Journal of Mobile, Embedded and Distributed Systems, vol. III, no. 4, 2011
ISSN 2067 – 4074

A Real-Time Face Recognition System Using Eigenfaces

Daniel Georgescu
Department of Economic Informatics and Cybernetics
University of Economic Studies, Bucharest
ROMANIA
dariusgeo@yahoo.com

***Abstract*:** A real-time system for recognizing faces in a video stream provided by a surveillance camera was implemented, having real-time face detection. Thus, both face detection and face recognition techniques are summary presented, without skipping the important technical aspects. The proposed approach essentially was to implement and verify the algorithm Eigenfaces for Recognition, which solves the recognition problem for two dimensional representations of faces, using the principal component analysis. The snapshots, representing input images for the proposed system, are projected in to a face space (feature space) which best defines the variation for the face images training set. The face space is defined by the 'eigenfaces' which are the eigenvectors of the set of faces. These eigenfaces contribute in face reconstruction of a new face image projected onto face space with a meaningful (named weight).The projection of the new image in this feature space is then compared to the available projections of training set to identify the person using the Euclidian distance. The implemented system is able to perform real-time face detection, face recognition and can give feedback giving a window with the subject's info from database and sending an e-mail notification to interested institutions.

Key-Words: Feature vector, eigenfaces, eigenvalues, eigenvector, face recognition, real-time

1. Introduction

The face is the primary focus of attention in the society, playing a major role in conveying identity and emotion. Although the ability to infer intelligence or character from facial appearance is suspect, the human ability to recognize faces is remarkable. A human can recognize hundreds of faces learned throughout the lifetime and identity familiar faces at a glance even after years of separation. This skill is quite robust, despite of large chances in the visual stimulus due to viewing conditions, expression, ageing, and distractions such as glasses, beards or changes in hair style.

Face recognition has become an important issue in many applications such as security systems, biometric authentication, human-computer interaction, criminal identification… Even the ability to merely detect faces, as opposed to recognizing them, can be important.

Face recognition has several advantages over other biometric technologies: it is natural, non-intrusive and easy to use. A face recognition system is expected to identify faces present in images and video automatically. It can operate in either or both of two modes: a) face verification (or authentica-tion), and b) face identification (or recognition) [1, 2]. In addition D.M. Blackburn [3] adds another task to a biometric system, the watchlist.

- Face verification: (Am I whom I claim I am?) this mode is used when the person provides an alleged identity. The system then performs a one-to-one search, comparing the captured biometric characteristics with the biometric template stored in the database. If a match is made the identity of the person is verified [1].
- Face identification: (Who am I?) this mode is used when the identity of the individual is not known in advance. The entire template database is then search for a match to the individual concerned, in a one-to-many search. If a match is made, the individual is identified [1]. It is important to note that a match does

not mean a sample that is identical to the template, but rather is within a given threshold [4, 2].

- The watchlist task: in the watchlist task the person does not claim any identity. The biometric sample of the individual is compared with the stored samples in a watchlist to see if the individual concerned is present in the watchlist [3, 5]. Examples of watchlist tasks could be comparing a flight passenger towards a database of known terrorists, or comparing a specific person with a list of missing persons. When a person is found that have a resemblance to one or more samples in the watchlist that is higher than the given threshold, the system should give an alarm and return the samples that triggered the alarm. When this alarm goes for an individual that is actually present in the watchlist and this person has the highest similarity score, it is called a *correct detect and identify*. An alarm that goes of even though the person is not present in the watchlist is called a *false alarm*, while the frequency which false alarms encounters is called the *false alarm rate* [3]. In an ideal system we want the false alarm rate to be 0% and the correct detect and identify rate to be 100%. However this is not possible, so we must compromise. The decision on whether to choose a system with a low false alarm rate and a medium correct detect and identify rate, or if we want a medium false alarm rate and a high correct detect and identify rate, depends on the usage of the system.

2. Face detection

A face recognition system includes, mainly, 4 modules: the face detection, image processing, extraction of face features and feature matching modules.

