Eye Localization in a Full Frontal Still Image

Ghassan Ahmad Ismaeel AL-Dabbagh
Assistant Lecturer – University Of Mosul/College Of Pharmacy
Clinical Laboratory Science Dept.

Abstract

In this paper the detection of human face and eye in still frontal color images is discussed. Firstly; the preprocessing required step is accomplished. It includes image resize, RGB to gray-scale conversion, image binarization, noise removing and small objects removing. Then a proposed algorithm is applied for face localization by detecting the face edges using the detection of the pixel color change in the binary image. Finally, the normalized cross correlation is applied to find the accurate position of eyes within the localized area of the face.

Keywords: Eye detection, Face detection, Face recognition, Image processing, Pattern recognition.

تحديد موقع العين في صور أمامية ثابتة

غسان أحمد إسماعيل الدباغ
مدرس مساعد – جامعة الموصل/ كلية الصيدلة
فرع العلوم المختبرية السريرية

المستخلص

في هذا البحث تم اقتراح خوارزمية للكشف عن وجوه وعيون الأشخاص في صور أمامية ملونة وثابتة. بداية تم إجراء المعالجات الأولية المطلوبة على الصورة لتهيئتها، إذ تم تغيير أبعاد الصورة ثم تحويل الصورة من صورة ملونة نوع RGB إلى صورة ذات التدرج الرمادي، ثم تحويلها إلى صورة ثنائية. بعد ذلك تم العمل على إزالة الضوضاء وإزالة الأجسام الصغيرة من الصورة الثنائية ثم تم تطبيق الخوارزمية المقترحة لتحديد مكان الوجه في الصورة واقتطاعها في صورة مستقلة عن طريق تحديد حافات الوجه باستخدام الكشف عن انقلاب لون النقطة الصورية (البكسل) في الصورة الثنائية. وأخيراً تم استخدام قياس معامل الترابط لتحديد موقع العين بدقة ضمن صورة الوجه التي تم تحديدها واقتطاعها في الخطوة السابقة.

Received: 19 – 6 - 2012 **Accepted: 12 – 12 - 2012**

1. Introduction

Face recognition as the front subject of pattern recognition and artificial intelligence has a broad application in the human-machine interface, the biometric information security and so on. The facial features localization is the key of face piece fitting and recognition [1].

The analysis of face images is a popular research area with applications such as face recognition, virtual tools, and human identification security systems [2].

As one of the salient features of the human face, human eyes play an important role in face recognition and facial expression analysis [3][4]. In fact, the eyes can be considered salient and relatively stable feature on the face in comparison with other facial features. Therefore, when detecting facial features, it is advantageous to detect eyes before the detection of other facial features. The position of other facial features can be estimated using the eye position [2].

Many methods and approaches was proposed over the years for face detection[5][6] but it is still a very challenging task because of variability in scale, location, orientation, and pose. Facial expression, occlusion, and lighting conditions also change the overall appearance of faces[5]. A novel algorithm for exact eye contour detection in frontal face image has been proposed by Vladimir and Anna[7]. Snake model has been proposed by Lam and Yan[8] for detecting face boundary. Wang, et al [9] have employed genetic algorithm to detect human faces by calculating the projection of each face candidate onto the Eigen faces space. Rowley, et al [10] have improved a frontal face detection system based on neural network.

This paper presents a robust face and eye detection algorithm for color frontal images. The idea of the method is to combine the respective advantages of two existing techniques, image processing with cross correlation and to overcome their shortcomings. Firstly, image processing techniques are used to locate the face region. After the location of face region is detected, an accurate detection of eyes is achieved by using cross correlation between the located face region and the previously prepared eye template. Results of experiments to the images show that the proposed approach is robust and quite efficient.

This paper is organized as follows: besides this introduction, section 2 presents the proposed algorithm for face detection, section 3 describes the eye locating process, section 4 presents the experimental results and discussion and the last section is the conclusion.

2. Proposed Algorithm

The algorithm steps is summarized below: -

1. Input the image of which face and eye detection has to be performed.
2. Resize image.
3. Convert color image into gray-scale image.
4. Binarize the gray-scale image using threshold.
5. Face top and width detection.
6. Noise Removal.
7. Small objects removing.
8. Finding the top of the head and the sides of the face accurately (as a reference).
9. crop the face area to narrow down the processing area.
10. using the cross correlation between the face area which is cropped from the original image and eye image (template) which prepared previously, the eye can be localized accurately.

