Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Algorithm Optimizations for Low-Complexity Eye Tracking

Shinji Yamamoto and Vasily G.Moshnyaga
Department of Electronics Engineering and Computer Science, Fukuoka University
8-19-1 Nanakuma, Jonan-ku, Fukuoka 814-0180, Japan

***Abstract*—This paper investigates techniques for reducing computational complexity of tracking eyes of computer user. By empirical evaluation of a number of techniques we define a technology capable of monitoring eyes of computer user in real time with high accuracy and very low computational overhead.**

***Keywords*—eye tracking, algorithm, complexity, optimization**

I. INTRODUCTION

A. Motivation

With the wide popularity of user centric applications, the role of smart and intelligent devices, capable of monitoring human eyes is increasing. In addition to "traditional" applications, such as human-computer interface, security, health care and commercial applications, eye tracking has been applied to new fields, such as of drowsy-driver detection [1], display energy management [2]. Up to date, all above mentioned applications, but the last one, have not been constrained by energy consumption. Therefore research efforts have been focused on delivering accurate eye-tracing in real-time. However, as demands to make the eye-tracking ubiquitous or embedded in portable devices grow, the needs to reduce energy consumption become very important.

Although it is easy for a human to detect and track eyes, performing it on hardware requires complex algorithms and many computations. Because each computation burns energy, lowering the computation complexity is considered to be one of the simplest and effective ways to reduce energy consumption.

This paper focuses on minimizing complexity of detecting and tracking eyes of PC user by a single camera. Based on application features [2], we derive optimizations, capable of reducing complexity of original eye-tracking algorithm by 84% while maintaining the same accuracy. Though we target a particular application, the techniques presented here are general and can be used in other applications.

B. Related Research

The techniques proposed so far for eye-tracking may be grouped into two categories. The first one makes use of infrared (IR) devices and exploits the reflective properties of pupils [3-4]. Since pupils become bright under IR-illumination, the eye tracking is accomplished by detecting and tracking pupils using the differential IR lighting scheme. The approach is simple and accurate but requires additional resources in the form of multiple IR sources and sensors.

The work was sponsored by The Ministry of Education, Culture, Sports, Science and Technology of Japan under the Knowledge Cluster Initiative (The Second Stage) and Grant-in-Aid for Scientific Research (C) No.21500063

The second approach attempts to track eyes with the aid of "ordinary" camera in the absence of any kind of IR devices. The techniques can be broadly classified into three groups: template matching, appearance-based and feature-based. The template-based techniques [5-6] are based on generic eye models and their recursive positioning, translation, rotation, and deformation to best fit the eye representation in the image. While these techniques can detect eyes accurately, they require good image contrast to converge correctly and are computationally expensive.

The appearance-based techniques [7-8] detect eyes based on their photometric appearance. These techniques usually employ large amount of training data representing the eyes of different subjects under different face orientations and illumination conditions. These data are used to train a classifier such as neural networks (NN) or support vector machine (SVM) and detection is achieved via classification. Since NN treat each frame independently, many redundant data have to be processed.

The feature-based techniques identify eyes by exploring their distinctive features, such as eye-corner points [9], color distribution of the eyes [10], edge and intensity of iris [11], intensity variation around the eyes [12], etc). Amid large variety of features available, the between-the-eyes point (BTE) [12,13] does not depend on illumination, face occlusion, eye closure and hence is considered as the most stable and robust. When the BTE point is found, the eyes can be easily located as two small darkest points on each side of the point. To detect the BTE point based on circle-frequency filter [12], however, is very computationally intensive. To accelerate the search, Kawato et al [13] proposed to use a less complex Six-Segment Rectangular (SSR) filter that reflects the bright-dark relations in between-the-eyes region of the image. However, even with this filter, the algorithm runs over 28Million operations, such as addition or subtraction per frame (640x480 pixels frame size). In the rest of the paper we address complexity reduction of this eye-tracking algorithm.

C. Related Research

The contribution of this paper is two-fold. First, we empirically analyze the complexity of eye detection and tracking algorithm [13] and present solutions capable of lowering it without affecting the accuracy of results. Second, we propose a modified algorithm that, as experiments show, achieves same accuracy as [13] while reducing the number of computations by 84%.