Face detection is the first step in a face recognition system. The performance of the entire face recognition system is influenced by the reliability of the face detection part. Given a set of images or a real-time video streaming, a reliable face detector should be able to identify and locate all the present faces regardless of their position, scale, orientation, age or expression.

2.1. Face detection – Canny egde algorithm

The purpose of edge detection in general is to significantly reduce the amount of data in an image, while preserving the structural properties to be used for further image processing.
J.F.Canny developed in 1986 [4] an algorithm that is optimal with regards to the following criteria:

1. Detection: The probability of detecting real edge points should be maximized while the probability of falsely detecting non-edge points should be minimized. This corresponds to maximizing the signal-to-noise ratio.
2. Localization: The detected edges should be as close as possible to the real edges.
3. Number of responses: One real edge should not result in more than one detected edge (one can argue that this is implicitly included in the first requirement).

1. The algorithm runs in 5 separate steps:

1. Smoothing / noise reduction: Blurring of the image to remove noise.
2. Finding gradients: The edges should be marked where the gradients of the image has
2. large magnitudes.
3. Non-maximum suppression: Only local maxima should be marked as edges.
4. Double thresholding: Potential edges are determined by thresholding.

ISSN 2067 – 4074

5. Edge tracking by hysteretic: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

Each step is described in the following subsections.

Smoothing

It is inevitable that all images taken from a camera will contain some amount of noise. To prevent that noise is mistaken for edges, noise must be reduced. Noise reduction is first realized by applying a Gaussian filter. The kernel of a Gaussian filter with a standard deviation of σ = 1.4, as shown in equation 1.

Here is an example of a 5x5 Gaussian filter applied to image I.

$$B = 1/159 * \begin{matrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{matrix} * I \quad (1)$$

The effect of blurring the test image with this Gaussian filter is shown in figure 1 b).

a) Original image b) After noise reduction

Figure 1. Lena [IMG3].

Finding gradients

The Canny algorithm basically finds edges where the grayscale intensity of the image changes the most. These areas are found by determining gradients of the image. Gradients at each pixel in the smoothed image are determined by applying what is known as the Sobel-operator. First step is to approximate the gradient in the x- and y-direction respectively by applying the kernels.

$$K_{GX} = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad K_{GY} = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix} \quad (2)$$

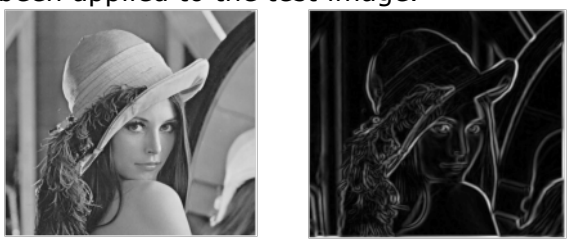
The gradient magnitudes (also known as the edge strengths) can then be determined as a Euclidean distance measure by applying the law of Pythagoras (equation. 3). It is sometimes simplified by applying Manhattan distance measure (equation 4) to reduce the computational complexity.

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (3) \qquad |G| = |G_x| + |G_y| \quad (4)$$

where :

- G_x / G_y are gradients;
- x / y are directions;

The Euclidean distance measure has been applied to the test image.



a) Original image b) Gradient magnitudes

Figure 2. Lena [IMG3]

Non-maximum suppression

The purpose of this step is to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges. Basically this is done by preserving all local maxima in the gradient image, and deleting everything else. The algorithm is for each pixel in the gradient image:

1. Round the gradient direction θ to nearest 45°, corresponding to the use of an 8-connected neighbourhood.
2. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient direction. I.e. if the gradient direction is north (theta = 90°), compare with the pixels to the north and south.

3. If the edge strength of the current pixel is largest; preserve the value of the edge strength.If not, suppress (i.e. remove) the value.