A diagram of the major functions of the face and eye detection algorithm is shown in Figure (1)

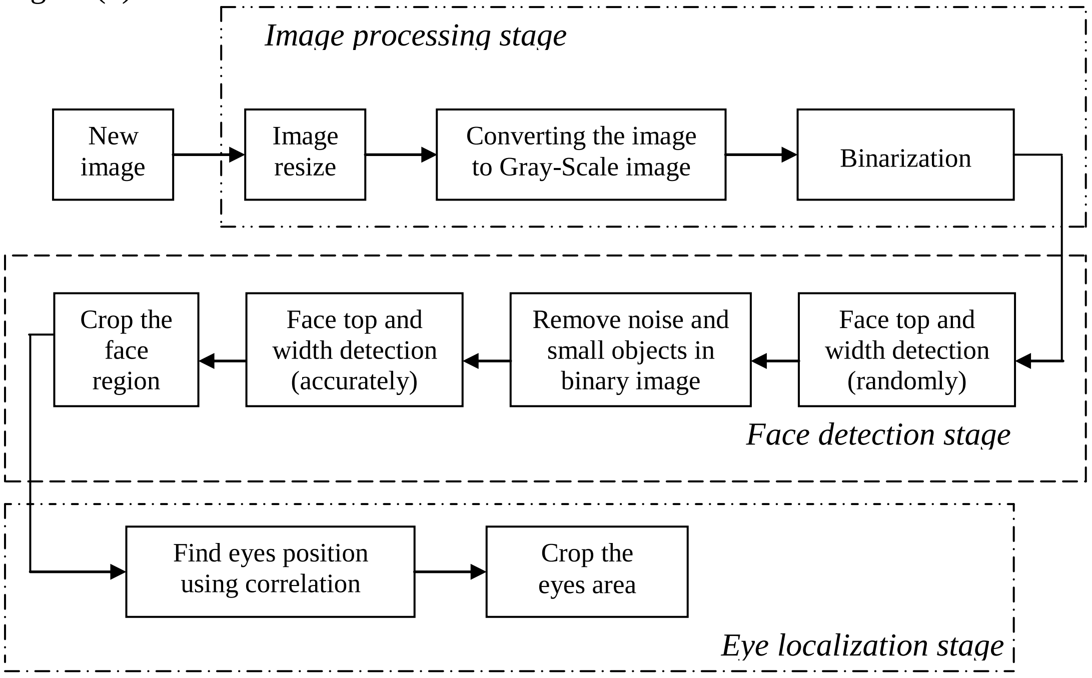


Image processing stage
New image
Image resize
Converting the image to Gray-Scale image
Binarization
Crop the face region
Face top and width detection (accurately)
Remove noise and small objects in binary image
Face top and width detection (randomly)
Face detection stage
Find eyes position using correlation
Crop the eyes area
Eye localization stage

Figure (1) Algorithm Steps

The following explains the face and eye detection procedure in the order of the processing operations. All images are generated in MatLab using the image processing toolbox. The input image is first resized as 640*480 for the purpose of normalization or compatibility.

2.1 Face Detection

The resized input image is then converted into gray-scale image as shown in Figure (2).

The following equation is used to convert the color image (RGB) to gray scale image [11] [12]:

$$gray = (R*299 + G*587 + B*114)/1000 \quad (1)$$

where R, G, B represent the red, green and blue color components, respectively.

