Xi'an Jiaotong-Liverpool University

西交利物浦大学

**DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING**

# EEE412 Image and Video Processing

## Lab3 - Eigenfaces


Student Name Zhiyu.Xu

Student ID 1926905


October 2019

# Contents

# Task 1 Orthonormal basis

## 1.1 Code and result

```
%Task 1 Determine whether the feature face matrix is an
orthogonal matrix
load('data_for_labC.mat');
faces=eignfaces_balk;
[m,n,l]=size(faces);
X=zeros(l,m*n);
%Reshape the matrix to a 101*135000 matrix
for i=1:l
x=faces(:,:,i);
X(i,:)=reshape(x,1,m*n);
end
%Determine whether the product of the matrix and the
transposed matrix is
%an identity matrix;
output_initial=X*X'/(m*n);
output=round(output_initial);
if output == eye(l,l)
    judgment=1; %Orthogonal
else
    judgment=0; %Not orthogonal
end
```

Code 1


Figure 1 Result

## 1.2 Explanation

The following formula can be obtained by the definition of an orthogonal matrix:

$$AA^T = E$$

$A^T$ is the transposed matrix of a, and e is the identity matrix. According to the requirements of Task1, we need to verify whether eignfaces_balk is orthogonal. But eignfaces_balk is a 450*300*101 3D matrix, so we need to reshape it to a 2D matrix of 101*135000. Finally verify that the matrix is multiplied by the product of the transpose of the matrix, whether it is an identity matrix. In code1, when the judgment is equal to 1, the measured eigenfaces are orthogonal.

# Task 2 Eigenfaces weights

## 2.1 Result and code

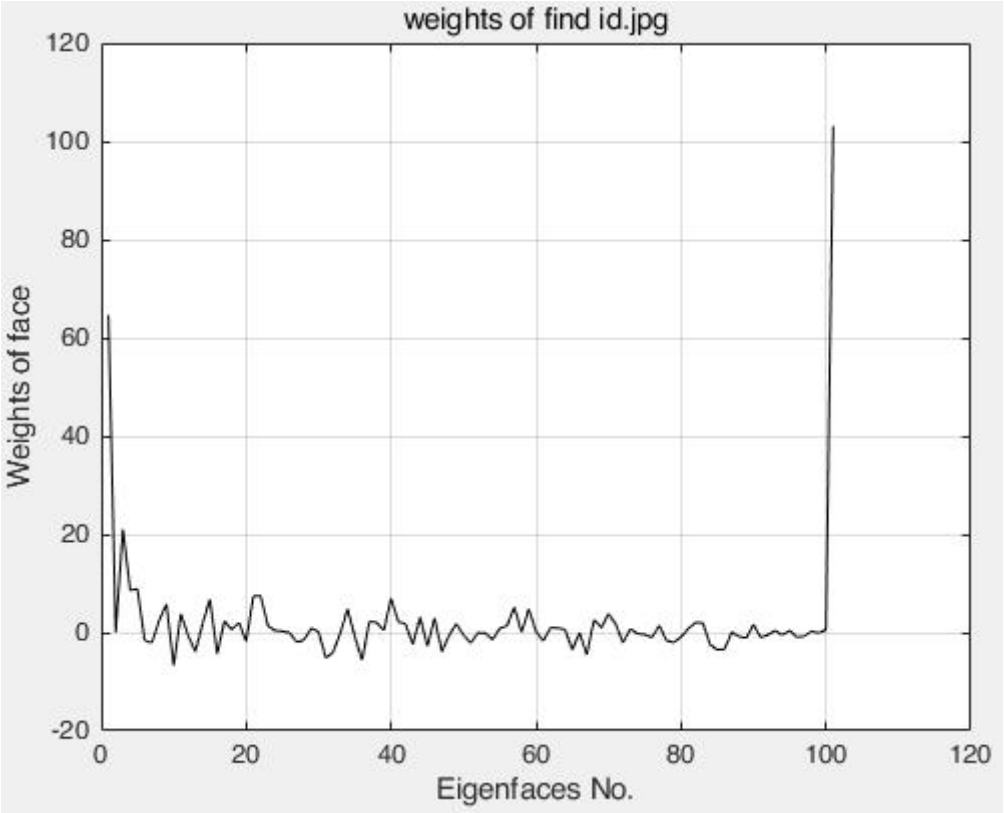


Figure 2 weights of find_id.jpg

```
%Task2 Evaluating the Eigenfaces weights of a face;
im=imread('find_id.jpg');
load('data_for_labC.mat');
weights_of_find_id=get_face_weights(im,eignfaces_blk);
plot(weights_of_find_id,'k');grid on;
title("weights of find id.jpg");
xlabel('Eigenfaces No.');ylabel('Weights of face');
```

