Fast Eye Detection Using Boundary Value and Template Based Method

A. Hemlata and Mahesh Motwani

***Abstract*—Human face can be identified by different biometric features, which are genetic properties of a human being. If the biometric features can be extracted from a facial image then they can be used for face recognition. This paper proposes a method to detect eye in human face by using boundary value analysis. Rejecting the area outside the boundary and performing further processing using template based approach on inside boundary values.**

***Index Terms*—Threshold, boundary value, normalisation, connected components.**

I. Introduction

Image processing is important in modern data storage and data transmission especially in progressive transmission of images, video coding (teleconferencing), digital libraries, and image database, remote sensing. It has to do with manipulation of images done by algorithm to produce desired images. Object detection is one of the most important preprocessing steps in object recognition and identification systems. This can be done by searching and indexing in still image or video containing object in various size, position and background.

One of the salient features of the human face [1], human eyes play an important role in face recognition and facial expression analysis. In fact, the eyes can be considered as salient and relatively stable feature on the face in comparison with other facial features. Therefore, when we detect facial features, it is advantageous to detect eyes before the detection of other facial features. Thus, eye position detection is important not only for face recognition and facial expression analysis but also for eye contour detection.

Using color characteristics is a useful way to detect eyes, YCbCr which its components give worthwhile information about eyes. The method [2] makes two maps according to its components and merge them to obtain a final map. Candidates are generated on this final map. By applying an extra phase on candidates to determine suitable eye pair. The extra phase consists of flexible thresholding and geometrical tests.

In Template Matching based Eye Detection in Facial Image [3], the template is correlated with different regions of the face image. The region of face which gives maximum correlation with template refers to eye region. The method is simple and easy to implement. The effectiveness of the method is demonstrated in both the cases like open eye as well as closed eye through various simulation results.

This proposed method presents eye detection using template based and boundary value analysis. This method works efficiently for image containing single facial image, the difficulty arise when we start processing on images containing multiple faces and side faces. The proposed algorithm can be modified further to improve and implement the system for side face and multiple faces.

Algorithm

1) Load the image
2) Preprocessing
   - Noise removal
   - Normalization of image
   - Binary image
1) Detecting boundary and analyzing boundary value
2) Connected components
3) Finding holes
4) Template of eyes is constructed by taking mean of 10 eye pair samples.
   - Testing hole pairs and comparing with eye template
   - Finding correlation between eye template and hole pair.
1) By repeated testing with sample data thresh hold value is computed.
2) Eye pairs are enclosed by rectangle.

II. Method

A. Preprocessing of Image

1) Noise removal

Digital images are prone to a variety of types of noise. Noise is the result of errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. There are several ways that noise can be introduced into an image, depending on how the image is created. For example: If the image is scanned from a photograph made on film, the film grain is a source of noise. Noise can also be the result of damage to the film, or be introduced by the scanner itself. If the image is acquired directly in a digital format, the mechanism for gathering the data (such as a CCD detector) can introduce noise. Electronic transmission of image data can introduce noise.

The adaptive filter behavior changes based on statistical characteristics of image inside the filter region Sxy is to calculate the noise performance Using Adaptive Filtering, the wiener2 function applies a Wiener filter (a type of linear filter) to an image adaptively, tailoring itself to the local image variance. Where the variance is large, wiener2

Manuscript received August 16, 2012; revised October 5, 2012.

A. Hemlata is with the Jabalpur Engineering college, Jabalpur, M.P. India (e-mail: a.hemlata@rediffmail.com).

M. Motwani was with Jabalpur Engineering college, Jabalpur, M.P., India. He is now with the Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (e-mail: mahesh.7@sify.com).

performs little smoothing. Where the variance is small, wiener2 performs more smoothing. The wiener2 function handles all preliminary computations and implements the filter for an input image. wiener2, however, does require more computation time than linear filtering. Wiener2 works best when the noise is constant-power ("white") additive noise, such as Gaussian noise.

2) Gray normalization

Given input gray image $I(i,j)$, normalized image $I'(i,j)$ image as shown in Fig. 1 is computed by the following equation [4]. :

$$I'(i,j)= M_0 +\sqrt{(V_0.(I(i,j)-M)^{2-} /V)}, \quad I(i,j)>M$$

$$I'(i,j)= M_0 -\sqrt{(V_0.(I(i,j)-M)^{2-} /V)}\ , \quad I(i,j)<=M$$

where, M, V: mean and variance of inputted image $I(i,j)$ Mo, Vo : Mean and variance of destination image $I'(i,j)$.