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

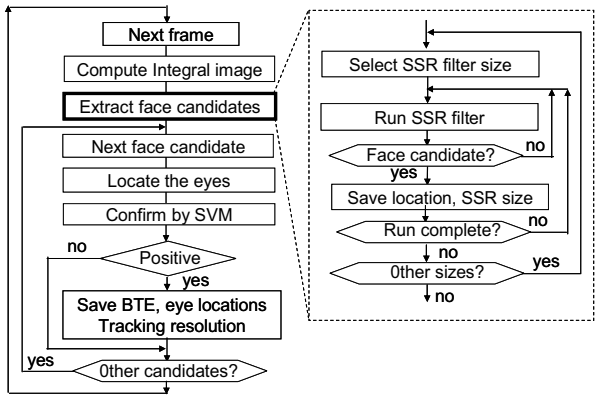

Fig.1. Flowchart of original eye tracking algorithm

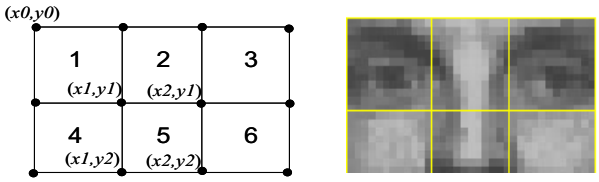

Fig.2. Illustration of the SSR filter.

The paper is organized as follows. In the next section we discuss original eye-tracking algorithm. Section 3 presents algorithmic optimizations and outlines the modified algorithm. Section 4 summarizes our findings and the future work.

II. The Original Eye Detection Algorithm

Before you begin to format your paper, first write and save the Fig. 1 shows the original face detection/tracing algorithm [13]. The system captures RGB video sequence and transforms the green component of each image frame *i(x,y)* into the integral image representation *I(x,y)* as follows:

$$S(x,y)=S(x,y-1)+i(x,y) \qquad (1)$$

$$I(x,y)=I(x-1,y)+S(x,y), \qquad (2)$$

where *S(x,y)* is the cumulative row sum, *S(x,-1)*=0, *I(-1,y)*=0.

The integral image is then scanned (pixel by pixel) by a six-segment rectangle (Fig.2), computing the sums of pixel values in each segment. For instance, the sum of pixels within rectangle *5* is defined as: *Sum(5)=I(x2,y2)+I(x1,y1)-I(x2,y1)-I(x1,y2)*

The SSR filter then compares the computed sums of the segments as follows:

$$Sum(1) < Sum(2) \text{ and } Sum(1) < Sum(4) \qquad (3)$$

$$Sum(3) < Sum(2) \text{ and } Sum(3) < Sum(6) \qquad (4)$$

If the above criteria (3)(4) are satisfied, the SSR is considered to be a candidate for Between-The-Eyes (BTE) pattern and two local minimum (i.e. dark) points each are extracted from the regions *1,4* and *3,6* of the SSR for left and right eye candidates. As the eye candidates are selected, the BTE pattern is normalized by scaling, rotation and histogram equalization and then fed to Support Vector Machine (SVM) for confirmation. Thus for one face candidate, at most 4 patterns are tested.

Table 1: Number of computations per frame (x10^3)

Function	Image size	
	320x240	640x480
Integral image computation	76.2	306.1
Face candidate extraction	5867.6	27869
Eye confirmation by SVM	82.9	147.3

The search is then repeated for 6 other SSR filter sizes (120x72, 80x48, 60x36, 40x24, 30x18 and 20x12) in order to extract eyes of smaller faces if any. For the next frame, the BTE location is predicted as $x_p=2x_1-x_2$, where x_p is the predicted position, x_1 and x_2 are its positions in the most and the second most previous frames, respectively. Any incorrect prediction starts the full search.

We implemented the algorithm in software and used specific instructions to count the number of computations performed. As results (Table 1) show, the face candidate extraction (see Fig.1, right side) accounts for most of computational load. Because the face/eyes detection is general (no restriction on the number of faces, face size, motion, and rotation), it implies the following:

1. Full SSR filter scan over the whole image frame.
2. Repeats the scan six times (for all filter sizes).

Although this full search is necessary to find multiple faces in an image in general case, it might be redundant in particular applications. Taking into account features of target application is the only way to reduce complexity of the task.

III. Complexity Reduction Techniques

A. Assumptions

In this study we focus on application [2] that uses eye-tracking to monitor gaze of computer user and lower the power consumption of computer display whenever the user detracts his/her eyes from its screen. In contrast to a general case, we assume that

1. The target object is a single PC user. The user sits in front of PC at a relatively close distance (50-70cm).
2. The camera is located at the top of display. When the user looks at display it faces the camera frontally.
4. The user's motion is slow relatively to the frame rate.
5. The background is stable and constant.

Based on the assumptions, we explore several optimizations.

