A LOW-POWER FPGA IMPLEMENTATION OF EYE TRACKING

Tomoaki Ando, Vasily G. Moshnyaga, Koji Hashimoto

Department of Electronics Engineering and Computer Science, Fukuoka University, 814-0180 Japan

ABSTRACT

The use of eye-tracking in energy-constrained applications requires systems capable of tracking human eyes without consuming large amount of power. In this paper we present a low-power implementation of real-time non-intrusive eye-tracking by single camera. Unlike existing hardware designs, our eye-tracker does not restrict the user nor requires computationally expensive tools for maintaining high quality tracking. The system provides over 88% eye detection accuracy at 8f/s image processing rate. We describe the hardware and present the results of its experimental evaluation.

Index Terms— eye tracking, FPGA design, low power

1. INTRODUCTION

With wide popularity of user centric applications, the role of smart devices, capable of monitoring human eyes is increasing. In traditional applications, such as human-computer interface, security, health care, commercial applications, etc., eye tracking has not been constrained by energy consumption. Therefore research efforts have been focused on delivering non-intrusive, non-restrictive and accurate eye-tracking in real-time. However, as demands to embed eye-tracking in portable devices grow, the need to reduce energy consumption becomes very important.

Unlike commercial eye-tracking systems, such as [1], powered from the wall, battery-operated portable devices have very limited energy budget. In order to be used portably, the eye-tracking must be optimized to reduce the energy drain from batteries, prolonging the operation time between recharges. Though portability is the main force behind the push for energy-conscious eye-tracking, other eye-tracking applications also require low-energy systems. One example is power management of computer display [2]. This new technology tracks eyes of display viewers and dims the display down to save energy whenever nobody looks at its screen. Lowering the energy consumption of eye-tracking system is of paramount importance for this application.

2. RELATED RESEARCH

The non-restrictive non-contact eye-tracking systems proposed so far may be grouped into two categories. The first one (e.g. [3-5]) makes use of infrared (IR) devices and exploits the reflective properties of pupils. The eye tracking here is simple, accurate but energy consuming due to multiple IR sources. Systems of the second group track eyes with the aid of "ordinary" camera in the absence of any kind of IR devices. Baluja et al. [6] advocate Artificial Neural Networks for eye tracking. One major drawback of this system is the number of eye images that are required to train the neural network. Also each frame is treated independently, which results in processing of redundant data. Smith and Pitas [7] use color predicates to locate face and the eyes using minima analysis. Due to high sensitivity to lighting conditions, this method, however, is not robust.

Pure hardware systems are typically implemented as Application Specific Integrated Circuits (ASIC). Compared to other technologies, ASIC have a high operating frequency resulting in better performance, low power consumption, high degree of parallelism and well established design tools. However, a large amount of development time is required to optimize and implement the designs. Besides the ASIC solutions are not flexible and can not be changed, resulting in high development costs and risk. The software implementation of eye tracking in DSP offers great deal of flexibility coupled with parallel data processing. The drawbacks of microprocessors are both higher power consumption, and inferior performance compared to custom ASIC. Configurable platforms, such as FPGA, combine the advantages from both pure hardware and pure software solutions. More specifically, the high parallelism and computational speed of hardware and the flexibility and short design time of software. Therefore majority of existing face and eye tracking systems [8-10] are implemented on programmable devices (FPGA, microcontrollers) integrated with embedded software based on multiprocessor platform. In a quest to achieve high accuracy of face-detection in real-time, these systems run complex video processing algorithms, which require a huge number of computations. For example, the system [10] initially scales an input 300x300 image into five images of 240x240, 180x180, 120x120, 60x60, and 20x20 pixels, respectively, and then processes them in parallel for histogram equalization, neural network (NN) classification, image rotation, brightness and contrast adjustment, NN-based face detection, etc. To perform over 163million multiply-accumulate operations per image frame, the system employs a wide network of processors and memories, consuming over 5W of power.

