C-Lab-3 Report u5752771

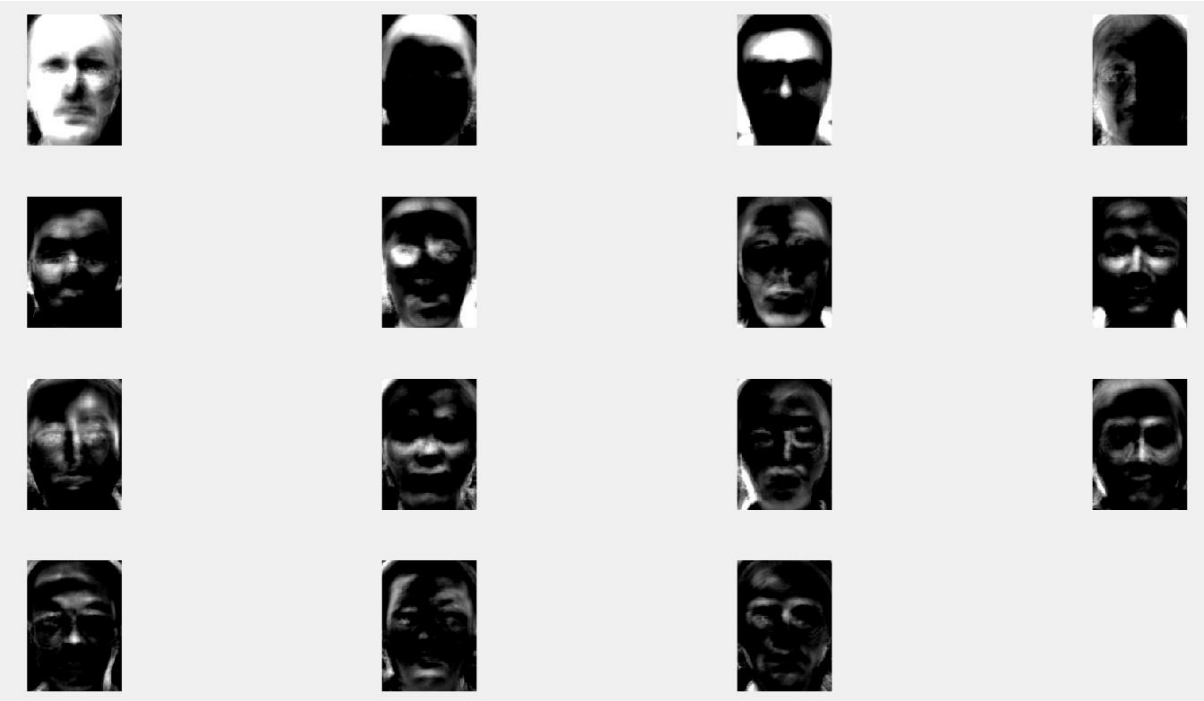
Task: Eigen faces:

This task perform face recognition by using principal component analysis.
First of all, all the image in the input directory are read and reshaped to raw data. After that the mean of raw data are calculated, all the raw data are subtracted from their mean. This subtracted data set are used to perform PCA. The mean face image is shown on the RHS.

Next, since the size of image is [180 130], which is a point in 180*130 dimension. The covariance is (180*130)*(180*130). This size of matrix is very hard for computer to process due to the limitation of computational speed. Thus it is necessary to perform fast Eigen value and vector calculation in order to realize the function.
For fast calculation method, Eigen values and Eigen vectors of gram matrix with size of 135 by 135 are evaluated. This is a much small size matrix. As the calculation below indicated, Eigen vector of A can be represented by $(B^T)^{-1}b$, where b is the eigenvector of the gram matrix. And the corresponding Eigen values can be calculated.

$$
\begin{aligned}
&Av = \lambda v \\
&\Rightarrow BB^T v = \lambda v \\
&\Rightarrow B^T BB^T v = B^T \lambda v \\
&\Rightarrow B^T BB^T v = B^T \lambda v \\
&\Rightarrow (B^T B)(B^T v) = \lambda (B^T v)
\end{aligned}
$$

After that all the Eigen values and Eigen vectors are found, they are sorted in descending order to allow the largest 15 Eigen values and their corresponding Eigen vectors are allocated. By using this selected Eigen vectors and Eigen values. Face from image set can be represented by much lower dimension with size of 15 by projecting the image data obtained above to the lower dimension space. 15 of the Eigen faces are shown below.