B. Filter Size Optimization

First we studied the influence of the filter size optimization on the detection rate and complexity. If we consider images of a person, which were taken by same camera from various distances, the interval between his/her eyes changes with the distance, as shown in Fig.3. Namely, the closer the camera, the larger is the interval. When the distance is about 50-70cm, the interval is about is 45-58 pixels. From experiments, we found for this distance range, the BTE of 55 pixels ensures almost 100% detection rate.

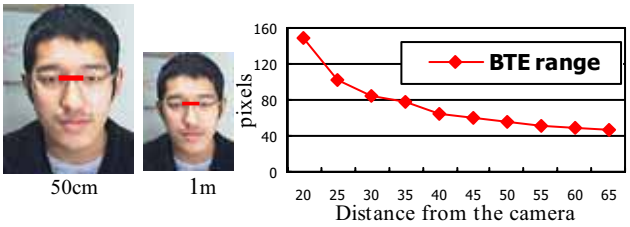

Fig 3: BTE variation with the user's distance from camera: user images captured from 50 cm and 1m (left); the dependency graph (right)

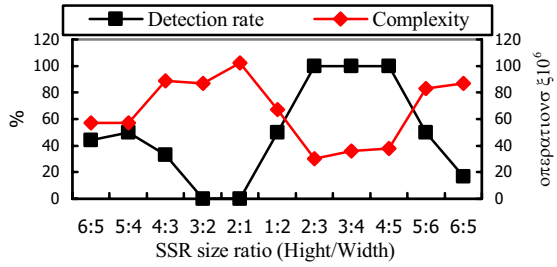

Fig 4: The effect of SSR filter ratio on the results

Next, we empirically estimated the most suitable SSR filter size ratio (*R=Height/Width)* for the selected BTE. We used 180 facial images (92x116 pixels, 256 gray levels) from FERET data base, re-sized when necessary to have the BTE interval in 45-58 pixel range. For each filter size we manually checked whether it covered two eyes, two eyebrows and cheekbone area, including nose. The ratio of the positive cases to the total number of tasted cases represented the detection ratio. Fig.4 shows the results obtained for different values of *R*. Clearly, the filter ratio (*R*=2:3) ensures minimal complexity at the highest detection rate. The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

C. Enlarging the SSR displacement

In the original algorithm (Fig.1), the SSR filter is scanned over the image in the raster-scan mode pixel by pixel, i.e. with the displacement (D) between two consequent SSR locations of 1 pixel. If we increase the distance (D), the detection rate almost does not change while the number of operations decreases significantly, as shown in Table 2. In this table, the detection rate shows the ratio of the number of times the eyes were detected in regions 1 and 3 of SSR to the total number of trials. We observe that the computational complexity decreases by a factor of 3 for D=2, and by a factor of 4.5 for D=3 without affecting the detection rate of the original (full scan) algorithm.

D. Background Subtraction

Background subtraction is a commonly used class of techniques for segmenting out objects of interest. The main goal is given a frame sequence (f_1 f_2, f_3,…,f_n) from a fixed camera, detect the foreground object with highest accuracy and minimum number of computations. Because in our target application, the object (i.e. the PC user) is not moving fast, the simplest solution to detect foreground is to find the difference between the current frame f*t* and the previous frame f_{t+1} that is larger than a threshold (*T*) [14], i.e. $|f_t - f_{t-1}| > T$.

Table 2: Effect of the SSR displacement on the results

SSR displacement in pixels	Detection rate (%)	Operations ($x10^6$)	Reduction factor
1	100	36.13	1
2	100	11.89	3.04
3	100	7.89	4.57
4	94	7.33	4.93

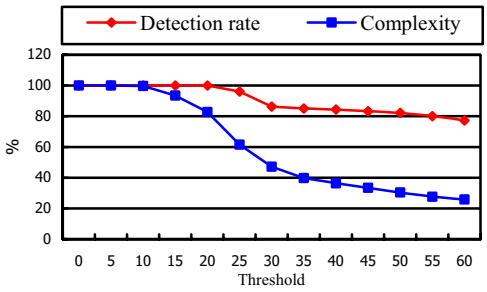

Fig 5: Detection accuracy and complexity vs. threshold

Fig. 5 shows variation of the detection rate and complexity normalized to the full search, empirically evaluated for different threshold values. The tests revealed that when *T<20*, the search area expanded almost over the whole frame because most of pixels were treated as foreground. This large search leads to 100% detection rate but costs many redundant operations. As the threshold increases, the search area shrinks, because fewer pixels belong to foreground. This reduces computational load of the algorithm but lowers the detection rate. For example, at *T*=60, the algorithm has 26% of original complexity and 77% detection rate. The value of *T*=25 provides a tradeoff with 40% of complexity reduction and 96% detection rate in comparison to the full search.