Code 2 Main code

```
function [weights_of_face] = get_face_weights(im,
eigenfaces_blk)
%UNTITLED2 Evaluating the Eigenfaces weights of a
face(im)
%The weights of the feature quantity is obtained by
separately
%calculating the coefficient of the measured image and
each eigenfaces;
```

```
[n,m,l]=size(eigenfaces_blk);
weights_of_face=zeros(1,l);
im=double(im);
for i=1:l
weights_of_face(1,i)=sum(sum(im.*eigenfaces_blk(:,:,i)
))/(m*n);
end
end
```

Code 3 Function of Evaluating the eigenfaces weights

## 2.2 Explanation

Figure2 is the plotting result of weights of find_id.jpg, the x-axis represents all numbers of eigenfaces(1 to 101), and the y-axis represents weight corresponding to each eigenface.

Code3 is the core code that implements evaluating weights, The main operations are as follows: The image pixels to be measured are multiplied by the pixels corresponding to each eigenfaces, and then averaged, that is, the weight of the eigenfaces.

# Task 3 Face generation

## 3.1 Result and code


Figure 3 Original Image and Image generated from weights

```
%Task3 Face generation from its  "weights";
im_OG=imread('find_id.jpg');
load('data_for_labC.mat');
weights_of_find_id=get_face_weights(im_OG,eignfaces_blk);
im=generate_face_from_weights(weights_of_find_id,eignfaces_blk)
;
subplot(1,2,1);imshow(im_OG);title('Original Image');
subplot(1,2,2);imshow(im);title('Generated Image from weights');
truesize;
```

Code 4 Main code

```
function[im]=generate_face_from_weights(weights_of_fac
e,eigenfaces_blk)
%Face generation from its  "weights;
%Superimpose the product of the weights of eigenfaces and
eigenfaces to get
%the original image;
[m,n,l]=size(eigenfaces_blk);
im=zeros(m,n);
```

```
im=double(im);
for i=1:l
im=im+weights_of_face(1,i)*eigenfaces_blk(:,:,i);
end
im=uint8(im);
end
```

Code 5 Function of generate_face_from_weights

## 3.2 Explanation

From Figure 3 we can see that the original image and the generated image are basically the same.

Code5 is the core code that implements generating a face from its "weights", the main operations are as follows: By superimposing the product of the eigenfaces and the eigenfaces weights to obtain the face image to be generated.

Through Task2 we get 100 parameters, multiply these features one by one by the eigenfaces, so we get a 450*300*101 three-dimensional matrix, and then we superimpose this three-dimensional matrix into a 450*300 two-dimensional matrix. And we get the last face image we need.

# Task 4 Recognizing an employee

## 4.1 Result and code

| ID | 96 |
|---|---|

Table 2 Identification result

```
%Task4 Recognizing an employee from his/her image;
im = imread('find_id.jpg');
load('data_for_labC.mat');
ID =
get_employees_ID_from_DB(im,employees_DB,eignfaces_bl
k);
```

Code 6 Main Code

```
function [ID] =
get_employees_ID_from_DB(im,employees_DB,eignfaces_bl
k)
%Recognizing an employee from his/her image;
%By calculating the Euclidean distance of the known
weights of face and
%each weights in the database; The ID corresponding to the
smallest weights
%distance is the ID that is found;
weights_of_face = get_face_weights(im,eignfaces_blk);
[~,m] = size(employees_DB);
Euclidean_distance = zeros(1,m);
for i = 1:m
    Euclidean_distance(i)=norm(weights_of_face-
    employees_DB(i).weights);
end
[ID] =
find(Euclidean_distance==min(Euclidean_distance));
ID = employees_DB(ID).id;
end
```

Code 7 Function of get employees ID

## 4.2 Explanation

By calculating the Euclidean distance of the known weights of face and each weights in the database; The ID corresponding to the smallest weights distance is the ID that is found.

# Task 5 Discuss the robustness

## 5.1 How to reject the image

The function of get employees ID(Code 7) designed in Task4 has a theoretical flaw: the system will give the closest face information regardless of whether the face being tested is the real face information in the database. Thus, the designed face recognition system loses its meaning.

For the above problem, we should not only find the minimum distance when the function judges Euclidean_distance, but also add a threshold judgment. If the minimum distance is greater than the modified threshold, the measured face does not belong to the database; if the minimum distance is less than the threshold Directly output the found ID information.

