International Journal on Artificial Intelligence Tools
Vol. 0, No. 0 (2000) 000—000

(Leave $1\frac{1}{2}$ inch blank space for Publisher)

FACE DETECTION AND VERIFICATION USING GENETIC SEARCH

GEORGE BEBIS SATISHKUMAR UTHIRAM
Department of Computer Science
University of Nevada
Reno, NV 89557

MICHAEL GEORGIOPOULOS
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816

Received (received date)
Revised (revised date)

We consider the problem of searching for the face of a particular individual in a two-dimensional intensity image. This problem has many potential applications such as locating a person in a crowd using images obtained by surveillance cameras. There are two steps in solving this problem: first, face regions must be extracted from the image(s) (*face detection*) and second, candidate faces must be compared against the face of interest (*face verification*). Without any *a-priori* knowledge about the location and size of a face in an image, every possible image location and face size should be considered, leading to a very large search space. In this paper, we propose using Genetic Algorithms (GAs) for searching the image efficiently. Specifically, we use GAs to find image sub-windows that contain faces and in particular, the face of interest. Each sub-window is evaluated using a fitness function containing two terms: the first term favors sub-windows containing faces while the second term favors sub-windows containing faces similar to the face of interest. Both terms have been derived using the theory of eigenspaces. A set of increasingly complex scenes demonstrate the performance of the proposed genetic-search approach.

Keywords: face detection, face verification, genetic algorithms, eigenfaces.

1. Introduction

Searching for the face of a particular individual in a two-dimensional image has many potential applications such as locating a person in a crowd from images taken by surveillance cameras. This problem contains two main sub-problems: *face detection* and *face verification*. Specifically, given an image, the first step is to extract all possible regions that might contain a face. The second step is to compare the extracted faces against the face of interest. Both of these problems are very challenging. Without any *a-priori* knowledge about the location and size of a face in an

image, we will have to consider almost every possible location and size, leading to a very large search space. In addition, changes to lighting conditions, facial expression and pose make the search space even larger and more complex.

For face verification, methods based on correlation [1] and neural networks [2] are quite common. For face detection, many methods consider color [3] [4] (e.g., skin color distribution) or motion [5] [6] information to find the face region(s) quickly without resorting to an exhaustive search. Such information, however, is not always available. Face detection methods can be classified into two main categories: methods based on facial features [7] [8] or face models [9] [10] (e.g., template) and methods based on face representations learned from a large number of examples (face images) using statistical approaches (e.g., eigenfaces) [11] [12] or neural networks [13] [14]. In general, methods in the second category are more practical since they are less time consuming and do not rely on special features or models.

The method of eigenfaces uses Principal Components Analysis (PCA) [15] to linearly project the image space to a low dimensional subspace (eigenspace). This subspace is defined by the principal components (eigenfaces) of the distribution of face images (i.e., the most important eigenvectors of the covariance matrix of the set of faces). Each face is represented as a linear combination of the eigenfaces. Given an image, sub-images of different size are extracted at every image location. To classify an image as a face, its distance from the eigenspace space is computed. In [13] [14], retinally connected neural networks examine sub-windows of the image to decide whether they contain a face. Extensive training using a representative set of both face and non-face examples is required for the neural networks to learn the concept of face.

In this paper, we investigate the idea of using Genetic Algorithms (GAs) [16] [17] to search the image for the face of interest. GAs are search procedures which have shown to perform well when considering large search spaces such as the one considered in this work. GAs operate iteratively on a population of structures, each one of which represents a candidate solution to the problem at hand. Each structure, is modified in a much the same way that populations of individuals evolve under natural selection. Variations among the individuals in the population result in some individuals being more fit than others (i.e., better solutions). In the past, GAs have been used to solve various difficult problems such as target recognition [18], facial feature extraction [19], and object recognition [20] [21].

Initially, the GA starts by extracting random sub-windows from the input image. A sub-window is retained in subsequent generations if it contains a face, especially if it is the face we are looking for. Each sub-window is evaluated using a fitness function which contains two terms. The first term favors sub-windows containing faces (*face detection term*) while the second term favors sub-windows containing faces similar to the face of interest (*face verification term*). Both terms have been derived using the theory of eigenspaces [11] [12] (eigenfaces). In the original eigenface approach, a single eigenspace was used, built from a large set of images from different individuals. Both face detection and recognition were performed using the same

eigenspace (i.e., by computing the "distance from face space" and "distance in face space"). Here, we use a separate eigenspace for each case.

The first eigenspace is built using images from different individuals, as in the original approach, and is used to define the first term of the fitness function (face detection term). There is a difference, however, between the original approach and the one used here: we do not use the original images but images filtered by Sobel operator [22]. Our experiments indicate that this preprocessing step enhances facial features (e.g., face contour, eyes, mouth, nose) making it more robust to distinguish between faces and non-faces. The second eigenspace is built using images of the face of interest under different lighting conditions, facial expression, and pose. The second term of the fitness function (face verification term) is defined using the distance from this eigenspace.

The approach proposed here has similarities with the approach of Swets et. al [23]. In that approach as well as in our approach, the GA searches for faces by extracting sub-windows from the image, however, there is not guarantee that the sub-windows fall inside the input image. To deal with this problem, they included an extra term to the fitness function to penalize those windows. This, however, adds unnecessary complexity to the fitness function and increases time requirements. Here, we use an improved encoding scheme which ensures that all sub-windows extracted by the GA fall inside the input image. Also, the extracted sub-windows were evaluated by computing the distance of each sub-window from the "mean" face of interest. Here, we employ a more powerful fitness function based on the theory of eigenspaces. An earlier version of this work appears in [24]

The rest of the paper is organized as follows: In Section 2, we provide brief review of the eigenface approach while Section 3 contains an introduction to genetic algorithms. Section 4 presents the proposed approach in detail including the preprocessing steps, eigenspace representation, encoding, and fitness evaluation. In Section 5, we present our experiments and we discuss our experimental results. Finally, Section 6 contains our conclusions.