Double thresholding

The edge-pixels remaining after the non-maximum suppression step are (still) marked with their strength pixel-by-pixel. Many of these will probably be true edges in the image, but some may be caused by noise or color variations for instance due to rough surfaces. The simplest way to discern between these would be to use a threshold, so that only edges stronger that a certain value would be preserved. The Canny edge detection algorithm uses double thresholding. Edge pixels stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.

a) Original image b) Thresholding

Figure 3. Lena [IMG3]

Edge tracking by hysteresis

Strong edges are interpreted as "certain edges", and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. The logic is of course that noise and other small variations are unlikely to result in a strong edge (with proper adjustment of the threshold levels). Thus strong edges will (almost) only be due to true edges in the original image. The weak edges can either be due to true edges or noise/color variations. The latter type will probably be distributed independently of edges on the entire image, and thus only a small amount will be located adjacent to strong edges. Weak edges due to true edges are much more likely to be connected directly to strong edges.

a) Original image b) Final output

Figure 4. Lena [IMG3]

2.2. Face detection in images

Given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and, if present, return the image location [6]. While this appears as a trivial task for human beings, it is a very challenging task for computers, and has been one of the top studied research topics in the past few decades. The difficulty associated with face detection can be attributed to many variations in scale, location, orientation (in-plane rotation), pose (out-of-plane rotation), facial expression, lighting conditions, occlusions and others.

a) b) c)

Figure 5. Face detection in images at different ages and with different expressions [IMG1]

Face detection systems can be based on several cues: skin color (color images and video), motion (real-time face detection/video), facial/head shape, facial appearance or a combination of these.

Enough successful face detection systems are appearance-based. In these systems the main problem is to classify each scanned sub-window as a face or a non-face [7]. A face/non-face classifier can be learned using a training set of images representing both faces and non-

ISSN 2067 – 4074

faces. This training is possible because pixels on a face are highly correlated, while those in a non-face sub-window present a high irregularity. Efforts were been made for constructing complex fast classifiers.
As a face detection solution for the proposed face recognition system, the Viola-Jones algorithm has been implemented.

2.3. The Viola-Jones face detection system

The Viola-Jones [8,9] face detection system is faster than the previously published solutions, being able to detect faces in 384x288 pixels images at 15 frames per second on a conventional 700 MHz Intel Pentium III. There are three main parts which together give the proposed solution.

The first part is a new image representation called an integral image that allows a very fast feature evaluation. The used features are reminiscent of Haar basis functions. For a fast processing of these features the integral image representation for images was introduced. The integral image can be computed from an image using a few operations per pixel. Once computed, any one of these Haar-like features can be computed at any scale or location in constant time.
The second part is a method for constructing a classifier be selecting a small number of important features using AdaBoost learning algorithm. Within an image sub-window the total number of Haar features is far larger than the number of pixels. Thus, the large majority of the available features is excluded and the focus is pointed on a small set of critical features.
Each stage of the boosting process can be viewed as a feature selection process.
In the third part more complex classifiers are combined in a cascade structure, increasing the speed of the detector by moving the most complex processing on promising regions of the image.

2.3.1. Features

Motivated by the work of Papageorgiou et al. [10], Viola and Jones use three types of simple features.

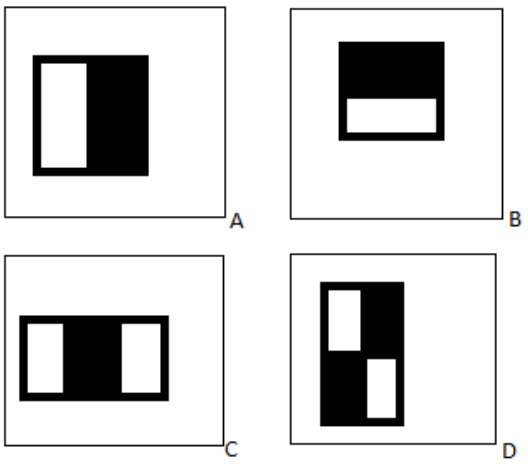


Figure 6. Haar wavelet-like feature: (A) and (B) – two-rectangle feature, (C) – three-rectangle feature, and (D) – four-rectangle feature. [9]

A simple feature is a scalar. It is calculated by summing up the pixels in the white region and subtracting those in the dark region.