Up to our knowledge there has been reported so far only one system [11], capable of performing real-time eye-tracking with less than 200mW power overhead. This system however requires the user to be positioned at 50-60cm from camera, which is quite a strong limitation.

In this paper we contribute to the previous research by introducing a new FPGA implementation of non-restrictive, non-contact low-power system capable of performing robust, real-time tracking of human eyes up to 1.5m distance. In the next section we describe the eye-tracking algorithm and its implementation in FPGA. Section 4 shows the results of experimental evaluation. Section 5 presents conclusions.

The work was sponsored by The Ministry of Education, Culture, Sports, Science and Technology of Japan under the Knowledge Cluster Initiative (The 2nd Stage) and Grant-in-Aid for Scientific Research (C) No.21500063

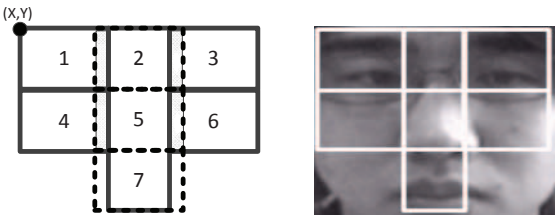

Fig.1. An illustration of the seven-segment rectangular template

3. PAGE TITLE SECTION

3.1. The eye-tracking algorithm

The system takes a gray-scale image produced by camera and outputs positions of human eyes. The eye-tracking algorithm has two distinctive features. First, it uses new seven-segment rectangular (SSR) template (see Fig.1) to track the between-the-eyes point (BTE), not eyes). In contrast to conventional six-segment rectangular template [10], the SSR contains an extra region (marked by 7 in the figure) that reflects the area between mouth and nose. Similarly to areas 1 and 3, this region is usually darker than the nose area 5 and therefore can be applied used to discriminate false candidates from the true BTE pattern. Second, the algorithm does not require computationally expensive tools (such as, Ada-Boost, SVM, Neural Networks, etc.) for BTE candidate selection. The false candidates are discriminated based on a set of simple rules that reflect bright-dark relations between regions in the BTE template.

Fig.2 shows flowchart of our eye-tracking algorithm. For the first frame or any frame in which eye locations are unknown, we extract the foreground image (which contains the face) and scan it to locate the BTE and eyes; otherwise we scan a small area (S) of ±8 pixels around the eye positions, found for the previous frame. For the chosen area, the algorithm first computes the integral image and then scans it by the Seven-Segment Rectangular to select the BTE candidate. If the BTE candidate is found, the system uses the BTE as a starting point to locate eyes. If eyes are detected, the user is assumed to be looking at screen; else it is not. Below we discuss the main steps in details.

3.1.1. Foreground extraction

The goal of this task is to reduce search area in the input image. Because a human face is not static and moving, we search only foreground area, which contains human face by default if any. The image foreground is extracted by computing difference between the current image frame F_t and the previous frame F_{t+1} that is larger than a threshold, T [12], i.e.$|F_t - F_{t-1}| > T_1$.

3.1.2. Integral image computation

The integral image representation I(x,y) is computed in one pass over the image i(x,y) based on the following equations [13]:

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

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

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

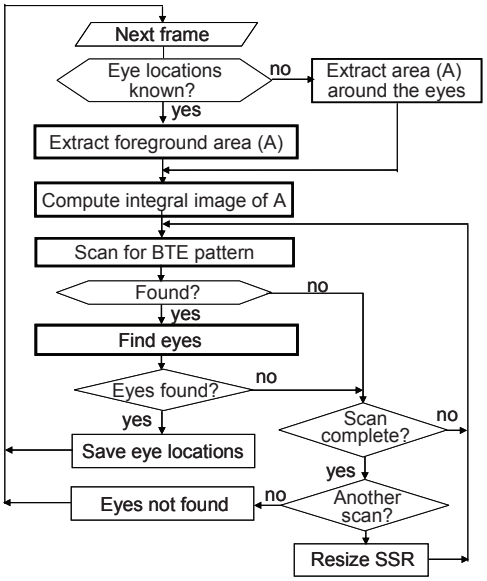

Fig.2. The eye-tracking algorithm

Computing the sums is simple; it requires only 1 addition, 2 subtractions and 4 lookups to calculate the sum of one SSR segment. For example, the sum of pixels within region *5* of Fig.1 is defined as: $S_5=I(x2,y2)+I(x1,y1)-I(x2,y1)-I(x1,y2)$. The produced integral image is scanned by the SSR rectangle of pre-defined size to locate the BTE pattern i.e. face candidate.

3.1.3. BTE pattern detection

To detect a BTE pattern in an image, we scan the SSR over the search area in a row-first fashion. At each scan location, the integral sum (S_j) of pixel values in each rectangular segment *(j)* is compared to the integral sums of other segments to select the true BTI candidate. (For the comparison, the integral sums of segments 2, 5 and 7 are defined from areas shown by plotted line in Fig.1). The comparisons are done as follows:

$$S_1 < S_2 \;\&\; S_1 < S_4 \quad (3)$$

$$S_3 < S_2 \;\&\; S_3 < S_6 \quad (4)$$

$$S_4 < S_5 \;\&\; S_6 < S_5 \quad (5)$$

$$S_7 < S_5 \;\&\; S_7 < S_4 \,||\, S_7 < S_6 \quad (6)$$

$$S_{3,6} < S_{2,5} \,||\, S_{1,4} < S_{2,5} \quad (7)$$

$$S_2 < S_5 + T_2 \;||\; S_{3,6} < S_{2,5} + T_3 \;||\; S_{1,4} < S_{2,5} + T_4, \quad (8)$$

Here $S_{i,j} = S_i + S_j$; T_2, T_3 and T_4 are the thresholds.

The criteria (3-8) reflect characteristic features of face geometry, i.e. the nose area (see regions 2 and 5 in Fig.1) is brighter than eye area (regions 1 and 3 respectively) and the mouse area (region 7); and the eye area is relatively darker than the cheekbone area (regions 4 and 6) (including nose). If the above criteria (8-13) are satisfied, the current SSR is considered to be a candidate for the BTE pattern (i.e. face candidate).

To verify the finding, we evaluate the bright-dark patterns in the lower half of segments 1 and 3 of the BTE region candidate. The upper half in each of these segments corresponds to eyebrows or hears (see Fig.3, a) and therefore ignored. Let E_1,E_2 and E_3 be three vertical (and horizontal) areas in the low half of segment (1 or 3) as shown in Fig.3(b)-(c) and R_1,R_2 and R_3 be the average values of pixels in these areas. Then the candidate is counted as

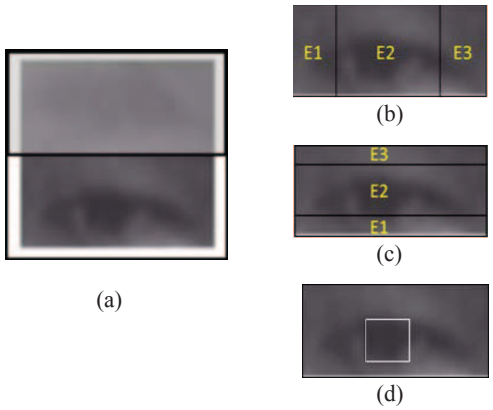

Fig.3. An illustration of eye detection: image corresponding to segment 3 of SSR (a); the image partition in vertical direction (b) and horizontal direction (c); the result of eye detection (d)