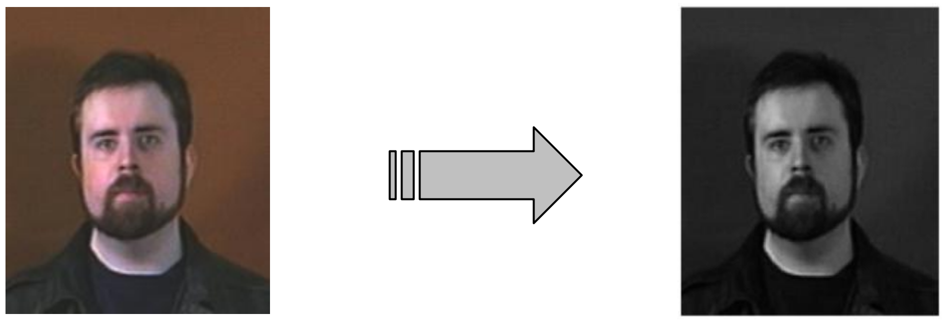


Figure (2) : RGB to gray-scale image conversion

Binarization step converts the resulting gray-scale image to a binary image. A binary image is an image in which each pixel assumes the value of only two discrete values. These values are 0 and 1, where 0 representing black and 1 representing white. With the binary image it is easy to distinguish objects from the background. The grayscale image is converted to a binary image via thresholding. The output binary image has values of 0 (black) for all pixels in the original image with luminance less than specified level (threshold), and 1 (white) for all other pixels. Thresholds are often determined based on surrounding lighting conditions. After observing many images of different faces under various lighting conditions a fixed threshold value of 160 is found to be effective. The criteria used in choosing the correct threshold is based on the idea that the binary image of the face should be majority white, allowing a few black blobs from the eyes, nose and/or lips. Figure (3) demonstrates optimal thresholded images. Figure (4) (a, b and c) illustrates binarization using threshold values of 120, 160 and 200, respectively for the same image. Figure 4b is an example of an optimal binary image for the face and eye detection algorithm so that the background is uniformly black, and the face is primary white. This will allow finding the edges of the face as described in the next steps.
The results of this method depends on the value of the current pixels as in the following equation [4][11][13] :

$$I_{new}(x,y) = \begin{cases} 1 & \text{if } I_{old}(x,y) \geq \textbf{threshold} \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where *Iold* represents a gray-scale matrix image pixel, and *Inew* represents binary matrix image pixel, the threshold represents a value which is chosen accurately by trying many values of threshold for a large number of images to find the optimum value (in the condition of the lighting of the images is approximately equal).

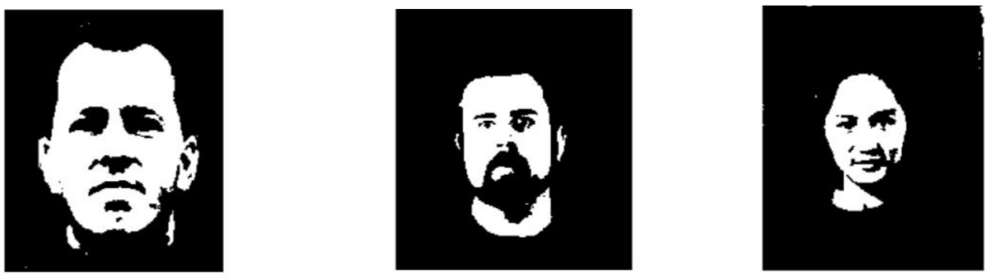
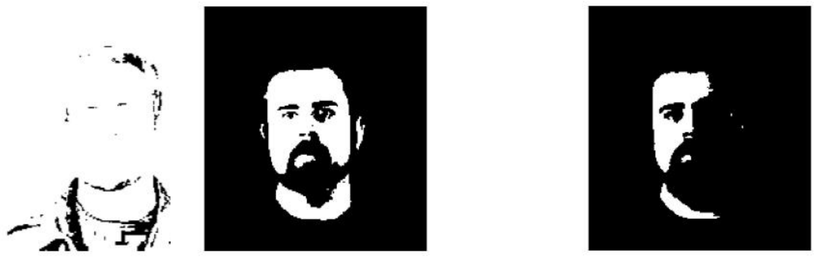


Figure (3) : Examples of optimal thresholded images with fixed threshold value of 160.



a) Threshold value 120. b) Threshold value 160. c) Threshold value 200.
Figure (4) : Examples of binarization using differentthresholds for the same image.

The process of converting an image from grayscale to binary image by a fixed threshold, although it achieved a high success rate but are deficient due to failure when lighting changes from one image to another, using variable threshold that adjusts with image lighting is the appropriate alternative to solve this problem.

The following algorithm can be used to obtain ***threshold*** value automatically:

1. Select an initial estimate for ***threshold***.
2. Segment the image using ***threshold***. This will produce two groups of pixels: G_1 consisting of all pixels with gray level values > ***threshold*** and G_2 consisting of pixels with values ≤ ***threshold***
3. Compute the average gray level values μ_1 and μ_2 for the pixels in regions G_1 and G_2.
4. Compute a new ***threshold*** value:

   threshold $= 1/2\ (\mu_1 + \mu_2)$

5. Repeat steps 2 through 4 until the difference in threshold in successive iterations is smaller than a predefined parameter T_o .

A good initial value for threshold is the average gray level of the image. The parameter T_o is used to stop the algorithm after changes become small in terms of this parameter [11].