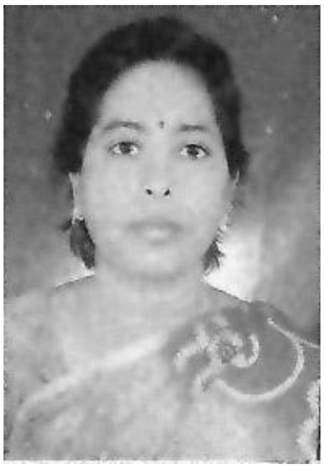

Fig. 1. Normalized image.

3) Binary image

A binary image is of class logical. Binary images in Fig. 2 contain only 0's and 1's. Pixels with the value 0 are displayed as black; pixels with the value 1 are displayed as white. In a binary image, also known as a bi-level image, each pixel assumes one of only two discrete values: 1 or 0. A binary image is stored as a logical array.

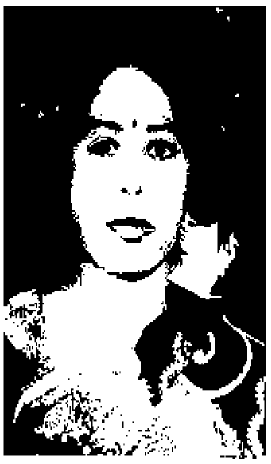

Fig. 2. Binary image.

B. Boundary Value

Boundary of an object may be an interior boundary or an exterior boundary. Interior boundary consists of pixels that belong to the object(s) itself and pixels of exterior boundary belong to background (So).

The boundary of S is S'

$$S'=S/S\ \partial K$$

where K is a binary structuring element for 8-connected boundary.

$bS'(r,c)= bS(r,c)\wedge\neg[\ (bS(r-1,c) \wedge bS(r,c-1) \wedge bS(r,c+1) \wedge bS(r+1,c)]$

	X	
X	$\underline{X}$	*X*
	X	

X	*X*	*X*
X	$\underline{X}$	*X*
X	*X*	*X*

The bwtraceboundary function returns the row and column coordinates of all the pixels on the border of an object in an image. You must specify the location of a border pixel on the object as the starting point for the trace.

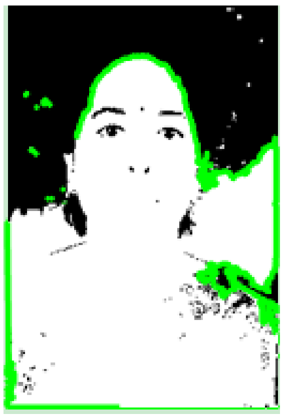

Fig. 3. Boundary marked image

The bwboundaries function returns the row and column coordinates of border pixels of all the objects in an image. For both functions, the nonzero pixels in the binary image belong to an object and pixels with the value 0 (zero) constitute the background.

C. Finding the Connected Components in an Image

In image processing, a connected components as in fig.4, algorithm finds regions of connected pixels which have the same value. For example, the image below contains 4 components: there are 2 red components, one blue component, and the white, background component [5]. The label image to the right colors each pixel according to the ID of its blob, identifying blob membership. The goal of connected components is to compute this label image.

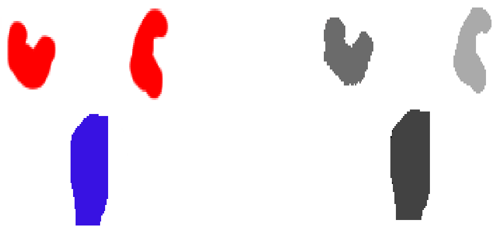

Fig. 4. Connected components.

For each pixel on the line, this implementation first checks to see if the pixel to the left has the same pixel value. If so, we know for sure that we're in the same blob, and the current pixel is so labeled. If the pixel at the top has the same value as the pixel to the left but not the same blob ID, we know at once that the pixels to the left and to the top are in the same region and that these regions should be merged. If a pixel is found with left and top neighbor having different pixel values, a new blob id is created and assigned to this pixel.

The algorithm continues this way, creating new blobs and merging them whenever they are discovered to be the same.

