Face recognition using eigenfaces

Muhammet Karakütük

January 21, 2014

Contents

1 Introduction

Naturally it is the face, which plays the most important role in the recognition of person. Whether we encounter a face on a photo, on a video or in reality: our brain has the ability to classify a face as known or unknown in split seconds. even blurred, rotated or scaled photos usually offer no challenge to our brain. Also the recognition performance on the distance is remarkably. So it would be quite useful to automate or computerize face recognition. So we are looking for a suitable technical procedures, which are able to look on images or videos for persons. From this it follows that for a practical detection method exists two requirements: on the one hand it should have the ability to detect faces in images and video (is there a face in an image? where is a face located in an image?), on the other hand it should have the ability to identify faces (is it a known face? who is it?).
However it is difficult to find a computer-based method, because faces are a natural class of things and so they are complex. after all, faces are multidimensional and they look is constantly changing with the mimic.threfore many detection methods limit the recognition to a well-founded two-dimensional form (photo) of faces. well-founded means that only frontal face without disturbing background is shown on the photo.
but also the recognition performance with deviations from this form (by rotation, skallierung, downsampling, etc.) is specifically studied and are important for information about the robustness of one method. another problem, which each face detection must process is the efficiency. To obtain profitable results sufficient information about a face must be collected in order to distinguis it from all other faces. That is one of the greatest challenges to the systems.

2 The general aspects of eigenface method

A. Turk and Alex P. Pentland (Face Recognition Using Eigenfaces, 1991) developed a method to recognize faces based on the method Karhunen-Loeve-Transformation (KLT), also called Principal-Component-Analysis (PCA). They were encouraged by Sirovich and Kirby, who used PCA to repräsent extremely efficiently images of faces.
The difference compared to many other approaches is that in the identification of a face by eigenfaces is not aimed to measure and compare the striking face features such as eyes, nose, lips or face shape as well as their relative position to each other. In contrast the eigenface method look at faces in its entirety and tries to encode the relevant information efficiently as possible in order to compare them with other equally encoded data.
The idea of this method is to capture the variation in a group of pictures - regardless of their individual characteristics. With their help it is able to encode and compare individual pictures. So mathematically expressed one wants to determine the principal components of a face distribution, or the eigenvectors of the covariance matrix of a range of faces. This range of faces is called "training set". They corresponds to the group of known faces, so called "training faces". Then the eigenvectors can be imagine as the distinctive features that characterize the variation between the faces. That means, that each face can be represent as a weighted linear combination of these eigenvectors.
Since we consider the images of faces as high-dimensional vectors and thus the eigenvectors reversed can be represent as images that have similarity with (at least ghostly) faces, these vectors are called eigenfaces.