2.3.2 The 'Integral image'

The rectangle features can be computed faster using an intermediate representation for the image, called integral image. The integral image at location x, y contains the sum of the pixels above and the left of x, y, as defined in [9]:

$$II(x,y) = \sum_{x' \le x, y' \le y} I(x', y'), \quad (5)$$

where:
- II(x, y) is the integral image, and
- I(x, y) is the original image

The integral image can computed in one pass over the original image with the following system of recurrence

$$S(x,y) = S(x,y-1) + I(x,y)$$
$$II(x,y) = II(x-1,y) + S(x,y) \quad (6)$$

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

Thus, using the concept of integral image, any rectangular sum from the original image can be computed using four array references.

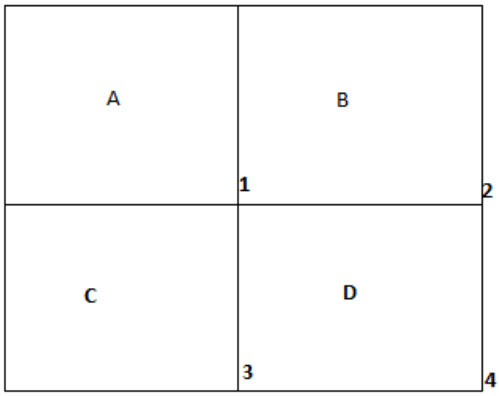


Figure 7. Source [9]

In figure 7, in point (1) the value of the integral image is the sum of the pixels of rectangle A. In point (2), the integral image's value is A+B. In point (3), it is A+C. In point (4), it is A+B+C+D. Thus, the rectangular sum in point D can be computed as 4+1 – (2+3) [9].

2.3.3. The AdaBoost learning algorithm

The second main contribution of the Viola-Jones method is the use of the AdaBoost learning algorithm for constructing strong classifiers.
The AdaBoost algorithm, in its original form, is used to combine a collection of weak classification functions in order to build a strong classifier, but in the Viola-Jones solution it will select the features, too.
The weak learning algorithm determinates the optimal threshold classification function and selects the single rectangle which best separates the positive and negative examples.

$$K_i(x) = \begin{cases} 1 & if \quad p_i f_i(x) < p_i \theta_i \\ 0 & otherwise \end{cases} \quad (7)$$

where:
- $K_i(x)$– weak classifier,
- $f_i(x)$ – feature,
- p_i – parity: indicates the direction of the inequality sign,
- Θ_i – threshold,
- x – a 24 x 24 sub-window of the original image.

A representation of the AdaBoost algorithm [9] is:

(1) The input
A training set of images is given: T = $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where

$$y_i = \begin{cases} 1, & \text{for positive examples} \\ 0, & \text{for negative examples} \end{cases} \quad (8)$$

(2) The initialization

The weights are initialized:

$$\omega_i^{(0)} = \frac{1}{2k} \text{ for the positive examples } (y_i = 1) \quad (9)$$

$$\omega_i^{(0)} = \frac{1}{2p} \text{ for the negative examples } (y_i = 0) \quad (10)$$

where k + p = n.

(3) The forward inclusion
For r = 1, …, T:

3.1 The weights are normalized:

$$\omega_r^{(i)} = \frac{\omega_r^r}{\sum_{j=1}^{n} \omega_r^j} \quad (11)$$

3.2 A classifier k_j, is trained for each feature f_j and it restricted to use a single feature. According to ω_r, the error is :

$$\varepsilon_j = \sum_i \omega_i \, | k_j(x_i) - y_i | \quad (12)$$

3.3 The classifier with the lowest error is chosen.
3.4 The weights are updated :

ISSN 2067 – 4074

$$\omega_{r+1}^{(i)} = \omega_{\mathrm{r}}^{(\mathrm{i})}\beta_{\mathrm{r}}^{1-\varepsilon_{\mathrm{i}}} \quad (13)$$

where:

- ε_i = 0, if example x_i is classified correctly
- ε_i = 1, otherwise
- $\beta_r = \frac{\varepsilon_r}{1-\varepsilon_r}$

(4) The output

The final classifier (strong classifier) is:

$$k(x) = \begin{cases} 1, & \text{if } \sum_{\mathrm{r}=1}^{\mathrm{T}} \alpha_{\mathrm{r}} k_r(x) \geq \frac{1}{2}\sum_{r=1}^{T} \alpha_r \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

where:

$$\alpha_r = \log\frac{1}{\beta_{\mathrm{r}}} \quad . \quad (15)$$

2.3.4 The cascade structure of strong classifiers

A two – rectangle classifier can be adjusted to detect 100% of the faces with a false positive rate of 40%. This means, a single classifier will not be enough to meet the system requirements. The solution for this is to implement a cascade of adjusted and trained classifiers.

