ECE 515

IMAGE ANALYSIS &   
COMPUTER VISION II

MID-TERM PROJECT

NAME: SHREYAS GAONKAR

UIN: 657613409

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF ILLINOIS AT CHICAGO

Display and print 18 face images.

clc;

close all;

clear all;

B = load('KL\_norm\_train.dat');

C = load('KL\_norm\_train\_2.dat');

Z = load('Test\_image.dat');

A = [B,C];

%Initialized the Temp variables to zero%

temp5=0;

distvector=0;

temp6=0;

temp7=0;

temp9=0;

m=0;

c=0;

figure;

a = reshape(A,40,40,18); %Reshaping the original Image in  
 40x40 format for display

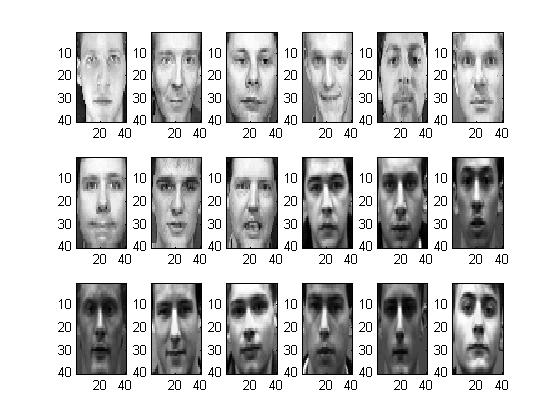
for n= 1:18

subplot(3,6,n);

imagesc(a(:,:,n));

colormap(gray); %This function is used to convert the colored imaged into grayscale image

end



Finding the Average Image.

for n=1:18

m=m+A(:,n);

end

%Displaying Average Faces%

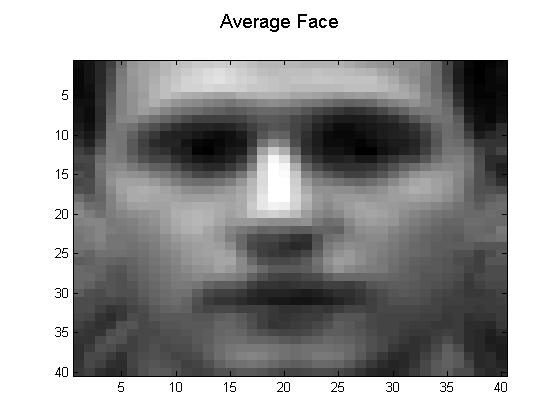
avg = m/18;

figure;

suptitle('Average Face');

imagesc(reshape(avg,40,40));

colormap(gray);   
%This function is used to convert the colored imaged into grayscale image



Displaying and printing the 9 Eigenvectors as Eigen Images

for n=1:18

cov=(A(:,n)-avg)\*(A(:,n)-avg)';

c=c+cov;

end

[eigenvector,eigenvalue] = eig(c);

for n=1:18

temp(:,n)=eigenvector(:,1582+n);

end

figure;

suptitle('Eigen Faces');

temp1 = reshape(temp(:,(1:18)),40,40,18);

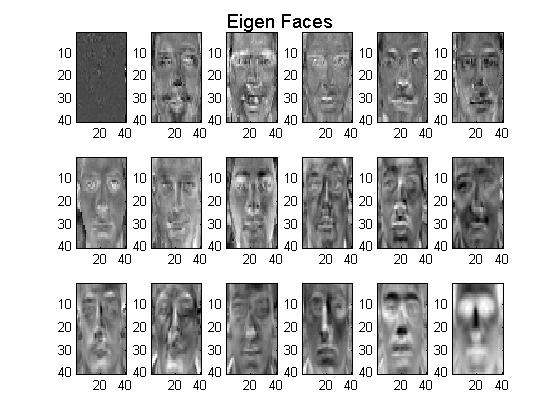
for n= 1:18

subplot(3,6,n);

imagesc(temp1(:,:,n));

colormap(gray);  
%This function is used to convert the colored imaged into grayscale image

end



Reconstructing all the 9 training images with 6 eigenvectors

flip = fliplr(eigenvector(:,1589:1600));

for n=1:18

weight(:,n)=(flip'\*(A(:,n)-avg));

recimg(:,n)=((flip\*weight(:,n))+avg);

end

figure;

suptitle('Reconstructed Image');

temp2 = reshape(recimg(:,(1:18)),40,40,18);  
 %Reconstructed Image is stored in temp2 just to reshape it while displaying