2. The method of eigenfaces

The method of eigenfaces is based on Principal Component Analysis (PCA) [15], a standard statistical technique for reducing the dimensionality of data while attempting to preserve as much of information as possible in terms of variance. The key idea is to represent each data in a low dimensional space defined by the most important eigenvectors (i.e., "eigenfaces") of the covariance matrix of the data distribution. A complete description of the eigenface approach can be found in [12]. Here, we just summarize the main ideas.

Representing each image $I(x,y)$ as a N x N vector Γ_i, first the average face Ψ is computed:

$$\Psi = \frac{1}{R}\sum_{i=1}^{R}\Gamma_i, \tag{1}$$

where R is the number of faces in the training set. Next, the difference Φ of each face from the average face is computed: $\Phi_i = \Gamma_i - \Psi$. Then, the covariance matrix is estimated by:

$$C = \frac{1}{R}\sum_{i=1}^{R} \Phi_i \Phi_i^T = AA^T, \tag{2}$$

where, $A = [\Phi_1 \Phi_2 ... \Phi_R]$. The eigenspace can then be defined by computing the eigenvectors u_i of C. Since C is very large (N^2 x N^2), computing its eigenvectors will be very expensive. Instead, we can compute v_i, the eigenvectors of $A^T A$, an R x R matrix. Then, u_i can be computed from v_i as follows (the details are given in [12]):

$$u_i = \sum_{j=1}^{R} v_{ij} \Phi_j, j = 1, ..., R \tag{3}$$

Usually, we only need to keep a smaller number of eigenvectors R', corresponding to the largest eigenvalues. Given a new image Γ, we subtract the mean ($\Phi = \Gamma - \Psi$) and we compute its projection:

$$\hat{\Phi} = \sum_{i=1}^{R'} w_i u_i, \tag{4}$$

where $w_i = u_i^T \Phi$ are the coefficients of projection.

An image is considered to be a face if the mean square error (called the *distance from face space (dffs)*) between its representation using the most important eigenvectors and its normalized counterpart (e.g., the difference of the input image and the mean image), is small. Also, an image is considered to be a face found in the data set if the error (called the *distance within face space (difs)*) between the coefficients of the eigenvectors used to represent the image and the face in the data set is small.

3. Background on Genetic Algorithms

This section contains a brief summary of the fundamentals of Genetic Algorithms (GAs). Goldberg [17] provides a great introduction to genetic algorithms and the reader is referred to this source as well as the survey papet of Srinivas and Patnaik [25] for further information.

GAs are a class of optimization procedures inspired by the biological mechanisms of reproduction. GAs operate iteratively on a population of structures, each one of which represents a candidate solution to the problem at hand, properly encoded as a string of symbols (e.g., binary). A randomly generated set of such strings forms the initial population from which the GA starts its search. Three basic genetic operators guide this search: selection, crossover, and mutation. The genetic search process is iterative: evaluating, selecting, and recombining strings in the population during each iteration (generation) until reaching some termination condition. The

basic algorithm, where $P(t)$ is the population of strings at generation t, is given below:

```
t = 0
initialize P(t)
evaluate P(t)
while (termination condition not satisfied) do
begin
  select P(t+1) from P(t)
  recombine P(t+1)
  evaluate P(t+1)
  t = t+1
end
```

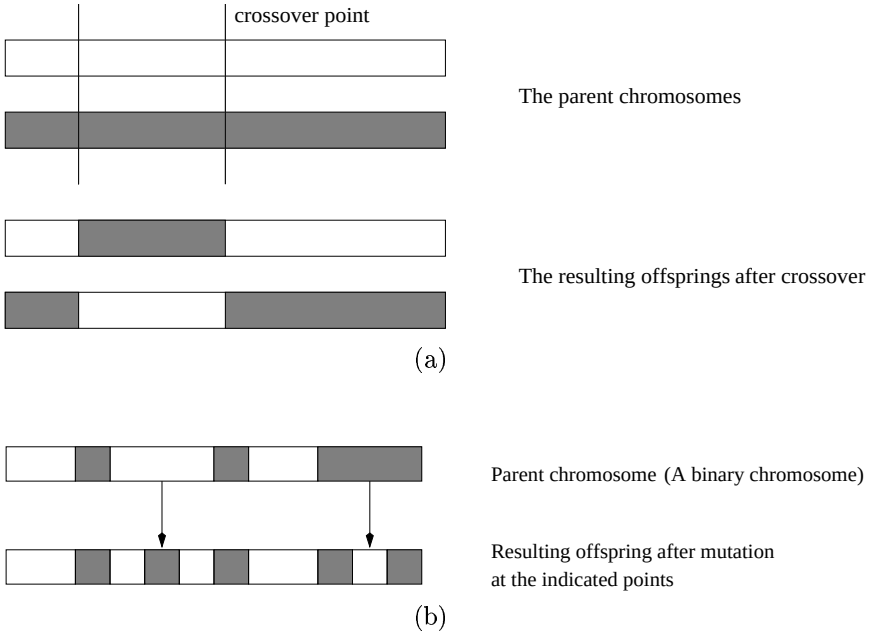

Fig. 1. Various genetic operators: (a) Crossover operator. (b) Mutation operator.

Evaluation of each string is based on a fitness function that is problem-dependent. It determines which of the candidate solutions are better. This corresponds to the environmental determination of survivability in natural selection. Selection of a string, which represents a point in the search space, depends on the string's fitness relative to that of other strings in the population. It probabilistically removes from the population those points that have relatively low fitness. Mutation, as in natural systems, is a very low probability operator and just flips a specific bit. (see Figure 1(b)). Mutation plays the role of restoring lost genetic material. Crossover in con-

trast is applied with high probability. It is a randomized yet structured operator that allows information exchange between points. Its goal is to preserve the fittest individuals without introducing any new value. Figure 1(a) illustrates a two-point crossover.

In summary, selection probabilistically filters out solutions that perform poorly, choosing high performance solutions to concentrate on or *exploit*. Crossover and mutation, through string operations, generate new solutions for *exploration*. Given an initial population of elements, GAs use the feedback from the evaluation process to select fitter solutions, generating new solutions through recombination of parts of selected solutions, eventually converging to a population of high performance solutions.