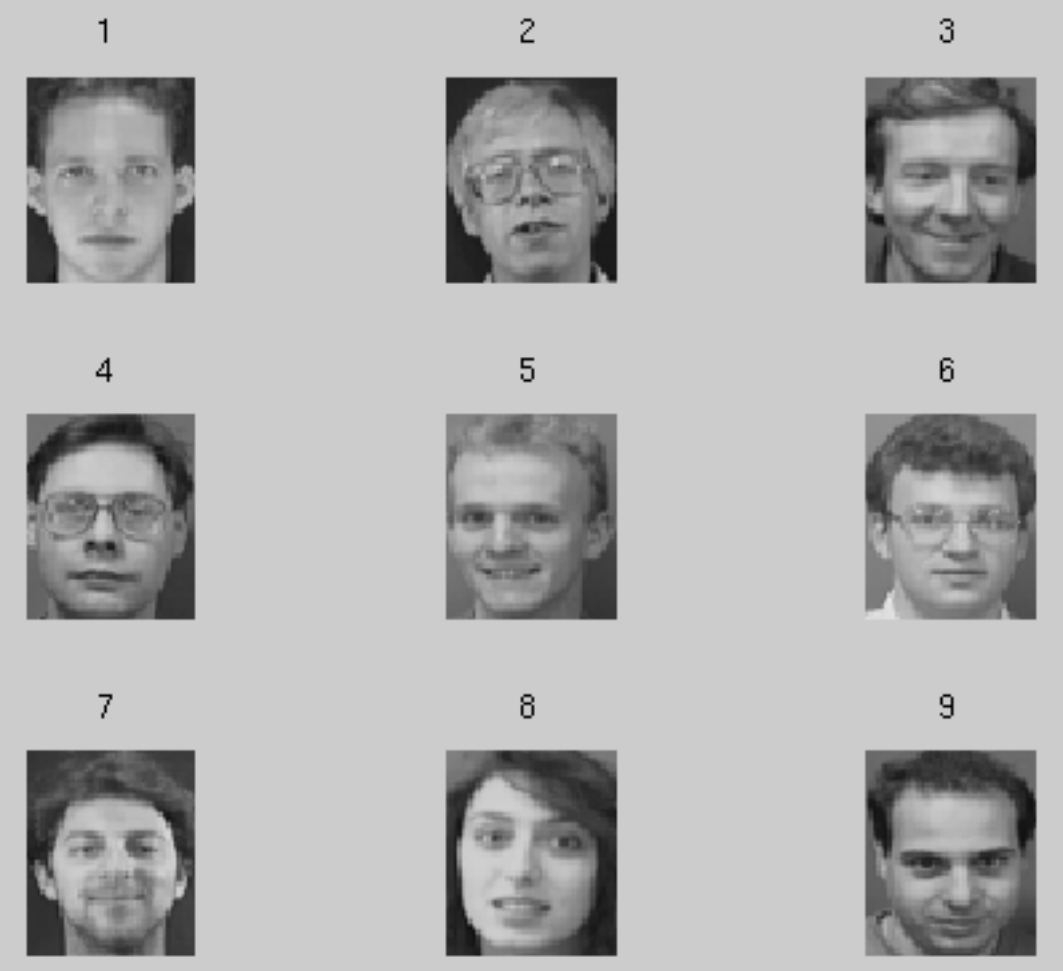

Figure 1: Face images from training set

3 Identification process

3.1 Preprocessing

Briefly mentioned above, we want to represent images as vectors: given is an image of height H and width W (both in pixels). Either this image can be described by the intensity pixel values of its H x W - field, or as a vector with the dimension of the number of pixels (H·W), where each pixel represents a component of the vector (see Figure 2).
An image of a standard size of 256 x 256 is then described as a vector of the dimension 65 536. Because on the whole the images of faces resembles each other, the result of the representation of many faces as points in a high-

dimensional space is: they are not randomly distributed throughout the space (the points form a cloud).
It is important to find the vectors that are most meaningful with respect to the distribution of face images in the space. These vectors are exactly the same eigenvectors described above with respect to the covariance matrix of the training set: the eigenfaces.

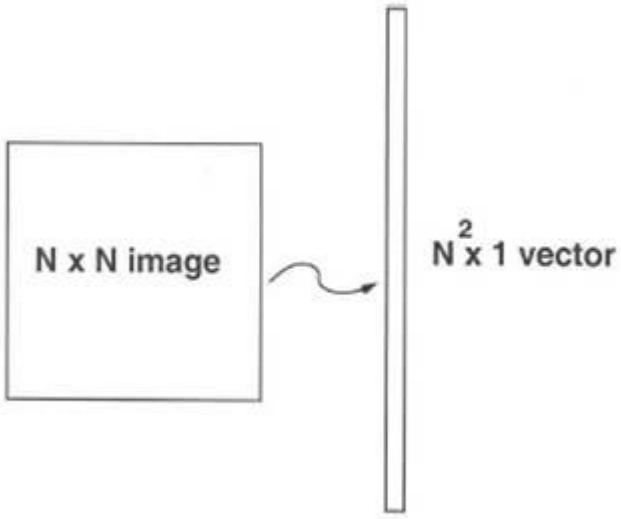

Figure 2: Vector representation of face images

3.2 Computation of eigenfaces

The training set consists of M face images $\Gamma_1, \Gamma_2, \cdots, \Gamma_M$. First the average face image ψ is generated from these M face images:

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i.$$

Figure 3: Average face image

From the average face image Ψ and the training set the difference images Φ_i are calculated:

$$\Phi_i = \Gamma_i - \Psi.$$

Now from these difference images the covariance matrix C is generated to find the M orthogonal eigenvectors (principal components):

$$\mathrm{C} = \Phi \cdot \Phi^T, \qquad \text{where } \Phi = [\Phi_1 \Phi_2 \cdots \Phi_M].$$

