제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

임베디드 시스템을 위한 눈 찾기 알고리즘

이영재, 김익동, 최미순, 심재창
안동대학교 컴퓨터 공학과
e-mail : kimchi1216@yahoo.com, {kid7, mschoi}@andong.ac.kr

A Simple Eye Detection Algorithm for Embedded System

Yung-Jae Lee, Ik-Dong Kim, Mi-Soon Choi, Jae-Chang Shim
Department of Computer Engineering,
Andong National University

abstract

Many of facial feature extracting applications and systems have been developed in the field of face recognition systems and its application, and most of them use the eyes as a key-feature of human face. In this paper we show a simple and fast eye detection algorithm for embedded systems. The eyes are very important facial features because of the attribution they have. For example, we know the darkest regions in a face are the pair of pupils, and the eyes are always a pair and parallel. Using such attributors, our algorithm works well under various light conditions, size of face in image, and various pose such as panning and tilting. The main keys to develop this algorithm are the eyes' attribution that we can usually contemplate and easily find when we think about what is the attribution that the eyes have. With some constraints of the eyes and knowledge of the anthropometric human face, we detect human eye in an image, and the experimental results demonstrate successful eye detection.

1. Introduction

Many applications for face detection and their algorithms have been developed up to now, and most of them have used human eyes, eyebrows, nose, and mouth as anthropometric measures and verification measures of the basis features on human face. C. Kotropoulos[1] used pair of the eyes, nose, and mouth as key features to verify whether objects in an image are a face or not using horizontal profile and vertical profile. S. A. Sirohey[2] proposed a localization method to segment a face from a cluttered background for face identification, and it also use facial features after finding face candidates using edge detector[3]. B. Heiseles[4] proposed a component based face detection. He made facial feature compoents to detect human face. R. Stiefelhagen[5] presented a non-intrusive model-based gaze tracking system using analysis of facial features based on skin-tone color-pixels. R. L. Hsu[6] also tried to find facial features based on skin-tone color and features' colorimetric attributions. W. Andrew [7] used skin-tone pixel and facial feature for face recognition system. He also used facial features to verify human face. But these systems and applications above are not enough efficient to be applied to embedded systems because algorithms for embedded systems have to be real-time operating systems or real-time applications, and use low memory resource.

Embedded systems are using a CPU made for a particular purpose. ARM processors have been used a lot these days for embedded system and also used for this paper. One thing we have to know before developing embedded system with ARM processor is that the ARM processors not support a proper floating point operation for image processing, if there are some floating point operations, the calculation of the floating point operation will need much more time than other processors used for personal computer or so. ARM9 and later version of ARM family support floating point operation that is not generally used for Embedded System yet. Another constraint of embedded System is its low memory resource for cost efficiency.

To fit on these constraints of embedded system, in this paper, we present a fast and simple eye detection algorithm which dose not need a large amount of mathematical calculation.

2. Eye detecting algorithm

We will mention an overview of our eye detection algorithm briefly. There are two pre-assumptions for the algorithm. The one is that one human face should be in the image, and the position of the face should be center of the image or at least around the center. The other is that under the illumination like gradient light coming from left to right or right to left can not detect the eyes because the iterative threshold method will not work well in this case. The algorithm first gets one image from CCD camera, and makes the image into a gray-scale image. The other procedures of the algorithm are only using the gray-scale image. After converting the image into gray-scale image, the algorithm uses the threshold method under changing light by changing the threshold value iteratively via the last of procedures until it finds good pairs of "eye candidate blobs". By filtering which is for eliminating eyebrow if there is one, only one eye candidate will be left. The iterative threshold will be presented in section 2.1. In section 2.2, how to search eye candidate blobs with binary image obtained as the result of 2.1 will be presented. In section 2.3, eliminating eyebrow by filtering will be presented. Experimental result images taken from a real embedded system will be shown in section 3.

2.1 Iterative threshold method

We are able to search the eye candidate blobs of a face by iterative threshold method[4] and chain code method. Under changing light condition, the threshold method changes its threshold value until the eye candidate blobs that are satisfied with the geometric eye constrains are found. We use a pair of eyes' distance, maximum eye size, minimum eye size, and a pair of the eyes' positions. Figure 1 shows the iterative threshold method, and Figure 2 shows the definition of eyes. The *ew* is eye's width, the *eh* is eye's height, the *es* is eye pixels' similarity, the *eb* is the distance of between eyes and eyebrow vertically, and the *ed* is the distance of between eyes horizontally.