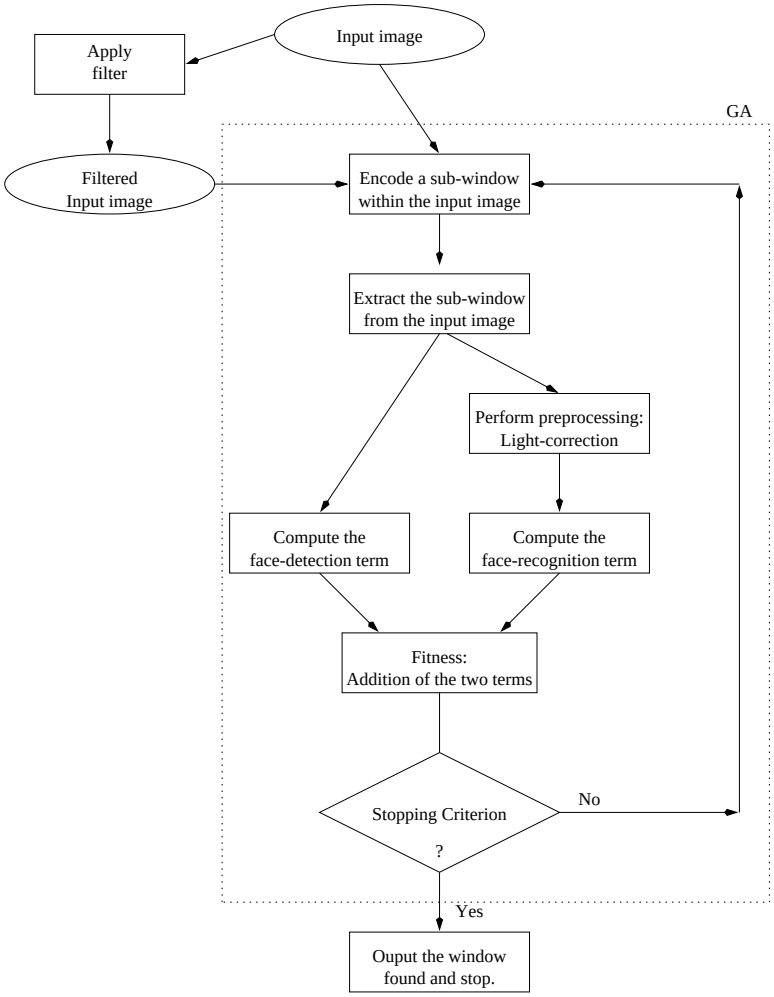

Fig. 2. The main steps of the proposed approach.

4. Methodology

In this section, we describe the genetic search approach. Figure 2 shows the main steps. In the following sections, we discuss the steps of the proposed approach in detail.

4.1. *Preprocessing*

To compute the eigenfaces, first the training images must be registered. The procedure used is similar to the one given in [13]. Specifically, the eyes, tip of the nose, and the corners and center of the mouth of each face were labeled manually. These points were then used to normalize each face to same scale, orientation and position. The normalization was performed by mapping the facial features to some fixed locations in an N x M image. The mapping was assumed to be an affine transformation, computed iteratively. The steps are described below:

Step1: Let $\bar{F}$ be a vector which contains the average positions of each labeled feature over all subimages. Initialize $\bar{F}$ with the feature locations in the first subimage F_1.

Step2: The feature coordinates in $\bar{F}$ are transformed so that the average locations of the eyes (P_1 and P_2) and tip of the nose (P_3) appear at predetermined locations (P_1^f,P_2^f,P_3^f respectively) in a N x M window (see Figure 3). A 48 x 40 window was used in our experiments. An affine transformation is used to register the images:

$$P_1^f = AP_1 + b \tag{5}$$

$$P_2^f = AP_2 + b \tag{6}$$

$$P_3^f = AP_3 + b \tag{7}$$

The above equations can be rewritten as

$$Pc_1 = p_x \tag{8}$$

$$Pc_2 = p_y \tag{9}$$

where

$$P = \begin{bmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \\ X_3 & Y_3 & 1 \end{bmatrix}$$

$$p_x = \begin{bmatrix} X_1^f \\ X_2^f \\ X_3^f \end{bmatrix}$$

$$p_y = \begin{bmatrix} Y_1^f \\ Y_2^f \\ Y_3^f \end{bmatrix}$$

$$c_1 = \begin{bmatrix} a_{11} \\ a_{12} \\ b_1 \end{bmatrix}$$

$$c_2 = \begin{bmatrix} a_{21} \\ a_{22} \\ b_2 \end{bmatrix}$$

(c_1 and c_2 are the parameters of the affine transformation).

Step3: For every subimage i in the training set, we compute the best affine transformation to align the features (eyes, tip of the nose) F_i with the average feature locations $\bar{F}$. Let's call the aligned feature locations F_i'.

Step4: Update $\bar{F}$ by averaging the aligned features F_i' for each subimage i.

Step5: If the error between $\bar{F}(t+1)$ and $\bar{F}(t)$ is less than a threshold, then stop; otherwise go to step 2.

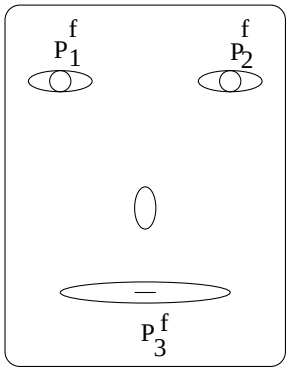

Fig. 3. A typical face showing the features of interest.

The alignment algorithm converges usually within seven iterations. For each subimage, it yields an affine transformation that maps that subimage to the 48 x 40 window. To avoid gaps in the normalized subimage, each point in the desired subimage was actually determined through the inverse affine transformation (see Figure 4). The inverse transformation is given by:

$$[X, Y, 1] = [X', Y', 1] \frac{1}{a_{11}a_{22} - a_{21}a_{12}} \begin{bmatrix} a_{22} & -a_{21} & 0 \\ -a_{12} & a_{11} & 0 \\ a_{21}b_2 - a_{22}b_1 & a_{12}b_1 - a_{11}b_2 & a_{11}a_{22} - a_{21}a_{12} \end{bmatrix}$$

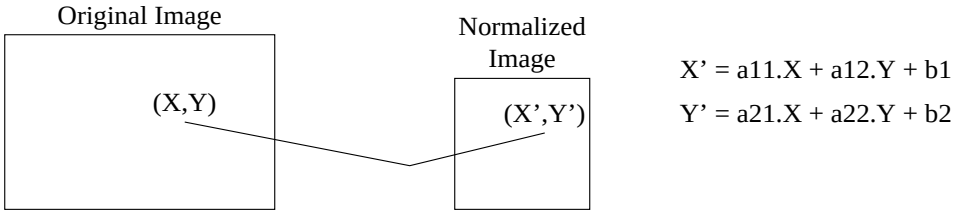

Fig. 4. Illustration of inverse mapping.