The two methods are tested with the same set of images and the results show that the adaptive binarization is more efficient and the system become more robust.

Figure (5) Shows the efficiency of the proposed method. Fig. (5a, 5b) is the original images, Figure (5c, 5d) is an optimum binary images thresholded at 187, 126, respectively using the adaptive threshold, Figure (5e, 5f) shows bad binary images thresholded at 160 using the fixed threshold and these images are failed in the face detection stage.

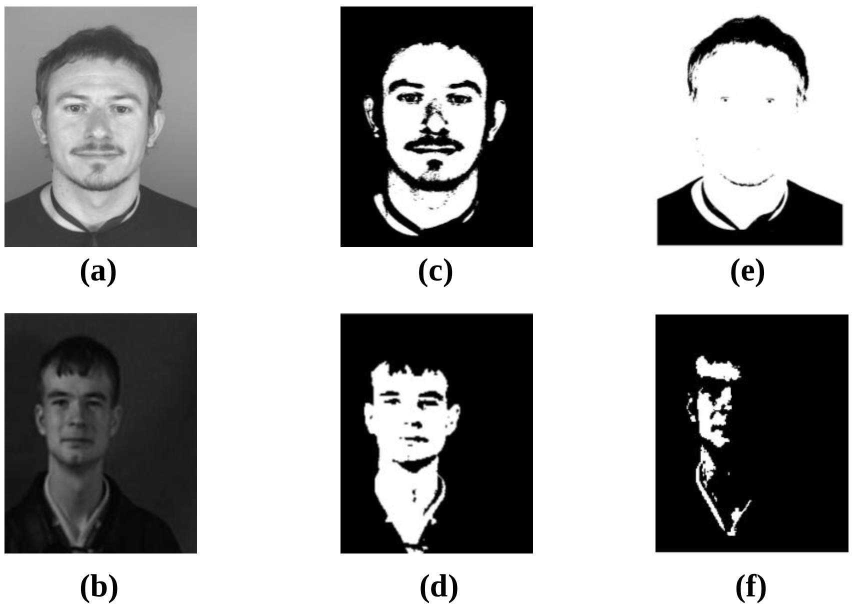


Figure (5): Shows a comparison between the two methods used to binarize the images (fixed threshold and adaptive threshold)

The next step in the eye detection function is determining the top and side of the face. This is important since finding the outline of the face narrows down the region in which the eyes fall, which makes it easier (computationally) to localize the position of the eyes. The first step is to find the top of the face. A starting point on the face should be identified firstly, followed by decrementing the y-coordinates until the top of the face is detected. Assuming that the person’s face is approximately in the center of the image (shifting the person's face within the image borders will not affect the system work), the initial starting point used is (100,240). The starting x-coordinate of 100 is chosen, to insure that the starting point is a black pixel (not on the face). The following algorithm describes how to find the actual starting point on the face, which will be used to find the top of the face.

1. Starting at (100,240), increment the x-coordinate until a white pixel is found. This is considered the left side of the face (L point).(not accurate)
2. If the initial white pixel is followed by 30 more white pixels, keep incrementing x until a black pixel is found.
3. Count the number of black pixels followed by the pixel found in step2, if a series of 30 black pixels are found, this is the right side (R point). (not accurate)
4. The new starting x-coordinate value (mid1) is the middle point of the left side and right side found in steps 1 and 3 above. Figure (6) explains the above algorithm. The starting point is (100,240) and mid1 is the middle point of L and R. Using the new starting point (mid1, 240), the top of the head can be found. The following is the algorithm to find the top of the head:

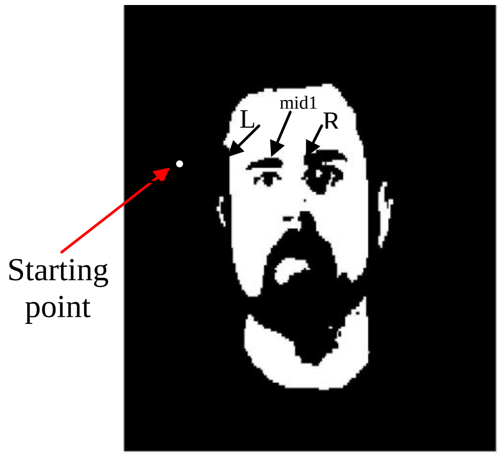


Figure (6)
Face top and width detection

1. Beginning at the new starting point, decrement the y-coordinate (i.e.; moving up the face).
2. Continue to decrement y until a black pixel is found. If y becomes zero (i.e. the top of the image is reached), set this point to the (***top***) of the head.
3. Check to see if any white pixels follow the black pixel.
4. If more than 30 white pixels are found, continue to decrement y.
5. If no white pixels are found, the ***top*** of the head is found at the point of the initial black pixel.