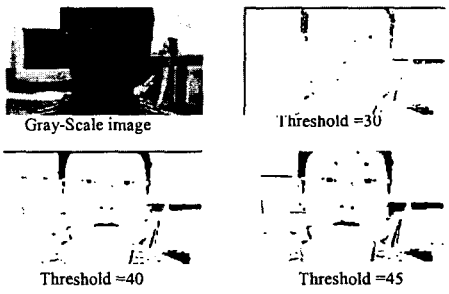


Figure 1. Binary Images of Iterative threshold

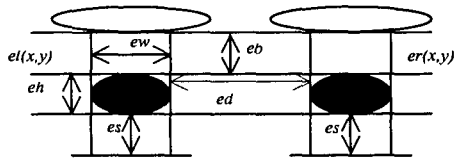
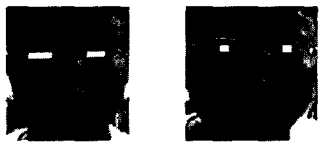


Figure 2. Definition of eye constraints

We don't separate pupils with sclera (white region of eye) from eye because we can't usually obtain high resolution image in order to separate them from the eye. When the eyes are small size, or not high resolution image, the region of pupil and sclera may be detected together shown in Figure 4. R. Stiefelhagen[6] and other face detection systems segmenting a region of the face candidates by the skin-tone pixel or the edge detector, but we don't. Also, we can't do a large amount of mathematical calculation because the ARM processors do not support floating point operation. By the reason, our algorithm is more simple and faster than those algorithms. Because we don't search for face region, we use more geometric constraints on facial feature. This will be presented section 2.2 and 2.3.



a) b)

Figure 3. Don't consider of pupils and scleras. a) pupils and scleras, b) pupils

2.2 Searching for the eye candidates

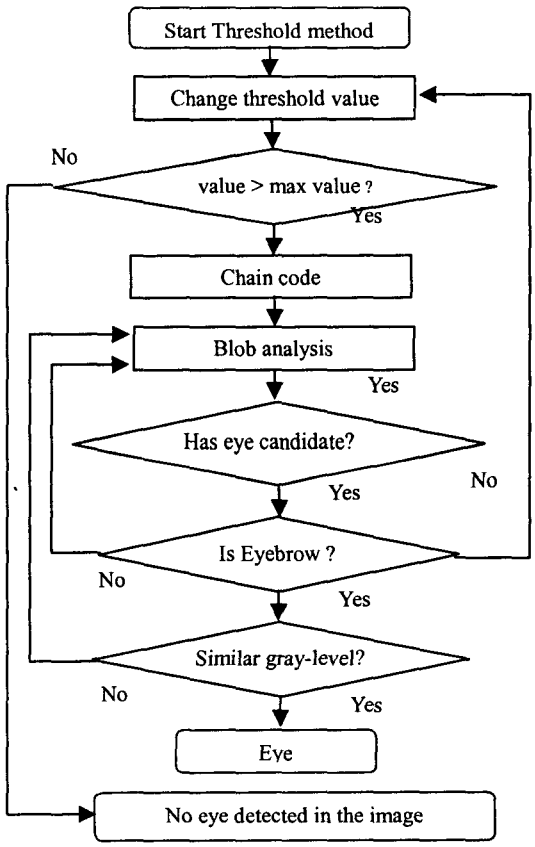


Figure 4. Flow of detecting Eyes

We get a binary image obtained by threshold method. Using chain code method, we can get the size, width, height, and positions of the blobs that interest us. We assume that the eye regions are the darkest regions on the face, and the pair of the eye regions might be always parallel. The blobs are analyzed by the definition of eye constraints, and if there is a pair of the regions fully satisfied with the eye constraints, the pairs will be ranked as an eye candidate blob. We use the following distance measures (1), (2) and parallel measure to measure the distance and parallelism (3):

$$Dx = | El(x) - Er(x) | \quad (1)$$

$$Dy = | El(y) - Er(y) | \quad (2)$$

Where *Dx*, *Dy* are the horizontal and vertical distances of the blobs, and *El*, Er are the position of left, right eye candidate blob each. If *Dx*, *Dy* dose not exceed a certain distance of *max(Dx)* and *max(Dy)*, this will be ranked as an eye pair, or will be rejected. If *Dx* and *Dy* are less than mix(*Dx*) and min(Dy), the pair of blobs will be rejected.