Figure 5 shows some examples of images before and after normalization. Each normalized image was then preprocessed to account for different lighting conditions and contrast. First, a linear function was fit to the intensity of the image. The form of the linear model used is shown below:

$$f(x, y) = ax + by + cxy + d \tag{10}$$

where $f(x, y)$ denotes the image and a,b,c,d are the coefficients to be determined. To solve for the coefficients, we use a least squares approach.

Fig. 5. Example showing images before (first row) and after (second row) normalization.

The result was subtracted out from the original image to correct lighting differences. Then, histogram equalization [22] was performed to correct for different camera gains and to improve contrast. Figure 6 shows some examples.

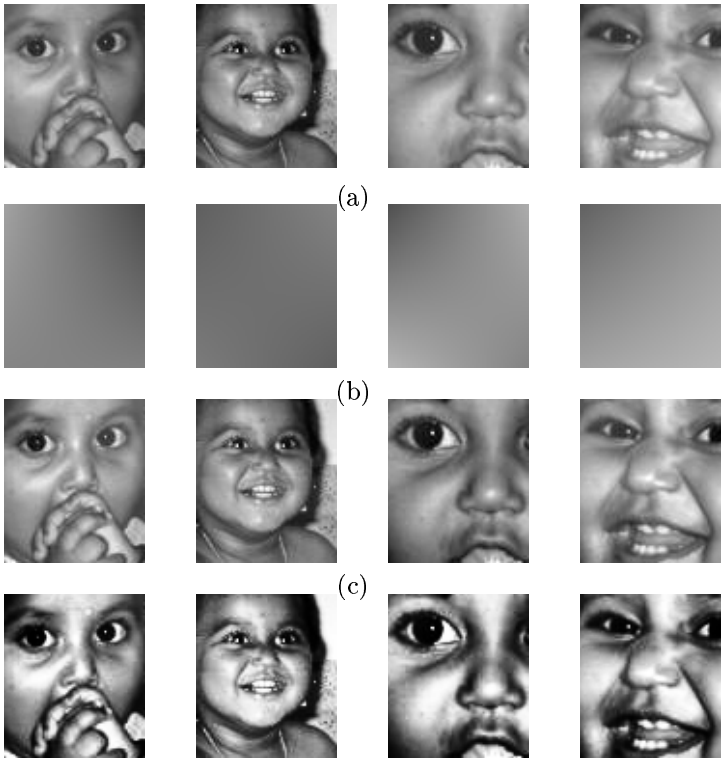

Fig. 6. The preprocessing steps: (a) original image, (b) linear fit, (c) light corrected image, (d) histogram equalized image.

4.2. *Eigenspace representation*

All the images in the training set were preprocessed as explained in the previous paragraph. The resulted images were used to compute the eigenspace representation. As we have already discussed, we are using two different eigenspaces: the first eigenspace is used for face detection and is built using images of different individuals. To improve face detection, we enhance the facial features of each face by computing the image gradient using the Sobel operator [22]. A 3 x 3 mask was used in our experiments. The resulted images were then thresholded (i.e., if a value was below 30, it was set to zero, otherwise, it remained unchanged) and used to built the first eigenspace. Figure 7 shows sample images and the first few eigenfaces. To build the second eigenspace, we used different images from the individual of interest. To allow for robustness, the images were obtained under different lighting conditions, facial expression, and pose.

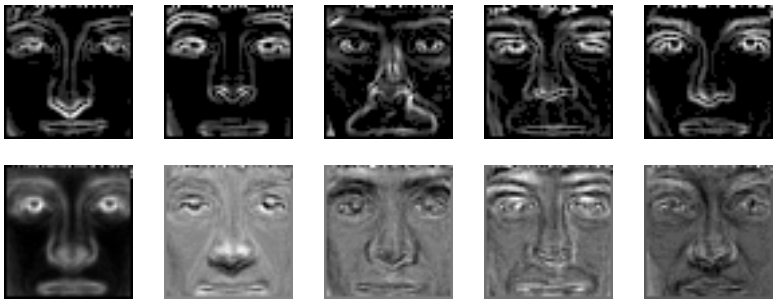

Fig. 7. The enhanced faces (first row) while and the mean and first four eigenfaces (second row).

4.3. *Encoding*

In our encoding scheme, each individual (chromosome) in the population represents a sub-window within the given input image. There are some constraints that need attention when encoding a rectangular window. First of all, to evaluate sub-windows of different size using the eigenface approach, we need to scale them down or up to the size of the images in the training set. Although we tried several different scaling algorithms [22], we used the nearest neighborhood scheme [22] since it performed as good as the other more sophisticated schemes but it is faster. Second, to avoid face distortions, we need to ensure that the sub-windows extracted by the GA have the same aspect ratio with that of the images in the training set. While trying to maintain the aspect ratio, we also need to make sure that each chromosome defines a window that lies within the bounds of the input image. Finally, during our experimentation, we found that the eigenface approach does not perform well when dealing with small sub-windows (it yields a large number of false positives when the image size is small). Thus, we added an additional constraint to the encoding scheme such that the GA could choose sub-windows not smaller than $MinX$ x $MinY$ ($MinX$=20 and $MinY$=20). These conditions are depicted in Figure 8.

