UNIVERSITE DE BOURGOGNE

FACE RECOGNITION "using PCA"

APPLIED MATHEMATICS PROJECT

VIBOT-10

1/2/2016

Submitted To:-

Dr. Désiré Sidibé

Submitted By:-

Raabid Hussain
Gourab Ghosh Roy

FACE RECOGNITION

Objective:-

The objective of this project was to implement a PCA based face-recognition technique on a given data-set. We were given a set of 320 x 240 images of different people. The main tasks to be performed on these images were as follows:-

- Image Normalization
- Build Train and Test databases
- Implement SVD & PCA
- Face Recognition
- Gender Classification

Introduction :-

Face recognition has been one of the main corner-stones of computer vision research throughout the previous decade. Up till now, most of the techniques have been furnished and widely accepted by everyone. One of these approaches involves PCA based classification of a given data-set, the result of which is then compared to a given database to recognize and differentiate between different faces. The matching involves comparison of various prominent features in the images. We implemented this technique followed by a k-nearest neighbor approach to perform gender-classification.

Methodology:-

We were provided with a set (5 images per person) of images of all the students in our class. These images were of size 320 x 240 pixels. Another set of text files corresponding to each image were provided containing locations of five distinct features of the face (left-eye, right-eye, nose, left-side of mouth and the right-side of mouth).

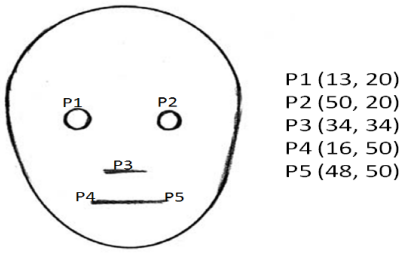

Figure 1: Features of the face

In normal face recognition software, the first task is to recognize that a face is present in the image. Thanks to the professor, we were given only the images that contained the face, so that we could skip this step.

The next step was to normalize the images. Since, every image was different from the other i.e. the location of the face on the images was not coherent among all the given images, this step was crucial in order to get sufficiently acceptable results. The key here is to find parameters for an affine transformation of the images in order to map them to pre-determined locations on a fixed window-size which in our case was chosen to be a window-size of 64 x 64 pixels. In order to accomplish this, a vector F was used to store the average location of each of the five facial features provided of all the images. F was initialized with the features of the first image. The best transformation was computed using SVD that aligned F with pre-determined positions, F^p (as mentioned in figure 1) on the smaller window. The transformation was defined by six parameters:-

$$F^p = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} F + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Since, we were given five features of each image; we had 5 similar equations for each face. This led to a system of 6 unknowns and 10 equations: a simple case of SVD transformation analysis. Once the transformation parameters are obtained, we applied the transformation on F to get F' (our normalized image). For every image the above process was repeated. The transformation parameters were calculated using an iterative process until the difference between F and F' was less than a threshold (in our case: norm(F-F') <= 2). The system converged in seven iterations on average. In order to avoid gaps/missed locations on the normalized images, we iterated the process through every pixel on the output image to find the corresponding point on the given/input image and then copied it over in the output image. This yielded far better normalized images. The normalized images were saved in a separate directory through the *'face_normalization.m'* file.

The next steps involved the use of the other mat file titled *'face_recogniction.m'*. The normalized images were dived into test images and train images. The norm that was followed was to put a front, and two side images of everyone in the train data set and the other two images were kept in the test data set. These were stored in different directories.

The next step is of PCA based recognition. In this an image I is treated as a vector X. This vector is obtained by concatenating the rows of the image I. Our 64 x 64 pixel images resulted in a 1 x 4096 vector. This showed signs of high co-relation between pixels, so we had the opportunity to reduce the high dimension of the vector to a smaller number of variables. We had a set of p training images, so we wrote our entire system of training data set as a pxd matrix D, every row of which corresponds to a single image:

$$D = \begin{bmatrix} I_1(1,1) & I_1(1,2) & \dots & I_1(1,N) & \dots & I_1(M,1) & I_1(M,2) & \dots & I_1(M,N) \\ I_2(1,1) & I_2(1,2) & \dots & I_2(1,N) & \dots & I_2(M,1) & I_2(M,2) & \dots & I_2(M,N) \\ & & \vdots & & & & \vdots & & \\ I_p(1,1) & I_p(1,2) & \dots & I_p(1,N) & \dots & I_p(M,1) & I_p(M,2) & \dots & I_p(M,N) \end{bmatrix}_{p \times d}$$