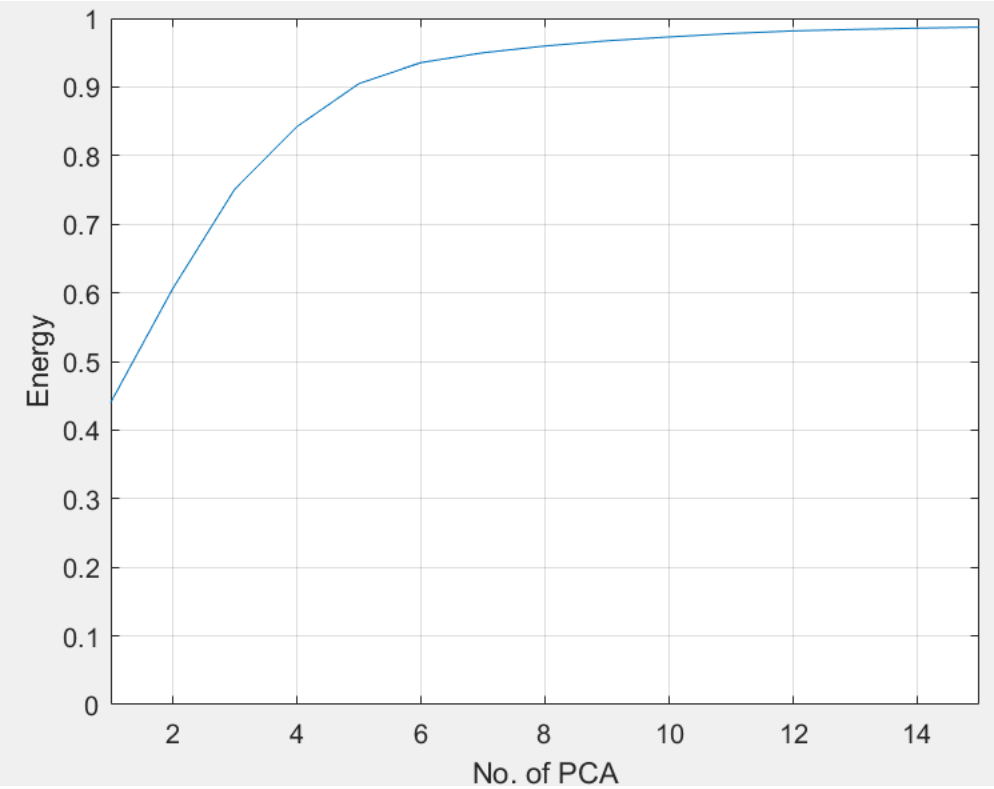
Now, image from testing set can be evaluated. First, a test image are projected to the lower dimension space. By exhaustively comparing the distance between the test image projection and the data projection, the image in training data set with closest distance can be found, which indicated that specific with closest distance is the most similar image in training set to the test image. The most similar three images of my face and all the other testing images are found and shown below.

C-Lab-3 Report u5752771

1. To calculate the energy captured by the first 15 PCA, following calculation is performed.

$$E_s = \langle x(n), x(n) \rangle = \sum_{n=-\infty}^{\infty} |x(n)|^2$$

Energies of each Eigen faces are calculated and sum together serving as normalizer. Each energy of Eigen face are calculated by taking dot product with itself and square afterward. According to the plotting, the first 15 PCA capture almost all the energy in the total number Eigen faces.