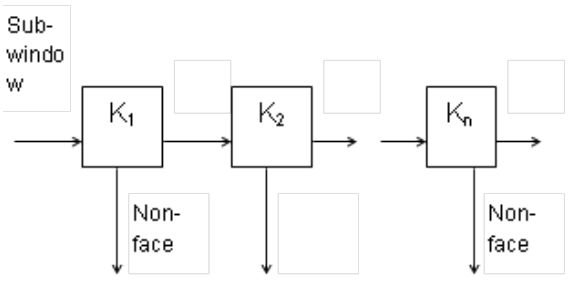


Figure 8. Cascade of strong classifiers [7]

In the above cascade structure, the simplest boos-ted classifiers reject the majority of negative sub-windows and detect the positive instances. The next classifier will complete a more difficult task than the previous. As the numbers of classifiers increases, the threshold is adjusted to minimize the false negatives and the detection rate grows.

3. Face Recognition Using Eigenfaces

The goal of a face recognition system is to discriminate input signals (image data) into several classes (persons), being important for a wide variety of problems like image and film processing, human-computer interaction, criminal identification and others.
The inputs signals can be highly noisy because of different lightning conditions, pose, expression, hair…. Nevertheless, the input signals are not completely random and even more, there are patterns present in each input signal. One can observe in all input images common objects like: eyes, mouth, nose and relative distances between these objects.
These common features are called *eigenfaces* [11] in the facial recognition domain (or *principal components* generally). They can be extracted out of the original image data through a mathematical technique called *Principal Component Analysis* (PCA).

3.1. Calculating Eigenfaces

Given a face image *I(x, y)*, it can be considered as a two-dimensional *NxN* array of intensity values. Also, an image can be normalized to a N^2 vector.
Thus, a typical image of size 256 by 256 becomes a vector of dimension 65.536, which can be thought as a point in a 65.536-dimensional space. Due to similarities of faces, a set of face images will map to a low-dimensional subspace of this huge space.

Using principal component analysis (or Karhunen – Loeve expansion), the vectors, which best account for the distribution of face images within the entire image space, are found. These vectors are those which have the largest corresponding eigenvalue. They compose a subspace within the entire image space, called by Turk and

Pentland [12] "face space". Because they are the eigenvectors of the covariance matrix corresponding to the original face images, and because they are face-like in appearance, they are referred by Turk and Pentland [12] as "eigenfaces".

If the training set of face images is Γ_1, Γ_2, …, Γ_M, then the average face of the set is defined by

$$\psi = \frac{1}{\mathrm{M}} \sum_{i=1}^{M} \Gamma_i \qquad (16)$$

Each face differs from the average face by the vector

$$\varphi_i = \Gamma_i - \psi \qquad (17)$$

This set of very large vector is the subject of PCA, which finds a set of M orthogonal vectors - v, which best describes the distribution of face images within the image space. The k-th vector, v_i, is chosen such that

$$\lambda_k = \frac{1}{M} \sum_{i=1}^{M} (v_k^T \varphi_i)^2 \qquad (18)$$

is a maximum, subject to

$$v_l^T v_k = \delta_{lk} = \begin{cases} 1, & \text{if } l = \mathrm{k} \\ 0, & \text{otherwise} \end{cases} \qquad (19)$$