The covariance matrix C is a symmetric (H·W) x (H·W) - matrix, for which reason the calculation of the H·W eigenvectors is difficult for a typically image size (256 x 256). Even new computers are not efficient, if they try to calculate the (65 536 x 65 536) - covariance matrix and its 65 536 eigenvectors. For this reason it is necessary to find a performable method to calculate the eigenvectors: This is achieved by solving a much smaller problem, the reduction of the (H·W) x (H·W) - matrix to an M x M matrix (M is the size of the training set, M $\ll$ H·W). So the eigenvectors of C' $= \Phi^T \cdot \Phi$ instead of C $= \Phi \cdot \Phi^T$ can be calculated.

Proof: According to the definition for the eigenvectors e_i of the matrix C' and its eigenvalues λ:

$$\begin{aligned}
& C' \cdot e_i = \lambda \cdot e_i && | \quad C' = \Phi^T \cdot \Phi \\
\Leftrightarrow \quad & \Phi^T \cdot \Phi \cdot e_i = \lambda \cdot e_i && | \quad \text{multiply from the left with } \Phi \\
\Leftrightarrow \quad & \Phi \cdot \Phi^T \cdot \Phi \cdot e_i = \lambda \cdot \Phi \cdot e_i && | \quad C = \Phi \cdot \Phi^T \\
\Leftrightarrow \quad & C \cdot \Phi \cdot e_i = \lambda \cdot \Phi \cdot e_i && | \quad \text{define } v_i = \Phi \cdot e_i \\
\Leftrightarrow \quad & C \cdot v_i = \lambda \cdot v_i
\end{aligned}$$

Conclusion: The eigenvalues λ of the covariance matrix C' are also the eigenvalues of the covariance matrix C. The eigenvectors v_i of C are obtained by multiplication of the eigenvectors e_i of C' with the matrix Φ: $v_i = \Phi \cdot e_i$.

Note:

1. With this method M eigenvectors of C are obtained. But the remaining eigenvectors can be neglected.

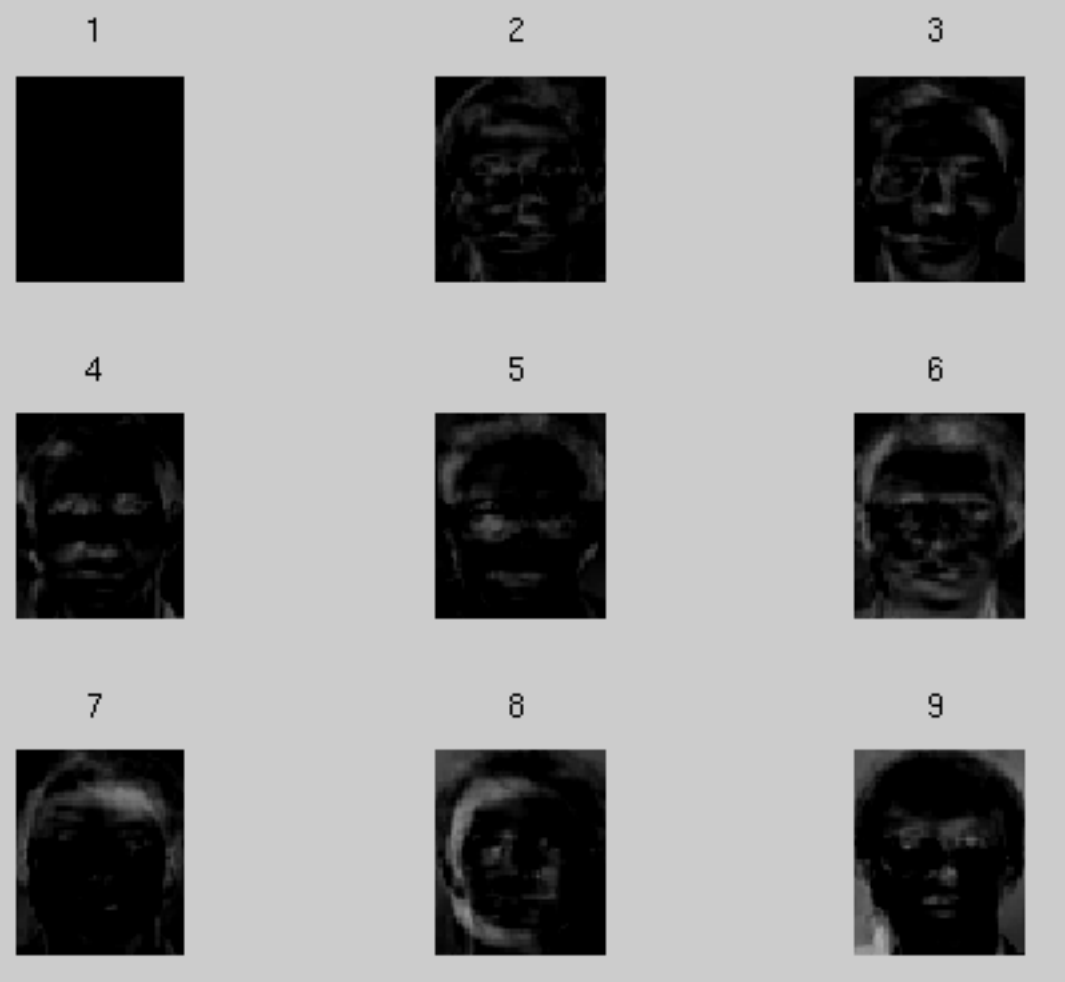

Figure 4: Eigenfaces of each image in the training set

3.3 Classification of a face image

As mentioned above the goal is to represent a face as a linear combination of eigenfaces. This means trying to reconstruct the input image using linear

combination of eigenfaces or at least to approximate it. In other words we search a coefficient w_i, which minimizes the following expression:

$$\|(\Gamma \cdot \Psi) - \sum_{i=1}^{M} w_i \cdot v_i\|,$$

where Γ is the input image, Ψ is the average image, v_i are the eigenfaces and $\|\|$ describes an euclidean norm (e.g. distance, inverse euclidean distance). These coefficient can be calculated by:

$$w_i = v_i(\Gamma \cdot \Psi), \qquad i \in \{1, 2, .., M\}$$

The constructed vector $W^T = [w_1 w_2 \cdots w_M]$ symbolizes the weighting of each eigenface representing the input image. Therefore w_i is called the weights of the face. By comparing these weights with the known faces (test faces) an analyze can be made wether the face is known or unknown. For this purpose the detection of the face, which describes the input image best, is necessary. The simplest method to do this is to find the class k, which minimizes the euclidean distance $\delta_k = \|W - W_k\|$ (W_k describes a vector, which represents the k-th face class). A face is classified into a face class k, if the distance is smaller than a defined value ε, so called threshold. This value corresponds to the highest tolerable deviation from the original face. Otherwise the input face image is classified as unknown.
The idea of the use of the weights of images for their comparison is based on the findings of Sirovich and Kirby. They had demonstrated that a series of images can be reconstructed quite well by their individual weights and a small quantity from standard images.

4 Experimental result

An implementation of face recognition using eigenfaces was implemented in Matlab (see section 5). A dataset of 400 images are available. Each image is in grey level and has dimension of 56x46. There are 40 subject in the dataset, where each of them gives 10 images, with frontal view with different gesture and face orientation (see Fig. 1).
Ideally the process should classify all faces (which are in the training set) as known and also it should classify a face into its correct face class. However for the success rate various factors are important:

1. the choice of threshold
2. number of test images of one person (in the training set)
3. the input image is not a face within the training set

4.1 Threshold

First the influence of the threshold on the result has been tested. For this purpose the training set contains 40 different faces and the threshold is changed at each pass. The result of the pass can be seen in Figure 5. It shows that the threshold has a big influence on the result of the face recognition. If the threshold is too small, a known input image could be classified into an unknown image. On the other hand if the threshold is to big all input images could be classified into a face class as well as images which are not in the training set. The selection of the threshold is important for the result of the system.