2. To reconstruct image with lower dimension representation, the below calculation are performed.

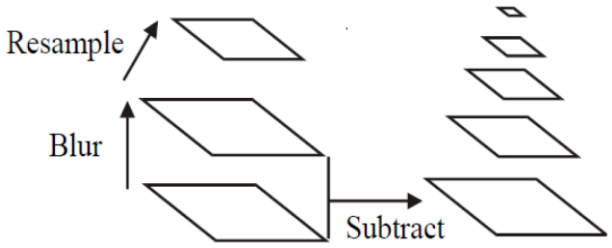
C-Lab-3 Report u5752771

$$\mathbf{x} \to (\underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_1}}_{a_1}, \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_2}}_{a_2}, \ldots, \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_K}}_{a_K})$$

$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K}$$

The results below are obtained by summing reconstructed face in lower dimension space with mean according to the descending order of Eigen value. For the first result, 16 principal components are required to be recognizable. For the second result, only required 12 principal components. This is due to the fact that the first test image is not strictly align with the mean face. In human perspective, is this hard to recognize the face due to the geometric difference of face mismatch. As more Eigen vectors are used in reconstruction, the result is quite reliable (third resulting image).

3. Face detection using Eigen face
   (1) First of all, built scale-space pyramid of the test image using difference of Gaussian pyramid



   (2) Perform normal correlation using the mean face and the pyramid, find the bright spots in the resulting images according to some threshold. At each scale, take a screen shot at that scaled test image. If everything works well, a set of images will be sliced out, some of them are faces, and some of them are not.
   (3) Find the dominant orientation in those images.
   (4) Rotate all training faces within some range (such as -35, -25, -10, 10, 25 and 35, totally 6 sets of training data). Obtain projection data from training faces using PCA algorithm.
   (5) Select another set of testing images contain both faces and non-faces to evaluation the distance, to obtain a threshold. Image within the threshold as considered a face.
   (6) Given the prior knowledge of threshold of faces obtained above. The sliced images can be evaluated. First match the dominant orientation with rotated angle. Find the closest match angle. Project the sliced image to lower dimension. Use the projection data at that angle and the test image projection data to evaluate the distance. If distance is less that threshold, the sliced image is considered as a face. Do the same process for all the sliced image

(7) Mark the faces with a bounding box on the original testing image. Hopefully box with different scale and orientation will be shown on the testing image, which indicate the location of possible faces.

C-Lab-3 Report u5752771

Code:

```
clc;clear all;close all;
%%%%%% data acquisition %%%%%%
input_dir = 'C:\Users\xiang\Dropbox\6528 computer Vision\lab\lab3\training_set_cropped';%input directory
[raw_data num_images] = raw_data_aquisition(input_dir); %raw data aquisition

%%%%%% training %%%%%%
mean_face = mean(raw_data, 2);                          %calculated mean of each rows
image_dims = [180, 130];                                %size of each iamge
data = raw_data - repmat(mean_face, 1, num_images);     %subtract mean from raw data
[evals evtrs]= fast_eigen_calculation(data);            %fast eigen vector and value calculation
[evalues evectors]  = sort_eigen(evals, evtrs);         %alocate the eigen vector according to decending
order of evalue
num_eface = 15;                                         %number of eigen faces to keep
evectors_tun = evectors(:, 1:num_eface);                    %discard the rest of the eigen faces
projection = evectors_tun' * data;                          %project data to those eigen vectors

%%%%%% classification test set %%%%%%
input_dir = 'C:\Users\xiang\Dropbox\6528 computer Vision\lab\lab3\test_set_cropped';%input directory
[test_img num] = raw_data_aquisition(input_dir);        %raw data aquisition
test_data = test_img - repmat(mean_face, 1, num);       %subtract mean face
test_projection = evectors_tun' * (test_data);              %project test image data to the eigen vectors
%%%%%% similar faces display %%%%%%
figure(1);title('test image and three of the most similar faces');
for i = 1:num
    distance = arrayfun(@(n) norm(projection(:,n) - test_projection(:,i)), 1:num_images);%distance
between test data and all the image
    [~, index] = sort(distance);                        %acending order of distance
    similar_faces = [];                                 %similar faces
    for j = 1:3                                         %for the closest distances
        temp = reshape(raw_data(:,index(j)), image_dims);%reshape data into image
        similar_faces = [similar_faces temp];           %cat the similar faces into 1 pic horizontally
    end
    subplot(ceil(sqrt(num)),ceil(sqrt(num)),i);
    imshow([reshape(test_img(:,i), image_dims) similar_faces]);                     %display
end

%%%%%% eigen faces display %%%%%%
figure(2);title('15 faces with largest eiven value');
for i = 1:num_eface
    subplot(ceil(sqrt(num_eface)), ceil(sqrt(num_eface)), i);
    eface = reshape(evectors_tun(:,i), image_dims);
    imshow(eface);
end
%%%%%% mean face display %%%%%%
figure(3);mface = reshape(mean_face, image_dims);
imshow(mface);
```

```
function [raw_data num_images] = raw_data_aquisition(input_dir)
```