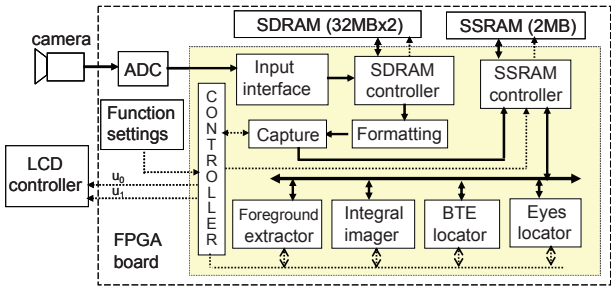

Fig.4. System architecture

true BTE pattern if *R4<R5 & R6 <R5*. Otherwise, the search continues. If no BTE candidate is found, we scale down the size of the SSR by a factor of $1/\sqrt{2}$ and repeat the search. If no BTE found even for the smallest SSR size, we look for another face area if any. If no more areas to search, the user is considered neither present in front of the camera/display nor viewing the display. Any true BTE candidate is supplied to eye detection.

3.1.4. Eye detection

The eye localization procedure is organized as a scan over the grey-scale representation of region E_2 in segments 1 and 3 of the BTE pattern for a continuous set of dark pixels (i.e. whose value is lower than a given threshold). For each detected region we then compute the circumference ratio $D= 4*\pi*A/L^2$, where A is the area of region, and L is the circumference length of the region. Since the higher D corresponds to a more true circle, we select those regions which have $0.3<D<1.0$ and $30<A<150$. Fig.3 (d) illustrates the result.

3.2. System implementation

The system was synthesized from Verilog HDL using Synopsis Design Compiler and implemented on a single board (DE2-70) which contains Analog-Digital Convertor (ADC), two SDRAMs of 32MB each, one 2MB SSRAM and Altera Cyclone-II 2C70 FPGA.

Table 1: Processing time

FPGA Unit	Clock cycles required	Processing time (ms)
Image capture/formatting	419,086	15.5
Foreground extractor	986,518	36.5
Integral Imager	494,150	18.3
BTE and Eye locators	1,312,400	48.6
Total	3,212,154	118.9

Table 2: Power consumption

Power	mW
Dynamic	76.78
Static	155.60
I/O	146.07
Total	375.45

Fig.4 outlines system architecture. The pattern here represents FPGA functionality. The board is connected to CCD image sensor (MTV-2510EM) through parallel I/O interface. The digitized input data, which initially is written to SDRAM, is assembled by Formatting and capture units into image frame of 320x240 pixels in size and saved temporally in SSRAM. The frame processing is done by the Foreground extractor, the Integral imager, the BTE locator and the Eye locator each of which implements a corresponding step described in subsections 3.1.1~3.1.4, respectively. Overall system control is done by CONTROLLER, while memory accesses are managed by local controllers. The system operates at 27MHz frequency and 3.3V external voltage. The eye-tracking implementation in FPGA consumed 8878 logic cells or 13% of FPGA cell budget, 28 multipliers and 53184 bits of SRAM.

Table 1 and 2 summarize the design characteristics in terms of clock cycles, processing time and power. The power consumption was evaluated by Altera Quartus II Power Analyzer from the logic synthesis results. As measurements show, the system requires 118ms to process a frame at 27MHz clock frequency; i.e. 8 fps image processing rate. The total power consumption of the eye-tracking design is 375mW.

4. EVALUATION

To evaluate accuracy of the eye-tracking system, we ran ten different tests each of which conducted by a different user in typical conditions of PC usage. Namely, the users were located at the distance from 40cm to up to 1.5m from display/camera; the room illumination was relatively good; when looking at display, the users faced camera frontally. In the experiment, the users were asked to conduct seven behavioral patterns: face the camera with eyes open and closed, face turned up and down, face turned to the right and to the left, and leave the PC. Each pattern was 5 sec long and repeated twice by each user.