Once the top of the person’s head is found, the sides of the face can also be found accurately. Below are the steps used to find the left and right sides of the face accurately.

1. Increment the y-coordinate of the ***top*** (found above) by 10 (A little distance far from the edge of the hair). Label this y1 = ***top***+10.
2. Find the center of the face using the following steps:
3. At point (mid1, y1), move left until 30 consecutive black pixels are found, this is the left side (Lx). at the first black pixel found.
4. At point (mid1, y1), move right until 30 consecutive black pixels are found, this is the right side (Rx), at the first black pixel found.
5. The centre of the face (in x-direction) is: (Rx – Lx)/2. Label this point as mid2.
6. Starting at the point (mid2, y1), find the top of the face again. This will result in a new y-coordinate, y2.

2.2 Noise Removal

The noise in binary image under study is usually due to the blobs of black pixels on the face, primarily in the eye, nose and lips. To fix this problem, an algorithm to remove the black blobs is developed as follows:

Starting at the top, at point (mid2,y2) move 5 pixels down, label this point as (mid2,y3) were y3=y2+5. From this point move left (decrement x) until black pixel meet, label this point as Lnew, then move right (increment x) until black pixel meet, label it as Rnew.
Calculate the horizontal distance between Lnew
and Rnew.
Divide the distance by 2, Label the result as Xnew.

From the point (mid2,y3) move left one pixel by decrementing (mid2) and set the pixel value to white, Repeat this process for (Xnew-25) times.

From the point (mid2,y3) move right one pixel by incrementing (mid2) and set the pixel value to white, Repeat this process for (Xnew-25) times.

The key to this is to stop before the left and right edges of the face; for this reason 25 pixels is left from each side, otherwise the information of where the edges of the face are, will be lost.

Repeat the last two steps for 250 value of y (250 is the approximate height of the face in the image).
Figure (7) illustrates the results of implementing noise-removing algorithm.

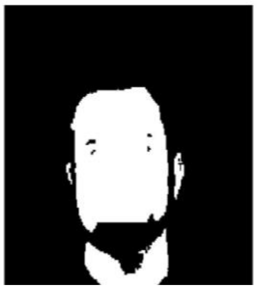


**Figure (7)
Binary image after implementing noise removing algorithm**

After this step the small black spots remaining in the face can be removed by another step which is called small objects removing.

2.3 Small Objects Removal

Binary image consists of a set of objects, each object consists of a set of connected pixels. This process removes associated components (objects) of the binary image, which the number of its pixels is equal to or less than a specific number to be determined in advance and this process is similar in its work for the filters that remove unwanted parts of the image as noise. The selected number of pixels in this application is 150. Figure (8) shows the binary image after implementing the small objects removing
algorithm. Finally after removing the black blobs on the face, the edges of the face can be found accurately using the point (mid2, y2).

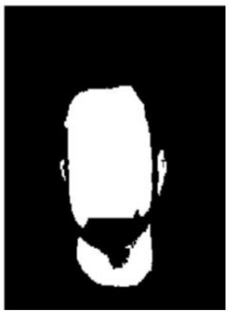


Figure (8) Binary image after implementing small objects removing

2.4 Accurate Face Edges

Starting at (mid2,y2), the following algorithm explains the accurate face edges detection:
1. Increment y-coordinate.
2. Move left by decrementing the x-coordinate, when 5 black consecutive pixels are found, this is the left side, add the x-coordinate to an array labeled ‘left_x’.

3. Move right by incrementing the x-coordinate, when 5 black consecutive pixels are found, this is the right side, add the x-coordinate to an array labeled ‘right_x’.
4. Repeat the above steps 250 times (250 is the approximate height of the face in the image).

The result of the face top and sides detection is shown in Figure (9), it is marked on the picture as part of the computer simulation. Figure (10) shows the cropped face image within these marks.