To define a sub-window, while at the same time satisfy the constraints discussed above, we encode three of the four points that define a window. We employed two different versions of encoding as shown in Figure 9 The selection of scheme1 is based upon the condition that the aspect ratio of the input image is greater than the aspect ratio of the images in the training set. If the opposite is true, scheme2 is used (the reason in discussed below). UL stands for the upper-left corner of the sub-window, and BR stands for the bottom-right corner of the defined sub-window. Let L be the length of each chromosome. Then, L is given by $L = 3m$, where m is the number of bits used to represent each point encoded in the chromosome. The following constraint should be satisfied:

$$2^m >= max(input\ image's\ height, input\ image's\ width) \tag{11}$$

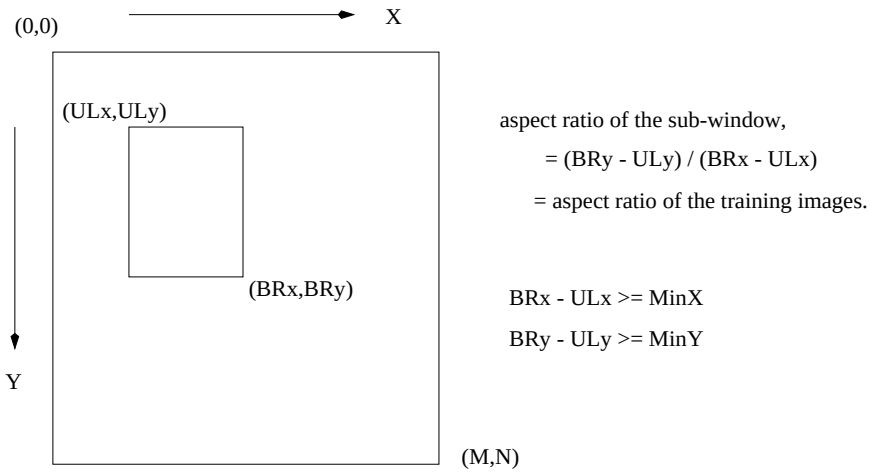

Fig. 8. The constraints that must hold true for a sub-window.

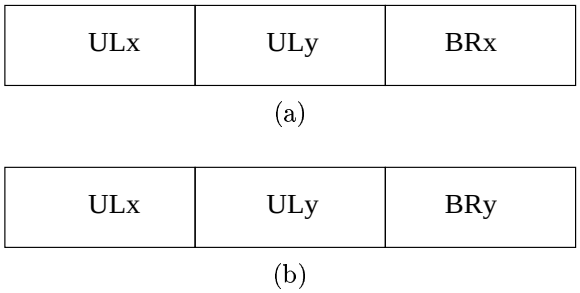

Fig. 9. Our encoding schemes, (a) scheme1, (b) scheme2.

Given that the sub-window lies within the image and that its aspect ratio is maintained, we need to compute the fourth point that completes the definition of the boundaries of the sub-window. The following discussion is based on Figure 8 and on the assumption that we use encoding scheme1 (Figure 9(a)). Let A be the aspect ratio of the images in the training set. In order for the sub-window and the training images to have the same aspect ratio, the fourth point, BR_y, that completes the definition of the window, is calculated as follows:

$$BR_y = A * (BR_x - UL_x) + UL_y \tag{12}$$

BR_y should satisfy the following inequality:

$$MinY <= BR_y < M \tag{13}$$

Substituting (12) in (13), we have:

$$MinY <= A * (BR_x - UL_x) + UL_y < M \tag{14}$$

or

$$c<=BR_x < d \tag{15}$$

where $c = (MinY - UL_y + A * UL_x)/A$ and $d = (M - UL_y + A * UL_x)/A$. BR_x should also lie inside the image, thus, it should satisfy the following inequality:

$$MinX<=BR_x < N \tag{16}$$

To decode BR_x, a linear mapping from $[0, 2^m - 1]$ to $[max(MinX, c), min(N, d)]$ is used. The other two points, UL_x and UL_y, are decoded using a linear mapping from $[0, 2^m - 1]$ to $[0, N - MinX]$ and from $[0, 2^m - 1]$ to $[0, M - MinY]$, respectively. Using (12) we can compute the value of BR_y. Similar calculations are performed when using encoding scheme2. To see when each scheme needs to be used, let us consider (15). From (15), we have that $MinX<=c$ and $d < N$. Let us assume for simplicity that $MinX = MinY = 0$. Then, $0<=c$ implies that $A < M/N$ while $d < N$ implies that $UL_x < N$ which is always true. Thus, if $A < M/N$ scheme1 is used, otherwise, scheme2 is used.

4.4. *Fitness Evaluation*

Each individual in the population needs to be evaluated. Based on its fitness it will be decided if it will survive in subsequent generations. As we have already discussed, the fitness function used here contains two terms. The first term denotes how close the sub-window is to the face space (face detection term). The second term, denotes how close the sub-window resembles the face of interest (face verification term). To compute the face detection term, we compute its $dffs_{detection}$ using the eigenspace built from different individuals. To compute the face verification term, we compute its $dffs_{verification}$ using the eigenspace built from different face images from the individual of interest. This has yielded better results than including the images of the face of interest in the first eigenspace, using the "distance in face space" as the second term of the fitness function.

The less the value of $dffs_{detection}$, the more the sub-window resembles a face; and the less the value of $dffs_{verification}$, the more it Resembles the face of interest. Thus, both of these values provide a measure of error. Since we need to maximize the fitness but minimize the error, our fitness function is given as:

$$Fitness = MAX - dffs_{detection} - dffs_{verification} \tag{17}$$

which changes the minimization problem to a maximization problem for the GA (MAX is a large constant value).

5. Experimental Results

We have used two training sets of faces in our experiments. The first set includes 38 images (see Figure 10(b) and is used to compute the face detection term of the fitness function. As we have already discussed, we do not use the original images for

building the eigenspace but the images obtained after applying the Sobel operator, followed by thresholding (see Figure 10(c)). The second set contains 20 face images from the individual of interest. Figure 10(a) shows one of the sets we used in our experiments. Different lighting, facial expressions and slight tilts were allowed to make verification more robust.