This gave us a matrix D of size dxd, since we had d variables. In order to compute PCA, we computed the co-variance matrix (after subtracting its mean) of D using:-

$$E = \frac{DD^T}{p-1}$$

This gave us a matrix E of size p by p, since we have p images. We then computed eigen-values and eigen-vectors of this matrix E. We multiplied by D' to get the eigen vectors we were looking for originally, serving as a good trick to reduce computational load. To reduce the computational load even more, we only kept the k largest singular values and their corresponding eigen vectors. The k principal components were then put into a dxk projection matrix V. Thus images were represented using PCA by:-

$$v = XV$$

This k-dimensional vector v was used to represent each image, a fair approximation of the original image. We applied this technique on all images to maintain a reduced dimension data set.

So, then we compared each image in the test data set with every image in the training data set to obtain the best 3 matching images. First, the test image I was represented by a corresponding vector X and it is projected onto a PCA based scheme as proposed above. To compute the identity of an image, we computed a similarity index using Euclidean distances between the test image and all the train images. The 3 images with the best similarity index were used to identify the image. Labels of names were assigned to each image in order to identify the person.

In order to quantify the results of our approach, we matched the labels of every test image with its best matched image in the train image data set and the accuracy was calculated using:-

$$Accuracy = (1 - \frac{wrong\ match}{total\ number\ of\ test\ images})*100$$

The next step was to classify the images on a gender basis. For this a nearest neighbor approach was used which involves assigning a gender label to each train image then comparing the gender of all the best three matches to determine the person's gender. In our case, since there were only two females, we decided not to assign separate labels for gender classification. Instead we decided on using the names as a gender label too. (i.e. if the 2/3 best matches contain images named 'PaolaArdon' or 'Armine', then the image is of a girl, otherwise a boy)

In order to show the output of the system, we displayed the best three images for every test image on matlab screen as well as its command window. First the gender of the person is mentioned, followed by the test image and the best three matches in the respective lines. A sample output screen for a test image is as follows:-

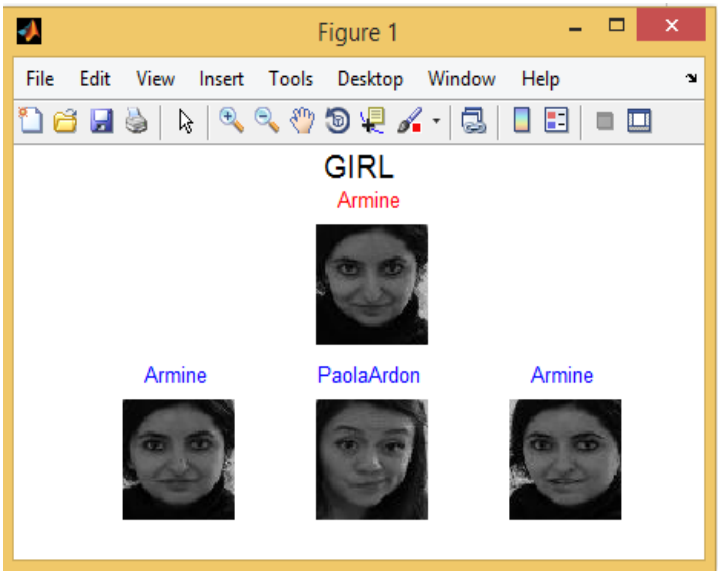

Figure 2: Sample output screen

The results are shown in graphical form in the following image which compares the k principal components with the accuracy of the system:-