$$P = | El(y) - Er(y) | \quad (3)$$

Where *P* is the parallel measure and *El, Er* are the position of left, right eye candidate blobs each. If *P* exceeds a certain parallel value, it's not the eye candidate blobs. Figure 4 shows a flow of detecting eyes. Initially, the image is binarized by the lowest threshold value, and we obtain each blob's size, position, width, height by chain code method. Using each blob's information, we obtain eye candidate blobs which are satisfied with the definition of eye constraints. If there are no more eye candidate blobs, the algorithm will change the value of threshold method, and if the value of threshold method exceeds the maximum threshold value, the algorithm will get another input image from CCD camera.

2.3 Eyebrow filtering

For every pair of the eye candidates we got from previous procedure, we search another pair of blobs vertically whether or not there is another pair of the eye candidates from the position of current eye candidates to other eye candidates. Because eyes and eyebrows in an image will frequently appear together in the image or one eyebrow and eyes after we have done the threshold method. As a result of such false detection, we try to eliminate eyebrow by filtering. If we get two pairs of the eye candidate blobs, we check the distance of the two pairs of blob vertically. If the distance of the eye candidate blobs is shorter than, or equal to the distance of *eb* in Figure 3, the eye candidate blob that we are currently trying to check is not the eyes, and the procedure will go back to blob analysis until only one pair of eye candidate blob is found. Figure 4 shows the false eye detections before eliminating eyebrows. Figure 5 a) shows a case of wrong eye detection. The white box represents the eyes. This will happen because of the inappropriate tolerance of eye parallel measure and *eb*. Figure 5 b), c) show a pair of eyebrows and eyes. This result will happen when we try to detect the eye candidates without the eyebrow filtering. Finally, we check the similarity of the pairs of eye candidates. We assume that a certain range (*es* in Figure2) of gray level intensity from eyes should be similar. If each *es* of eye candidates are not similar each other, the eye candidates are not the eyes. If no eyes are detected, go back to the procedure of the using threshold method, and increase the value of the threshold method by 2, and if the increased value of the threshold method exceeds the maximum value of the threshold method, the procedure will go back to input image in Figure 4. We use the following similarity measure (4) to check the pair of the eye candidate is the eyes or eyebrows.

a) b) c)

Figure 5. False eye detections. a) one eye and one eyebrow, b) two eyes and two eyebrows, and c) two eye brows only

$$ES = | ESl - ESr | \quad (4)$$

$$ESl = \left(\sum_{j=el(x)}^{el(x)+width} \sum_{i=el(y)+height}^{el(y)+height+es} El(j,i) \right) / total\_els \quad (5)$$

$$ESr = \left(\sum_{j=er(x)}^{er(x)+width} \sum_{i=er(y)+height}^{er(y)+height+es} Er(j,i)) \right) / total\_ers \quad (6)$$

Where *ES* is the eye similarity, *ESl* is the sum of the left eye's similarity, and *ESr* is the sum of the right eye's similarity. The total pixels of *els* and the total pixels of total_*ers* are the total number of pixels searched *es* each. If *ES* is in the range of the eye similarity, the pair of the eye candidates is the eyes we have looked for.

3. Experimental Results

We have made this experiment upon both PC and the embedded system made by Face3d company[8].

3.1 Experimental Environment

Figure 6 shows the embedded system of face3d. We can see the continuous image by LCD screen. The Touch Pad is for users to put in their ID and pin number. From the speaker, some voice guidance comes out for every procedure. While the system is sleep mode, any movements detected by IR Sensor will wake up the system. Embedded Linux, main operating system, is used for this system. The main operating system is composed of BootLdr, Linux kernel, and file system. This system uses 4M RAM for the file system. Figure 7 shows images of experimental results.