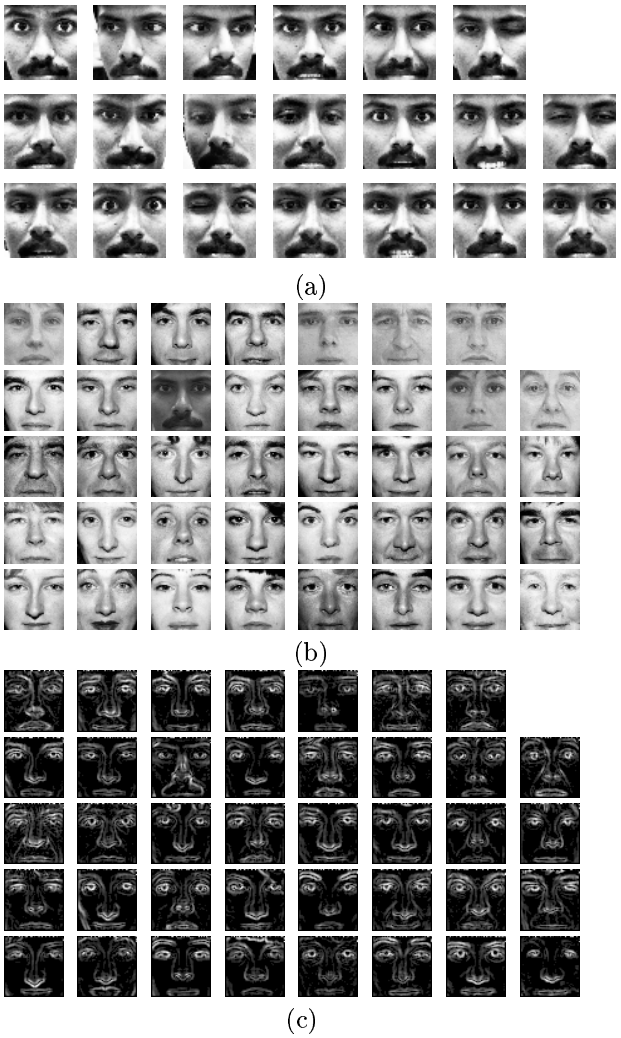

Fig. 10. (a) Lighting-corrected, histogram-equalized training set used for face verification, (b) original images used for face detection (c) images processed with the Sobel operator, followed by thresholding.

Our selection strategy was cross generational. Assuming a population of size P, the offspring double the size of the population and we select the best P individuals from the combined parent-offspring population for further proces sing [18]. This kind of selection does well with small populations and leads to quick convergence

(sometimes prematurely). We also linearly scale fitnesses to try maintain a constant selection pressure. A simple two-point crossover and point mutation were employed in our experiments. The crossover probability used was 0.95, the mutation probability was 0.05 and the scaling factor was 1.2. The MAX constant in the fitness function was set to 18000.

We have tested our algorithm on several scenes. Here, we show results on seven scenes of increasing complexity (scene1 - scene7) with the face of interest being present in all of them (see Figure 11). The population size was set to 100 for scene1 and scene2 and to 150 for all other scenes. On the average, 40 generations were required for the algorithm to find the face of interest. For each scene, we tested our approach 10 times with different random seeds. Performance plots indicate that the GA gets close to the correct solution quickly and then spends most of its time making little progress.

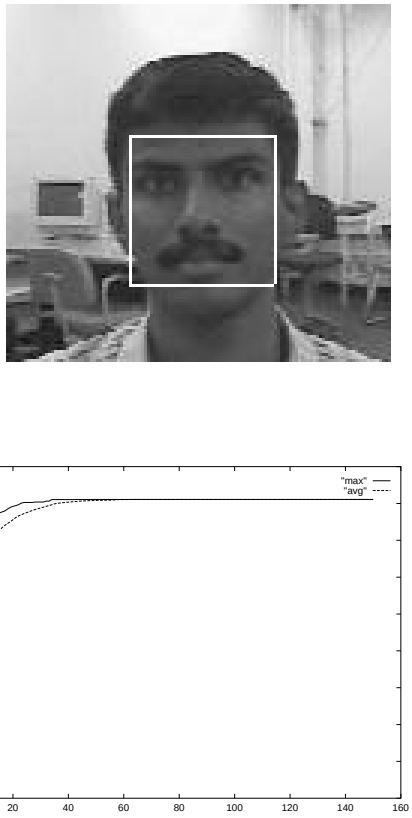

Fig. 11. Scene1 (Size: 120 x 128) and its performance plot.

Scene1 is relatively simple containing only the face of interest. What is interesting about this image is that it is not a very recent picture of the person of interest.

The GA was able to find the face in all cases (see Figure 11). Next, we tested GA's ability to find the face of interest, assuming different size. Figure 12 shows scene2 with the face of interest being very close to the camera. The GA converged to the face in all experiments. To test GA's robustness against occlusion, we tested it on Scene3 where the person of interest is wearing glasses. As seen in Figure 13, the GA performed a good job finding the face. It should be mentioned that in all three cases, the fintness of the solutions found was fairly high. We have also tested images containing different faces. The GA converged to these faces as well, however, the fitness of the solutions was much lower.

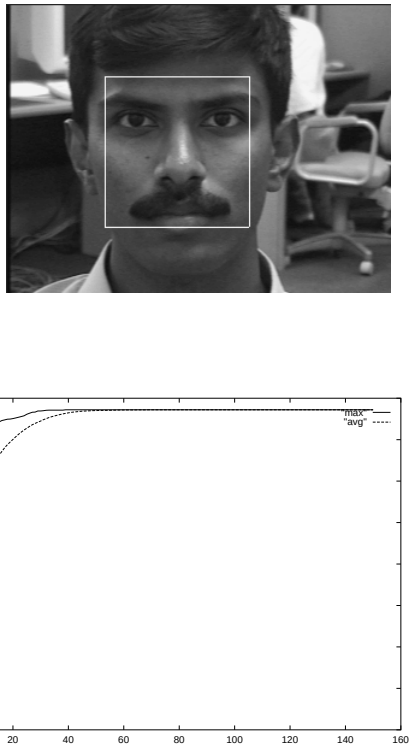

Fig. 12. Scene2 (Size: 240 x 320) and its performance plot.