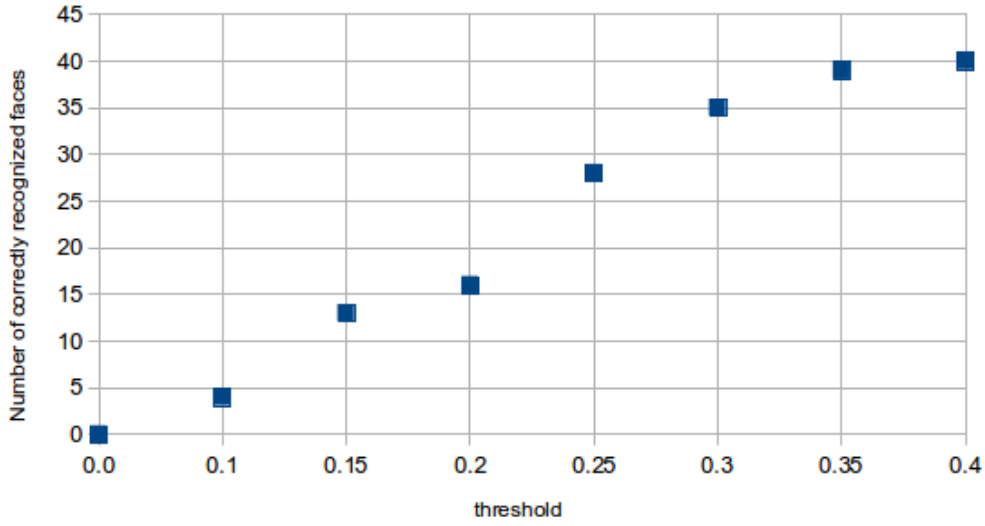

Figure 5:

4.2 Number of test images of one person

Also the number of test images of one person in the training set has a big influence on the result, too. In this test the threshold is 0.26 and the number of test images of one person is changed at each pass. Figure 6 shows the result of the pass. The more images of one person is in the training set the merrier high is the success rate of the system.

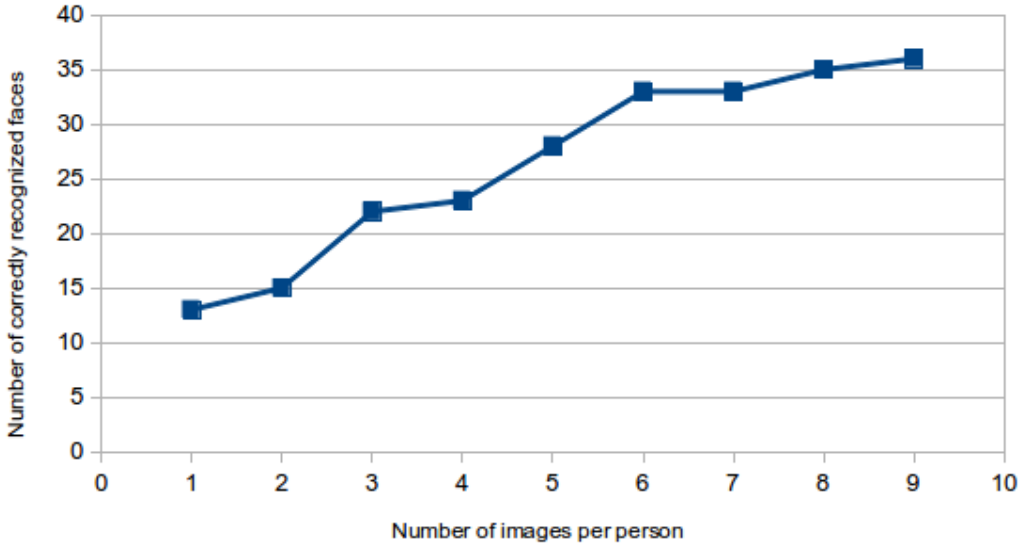

Figure 6:

4.3 Testing an unknown Face

In this test case faces were tested, which are not in the training set. The training set contains 35 x 9 face images, also 315. Therefore 5 face images of different persons are available for testing. The threshold is 0.15.
The result of this test case is, that 4 of the 5 faces are classified correctly into unknown faces. But one face is classified as a known face, hence the failure of the system (having the mentioned properties) is 20%.

All in all the eigenface method is a very workable approach because it is effective and efficient. Also in some sense it is capable of learning because it allows always to add new faces into the training set. For the success rate of the eigenface method is the selection of a capable threshold and the training set very important.