E. Background Subtraction

Skin-color is a well known-technique to filter out the non-faces in image frames. To define skin we use the following criteria [15]: 0.55<R<0.85, 1.15<R/G<1.19, 1.15<R/B<1.5, 0.6<(R+G+B)<1.8.

Furthermore, to speed-up the face-area extraction, we use two filters. The first one limits the size of the head in reasonable range. The second one verifies that the face contains a minimum of 25% of skin colored pixels.

Fig.6 shows the results of applying the optimization to the results of the background extraction. As one can see, at T=25, the optimization lowers complexity by 23% with the detection rate loss of 7%.

F. Frame rate reduction

Next we evaluated the influence of the frame rate on detection complexity. Regarding to the frame rate of camera

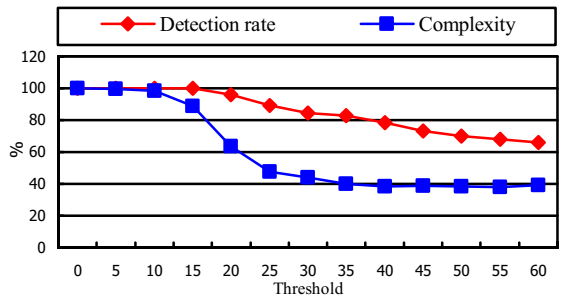

Fig 6: Results of joint background extraction and skin-color based search reduction

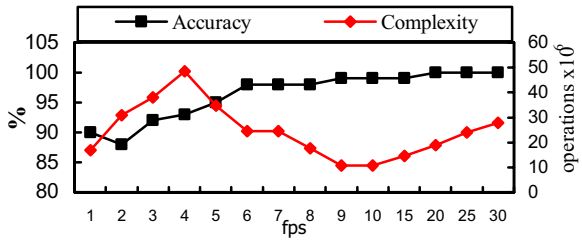

Fig 7: Detection accuracy and complexity vs. frame rate

readings (30 fps), the movement of PC user is very slow. Hence, tracing the PC user by using the highest frame rate is usually redundant because the lower rate brings the same results with fewer computations.

Fig. 7 shows the results obtained for 3min. long user detection at different frame rates (measured in frames per second). As we process less frames per second, both the total number of operations performed per second and the detection accuracy decrease though the number of computations per frame increases due to larger difference between the frames and frequent mis-prediction of eye locations. Clearly, the 10 fps rate ensures minimum computations at maximum accuracy.

G. The Modified Algorithm

Based on the proposed optimizations, we modified the original eye tracking algorithm, as shown in Fig.8. For initial frame or frames, in which the BTE point is unknown, the algorithm runs background extraction and skin-color segmentation to reduce thesearch area (S) to face. Else, it searches only a small area (S) of ±8 pixels around the BTE point. For the chosen area (S), the algorithm computes the integral image and scans it by the SSR filter with D=3 to accelerate the BTE search. The eye-location and verification steps are done as in the original algorithm.

In comparison to original formulation, our algorithm not only scans the SSR over a smaller image area, but scans it faster. Fig.9 illustrates the search area reduction of our algorithm: dashed line shows the area defined by background extraction; dotted line depicts the area obtained by skin-color-segmentation; the plain (dark line) shows the area around the previous BTE point; white crosses show the computed locations of eyes.