Code show as below:

```
function [ID] =
get_employees_ID_from_DB_new(im,employees_DB,eignface
s_blk,Threshold)
%Recognizing an employee from his/her image;
%By calculating the Euclidean distance of the known
weights of face and
%each weights in the database; The ID corresponding to the
smallest weights
%distance is the ID that is found;
weights_of_face = get_face_weights(im,eignfaces_blk);
[~,m] = size(employees_DB);
Euclidean_distance = zeros(1,m);
for i = 1:m
Euclidean_distance(i)=norm(weights_of_face-employees_D
B(i).weights);
end
A = Euclidean_distance==min(Euclidean_distance);
B = min(Euclidean_distance);
%set threshold;
if B <= Threshold
    ID = employees_DB(A).id;
else
    ID = 0;
    disp('Warning!! face recognition error!');
end
end
```

Code 8 function of getting employees ID with threshold

## 5.2 Query image with noise

In this case, we chose adding Gaussian Withe Noise and Salt & Pepper noise to the query image respectively. Then add Gaussian white noise with variance of 100 , Gaussian white noise with variance of 500, 20% Salt & Pepper noise and 65% Salt & Pepper noise to the original image respectively. The result is shown below:

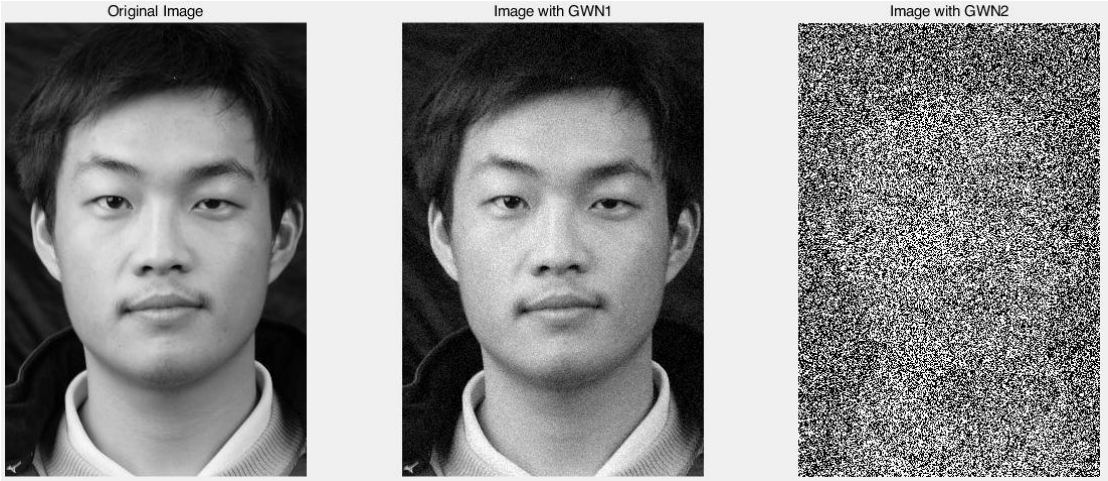


Figure 4 Image with GWN

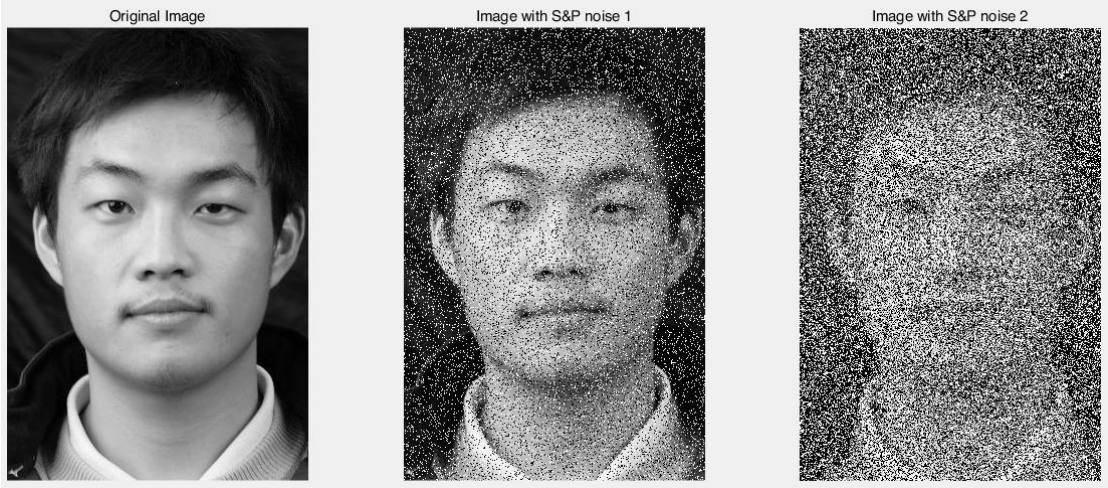


Figure 5 Image with S&PN

And The identification ID results are shown in the following table:

| Images | ID number |
|---|---|
| Im_original | 96 |
| Im_GWN1 | 96 |
| Im_GWN2 | 0 |
| Im_SPN1 | 0 |
| Im_SPN2 | 0 |