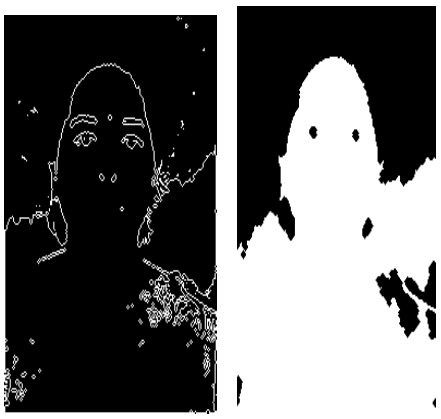
Fig. 5. Image with connected components and holes

D. Holes

The image contains single facial image or object, the connected component detect region. Each region is labeled. And processed one bye one. Holes as shown in Fig. 5 are detected from each region. If the image contains hole then this may be eyes , nose , lips or something else. To detect the holes from connected component

It performs a flood-fill operation on background pixels of the binary image, starting from the points specified in locations. If locations is a P-by-1 vector, it contains the linear indices of the starting locations. If locations is a P-by-ndims matrix, each row contains the array indices of one of the starting locations. The function fills holes in the binary image . A hole is a set of background pixels that cannot be reached by filling in the background from the edge of the image, then fills holes in the intensity image I. In this case, a hole is an area of dark pixels surrounded by lighter pixels.

E. Eye Template Match

Holes are located and matched with eye template. Template of eyes as shown in fig. 6 are constructed from collecting samples of eye from 10 faces and taking mean of these eyes to get the eye template. This eye template is used to detect eyes in images in each region by calculating the correlation coefficient between two.

The value of the correlation coefficient [6] is always between -1 and +1:

$$-1 \leq \rho \leq +1$$

When the two distributions are known only through a sample, the common estimate of the Correlation Coefficient is:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 (y_i - \bar{y})^2}}$$

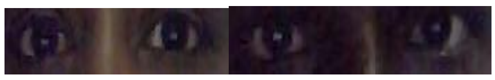
Fig. 6. Eye Template.

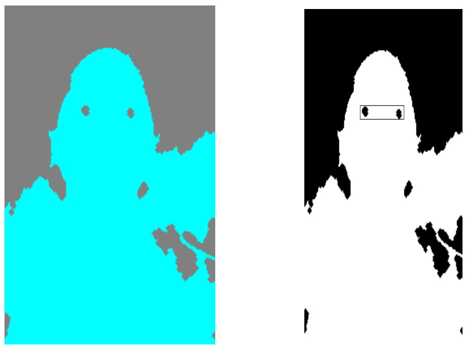
Fig. 7. Eye template matching.

A threshold value is detected by repeatedly performing the sample testing to get the accurate eye region shown in Fig. 7. If the coefficient is greater than threshold value then the region is assumed to be as eye and now mark the eye region by bounding box.

III. Experimental Results

The experiment shows reasonably good result for locating eye on image containing frontal face. We implemented the method in Matlab 7.0 and tested the method on 50 images. The accuracy of detection is found to be 90% for the images having single frontal face. Test results are shown in Fig. 8. As eye detection first step towards face detection and recognition system, this method can be further modified to detect facial features in images containing multiple faces and side faces.

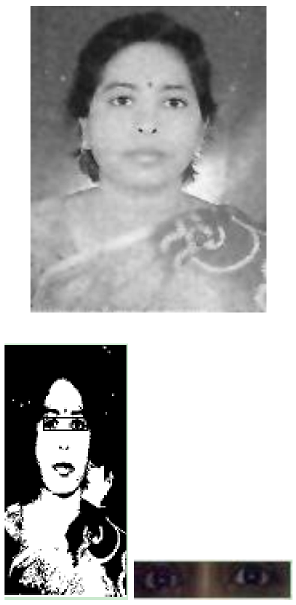
Fig. 8. Eye enclosed in rectangle.

REFERENCES

[1] K. Peng, L. Chen, S. Ruan, and G. Kukharev, “A robust algorithm for eye detection,” *JCS&T* vol. 5, no. 3, Oct. 2005.

[2] J. A. Nasiri, S. Khanchi, and H. R. Pourreza, “Eye detection algorithm on facial color images,” *Second Asia International Conference on Modelling and Simulation, IEEE*. 2008

[3] Nilamani and M. N. Mohanty, “Template matching based eye detection in facial image,” *International Journal of Computer Applications* vol. 12, no. 5, December 2010.

[4] T. Mondal, A. Nath, A. Das, and M. Banerjee, “An approach of face detection using geometrical definition of human face,” *NCCI 2010 -National Conference on Computational Instrumentation CSIO Chandigarh*, India, pp. 19-20 March 2010.

[5] X.-Y. Feng, “Eyes location by neural network-based face segmentation,” *IJCSNS International Journal of Computer Science and Network Security*, vol. 6, no. 5A, May 2006.

[6] S. Anila and N. Devarajan, “Simple and fast face detection system based on edges,” *International Journal of Universal Computer Sciences* vol. 1, no. 2, pp. 54-58, 2010.

[7] Q. Wang and J. Yang, “Eye detection in facial images with unconstrained background,” *Journal of Pattern Recognition Research* vol. 1, pp. 55-62, 2006.