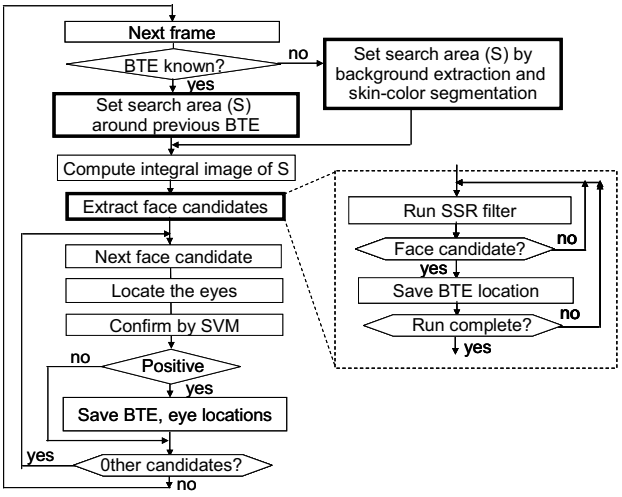

Fig.8. The modified eye-tracking algorithm

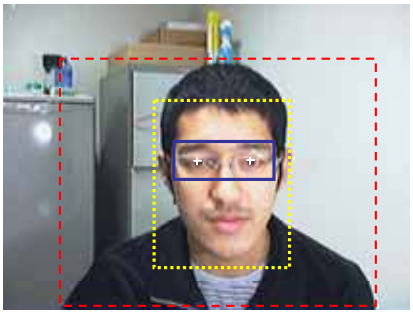

Fig.9. An illustration of the search area reduction

Table 3: Results of evaluation on test sequences

test	Frames	True/False		Accuracy (%)		CFR (%)
		Or.	Mod.	Or	Mod.	
1	151/16	142/9	14/1	94	93	84
2	240/25	217/23	22/2	88	87	87
3	100/10	99/1	10/0	99	100	82
Av.	164/17	153/11	15/1	94	94	84

Additionally to algorithm modification, we propose to run the eye-tracing at low (10fps) frame rate. Fig.10 profiles the amount of computations performed per frame by the modified algorithm (10 fps) and the original one (30fps) in a test, which had the PC user looking at display (camera) and for 123 frames then not looking at the display (camera) at all. The peaks in profile of the original algorithm reflect eye-blinks and miss-predictions. By analyzing the performance on all 151 frames we found that the modified algorithm traced the human eyes by only 1% less correctly than the original one (94% accuracy), while taking only 16% of computations.

Table 3 shows comparisons on 3 different tests. Here, columns marked by 'Or.' and 'Mod.' reflect the original and the modified algorithms, respectively; CRF is complexity reduction factor, calculated as ratio of the total number of

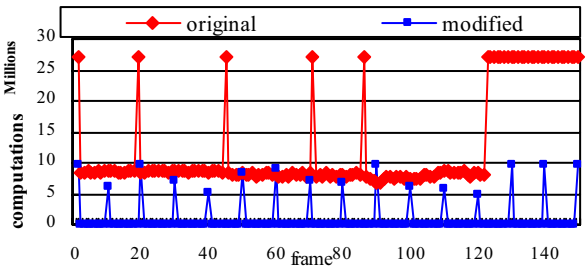

Fig.10. Complexity comparison per frame

computations taken by the modified algorithm to those taken by the original algorithm; 'Av.' is the average. We see that proposed modifications decrease complexity of the original algorithm by 84% on average without affecting accuracy of eye-tracking.

IV. Conclusion

In this paper we have studied computational savings in eye/face monitoring due to the reduction of the scan area. Combining all the discussed techniques produces an eye-tracking system with satisfactory accuracy and significantly better performance than the original one. In this study we have not tacked the computational complexity of the SVM-based decision making. Because the SVM searches a huge space, it is imperative to investigate ways to reduce its complexity of the search. A comparative analysis with other eye-tracing systems will also be conducted in the near future.

References

[1] M.Kutila, "Methods for machine vision-based driver monitoring applications", VTT Research, 2006, Available from http://www.vtt.fi/publications/index.jsp

[2] V.G.Moshnyaga, E.Morikawa, "LCD display energy reduction by user monitoring", Proc. IEEE ICCD, 2005, pp.94-97.

[3] Y.Ebisawa, "Improved video-based eye-gaze detection method", *IEEE Trans. Instrum. Meas.*, vol.47, no.2, pp.948-955, 1998

[4] C.Morimoto, M.Flickner, "Real-time multiple face detection and using active illumination." Proc.IEEE Int. Conf. Automatic Face and Gesture Recognition, 2000.

[5] X.Liu, et al., "Real-time eye detection and tracking for driver observation under various lighting conditions", IEEE Intelligent Vehicle Symp. 2002

[6] K.M.Lam, H.Yuan, Locating and extracting the eye in human face images, *Pattern Recognit*., vol.27, pp.99-111, 1998

[7] W.Huang and R.Mariani, "Face detection and precise eye location", Proc. Int. Conf. Pattern Recognition, 2000

[8] M.Reinders, et al, Locating facial features in image sequences using neural networks. Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition, 1997

[9] G.C.Feng, P.C.Yuen, "Variance projection function and its application to eye detection for human face recognition", *Int. J. Computer Vision*, vol.19, pp.899-906, 1998.

[10] S.Amarnag, R.S.Kumaran, J.N.Gowdy, "Real time eye tracking for human computer interface"s, Proc. ICME 2003, vol.3, pp.557-560.

[11] Y.Tian, T.Kanade, J.F.Coin, "Dual-state parametric eye-tracking", Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition, 2000.