The vectors v_k and the scalars λ_k are the eigenvectors and eigenvalues, respectively, of the covariance matrix:

$$C = \frac{1}{M} \sum_{i=1}^{M} \varphi_i \varphi_i^T = AA^T \qquad (20)$$

where

$$A = [\varphi_1 \varphi_2 \ldots \varphi_M] \qquad (21)$$

However, the covariant matrix C is $N^2 xN^2$ and there should be determined N^2 eigenvectors and eigenvalues. Computationally, this is a not very efficient as most of these eigenfaces (eigenvectors) are not useful for the proposed task (having low-weights in reconstructing the original image).
Consider the eigenvectors x_i of A^TA such that

$$A^T A x_i = \mu_i x_i \qquad (22)$$

multiplying both sides by A, results

$$AA^T A x_i = \mu_i A x_i \qquad (23)$$

where can be visible that Ax_i are eigenvectors of covariant matrix $C = AA^T$.
Following this analysis, $L = A^TA$, a MxM matrix, is constructed, where

$$L_{mn} = \varphi_m^T \varphi_n \qquad (24)$$

and x_i – M eigenvectors of matrix L are calculated.
These vectors determine linear combinations of the M training set face images to form the eigenfaces v_l.

$$v_l = \sum_{r=1}^{M} x_{lr} \varphi_r \qquad \mathrm{r} = 1, \ldots, \mathrm{M} \qquad (25)$$

At this point the calculations are reduced from the number of pixel in images – N^2 to the number of images in the training set – M (in practice $M << N$).

3.2. Classifying face images using eigenfaces

It was proved that, in practice, a small number of eigenfaces is sufficient for identification – those vectors with largest associated eigenvalues.

The process of classification of a new (unknown) face Γ_{new} to one of the classes (faces) proceeds in two steps.

Let be M' the significant eigenvectors of the L matrix. The first step is to transform the new face image into its

ISSN 2067 – 4074

eigenface components, to project the new face image into "face space". The resulting weights form the weight vector Ω_{new}^T

$$\omega_i = x_i^T(\Gamma_{new} - \psi) \qquad i = 1...M' \tag{26}$$

and

$$\Omega_{new}^T = [\omega_1\ \omega_2\ ...\ \omega_{M'}] \tag{27}$$

Ω_{new}^T describes the contribution of each eigenface in representing the input image, as in figure 9.

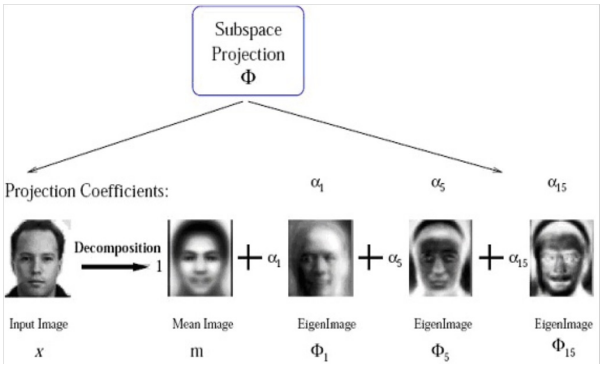


Figure 9. Face image projection into face space

The easiest way to determine which face class /cluster provides the best description of an input face image is to find that class which mini-mizes the Euclidian distance. The Euclidian distance allows to construct cluster of faces such that similar faces are assigned to one cluster.

3.3. The Euclidian distance

The Euclidian distance between a new face image, Γ_{new} and a *k*th face class is

$$\xi_k = \sqrt{\| \Omega - \Omega_{new}^T \|} \tag{28}$$

A face is classified as belonging to face class *k*th the minimum ξ_k is below a chosen threshold θ.

There can be four possible cases for a new face image and its eigenvector:

a) it can be near face space and near a face class - the subject is recognized and identified;
b) it can be near face space but not near a known face class - the subject is an unknown individual;
c) it can be distant from face space and near a face class - it is not a face image;
d) it can be distant from face space and distant from face class - it is not a face image.

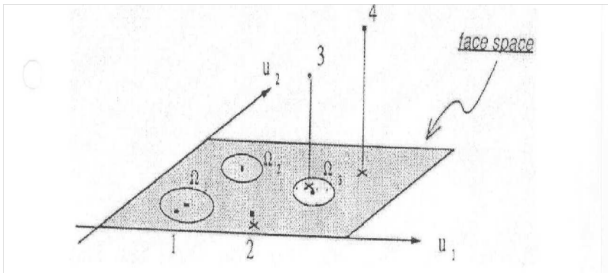


Figure 10. Projection of 4 new face images in a face space with three known individuals [12].