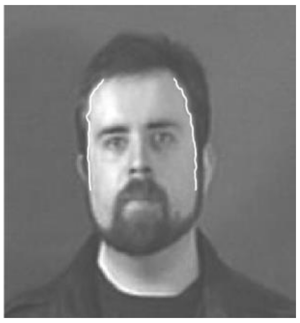


Figure (9) Accurate face edges found after implementing the last algorithm

Figure (10) The cropped face image

3. Eyes Locating

Now after the face boundaries are found, the face region cropped in a separate image to narrow down the area of where the eyes exist, leading to reduce the computational time for the next step significantly.

3.1 Matching by correlation [11][12][14][15]

Correlation is commonly exploited to measure the similarity between a stored template and the window image under consideration. Templates should be deliberately designed to cover variety of possible image variations[3]. Obviously, the first obligatory (essential) step for a template matching is to create a template. It’s easy to find out eye templates which can be obtained from a real face image by crop it manually.
Assuming that the original image *f(x, y)*, and the sub-image *w(x, y)*, the correlation between *(f)* and *(w)* In it is simplest form is performed according to the following equation [11] :

$$c(x,y)=\sum_{s}\sum_{t} f(s,t)w(x+s,y+t) \qquad (3)$$

For x = 0, 1, 2, ..., M - 1, y = 0, 1, 2, ….., N - 1, and the summation is taken over the image region where *w* and *f* overlap.
The point that gives the highest value for *c(x,y)* will be the center of the sub image being searched for within the mother (main) image, and this operation is called the *cross-correlation* between *f* and *w*.
Figure (11) illustrates the procedure, assuming that the origin of *f* is at its top left and the origin of *w* is at its center. For one value of *(x, y)*, say, *(x_n, y_n)* inside *f*, application of Eq.(3) yields one value of *c*. As *x* and *y* are varied, *w* moves around the image

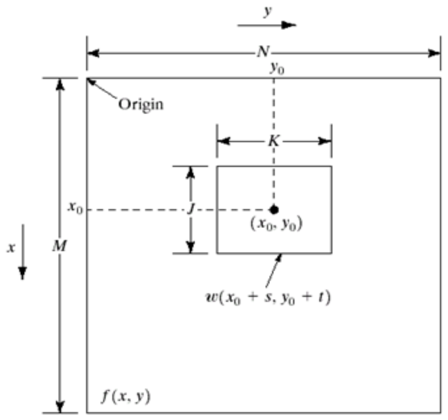


Figure (11) Arrangement for Obtaining the correlation of *f* and *w* at point (x_o,y_o)

area, giving the function *c(x,y)*. The maximum value(s) of *c* indicates the position(s) where *w* best matches *f*. *(The higher the similarity between two statistical sets, the larger the correlation coefficient will be generated*).

For image-processing applications in which the brightness of the image and template can vary due to lighting and exposure conditions, the images can be first normalized.
The correlation function given in Eq.(3) has the disadvantage of being sensitive to changes in the amplitude of *f* and *w*. For example, doubling all values of *f* doubles the value of *c(x, y)* An approach frequently used to overcome this difficulty is to perform matching via the correlation coefficient, which is defined as [11] :

$$r(x,y)=\frac{\sum_s\sum_t\left[f(s,t)-\bar{f}(s,t)\right][w(x+s,y+t)-\bar{w}]}{\sqrt{\sum_s\sum_t[f(s,t)-\bar{f}(s,t)]^2\sum_s\sum_t[w(x+s,y+t)-\bar{w}]^2}} \tag{4}$$

where x = 0, 1, 2,..., M – 1 , y = 0, 1, 2, ….., N-1, $\bar{w}$ is the average value of the pixels in *w* (computed only once). $\bar{f}$ is the average value of *f* in the region coincident with the current location of *w*, and the summations are taken over the coordinates common to both *f* and *w*. The correlation coefficient *y(x, y)* is scaled in the range -1 to 1. independent of scale changes in the amplitude of *f* and *w*.

4. Experimental Results and Discussions

In this section, the experimental results of the proposed algorithms is presented.
The proposed algorithm is implemented in MatLab 6.5 running on a computer with 1.6 GHz Atom processor and 1 GB RAM. The images utilized in the experiments are color images with size of 180×200 under various illumination condition and was obtained from the publicly available database of Computer Vision Science Research Projects[16].
Figure (12) shows example images used in the experiments.
The success rate of proposed algorithms for 278 image used is as follows :
- For adaptive threshold : 95.7 % (266 images succeed and 12 images failed).
- For fixed threshold : 73.4 % (204 images succeed and 74 images failed).