Scenes 4-7 were used to test GA's performance in the presence of other faces. In the case of scenes 4 and 5, the faces present resemble the face of interest as they are from the same ethnic group and all have moustache. In all ten experiments, the face of interest was found correctly. Scenes 6 and 7 contain faces from other races. In the case of scene 6, the GA converged to the face of interest 6 out of 10 times. The other four times it converged to the face shown in Figure 16. Our analysis indicated that the two solutions have close fitnesses. In the case of scene

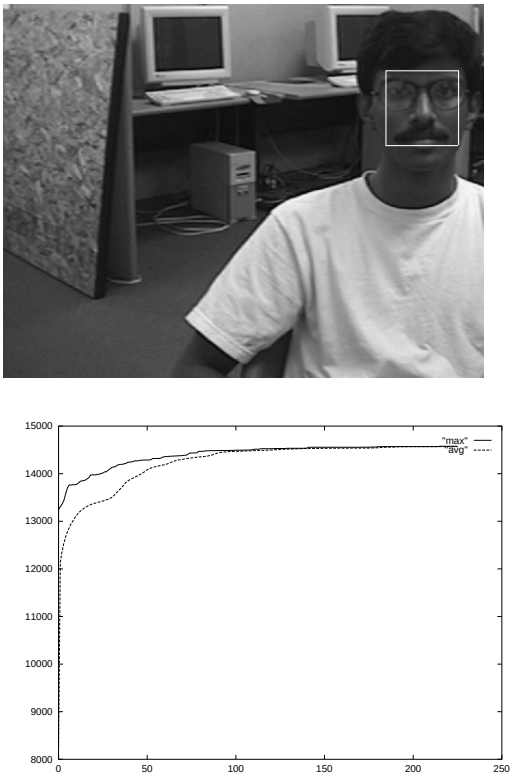

Fig. 13. Scene3 (Size: 350 x 444) and its performance plot.

7, the GA converged to the face of of interest 9 out of 10 times. The face to which it converged one time is shown in Figure 17. Our analysis indicated again that the fitnesses were close (but not as close as in the case of scene 6). These two cases indicate that more powerful fitness functions are required. We elaborate on this issue in the next section. Nevertheless, we were able to achieve 100% accuracy by increasing the population size.

The total number of sub-windows that the GA explores is much less compared to the entire search space. To compute this number, we consider the product of population size and number of generations it took for the GA to converge to the correct face. To estimate the efficiency gained, we computed the number of all possible sub-windows assuming that the smallest sub-window is of size 20 x 20 (since the GA does not consider smaller sizes than this). Table 1 shows the ratio of sub-windows searched by the GA over the total number of possible sub-windows. Clearly, the GA was able to find the face of interest by searching only a very small portion of the solution space.

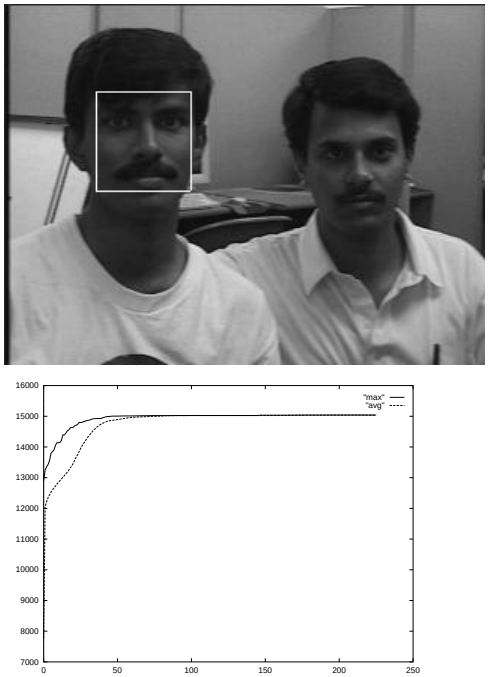

Fig. 14. Scene4 (Size: 240 x 320) and its performance plot.

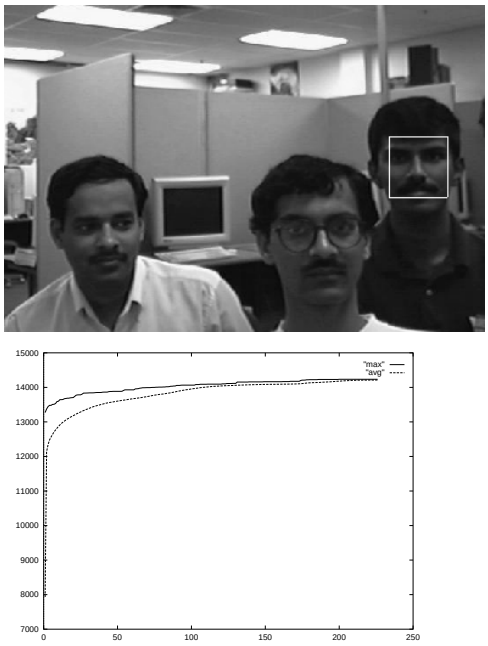

Fig. 15. Scene5 (Size: 269 x 395) and its performance plot.

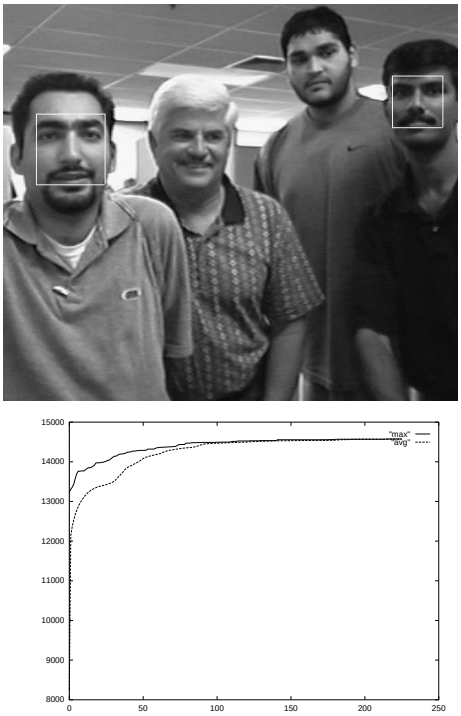

Fig. 16. Scene6 (Size: 438 x 497) and its performance plot.