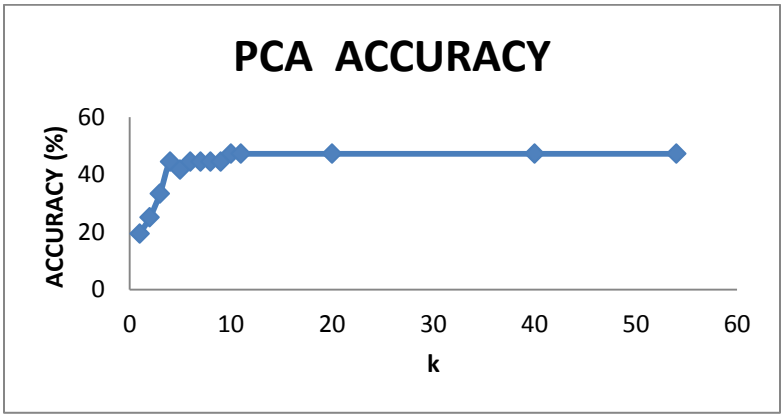

Figure 3: Graph of Accuracy against k

The following table lists the results of gender classification:-

Actual\Experiment	Male	Female
Male	32	0
Female	2	2

Figure 4: Results of gender classification

Since we were given a set of 36 images with four images of girls and thirty-two images of boys, the nearest neighbor technique yielded reasonable results which just a mismatch of 2 images (of the same person though).

Conclusion:-

In this project a PCA based faced recognition system was successfully implemented using linear algebra studied in lectures. The maximum accuracy reached was 47.22%. The classification accuracy increases as we increase the number of principal components up to a maximum, after which it becomes constant and does not change with that number when the variance in the input data needed for this classification task has been captured. This accuracy can be sought to be highly dependent on the nature of input images and their division between test and train images. All in all, it was fun to learn a cool computer vision application using only linear algebra.

Annex:-

Our code is dived into two steps. The normalized images are contained in the normalized images folder. These were obtained after running the face_normalization.m code. These images are then used by the face_recognition.m code to perform PCA and split into test and train datasets. In order to run the code, the face_recognition.m file can be alone run to check the results.

The content of the two files is as follows:-

face_normalization.m

```
%Warning: in order to run this file, first set the exact path of the
%directory of un-normalized images in line 8 of this code

% SVD for Images
clc
clear all
%change the path to where the images are on your computer

directory =  'C:\\all_faces\';
%reading all the images
filenames = dir(strcat(directory,'*.jpg'));
filenames = {filenames.name};
len = length(filenames);
MaxIter = 20;
s = zeros(6,len);
filename = char(filenames(1));
filename = strrep(filename,'.jpg','.txt');
filename = strrep(filename,'.JPG','.txt');
[x, y] = textread(strcat(directory,filename),'%u %u', 'delimiter', '\n');
F = [];
for j = 1:5
    F = [F; x(j); y(j)];
end
%finding transformation indices
b = [13 20 50 20 34 34 16 50 48 50]';
A = [];
for j = 1:5
    A = [A; F(2*j-1) F(2*j) 0 0 1 0; 0 0 F(2*j-1) F(2*j) 0 1];
end
sprime = pinv(A)*b;
Fnew = [];
for j = 1:5
    temp = [sprime(1) sprime(2); sprime(3) sprime(4)]*[F(2*j-1);F(2*j)] +
[sprime(5); sprime(6)];
    Fnew = [Fnew; temp];
end
F = Fnew;
pA = zeros(6,10,len);
%normalizing
for iter=1:MaxIter
    Fold = F;
```

```
    Ftotal = zeros(10,1);
    for i=1:len
        filename = char(filenames(i));
        filename = strrep(filename,'.jpg','.txt');
        filename = strrep(filename,'.JPG','.txt');
        [x, y] = textread(strcat(directory,filename),'%u %u', 'delimiter',
'\n');
        Fi = [];
        for j = 1:5
            Fi = [Fi; x(j); y(j)];
        end
        b = F;
        if (iter == 1)
            A = [];
            for j = 1:5
                A = [A; Fi(2*j-1) Fi(2*j) 0 0 1 0; 0 0 Fi(2*j-1) Fi(2*j) 0
1];
            end
            pA(:,:,i) = pinv(A);
        end
        s(:,i) = pA(:,:,i)*b;
        Fnew = [];
        for j = 1:5
            temp = [s(1,i) s(2,i); s(3,i) s(4,i)]*[Fi(2*j-1);Fi(2*j)] +
[s(5,i); s(6,i)];
            Fnew = [Fnew; temp];
        end
        Ftotal = Ftotal + Fnew;
    end
    F = Ftotal/len;
    if (norm(F-Fold) <= 2)
        break;
    end
end

disp('SVD Normalization Converged in Iteration :')
disp(iter)
%splitting into test and train images
out_directory = 'C:\\normalized_faces\';
for i=1:len
    filename = char(filenames(i));
    I = imread(strcat(directory,filename));
    I = rgb2gray(I);
    [m, n] = size(I);
    I0 = zeros(64,64);

    for k=1:64
        for j=1:64
            f = round(([s(1,i) s(2,i); s(3,i) s(4,i)])\([j;k] - [s(5,i);
s(6,i)]));
            if ((f(1) > 0) && (f(1) <= n) && (f(2) > 0) && (f(2) <= m))
                I0(k,j)=I(f(2),f(1));
            end
        end
    end
```

```
    I0 = uint8(I0);
    imwrite(I0,strcat(out_directory,filename),'jpg');
end
```