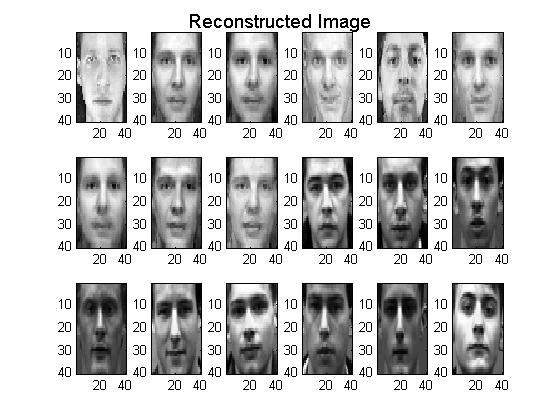
for n= 1:18

subplot(3,6,n);

imagesc(temp2(:,:,n));

colormap(gray); %This function is used to convert the colored imaged into grayscale image

end



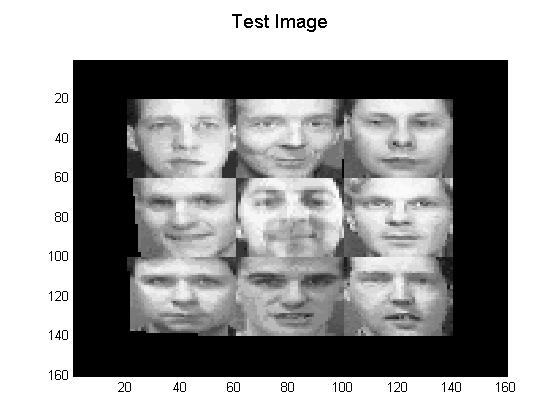
Displaying Test Image

figure;

suptitle('Test Image');

imagesc(Z);

colormap(gray);   
%This function is used to convert the colored imaged into grayscale image



In processing the test image, you need to scan the image with a 40 × 40 window. The part of the image within the window is first normalized by making it zero mean and unit variance to match the statistics of the test image.   
Is this necessary? Why or why not? Please provide a well-thought out argument based on the methods used for detection and recognition.

A. Yes, It is necessary. The main reason is that in detection and recognition, we have to compare the test images. The dimensions of the image should match up with that of the average image and respective Eigen vector. It is impossible to recognize and detect the image if the image is not normalized or scaled.

Why is it necessary to subtract from W?

1. It is necessary since we are given different images to construct Eigen-vectors. Each image is unique and unlike each other and hence we need subtract the average image to get correct coefficient matrix.

Source code for the detection algorithm

for i=1:121 %for1

for j=1:121 %for2

window(:,:)=Z(i:i+39,j:j+39); %Window%

M = mean(mean(window)); %Mean of the window

diff=window-M;

SD = sqrt(sum(sum(diff.^2)));

NW = diff/SD;

%Normalized Window will have DC Value of 0 and variance of 1

output=(reshape(NW,1600,1)-avg);

%y

for k=1:12

y(i,j,k)=dot(flip(:,k),output);

end

%error square

temp8=zeros(1,12);

for k=1:12

temp8(:,k)=(y(i,j,k).^2);

end

temp9=sum(temp8); %temp9 is the sumation of y(i)^2 of 12 values

errorsq=((norm(output).^2)-temp9);

%Lamba i for 12 values

temp3=eigenvalue(1583:1600,1583:1600);

temp4=zeros(1,18);

for i1=1:18

for j1=1:18

if i1 == j1

temp7(:,i1)=temp3(i1,j1); %Lambda i

end

end

end

temp5=sum(temp7(:,13:18)); %Summation Lambda i for last 6 values

rho=(1/6)\*(temp5);

%distance vector

temp12=(y(i,j,:).^2);

for k=1:12

temp11(i,j)=sum((temp12)/fliplr(temp7(:,k)));

end

distvector(i,j)=abs(temp11(i,j)+(errorsq/rho));

end% for1 ends

end% for2 ends

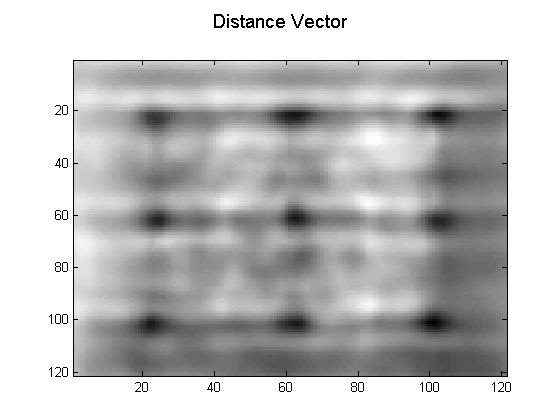
figure;

suptitle('Distance Vector');

imagesc(distvector);

colormap(gray);   
%This function is used to convert the colored imaged into grayscale image

Display and print the distance image.