5 Implementation in Matlab

Listing 1: Implementation of Face Recognition using eigenfaces in Matlab

```
% Face recognition using eigenfaces
%
% This script takes M face images from FaceData_56_46 as the training set.
% Then in the first part the eigenfaces of the training set is calculated.
% In the second part a given face (test_image) will be verified with the
% help of the training set. If the person is an authorized person, the
% script will show the recognized face and for an unauthorized person it
% will show an error message.

close all; % close all windows
clear all; % clear all variables
clc;       % clear the console

%FaceData: load all faces from file. the file contains 40 different persons
% where each person has 10 different face images.
load('FaceData_56_46.mat');

M_1 = 9; % number of different persons
M_2 = 6; % number of test images per person
M = M_1 * M_2;   % total number of test images


% take an test image to get the height and weight
TMP_Image = FaceData(M_1,M_2).Image;
[P,Q] = size(TMP_Image);
R = P*Q;

A=zeros(R,M); % training matrix;
A_column_index = 1;
for ii=1:M_1
    for jj=1:M_2
```

```
32          Im = (FaceData(ii,jj).Image);
33
34          % transform matrix into an vector
35          Gamma_i = reshape(Im,R,1);
36
37          A(:,A_column_index) = double(Gamma_i);
38          A_column_index = A_column_index + 1;
39      end
40  end
41
42
43  % computing the average face vector Psi
44  sum = 0;
45  for ii=1:M
46      sum = sum + A(:,ii);
47  end
48  Psi = sum/M;
49
50
51  % subtracing the mean face from each face images in the
        training set:
52  % Phi_i = Gamma_i - Psi
53  Phi = zeros(R,M);
54  for ii=1:M
55      Phi(:,ii) = A(:,ii) - Psi;
56  end
57
58
59  % computing the covariance matrix C:;
60  C = Phi' * Phi;
61
62  [eigenvectors, eigenvalues] = eig(C);
63
64
65  % computing eigenfaces, the eigenvectors of Phi*Phi'
66  eigenfaces = Phi * eigenvectors;
67
68
69
70  weights = zeros(M,M);
71  for jj=1 : M
```

```
72      for ii=1 : M
73          weights(ii,jj)= eigenfaces(:,ii)' * (A(:,jj)-
                Psi);
74      end
75  end
76
77
78  % ------------------------------------------------------------
79  % ------------------------------------------------------------
80  % ------------------- Face Recognition -----------------------
81  % ------------------------------------------------------------
82  % ------------------------------------------------------------
83
84  % input image
85  test_image = FaceData(5,10).Image;
86
87  figure,
88  subplot(3,2,3);
89  imshow(test_image);title('test_face');
90
91  % transform the test_image matrix into a vector
92  v_test_image = double(reshape(test_image,R,1));
93
94  test_weights = eigenfaces' * (v_test_image - Psi);
95
96  % use inverse Euclidean distance
97  similarity_score = arrayfun(@(n) 1 / (1 + norm(weights
        (:,n) - test_weights)), 1:M);
98
99  % find the image with the highest similarity
100 [match_score, match_index] = max(similarity_score);
101
102
103 % decision parameter
104 a=max(similarity_score) * 0.25;
105 b=min(similarity_score);
106 if(b>a)
107     msgbox('UNKNOWN_FACE','error','error');
108     subplot(3,2,4);
109     imshow(zeros(P,Q));title('unknown_face');
110 else
```

```
111     % display the result
112     subplot(3,2,4);
113     imshow(uint8(reshape(A(:,match_index), P,Q)));title
            ('recognized face');
114 end
```

6 References

- M. Turk and A. Pentland, "Face recognition using eigenfaces", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pages 586-591, 1991.

- V. Perlibakas, "Distance Measures for PCA-based Face Recognition", Pattern Recognition Letters, Vol. 25, No. 6, pp. 711 – 724, April 2004

- A.J. O'Toole, H. Abdi, K.A. Deffenbacher, and D. Valentin, "A low-dimension representation of faces in the higher dimensions of the space", Journal of the Optical Society of America A, Vol. 10, pp. 405-411, 1993.

- Mrs. Sunita Roy and Prof. Samir K. Bandyopadhyay, "Face recognition using Eigen face based technique utilizing the concept of principal component analysis"