EEE412-IMAGE PROCESSING-LAB3

Author: Zhipeng Ye

Student ID: 1926908

Faculty: MSc Multimedia Telecommunications

Data: 21th October 2019

Orthonormal basis. (**15 marks**)

The variable eigenfaces_blk contains 100 eigenfaces, verify that these eigenfaces are orthogonal, and if they are not orthonormal, please normalized them to orthonormal.

verify that these eigenfaces are orthogonal

Code

```
clear
clc
load('data_for_labC.mat')

[m, n, k] = size(eignfaces_blk);

% k-1 means 101-1=100

for i = 1:k-2
    for j = i+1:k-1

%         check the result of them multiplying to each other is 0
        matrix = eignfaces_blk(:,:,i).*eignfaces_blk(:,:,j);

        check_value = int16(sum(sum(matrix)));

%         compare two value in matlab must use this formula
        if (abs(check_value - 0) > 0.000001)
            sprintf('i:%d,j:%d', i,j)
            disp('These eignface are not orthogonal!')
        end

    end
end
```

Result

They are all orthogonal.

Analyze

The orthogonal vectors means two vectors are prendicular to each others. Suppose $\vec{a}, \vec{b}$ are two orthogonal vectors , then $\vec{a} * \vec{b} = 0$.

normalized them to orthonormal

Code

```
clear
clc
load('data_for_labC.mat')

[m, n, k] = size(eignfaces_blk);

eignfaces_blk_copy = zeros(m,n,k-1);

% Standard orthogonalization
for i = 1:k-1

    norm_value = sqrt(sum(sum(eignfaces_blk(:,:,i).^2)));

    if (abs(norm_value - 1) > 0.01)

        disp(norm_value)
        disp('it is not a orthonormal')
        eignfaces_blk_copy(:,:,i)  = eignfaces_blk(:,:,i)./norm_value;
    end

end

% verify
count = 0
for i =1:k-1

    norm_value = sqrt(sum(sum(eignfaces_blk_copy(:,:,i).^2)));

    if(abs(norm_value - 1)<0.001)
        count = count+1;
        disp('it is a orthonormal');
    else
        disp('it is not a orthonormal')
    end

end
sprintf('The number of orthnologinal = %d',count)
```

Result

The copy of eignfaces_blk become orthonormal.

Analyze

We can use vector module to make vector orthonormal by the transformation of $\vec{a}/|\vec{a}|$.

Evaluating the Eigenfaces weights of a face. (**20 marks**)

When an image of a face is presented to the system for classification/ recognition, its own weights are found by projecting the image onto the collection of eigenfaces. This provides a set of weights describing this particular face.

Write a Matlab function which evaluates the weights of a face. The function should have the following declaration:

```
function [weights_of_face] = get_face_weights(im, eigenfaces_blk);
```

Code

```
function weights_of_face = get_face_weights(im, eignfaces_blk)
```

```
im = double(im)

[m,n,k] = size(eignfaces_blk);

weights_of_face = [];

im_reshaped = reshape(im,[m*n,1]);

for i = 1:k
    eignface_blk_reshaped = reshape(eignfaces_blk(:,:,i),[m*n,1]);
%     because these eignfaces are orthogonal
    weight = eignface_blk_reshaped'*im_reshaped
/(eignface_blk_reshaped'*eignface_blk_reshaped);
    weights_of_face = [weights_of_face weight];
end

end
```

Analyze

We can use the property of orthogonality to get weights, Because of this formula.