Detection Algorithm

n=20; %Window

for i=1:121-n+1 %for1

for j=1:121-n+1 %for2

window1(:,:)=distvector(i:i+n-1,j:j+n-1);

M1 = min(min(window1)); %Mean of the window

for i1=i:i+n-1

for j1=j:j+n-1

if (distvector(i1,j1)>M1)

distvector(i1,j1)=255; %Assign higher value for all pixel whose value is greater than the minimum of the window, else keep the pixel value intact

end

end

end

end

end

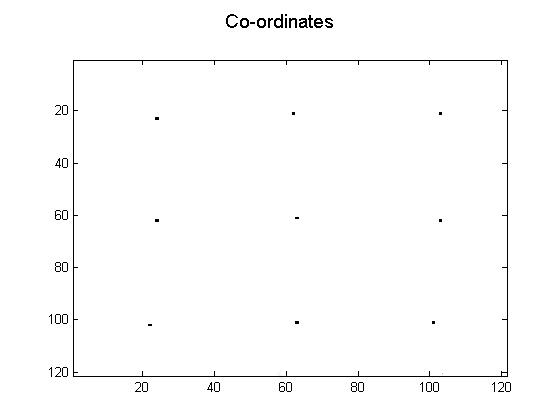
figure;

suptitle('Co-ordinates');

imagesc(distvector);

colormap(gray);   
%This function is used to convert the colored imaged into grayscale image

Co-ordinates of the detected faces



Co-ordinates:

P1 = 23,24

P2 = 21,62

P3 = 21,103

P4 = 62,24

P5 = 61,63

P6 = 62,103

P7 = 102,22

P8 = 101,63

P9 = 101,101

KL – Coefficient Matrix

16.0497 -0.3805 0.4844 -0.4585 0.0243 0.0259

15.9163 -0.4762 0.3803 -0.2159 0.0313 -0.0847

15.8759 -0.4245 0.4058 -0.2374 0.0363 -0.0323

15.9649 -0.4207 0.5163 -0.4237 -0.0582 -0.0374

15.9165 -0.2815 0.6086 -0.1330 -0.0537 -0.0710

15.9813 -0.3586 0.4183 -0.3216 0.0639 -0.0211

15.9377 -0.3023 0.2934 -0.2326 -0.1541 -0.1069

15.9188 -0.3942 0.6932 0.0381 0.2263 0.0001

15.8103 -0.3400 0.4393 -0.1700 0.1259 -0.1299

Distance of test faces versus training faces in the file

0.1930 0.2682 0.2502 0.1989 0.2260 0.2923 0.2389 0.2830 0.3265

0.2902 0.0967 0.1416 0.3564 0.2912 0.1840 0.1559 0.2145 0.1624

0.3103 0.1473 0.1365 0.3758 0.2863 0.2204 0.1551 0.225 0.1778

0.2445 0.2280 0.2272 0.2184 0.2295 0.2723 0.2502 0.3117 0.2969

0.4309 0.3619 0.3576 0.4936 0.2781 0.3758 0.3730 0.3872 0.2892

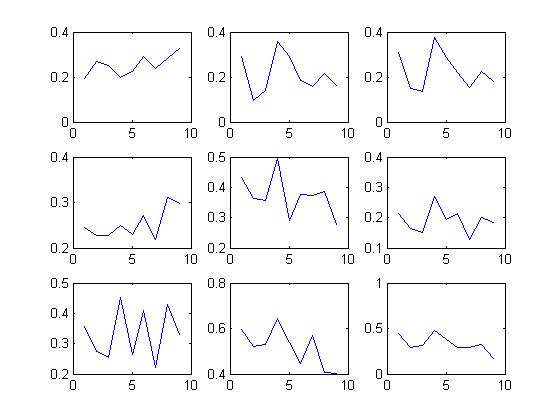
0.2132 0.1654 0.1524 0.2707 0.1938 0.1281 0.2142 0.2003 0.1841

0.3576 0.2728 0.2531 0.4528 0.2647 0.4095 0.2224 0.4306 0.3287

0.5955 0.5228 0.5313 0.6429 0.5403 0.4470 0.5712 0.4083 0.4002

0.4436 0.2885 0.3140 0.4776 0.3779 0.2881 0.2914 0.3200 0.1681

Ploting the distance of Test Image with Training Image



Project Source Code

clc;

close all;

clear all;

B = load('KL\_norm\_train.dat');

C = load('KL\_norm\_train\_2.dat');

Z = load('Test\_image.dat');

A = [B,C];