3.2 Experimental Results

We have developed a simple and fast eye detection algorithm using iterative threshold method for embedded system.

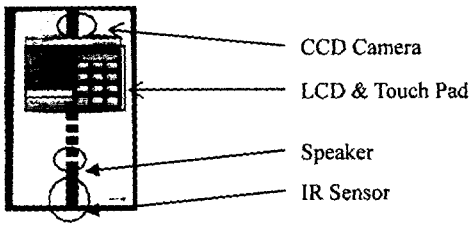


Figure 6. Embedded System

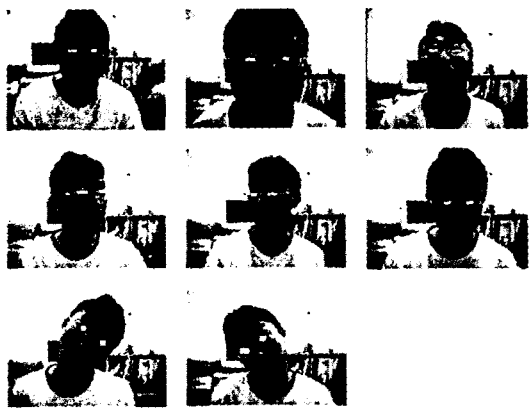


Figure 7. Images of experimental results

It requires only one person's face in the image from CCD camera, and it shows a strong phase on rotation, various lights, the size of human in the image, and pose such as panning and tilting. Moreover, the speed of detecting eyes depends heavily on the hardware's specifications such as ability of CPU, CCD camera, and memory. When we had applied this algorithm to face3d's embedded system, this algorithm detected the eyes from input image within 1 second. From CCD camera, we get the size of 320*240 as input image, but we don't fully use the size of input image. We use only 160*120 input images because this algorithm is made for the purpose of door lock authentication systems. Usually door lock systems only need a face for allowing user to enter the door, so we use the constraint of that only one human face has to be appeared at the center of input image or around center. Also using the 160*120 size of input image, we are not necessary dealing with objects in outer regions of image. Even though this algorithm is made only for embedded system, simple and fast, this application can not do the eye detection successfully when there is a light coming from left to right or its opposition or if too small of a face appears in the image. Our simple application gives us steady successful results. Moreover, the algorithm is using a small amount of system's recourses, so it is able to be applied to any system using camera to detect human's eyes.

4. Conclusions

The algorithm is originally made for the purpose of face3d's door lock authentication system based on embedded system. We have detected one pair of eyes by iterative threshold method, chain code method, blob analysis, and check gray level similarity. This algorithm is much faster when the last threshold value is put into initial threshold value of next search.

References

[1] C. Kotropoulos, A. Tefas, and I. Pitas, "Frontal Face Authenticaion Using Variants of dynamic Link Matching Based on Mathematical Morphology", proc. Int'l Conf. Acoustics, Speech and Signal Processing, pp. 122-126, 1998.

[2] S. A. Sirohey, "Human Face Segmentation and Identification", Technical Report CS-TR-3176, Univ. of Maryland, 1993.

[3] R. Brunelli and T. Poggio, "Face Recognition: Features versus Templates", IEEE Transations on Pattern Analysis and Machine Intelligence, Vol. 15, No 10, October 1993.

[4] B. Heisele, T. Serre, M. Pontil and T. Poggio. Component-based face detection. In Proceedings of the IEEE Computer Society Conference on Computer vision and Pattern Recognition, IEEE Computer Society Press, vol 1,657-662, 2001.

[5] Rein-Lien, Mohamed Abdel-Mottaleb, Anil K. Jain, "Face Detection in Color Image", IEEE Transactions on pattern analysis and machine intelligence, vol. 24, no. 5, May 2002.

[6] Rainer Stiefelhagen, Jie Yang, Alex Waibel, "A Model-Based Gaze Tracking System", IEEE international Joint Symposia on Intelligence , Proceedings of the 1996.

[7] Andrew W. Senior, "Face and Feature finding for a face recognition system", In proceedings of the Second International Conference on Audio- and Video-based Biometric Person Authentication, pp. 154-159. Washington D.C. March 1999.

[8] www.face3d.com