Since the threshold is set in function of Code8, when the ID is equal to 0, the database rejects the query image. Therefore, according to the information in the above

table, only the original image and the image with the Gaussian white noise with a variance of 100, the face information of the 96th is successfully recognized.

In order to solve the above problem, it is necessary to add a filter to the function. Here, SPN is taken as an example, so we set a median filter in the front of the function.

Results are shown in the following table:

| Images | ID number |
|---|---|
| Im_SPN1_filter | 96 |
| Im_SPN2_filter | 0 |

As can be seen from the above table, when the function is added to the 5X5 median filter, an image with 20% S&P noise is added and successfully recognized. However, if there is too much noise and it becomes dominant in the image information, the system cannot recognize it, it's all depending on the effect of the filter.

By visually observing the image information in figure5, it is difficult to distinguish the face from the third picture, so we can consider classifying the image with too much noise as an unrecognizable image.

The code is as follows:

```
%Task5 Discuss the robustness
im_OG = imread('find_id.jpg');
load('data_for_labC.mat');
%% Adding Gaussian Withe noise
im_GWN1 = imnoise(im_OG, 'gaussian', 0, 10^2/255^2);
im_GWN2 = imnoise(im_OG, 'gaussian', 0, 500^2/255^2);
%% Adding Salt & Pepper noise
im_SP1 = imnoise(im_OG, 'salt & pepper',0.2);
im_SP2 = imnoise(im_OG, 'salt & pepper',0.65);
%% Get employees ID
ID =
get_employees_ID_from_DB(im_OG,employees_DB,eignfaces
_blk);
ID_GWN1 =
get_employees_ID_from_DB_new(im_GWN1,employees_DB,eig
nfaces_blk,5);
ID_GWN2 =
get_employees_ID_from_DB_new(im_GWN2,employees_DB,eig
nfaces_blk,5);
ID_SP1 =
get_employees_ID_from_DB_new(im_SP1,employees_DB,eign
faces_blk,5);
ID_SP2 =
get_employees_ID_from_DB_new(im_SP2,employees_DB,eign
faces_blk,5);
%% Get employees ID(with filter)
```

```
ID_SP1_filter =
get_employees_ID_from_DB_new1(im_SP1,employees_DB,eig
nfaces_blk,5);
ID_SP2_filter =
get_employees_ID_from_DB_new1(im_SP2,employees_DB,eig
nfaces_blk,5);
%% Plot images
figure(1)
subplot(1,3,1);imshow(uint8(im_OG));title('Original
Image');
subplot(1,3,2);imshow(uint8(im_GWN1));title('Image
with GWN1');
subplot(1,3,3);imshow(uint8(im_GWN2));title('Image
with GWN2');
truesize;
figure(2)
subplot(1,3,1);imshow(uint8(im_OG));title('Original
Image');
subplot(1,3,2);imshow(im_SP1);title('Image with S&P
noise 1');
subplot(1,3,3);imshow(im_SP2);title('Image with S&P
noise 2');
truesize;
```

Code 9 Main code

```
function [ID] =
get_employees_ID_from_DB_new1(im,employees_DB,eignfac
es_blk,Threshold)
%Recognizing an employee from his/her image;
%By calculating the Euclidean distance of the known
weights of face and
%each weights in the database; The ID corresponding to the
smallest weights
%distance is the ID that is found;
im_new = medfilt2(im,[5 5]);%add 5X5 median filter
weights_of_face =
get_face_weights(im_new,eignfaces_blk);
[~,m] = size(employees_DB);
Euclidean_distance = zeros(1,m);
for i = 1:m

Euclidean_distance(i)=norm(weights_of_face-employees_D
B(i).weights);
end
A = Euclidean_distance==min(Euclidean_distance);
B = min(Euclidean_distance);
%set threshold;
if B < Threshold
    ID = employees_DB(A).id;
```

```
else
    ID = 0;
    disp('Warning!! face recognition error!');
end
end
```

Code 10 function of getting employees ID with 5X5 median filter