Fig.5 illustrates the detection results displaying the locations of true BTI and eyes. The white rectangle in each image outlines the area searched. Experiments showed that ordinary pairs of glasses have no bad effect on the performance. The system correctly distinguishes open and close eyes tracking the eyes correctly when the face inclination in vertical or horizontal directions is less than

Fig.5. Examples of eye tracking images

Table 3: Power consumption

Case	State	Number of success	Accuracy (%)
1	Frontal face, open eyes	186	89
2	Frontal face, close eyes	185	88
3	Face turned right	206	98
4	Face turned left	206	98
5	Face turned up	189	90
6	Face turned down	189	90
7	No user in front of camera	210	100

Table 4: Comparison

Design	FPGA	Freq. MHz)	Frame size	Power (W)	Tracking distance	Frame Rate	Accuracy (%)
[11]	Xilinx XC3S 250E	48	160x 120	150	50-60 cm	10	88%
Ours	Altera Cyclone II 2C70	27	320x 240	375	40-150 cm	8	88%

15°. With the larger inclination, the misdetection rate increases. Also, in some face orientations, frame of pair of glasses can hide a part of eye ball, causing the system to loose the eye.

Table 3 summarizes the results in terms of the number of frames for which the eye-tracking was successful. The case 1 in the table reflects the number of successes for true eye gazing, and therefore actually can be interpreted as the true-positives. All the other cases reflect situations when the users are not looking at camera. Therefore successes in these cases reflect "no eyes" detection. (i.e. false negatives). Although the detection ratio depends on the case, the accuracy of the decisions is quite high: more than 88% accuracy.

Table 4 compares our implementation with the related design [11]. Although our system consumes a little more power than the prior design, it operates on larger frames enabling user tracking at larger distances.

5. CONCLUSION

In this paper we presented a prototype FPGA design for detecting and tracking eyes of computer user. Experiments show that the design is capable of detecting eyes in real-time with 88% accuracy while consuming 375mW of power. However, this power figures can be reduced even further should custom design of both chip and board performed. We are currently working on custom hardware design.

6. REFERENCES

[1] Eye Tracking by Tobii, available from www.tobii.com

[2] V.G.Moshnyaga, The use of eye-tracking for PC Energy management, Proc. ETRA 2010, pp.113-116, 2010

[3] T.Ohno, N.Mukawa, S.Kawato, Just blink your eyes: a head-free gaze tracking system. Proc. CHI 2003, pp.950-951.

[4] K.R.Park, J.Kim, Real-time facial and eye gaze tracking system, IEICE Trans. Inf.&Syst., vol.E88-D, no.6, pp.1231-1238, June 2005.

[5] Q.Ji, Z.Zhu, Eye and gaze tracking for interactive graphic display. ACM Proc. Int. Symp. On Smart Graphics, pp.2002.

[6] S.Baluja and D.Pomerleau, Non intrusivegaze tracking using artificial neural networks, Technical report CMU-CS-94-102.

[7] P.Smith et al, Monitoring head/eye motion for driver alertness using one camera. Proc. Int.Conf. Pattern Recognition, 2000.

[8] T.Theochrides, G.Link, N.Vijaykrishnan, et al.Embedded Hardware Face Detection, EEE Int. Conf. VLSI Design,, 2004.

[9] V.Jeanne, F-X.Jegaden, R.Kleihorst, et al, "Real-Time Face Detection on a "Dual-Sensor" Smart Camera Using Smooth-EdgesTechnique", Workshop on Distributed Smart Cameras, 2006.

[10] S.Kawato, N.Tetsutani, K.Osaka, Scale-adaptive face detection and tracking in real time with SSR filters and support vector machine, IEICE Trans.Inf.&Systems, E88-D, no.12, pp.2857-2863, Dec.2 005

[11] V.G.Moshnyaga, K. Hashimoto, T.Suetsugu and S.Higashi, A Hardware System for Tracking Eyes of Computer user, Proc. of the Int. Conf. Computer Design, pp.125-130, 2009.

[12] D.Douxchamps, N. Campbell, Robust real time face tracking for the analysis of human behavior, in Machine Learning for multimodal Interaction, LNCS 4892, pp.1-10, 2008.

[13] P.Viola, M.Jones, Rapid Object Detection Using a Boosted Cascade of Simple Features, IEEE Conf. on Computer Vision and Pattern Recognition, 2001