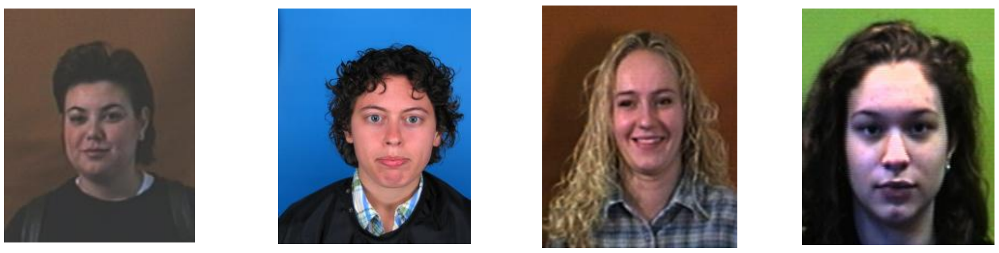


Figure (12) Example images used in the experiments

Table (1) shows the running times of the proposed algorithms on each stage. The total execution time of the proposed algorithm is about 4.16 second on average when the fixed threshold is used and about 4.37 seconds when adaptive threshold is used. It's remarkable that this execution time is reckoned for a program written in MatLab. Execution time for the 1st stage (image processing) is 0.23 seconds with fixed threshold and 0.34 seconds with adaptive threshold, and for the 2nd stage (face detection) is 0.82 seconds and for the 3rd stage (eye locating) is about 3.11 seconds.

Table (1) Running times of the proposed algorithms in seconds

Stage	Execution time (sec) (fixed threshold)	Execution time (sec) (adaptive threshold)
Image Processing	0.23	0.44
Face Detection	0.82	0.82
Eye Locating	3.11	3.11
Total	4.16	4.37

Figure (13) shows examples of images for which the proposed algorithm could correctly detect face, locate eyes and crop them in separate images.

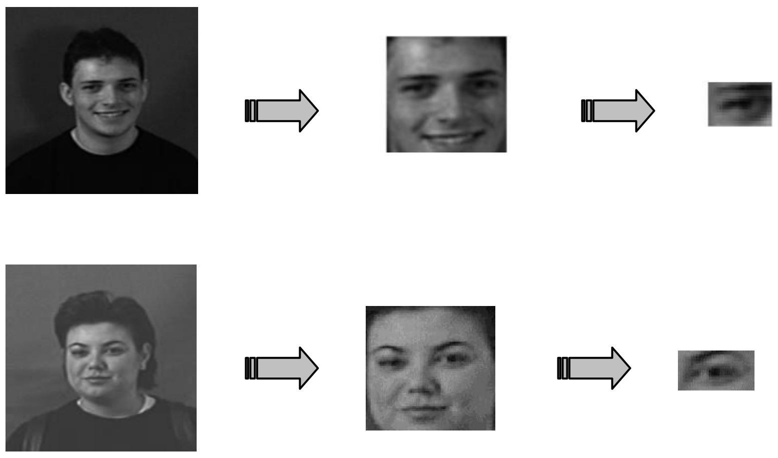


Figure (13) Examples of eyes and faces located correctly

In order to test the performance of the proposed method, a comparison of proposed method and other different methods is shown in Table (2).

Table (2) Comparison of different methods

Reference No.	Method	Success rate	Program used	CPU Speed	Excution time
[3]	Ratio Local Binary Pattern RLBP	100%	MatLab	2.8 GHz	Not mentioned
[6]	Dynamic Time Wrapping DTW	95%	MatLab	2 GHz	Not mentioned
[12]	Skin detection and morphological operations	90%	MatLab	2 GHz	15-20 seconds
Proposed algorithm in this paper	Image processing with template matching	95.7%	MatLab	1.6 GHz	4.37 seconds

As shown in Table (2), the proposed eye localization method achieves high rate comparing with other existing methods. Among 278 cases in the experiment, 95.7% cases are correctly localized. From the above analyses and comparison, it can be found that the proposed method is effective and robust.

5. Conclusions

An approach to accurate face and eye detection for frontal color and still images has been presented in this paper, combining image processing techniques with cross correlation. The proposed algorithm firstly makes use of image processing techniques to detect the face region. The precise locations of eyes are then detected by performing template matching (cross correlation).

The proposed method has been tested by images from Computer Vision Science Research Projects face database. Experimental results show that this method works well. For 278 images, the detection accuracy is 73.4% when the fixed threshold is used and 95.7% when the adaptive threshold is used. In addition, the average execution time of proposed algorithms (4.16 seconds and 4.37 seconds) for the fixed and adaptive threshold, respectively is acceptable, shows that this approach is also efficient.