%Initialized the Temp variables to zero%

temp5=0;

distvector=0;

temp6=0;

temp7=0;

temp9=0;

m=0;

c=0;

figure;

a = reshape(A,40,40,18); %Reshaping the original Image in 40x40 format for display

for n= 1:18

subplot(3,6,n);

imagesc(a(:,:,n));

colormap(gray); %This function is used to convert the colored imaged into grayscale image

end

for n=1:18

m=m+A(:,n);

end

%Displaying Average Faces%

avg = m/18;

figure;

suptitle('Average Face');

imagesc(reshape(avg,40,40));

colormap(gray); %This function is used to convert the colored imaged into grayscale image

for n=1:18

cov=(A(:,n)-avg)\*(A(:,n)-avg)';

c=c+cov;

end

[eigenvector,eigenvalue] = eig(c);

for n=1:18

temp(:,n)=eigenvector(:,1582+n);

end

figure;

suptitle('Eigen Faces');

temp1 = reshape(temp(:,(1:18)),40,40,18);

for n= 1:18

subplot(3,6,n);

imagesc(temp1(:,:,n));

colormap(gray); %This function is used to convert the colored imaged into grayscale image

end

flip = fliplr(eigenvector(:,1589:1600));

for n=1:18

weight(:,n)=(flip'\*(A(:,n)-avg));

recimg(:,n)=((flip\*weight(:,n))+avg);

end

figure;

suptitle('Reconstructed Image');

temp2 = reshape(recimg(:,(1:18)),40,40,18); %Reconstructed Image is stored in temp2 just to reshape it while displaying

for n= 1:18

subplot(3,6,n);

imagesc(temp2(:,:,n));

colormap(gray);   
%This function is used to convert the colored imaged into grayscale image

end

figure;

suptitle('Test Image');

imagesc(Z);

colormap(gray); %This function is used to convert the colored imaged into grayscale image

for i=1:121 %for1

for j=1:121 %for2

%Window%

window(:,:)=Z(i:i+39,j:j+39);

M = mean(mean(window)); %Mean of the window

diff=window-M;

SD = sqrt(sum(sum(diff.^2)));

NW = diff/SD;

%Normalized Window will have DC Value of 0 and variance of 1

output=(reshape(NW,1600,1)-avg);

%y

for k=1:12

y(i,j,k)=dot(flip(:,k),output);

end

%error square

temp8=zeros(1,12);

for k=1:12

temp8(:,k)=(y(i,j,k).^2);

end

temp9=sum(temp8); %temp9 is the sumation of y(i)^2 of 12 values

errorsq=((norm(output).^2)-temp9);

%Lamba i for 12 values

temp3=eigenvalue(1583:1600,1583:1600);

temp4=zeros(1,18);

for i1=1:18

for j1=1:18

if i1 == j1

temp7(:,i1)=temp3(i1,j1); %Lambda i

end

end

end

temp5=sum(temp7(:,13:18)); %Summation Lambda i for last 6 values

rho=(1/6)\*(temp5);

%distance vector

temp12=(y(i,j,:).^2);

for k=1:12

temp11(i,j)=sum((temp12)/fliplr(temp7(:,k)));

end

distvector(i,j)=abs(temp11(i,j)+(errorsq/rho));

end% for1 ends

end% for2 ends

figure;

suptitle('Distance Vector');

imagesc(distvector);

colormap(gray); %This function is used to convert the colored imaged into grayscale image

n=20;

for i=1:121-n+1 %for1

for j=1:121-n+1 %for2

window1(:,:)=distvector(i:i+n-1,j:j+n-1);

M1 = min(min(window1)); %Mean of the window

for i1=i:i+n-1

for j1=j:j+n-1

if (distvector(i1,j1)>M1)

distvector(i1,j1)=255;

end

end

end

end

end

figure;

suptitle('Co-ordinates');

imagesc(distvector);

colormap(gray); %This function is used to convert the colored imaged into grayscale image

[row, col] = find(distvector<255);

rowcol=[row,col];

rowarranged=[23;21;21;62;61;62;102;101;101];

colarranged=[24;62;103;24;63;103;22;63;101];

rowcolarrange=[rowarranged,colarranged];

for i=1:9

img=Z(rowarranged(i):rowarranged(i)+39,colarranged(i):colarranged(i)+39);

M2=mean(mean(img));

diff1=img(:,:)-M2;

SD2=sqrt(sum(sum(diff1.^2)));

NW2(:,:,i)=diff1/SD2;

subplot(3,3,i);

imagesc(NW2(:,:,i));

colormap(gray);

end

for j2=1:9

oldweight(:,:,j2) = (A(:,j2)-avg)'\*(fliplr(eigenvector(:,1595:1600)));

end

oldweight=squeeze(oldweight);

for j2=1:9

newweight(j2,:)=((reshape(NW2(:,:,j2),1600,1))-avg)'\*(fliplr(eigenvector(:,1595:1600)));

for k2=1:9

final(j2,k2)=norm(newweight(j2,:)-(oldweight(:,k2))');

end

end

disp(newweight);

disp(final);

figure;

for i1=1:9

for j1=1:9

subplot(3,3,i1);

plot(final(i1,:)); %Distance of Test Image with Training Image

end

end