3.4. Face reconstruction using eigenfaces

Using its features vector and the eigenfaces obtained from the training set of images an image can be approximately rebuilt.

$$\Gamma' = \Psi + \Phi_f \tag{29}$$

where

$$\Phi_i = \sum_{i=1}^{M'} w_i u_i \tag{30}$$

is the projected image.

From equation 29 we can consider that the input face is rebuilt by adding each eigenface with a contribution (weight) w_i in equation 30, to the average of training set images. The degree of the fit (or 'rebuild error ratio') can be expressed by means of the Euclidian distance between the original and the

reconstructed face image as given in equation 31.

$$\text{Rebuild error ratio} = \frac{\| \Gamma' - \Gamma \|}{\| \Gamma \|} \quad (31)$$

It has been observed that, rebuild error ratio increases as the training set members differ heavily from each other[13]. This is due to the addition of the average face image. When the members differ from each other (especially in image background) the average face image becomes messier and this increases the rebuild error ratio. There are four possibilities for an input image and its pattern vector:

1. Near face space and near a face class,
2. Near face space but not near a known face class,
3. Distant from face space and near a face class,
4. Distant from face space and not near a known face class.

In the first case, an individual is recognized and identified. In the second case, an unknown individual is presented. The last two cases indicate that the image is not a face image. Case three typically shows up as a false classification. It is possible to avoid this false classification in this system by using eq. (31) as

$$\frac{\| \Phi - \Phi_f \|}{\| \Phi_f \|} \leq \Phi_k \quad (32)$$

where Φ_k is a user defined threshold for the face presence in the input face image belonging to k-th face class.

4. A comparative study of PCA, LDA and ICA

Experiments were done by Ö. Toygar and A. Acan[14] using PCA, LDA and ICA on color face images. The Libor Spacek's Collection of Facial Images [IMG2] was used and includes 7900 color images of faces of 395 individuals. Each subject has 20 image samples in the database. In their first experiment the three methods were tested against the training face images. The experiment was done with 25 and 50 training images separately to see the effect of using different number of training images. The results demonstrating the recognition performance of the three techniques are presented in the table 1. The results show that whenever the number of training images is increased, the thresholds must be increased to obtain better results.

Table 1. Performace results of PCA, LDA and ICA [14]

Number of Training Faces	Method	Threshold	Success rate (Percentage)
25 (5x5)	PCA	3000	**93**
	LDA	500	85
	ICA	60	36
50 (10x5)	PCA	3000	79
		4000	89
		5000	**93**
	LDA	500	45.5
		800	67
		1000	79.5
		1200	**85**
	ICA	60	32
		100	32
		500	**32**

Different experiments were done with partial occlusion of faces. Partial occlusion of original faces was obtained for six different cases. The training face images were cut from left, right, up and down parts and six different databases were formed with these training images. The six different partial occlusions applied are shown in fig.11

Original image

I

II

III

ISSN 2067 – 4074

Figure 11. Face images of six partial oclussion cases [14]

Case I: The left part of the image is cut vertically until the left ear of the face is lost.
Case II: The upper part of the image is cut horizontally until the hairs are lost, but the forehead will be included.
Case III: The bottom part of the image is cut horizontally until the bottom edge of the chin is reached.
Case IV: The right part of the image is cut vertically until the right edge of the right eye is reached.
Case V: The upper part of the image is cut horizontally until the upper part of the eyebrows is reached.
Case VI: The bottom part of the image is cut horizontally until the bottom edge of the lip is reached.
These tests were done with 25 and 50 images for each case using PCA, LDA and ICA. Their results are presented in table 2.

Table 2. Recognition results on images with partial occlusion [14]

Method	No. of training faces	Case I %	Case II %	Case III %	Case IV %	Case V %	Case VI %
PCA	25	96	60	100	100	32	100
	50	33	37	90	58	21	71
LDA	25	0	0	0	20	0	0
	50	4	1	7	0	1	1
ICA	25	28	24	4	20	20	20
	50	10	20	2	10	20	10

4. Practical implementation

Figure 12. Real-time face detection

The proposed system simulates an airport or a frontier surveillance system. It is supposed that 2 actors will interact with the system. The subjects will be enrolled into database by a police officer, which can use as input data the information from a car licenses or identity cards database.