After comparing and analyzing the detection results, it found that :

1. The false detection in the images failed is mainly due to the failure in face detection which leads to a false template matching.
2. All the images pass the 2nd stage (face detection) can pass the 3rd stage (eye locating) correctly, that means all the failure is because face detection failure.
3. The template matching is excellent and accurate method for detection although the long time in execution.
4. The system becomes more efficient and robust by using the adaptive threshold.
5. The high difference between the results obtained by fixed threshold (low success rate) and adaptive threshold (high success rate) is due to appropriate binary image obtained by adaptive threshold method which allow to detect the face edges accurately in the next stage.
6. Comparing the results of this paper with previous works shows that the proposed algorithms are efficient and robust.

References

[1] He Yan, Jianwei Li, and Ruxi Xiang, **"Robust Facial Features Localization on Rotation Arbitrary Multi-View Face in Complex Background"**, Journal Of Computers, Academy Publisher, VOL. 6, No. 2, February 2011, pp. 337-342.

[2] Kun Peng and Liming Chen, **"A Robust Algorithm for Eye Detection on Gray Intensity Face without Spectacles"**, JCS&T VOL. 5, No. 3, October 2005**.**

[3] Wencheng Wang, Faliang Chang, Guoqiang Zhang and Xiaoyan Sun,**"A Precise Eye Localization Method Based on Ratio Local Binary Pattern",** Journal of Convergence Information Technology, VOL. 6, No. 1, January 2011.

[4] ChengzheXu, Tauseef Ali, and Intaek Kim, "**A Robust Approach to Automatic Iris Localization"**, Journal of the Optical Society of Korea, VOL. 13, No. 1, March 2009, pp. 116-122.

[5] Hsuan M. G. Y., Kriegman D. J. and Ahuja N., **"Detecting Faces in Images: A Survey"**, IEEE Transactions On Pattern Analysis And Machine Intelligence, No. 1(24), 2002, pp. 34-57.

[6] Adwan S. and Arof H., "**A New Approach for an Efficient DTW in Face Detection through Eyes Localization"**, Electronics and Electrical Engineering, ISSN 1392 – 1215, No. 2(108), 2011.

[7] Vladimir Vezhnevets and Anna Degtiareva, **"Robust and Accurate Eye Contour Extraction"**, International Conference on Computer Graphics and Vision - GRAPHICON, 2003.

[8] Lam K., Yan H.**, "Locating And Extracting The Eye In Human Face Images",** Pattern Recognition, No. 29(5), 1996, pp. 771–779.

[9] Wang, K. W., Lam K., Siu Wan, **"An Efficient Algorithm For Human Face Detection And Facial Feature Extraction Under Different conditions"**, Pattern Recognition, No. 10(34), 2001, pp. 1993 –2004.

[10] Rowley H., Baluja S., Kanade T.**, "Neural Network-Based Face Detection"**, IEEE Trans. On Pattern Analysis and Machine Intelligence, No. 1(20), 1998, pp. 23 –57.

[11] Rafael C. Gonzalez and Richard E. Woods, **"Digital Image Processing"**, New Jersey: Prentice-Hall, Inc., 2nd Edition, 2002, ISBN: 0-201-18075-8, pp.701-703.

[12] Tanmay Rajpathaka, Ratnesh Kumarb and Eric Schwartzb, **"Eye Detection Using Morphological and Color Image Processing",** Florida Conference on Recent Advances in Robotics, FCRAR, 2009.

[13] **"Image Processing Toolbox For Use with MATLAB",** The Math. Works Inc., MA. USA, 2002.

[14] Elham Bagherian, Rahmita.Wirza.Rahmat and Nur Izura Udzir, **"Extract of Facial Feature Point",** IJCSNS International Journal of Computer Science and Network Security, VOL. 9, No. 1, January 2009.

[15] Qiu Chen, Koji Kotani, Feifei Lee, and Tadahiro Ohmi, "**An Accurate Eye Detection Method Using Elliptical Separability Filter and Combined Features"**, IJCSNS International Journal of Computer Science and Network Security, VOL. 9, No. 8, August 2009, pp. 65-72.

[16] Computer Vision Science Research Projects , **The Database of Faces**, at http://cswww.essex.ac.uk/mv/otherprojects.html by Dr. Libor Spacek.

The work was carried out at the University of Mosul