C-Lab-3 Report u5752771

```
filenames = dir(fullfile(input_dir, '*.jpg'));          %read all the jpg image
num_images = numel(filenames);                          %number of image received
raw_data = [];                                          %array to hold pix after reshape
for i = 1:num_images
    filename = fullfile(input_dir, filenames(i).name);  %read the filename of n'th image in the set
    img = imread(filename);                             %read the image in matlab workspace
    img = im2double(img);                               %convert image to double class
    img = rgb2gray(img);                                %convert image to grayscale
    raw_data(:, i) = img(:);                            %reshape image and store as raw data,row by row
end
end
```

```
function [evalues evectors]= fast_eigen_calculation(data)
B = data';                                              %transpose of data
covarianceB = B*B';                                     %much smaller size of covariance
[evectorsB evaluesB]=eig(covarianceB);                  %eigenvector and eigenvalue of the covariance
matrix
evectors = data*evectorsB;                              %fast way to calcualted the eigenvalue of the
much larger matrix
temp = pinv(evectors)*data;                             %use inverse eigen vector to calculate the eigen
value
temp = temp*data';
evalues = temp*evectors;
```

```
function [value temp] = sort_eigen(evalues, evectors)
evtr = evectors;                                        %matrix to store sorted eigen vector
eval = [];                                              %matrix to store sorted eigen value
for i=1:size(evectors,2)                                %for each eigen vector
        eval=[eval evalues(i,i)];                       %rearrange the eigen vector in the form of array
end
[value index]=sort(eval);                               %ascending order
index = fliplr(index);                                  %decending order of eigen value
value = fliplr(value);                                  %cooresponding index
len=length(index);                                      %lenth of index
temp=zeros(size(evtr));                                 %to sort the eiven vectors
for i=1:len
    temp(:,index(i))=evtr(:,i);                         %alocate the eigen vector according to decending
order of evalue
end
```

```
%%%%%% energy captured by the first 15 eigen values %%%%%%
clc,clear, close all;
load('clab3_data');
total_energy = sum(arrayfun(@(n) ((evectors(:,n)')*evectors(:,n))^2, 1:size(evectors,2)));%calculate
total energy
nrd_energy =  (arrayfun(@(n) ((evectors(:,n)')*evectors(:,n))^2, 1:size(evectors,2)))/
total_energy;  %normalized one each energy of eigen face
figure, plot(cumsum(nrd_energy));                       %plot the cumilative function
xlabel('No. of PCA'),ylabel('Energy');xlim([1 15]), ylim([0 1]), grid on;
```

```
%%%%%%% eface stacking  %%%%%%%%
clc,clear, close all;
load('clab3_data');                                     %load clab3 stored data
num_test = 20;                                          %unmber of PCA concerned
reconstructed_faces = [];
a = [];
b = [];
reconstructed_face = zeros(image_dims);
mean = reshape(mean_face, image_dims);
sum_face = zeros(image_dims);
projection_test = zeros(size(data,1),1);
data_num = 8;                                           %reconstructed the 8th image in the training set
test_image = reshape(raw_data(:,data_num), image_dims);
for i = 2:2:num_test
    evector_test = evectors(:, num_test);
    projection_test = normc(data(:,data_num) * evector_test' *evector_test);%one reconstruction
```

C-Lab-3 Report u5752771

```
    sum_face = sum_face + reshape(projection_test, image_dims);%sum with all the perivious reconstruction
    a =  [a mean];
    b = [b sum_face];
end
figure,imshow([a+b test_image]);        %stack with mean face, along with original face, display
```