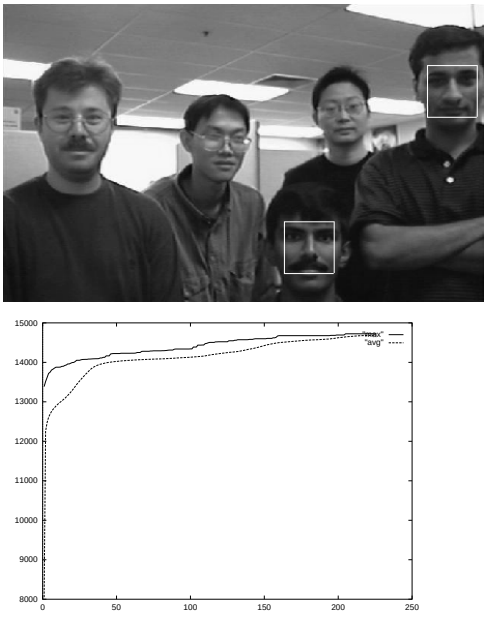

Fig. 17. Scene7 (Size: 288 x 464) and its performance plot.

Table 1. Summary of results.

1	120x128	8,051,400	3200	0.000397
2	240x320	673,688,000	4000	0.00000594
3	350x444	3,549,060,000	6975	0.00000197
4	240x320	673,688,000	14625	0.0000217
5	269x395	1,454,510,000	11250	0.0000077
6	438x497	7,623,950,000	17100	0.0000022
7	288x464	2,466,860,000	21150	0.00000857

6. Conclusions

We have proposed using genetic algorithms to search for the face of a particular individual in an image. Specifically, we used GAs to search for sub-images that might contain the face of interest. To evaluate each sub-window, we proposed a fitness function derived from the theory of eigenspaces. Also, we proposed an encoding scheme which ensures that the sub-images extracted lie inside the input image and have the same aspect ratio with the training images. Our experimental results demonstrate that the GA approach is promissing.

The current work has certain limitations: we have considered mostly frontal face images, without significant changes in lighting conditions facial expression and pose. These limitations are in fact limitations of the eigenspace approach used to derive the fitness function and not of the genetic search approach. We allowed for robustness by employing a separate eigenspace for the face of interest, however, more powerful approaches are need in defining better fitness functions. In the future, we plan to investigate improved methods such as the methods of Fisherfaces [26], Discriminant Analysis [27] and Independent Component Analysis (ICA) [28]. It should be mentioned that in several practical applications (i.e., criminal identification), a large image gallery of the face of interest might not be available. In such cases, we might be able to generate a synthetic image gallery [29].

References

[1] Matas J. Jonsson K. and Kittler J. Fast face localisation and verification. *Image and Vision Computing*, 17:575–581, 1999.

[2] Winter R. Verification of personal identity using facial images. *SPIE Conference*, 2277:63–70, 1994.

[3] Yang G. and Waibel A. A real-time face tracker. *Workshop on Applications of Computer Vision*, pages 142–147, 1996.

[4] Wu H. Chen Q. and Yachida M. Face detection from color images using a fuzzy pattern matching method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6):557–563, 1999.

[5] Wang C. and Brandstein M. A hybrid real-time face tracking system. *Proceedings of ICASSP*, pages 3737–3740, 1998.

[6] Eleftheriadis A. and Jacquin A. Automatic face location detection for model-assisted

rate control in h.261-compatible coding of video. *Signal Processing: Image Communication*, 7(4–6):435–455, 1995.

[7] Huang C. and Chen C. Human facial feature extraction for face interpretation and recognition. *Pattern Recognition*, 25(12):1435–1444, 1992.

[8] Brunelli R. and Poggio T. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[9] Birchfield S. An elliptical head tracker. *1st Asilomar Conference*, pages 1710–1714, 1997.

[10] Kwon Y. and da Vitoria Lobo N. Face detection using templates. *Computer Vision and Pattern Recognition Conference*, pages 764–767, 1994.

[11] Kirby M. and Sirovich L. Application of the karhunen-loe'ye procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[12] Turk M. and Pentland A. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

[13] Baluja S. Rowley H. and Kanade T. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[14] Sung K. and T. Poggio T. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[15] Press W. et. al. Numerical recipes in c: the art of scientific programming. *Cambridge University Press*, 1990.

[16] Holland J. Adaptation in natural and artificial systems. *The University of Michigan Press*, 17:Ann Arbor, 1975.

[17] Goldberg D. Genetic algorithms in search, optimization and machine learning. *Addison-Wesley*, pages Reading, MA, 1989.

[18] Katz A. and Thrift P. Generating image filters for target recognition by genetic learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):906–910, 1994.

[19] Lin C. and Wu J. Automatic facial feature extraction by genetic algorithms. *IEEE Transactions on Image Processing*, 8(6):834–844, 1998.

[20] Bebis G. Louis S. and Varol Y. Using genetic algorithms for model-based object recognition. *International Conference on Imaging Science, Systems, and Technology*, pages 1–6, 1998.

[21] Bebis G. Louis S. and Fadali S. Using genetic algorithms for 3d object recognition. *11th International Conference on Computer Applications in Industry and Engineering*, pages 13–16, 1998.

[22] R. Crane. A simplified approach to image processing. *Prentice Hall*, 1997.

[23] Punch B. Swets D. and Weng J. Genetic algorithms for object localization in a complex scene. *International Conference on Image Processing*, II:595–598, 1995.

[24] Bebis G. Uthiram S. Georgiopoulos M. Genetic search for face detection and verification. *IEEE International Conference on Intelligence, Information and Systems*, pages 360–368, 1999.

[25] M. Srinivas and L. M. Patnaik. Genetic Algorithms: A Survey. *IEEE Computer*, 27(6):17–26, June 1994.

[26] Belhumeur P. Hespanha J. and Kriegman D. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[27] . Swets D. Weng J. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

[28] Bartlett M. and Sejnowski T. Independent components of face images: A representation

for face recognition. *4th Joint Symposium on Neural Computation Proceedings*, 1997.

[29] Beymer D. and Poggio T. Face recognition from one example view. *International Conference on Computer Vision*, pages 500–507, 1995.