z_k is a normalization factor of W_{k+1} and $\alpha_k = \frac{1}{2} In\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$.
- After T iterations, the final strong classifier is given by the sign decision function $d(x) = \sum_{i=1}^{N} \alpha_i h_i(x)$.

C. Real-time Performance Measurement

In order to evaluate the robustness of the algorithm under real world conditions, we analysis the performance of our system on four video sequences recorded under different environments: indoor, outdoor, inside-car, real driving. It

should be pointed out that camera for these sequences cannot be directly exposed to light sources in case of over exposure. Lighting conditions vary in these videos. Indoor video was recorded in the laboratory with multiple artificial lights while outdoor video was taken in a sunshine afternoon. Then in a still car, lights only come from front and two sides. At last, we test our algorithm in real driving condition. For first three sequences mentioned above, lighting conditions do not vary too much as background maintains the same all the time while in real driving environment, lights change greatly. Table III gives the detection results of these videos.

TABLE II. DETECTION RESULT OF TESTING SAMPLES

Detection error	*Local Binary Pattern*	*Color Correlogram*
False Positive(FP)	25/812=3.08%	20/812=2.46%
False Negative(FN)	32/1618=1.98%	12/1618=0.74%
Overall error	57/2430=2.35%	32/2430=1.32%

TABLE III. DETECTION RESULT OF VIDEO SEQUENCES

Successful detection rate	*Eye detected rate**	*Open State(TP)*	*Close State(FN)*
Indoor video (3026 eyes)	99.17% (3001/3026)	97.34% (2598/2669)	97.11% (323/332)
Outdoor video (3278 eyes)	89.51% (2934/3278)	96.11% (2074/2158)	98.5% (764/776)
Inside-car video (1512 eyes)	96.83% (1464/1512)	96.16% (1277/1328)	94.85% (129/136)
Real-driving (2204 eyes)	81.86% (1886/2304)	93.30% (1685/1806)	81.25% (65/80)

*: Percentage of eyes being successfully detected.

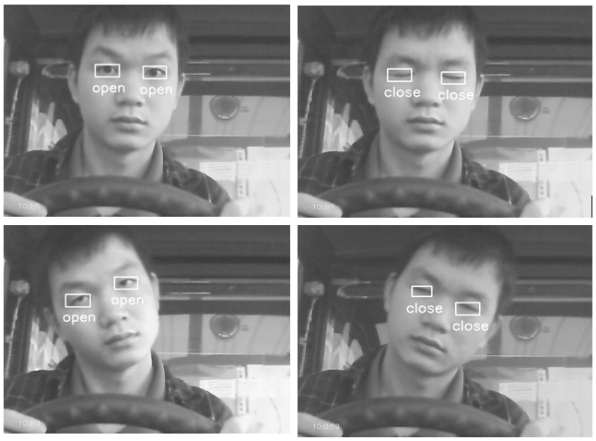

Figure 3. Some correct detection results of video sequences

According to Table III, successful rate of eye detection affects the overall performance of the system in some degree. Only 89.51% eyes in the outdoor video sequence are successfully detected as a result of the bright sunshine. Camera tends to expose with too much light and hence results in images with white regions. Despite that, accuracies of eye state detection based on the detected eyes still turn out to be excellent. For in-door and in-car sequences, our algorithm shows very promising performance as the lighting conditions are relatively better. In real-driving video, quality of input images degrades extremely as unexpected lighting changes and the vibrations of cars and camera. Camera tends to take seconds to stabilize capture environment when big vibrations happen. Consequently, eye detection rate declines to only 81.86% due to poor input image. Even detected, some eyes are so motion blurred that even human cannot easily distinguish the eye states. Even though, our algorithm still shows acceptable performance. Better camera would facilitate and improve our system. In Figure 3, some correct detection results from these videos are presented. Certain head rotations and sudden movements can only make limited effects to the performance of our algorithm. Compared to cropped images, errors for real time application mostly come from motion blur and partial closure of eyes. This could be seen from some incorrect results in Figure 4.

D. Real Time Fatigue Measurement

Percentage eye closure (Perclose) has been widely used for driver fatigue detection among video-based techniques. Perclose is calculated as the ratio of time when both eyes are closed within predetermined time duration. For video sequences, we can divide the number of eye-closed frames by the total number of frames (i.e. 250 frames or 10 seconds). Perclose value increases as the driver's fatigue level increases.