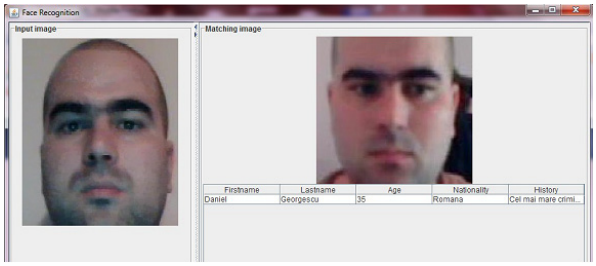


Fig. 13 System feedback

On the other part, a surveillance agent will interact with the system through the detection/recognition modules.

5. Conclusions

The proposed approach is robust, simple, and easy and fast to implement compared to other algorithms. It provides a practical solution to the recognition problem. We are currently

investigating in more detail the issues of robustness to changes in head size and orientation. The main focus is to migrate the processing to a distributed system and implement a neural network for each subject. These are required by the large number of subjects in a real face recognition system and by the expected fast speed to recognize the subjects.

References

[1] D. Maltoni, D. Maio, A.K. Jain, and S. Prabhakar, Handbook of Fingerprint recognition. Springer Verlag, New York, USA, 2003.

[2] A. K. Jain, A. Ross, and S. Prabhakar, *An Introduction to Biometric Recognition*. IEEE Transactions on circuits and systems for video technology, Vol.14 No.1, January 2004.

[3] D. M. Blackburn. Biometrics 101, version 3.1. Federal Bureau of Investigation, March 2004

[4] J. F. Canny *A Computational Approach to Edge Detection*, IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. Pami-8, No. 6, November 1986.

[5] P. J. Phillips, P. Grother, R. J. Michaels, D. Blackburn, E. Tabassi, and M. Bone. *Face Recognition Vendor Test 2002: Evaluation Report*. Online: http://www.frvt.org, March 2003.

[6] M.-H. Yang, D. J. Kriegman, and N. Ahuja, *Detecting faces in images: A survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol. 24, No.1, pp. 34–58, 2002.

[7] Stan Z. Li, Anil K. Jain, Handbook of Face Recognition, Springer Science+Business Media, LLC, New York, USA, 2005.

[8] P. Viola, M. Jones, *Rapid object detection using a boosted cascade of simple features*, IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Hawaii, 2001.

[9] P. Viola, M. Jones, *Robust real-time object detection*, Second International Workshop on Statistical And Computational Theories Of Vision – Modelling, Learning Computing, And Sampling, Vancouver, 2001.

[10] C. P. Papageorgiou, M. Oren, T. Poggio, *A General Framework for Object Detection*, International Conference on Computer Vision (ICCV), pp. 555-562, India, 1998.

[11] M. A. Turk, A. P. Pentland, *Face recognition using eigenfaces*, IEEE Conference on Computer Vision and Pattern Recognition - CVPR, pp 586-591, 1991.

[12] M. A. Turk and A. P. Pentland, *Eigenfaces for recognition*, Journal of Cognitive Neuroscience, Vol. 3, No. 1, pp. 71-86, Massachusetts Institute of Technology, 1991.

[13] I. Atalay, Face recognition using eigenfaces, M. Sc. Thesis, Technical University Institute of Science and Technology, pp -58-59, Istanbul, 1996

[14] Ö. Toygar, A. Acan, *Face Recognition Using PCA, LDA And ICA Approaches On Colored Images*, Journal Of Electrical & Electronics Engineering, Vol.3, No.1, pp. 735-743, Turkey, 2003.

Face images databases

[IMG1] The Yale Face Database. Online: http://cvc.yale.edu/projects/yalefaces/yalefaces.html

http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html

[IMG2] Spacek L., "Description of Libor Spacek's Collection of Facial Images", 1996, Online:

http://cswww.essex.ac.uk/mv/allfaces/index.html

[IMG3] Lena

http://www.ee.columbia.edu/~sfchang/course/dip/images/lena.jpg