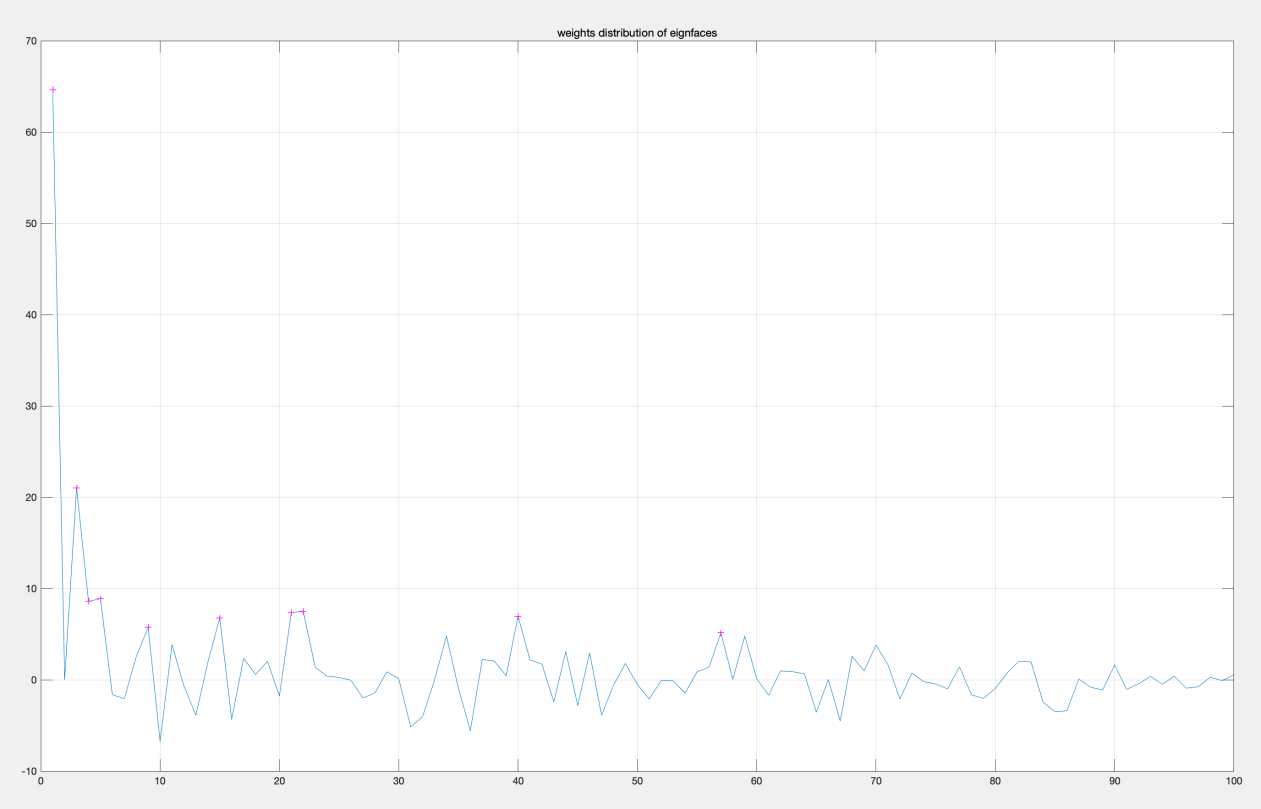
$$\begin{aligned} \vec{A} &= w_a * \vec{a} + w_b * \vec{b} + w_c * \vec{c} + \dots \\ \vec{a}' * \vec{A} &= w_a * \vec{a}' * \vec{a} + 0 + 0 + \dots \\ w_a &= \frac{\vec{a}' * \vec{A}}{\vec{a}' * \vec{a}} \end{aligned}$$

Use the function get_face_weights to find the weighting parameters for the image *find_id.jpg* . Plot these weighting parameters, and comment on whether this plot carries any information about the employee's face or not.

Code

```
clear
clc
load('data_for_labC.mat')

im = imread('find_id.jpg')

% get the top 100 weights
eignfaces_blk_copy = eignfaces_blk(:,:,1:end-1);

weights_of_face = get_face_weights(im, eignfaces_blk_copy);

ids = 1:100

plot(ids,weights_of_face)

title('weights distribution of eignfaces')

grid on
hold on

% find the eignfaces which has a big weights
ids_max = find(weights_of_face>=5)

ids_max_value = weights_of_face(ids_max)

plot(ids_max, ids_max_value, 'm+')
```

Result



Analyze

This weights carry information of employee's face. We can see from result, some eignfaces own larger weights and some eignfaces have litter weights. The principle of this technology is PCA or SVD. Eignface means eigenvector and weights mean eigenvalue.

Face generation from its "weights". (**20 marks**)

Write a Matlab function which generates a face from its weights. The function should have the following declaration:

```
function [im] = generate_face_from_weights(weights_of_face, eigenfaces_bl
```

Use the obtained weights_of_face in the previous task (i.e.,task2) to synthesise the image of the face using the function generate_face_from_weights.