face_recognition.m

```
% PCA for images

clc
clear all
close all;
%reading all the images
train_dir = 'train_faces\';
test_dir = 'test_faces\';
train_filenames = dir(strcat(train_dir,'*.jpg'));
train_filenames = {train_filenames.name};
test_filenames = dir(strcat(test_dir,'*.jpg'));
test_filenames = {test_filenames.name};
train_length = length(train_filenames);
test_length = length(test_filenames);
train_data = [];
test_data = [];
train_labels = cell(train_length,1);
test_labels = cell(test_length,1);
indextrain = 1;
indextest = 1;
%assigning name labels to each image
for i=1:train_length
    filename = char(train_filenames(i));
    I = imread(strcat(train_dir,filename));
    I = reshape(I, 1, 64*64);
    label = strrep(filename,'.jpg','');
    label = strrep(label,'.JPG','');
    label = strrep(label,'1','');
    label = strrep(label,'2','');
    label = strrep(label,'3','');
    label = strrep(label,'4','');
    label = strrep(label,'5','');
    label = strrep(label,'_','');
    train_labels{indextrain} = label;
    indextrain = indextrain + 1;
    train_data = [train_data; double(I)];
end
for i=1:test_length
    filename = char(test_filenames(i));
    I = imread(strcat(test_dir,filename));
    I = reshape(I, 1, 64*64);
    label = strrep(filename,'.jpg','');
    label = strrep(label,'.JPG','');
    label = strrep(label,'1','');
    label = strrep(label,'2','');
    label = strrep(label,'3','');
```

```
    label = strrep(label,'4','');
    label = strrep(label,'5','');
    label = strrep(label,'_','');
    test_labels{indextest} = label;
    indextest = indextest + 1;
    test_data = [test_data; double(I)];
end


% PCA

Npca = 10;

D = train_data;
[p, d] = size(D);
meanD = mean(D);
D = D - repmat(meanD,p,1);

C=(1/(p-1))*(D*D');
[U,E] = eig(C);
V = D'*U;

[t,k] = sort(diag(E),'descend');
knew = k(1:Npca,:);
Vnew = V(:,knew');
datanew = D*Vnew;

[N,~] = size(test_data);
error = 0;
%finding the best matches
for i = 1:N
    FeatureQ = test_data(i,:) - meanD;
    FeatureQ = FeatureQ*Vnew;
    t = [];
    for j = 1:p
        t = [t; norm(FeatureQ-datanew(j,:))];
    end
    [val,ind] = sort(t);
    disp('Test label : ')
    disp(test_labels(i))
    disp('Top 3 Matched labels : ');
    disp(train_labels(ind(1:3)))
%output screens for all images
    hFig = figure;
    set(hFig, 'Position', [250 250 450 250])
    set(gcf,'color','w');
    subplot(2,3,2);
    imshow(uint8(reshape(test_data(i,:),64,64)));
    title(test_labels(i),'Color','r');
    for dis=2:4
        subplot(2,3,dis+2);
        imshow(uint8(reshape(train_data(ind(dis-1),:),64,64)));
        title(train_labels(ind(dis-1)),'Color','b');
    end
%finding accuracy of the system
```

```
    if (strcmp(train_labels(ind(1)),test_labels(i)) ~= 1)
        error = error + 1;
    end
%gender classification
    girl=0;
    for g=1:3
        if (strcmp(train_labels(ind(g)),'Armine') == 1 ||
strcmp(train_labels(ind(g)),'PaolaArdon') == 1)
            girl=girl+1;
        end
    end
    if girl<=1
        disp(sprintf ( '\t Boy') )
        suptitle('BOY');
    else
        disp(sprintf ( '\t Girl') )
        suptitle('GIRL');
    end
end
%displaying accuracy on matlab screen
acc = (1-error/N)*100;
disp('Accuracy % : ')
disp(acc)
```