In the experiment, we test our system using a simulated driving video sequence of 14457 frames (about 11 minutes) as the successful eye detection rate are much higher in order to reduce the effects from eye detection part. In this video, we mimics the real-world driving behaviors and fatigue symptoms, intentionally adding more sudden head downward movements, yawning and eye blinks in the second half part. Perclose value is automatically checked every 10 seconds. Perclose analysis of this simulated fatigue process of driving is shown in Figure 5. According to the curve, we choose 0.3 as the optimal perclose threshold. This optimal threshold varies individually due to personal habit and can be obtained by observing a certain period of time. Whenever perclose value exceeds the threshold, a warning signal will be automatically sent out to keep the driver attentive.

V. CONCLUSION

In this work, we proposed a new approach to detect eye states for fatigue awareness system based on classification using a common webcam. It turns out to be either more robust or faster than previous methods. Instead of direct detection by geometrical information, we utilize a discriminative feature representation for eyes and construct an eye-state classifier via using Adaboost algorithm. Our algorithm shows comparable or higher accuracy and efficiency when compared with previously reported works under same experimental conditions. In real time, our algorithm performs fast and well even when sudden head movements, head orientation and certain illumination changes happen.

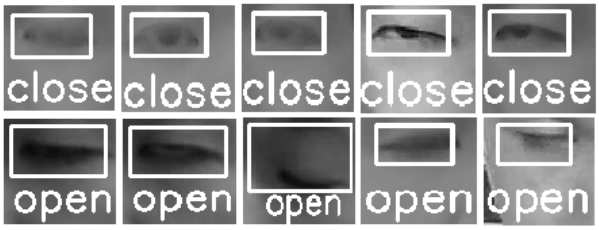

Figure 4. Some incorrect results of eye state detection

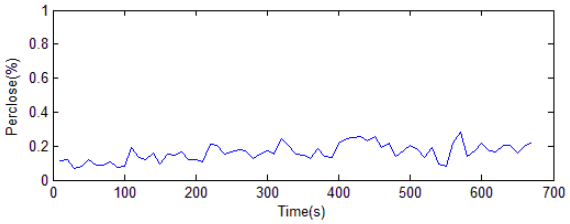

Figure 5. Perclose estimation of a 11 minutes' video sequence

REFERENCES

[1] Q. Wang and J. Y. Yang, "Eye location and eye state detection in facial images with unconstrained background", Journal of Information and Computing Science, Vol. 1, No. 5, pp. 284-289, 2006.

[2] Y. L. Tian, T. Kanade and J. F. Cohn, "Eye-state action unit detection by gabor wavelets", Int. Conf on Multimodal Interfaces 2000, LNCS 1948, Vol. 1948, pp. 143-150, 2000.

[3] R. Senaratne, D. Hardy, B. Vanderaa, S. Halgamuge and D. Liu et al., "Driver Fatigue Detection by Fusing Multiple Cues", Lecture Notes In Computer Science Vol. 4492, pp. 801-809, 2007.

[4] T. Ojala, M. Pietikainen, and M. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Trans. PAMI, vol. 24, pp. 971-987, 2002.

[5] C. Xu, Y. Zheng and Z. F. Wang, "Efficient eye states detection in real-time for drowsy driving monitoring system", IEEE Conf. on ICIA, vol. 1-4, pp. 170-174, 2008.

[6] J. Huang, H. Wechsler, Visual routines for eye location using learning and evolution, IEEE Trans. Evolut. Comput. 4 (1) (2000) 73–82.

[7] A. Pentland, B. Moghaddam, T. Starner, "View-based and modular eigenspaces for face recognition", Proc of the IEEE Int. Conf. on Computer Vision and Pattern Recognition, Seattle, WA, 1994, pp. 84–91.

[8] A.L. Yuille, P.W. Hallinan, D.S. Cohen, "Feature extraction from faces using deformable templates", Int. J. Comp. Vision 8 (2) (1992) 99–111.

[9] P. Viola and M. Jones, "Rapid object detection using a Boosted cascade of simple features," Proc. of IEEE Conf. on CVPR, pp. 511 C518, 2001.

[10] J. Huang, S. R. Kumar, M. Mitra, W. Zhu, and R. Zabih, "Image indexing using color correlograms," in Proc. IEEE CVPR'97, June 1997, pp. 762–768.

[11] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119–139,August 1997.

[12] P. J. Phillips, H.Wechsler, J. Huang, and P. Rauss, "The FERET database and evaluation procedure for face recognition algorithms," Image Vis. Comput. J., vol. 16, no. 5, pp. 295–306, 1998.

[13] C. M. Bishop, Neural Networks for Pattern Recognition: Oxford Univ. Press, 1995.

[14] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," Knowledge Discovery and Data Mining, vol. 2, no. 2, pp.121–167, 1998.