Code

```
function im = generate_face_from_weights(weights_of_face, eignfaces_blk)

    [m,n,k] = size(eignfaces_blk);
    [q,w] = size(weights_of_face);

    if(w~= k)
       error('parameter error')
    end

    sum = zeros(m,n);

    for i = 1:k
       sum = sum + eignfaces_blk(:,:,i)*weights_of_face(i);
    end

    im = uint8(sum);
end
```

```
% I think it isn't enough to just use 100 basis to describe 450*350 pixels, because if we
want
% to describe R^n vector space, we must use n basis which is independent to generate whole
vector
% space. The 100 basis just can generate a subspace.
```

Result

Comment on how 100 parameters are enough to describe an image of a face which has 450X300 pixels.

I think 100 parameters can't describe an image of a face which has 450*300 pixels. Although we can use little Eigenvector and eigenvalue to recognize the specfic face. but 100 vector can't represent whole $R^{450*300}$ vector space in theory. But in this experiment, it is ok to implement it.

Recognizing an employee from his/her image. (**25 marks**)

To recognize someone from his (/her) face's photo the weights of this face need to be compared against all weights in the database to find the closest match.

The Euclidean Distance between two vectors is a simple approach for finding two nearest neighbors; this metric could be used in this task to find the closest face in the database.

Write a Matlab function which finds the employee's ID starting from his/her photo. The function should have the following declaration:

```
function [ID] = get_employees_ID_from_DB (im, employees_DB, eigenfaces_blk);
```

The first thing you need to do is to use the function get_face_weights to generate the weighting parameters for the given face.

Then you need to compare the face weights with all the employees' record in the employees_DB to find the closest to the given one. Output the ID of the closest face.

Find the employee's ID of the image find_id.jpg, and write it down in your report.

Code

```
function ID = get_employees_ID_from_DB (im, employees_DB, eigenfaces_blk)
```

```
    [m,n,k] = size(eigenfaces_blk);

%     get weights of face
    weights_of_face = get_face_weights(im,eigenfaces_blk);
    employer_distance_list = [];
    for i = 1:k
        employer_weights = employees_DB(i).weights;
        employer_weights = employer_weights(1:end-1)
        Euclidean_distance = sqrt(sum((weights_of_face - employer_weights).^2));
        employ.id = i;
        employ.distance = Euclidean_distance;
        employer_distance_list = [employer_distance_list,employ];
    end

%     find the closest one
    [employer_distance_list_sorted, index] = sort([employer_distance_list.distance]);

    ID = employer_distance_list(index(1)).id
end
```

Result

```
ID = 96
```

Analyze

Firstly, I use Euclidean distance to compare 2 faces, then use struct to record result,Finally I use sort function to get the cloest one.

Discuss the robustness of the eigenface-based image recognition algorithm under various working conditions, like with salt & pepper noise. How can we make the algorithm more robust? You can thinking about this question following these issues (but not limited), like how to reject the image out of the data base or what should we do if the query image is with noise? (**20 marks**)

Code

```
im_noise = imnoise(im,'salt & pepper',0.6)
% change return result for testing
IDs = get_employees_ID_from_DB(im_noise, employees_DB,eignfaces_blk_copy)
IDs = IDs(1:5)
```

Result & analyze

We can find the face recognition algorithm will fail when noise become more. Here are some solution to solve this problem. Firstly, we must reduce noise before face recognition. Secondly, we can create more eignface and expand the data set to describe image clearly. Thirdly, Euclidean distance maybe not the perfect distance to compare two picture with noise such as 'salt & pepper' because the image with 'salt & pepper' will have a lot of 0 and 255. I think we can use **Normalization** to make data at $[0\ 1]$ which can reduce the effect of 'salt& pepper' noise. Fourthly, we can take more pictures of a person before recognization and vote from multiple reslut. This means if one picture get results of ID = 96, ID = 88, ID = 88, ID = 99, So we can choose 88 as the result. Finally, we can try other algorithm instead of PCA. such as CNN to get better result.

Conclusion and discussion

In this experiment, I review some concept such as orthogonal and orthonormal. Moreover, vector is a good tool to represent image, so this experiment ask us to use PCA (Eigen value and Eigen vector) to recognize face. I am inteseted in other face recognition algorithms such CNN & SVM. I will survey these technology after class.