INTRODUCTION

This experiment guides students through the functions of MATLAB regarding image transformation. It helps students to build an intuitive understanding of facial recognition with Principal components analysis(PSA) and eigenfaces，leading to further study on image processing technics.

RESULTS&EXPLAINATIONS

**TASK1**

**(1)**

**Orthogonality is a generalization of vertical in the inner product space of vectors. Two vectors in an inner product space are said to be orthogonal if their inner product is zero, which indicates that the two vectors are independent of each other.** **For matrices, a matrix is said to be orthogonal if its cross product with its transpose is a unitary matrix.**

**In machine learning, the idea of orthogonality is often used for data dimensionality reduction. For data represented as vectors, each dimension (direction) represents a feature (factor/texts/pixels that influences the variations with the data set). Features that do vary similarly are consolidated by using orthogonal basis to re-represent the data set (like what orthogonal decomposition in mechanics analysis does) to obtain a new data set that retains maximum variance.** **Doing so highlights the variability of features within the data set and reduces the complexity of the algorithm without losing too much information.**

**(2)**

**Two vectors , are orthogonal if their dot product equals to zero.**

**The essential function is:**

round(dot(V1,V2))

**Where V1 and V2 are converted vector from eigenfaces A and B**

V1 = A(:,1);

V2 = B(:,1);

for a = 2:300

V1 = [V1;A(:,a)];

end

for b = 2:300

V2 = [V2;B(:,b)];

end

Should round(dot(V1,V2)) be zero, the faces are (almost) orthogonal.

**(3)**

**To check if the vector**  **is a unit vector, calculate the length**

**If , is a unit vector.**

**To normalized the vectors, should use a factor of where are the number of columns and rows in an eigenface matrix.**

Thus, the essential code is:

%check if it is a unit vector

if dot(V,V) == 1

eignfaces\_blk\_norm(:,:,i) =A;

else

%if not, normalize it

B = normalize(reshape(A,[450,300]));

eignfaces\_blk\_norm(:,:,i) = B;

**Where A is the eigenface matrix extracted from** eignfaces\_blk, looping from i to 100. V is the vector converted from A. Use the stock normalize function to normalize the matrix and add it into eignfaces\_blk\_norm

**TASK2**

**(1)**

**The essence of finding the weights is to find out how well the original image is represented in the eigenspace, a process which, according to Principal components analysis, goes through the following steps.**

**1. Pre-processing the data**

**transforming the picture and the eigenface matrix into vectors (by connecting each column)**

**The essential code is listed below:**

face\_vector = [];

covariance\_vector = [];

face\_vector = [face\_vector;eignface(:,k)];

**2. Image decentration**

**Since the original image is already known to consist of eigenfaces, the image data can be obtained by averaging all eigenfaces to obtain a consistent representation in all feature spaces, which is equivalent to the mean face**

**The essential code is listed below:**

B = zeros(450,300);

for i=1:100

B = B + eignfaces\_blk(:,:,i);

end

Meanface = B/100;

**3. Eigenvalue decomposition and reprojection**

**The difference between the actual image data and the average face (covariance matrix** **) is transformed into each eigenspace in turn, taking out the projection discrepancies** **, aka the weights by taking advantage of the orthogonality of , for unit vector in ,dot product with result in other vectors’ nullification, leaving only corresponded weight with a level of parameter from the mean eigenfaces.**

**The essential code is listed below:**

covariance\_matrix = A-Meanface;

where A is the original image matrix

weights\_of\_face(1,j) = dot(face\_vector,covariance\_vector)+0.01;

**(2)**

**Use the stem function to plot the discrete weights sequence, shown below**

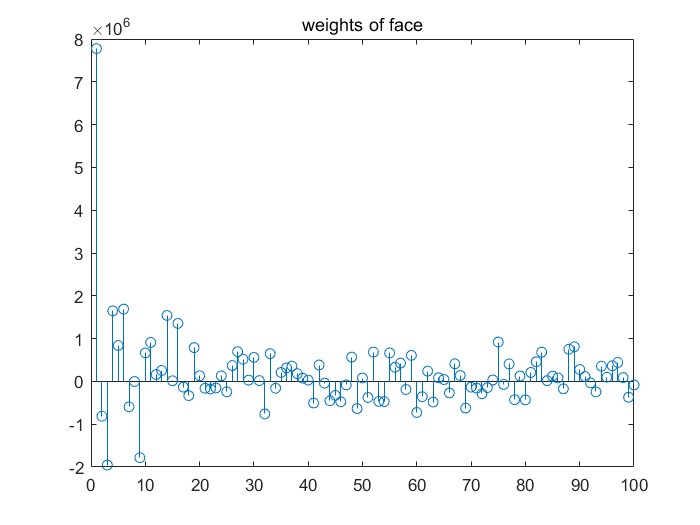


Figure 1-data sequence of weights

**The data for these weights clearly reflect the component weightings that the individual features (eigenfaces) as to the overall picture.** **In theory, any vectors in a specific space can be represented by a linear combination with unique weights of each basis. Consequently, the stem plot shown on Figure. 1 indicates a unique weight representation about the face, which means the stem plot and the face are One-to-One-Corresponding. Furthermore, a larger value of weight parameter indicates that this eigenface is a more dominant basis than others, results in greater similarity.**

**TASK3**

**(1)**

**It is easy to know that with all image features and all feature representation states a reproduction a picture****could be achieved.**

**Here eigenfaces are features under given conditions , and the weighting reflects the expression state of the corresponding feature. Hence, a synthesized image could be generated.**

**In the case of vectors that are orthogonal to each other, the mapping conditions of the original image in their respective spaces can be easily calculated, so eigenfaces made to be orthogonalize with each other easily allows principal component analysis and feature extraction.**

**(2)**

**The** **synthesize the image of the face, using weights obtained from task2, is shown below**

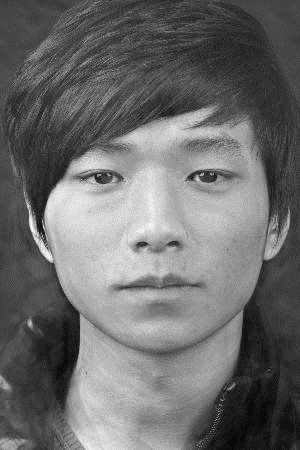


Figure 2-face generated using Weights\_of\_face

**(3)**

**The PSNR was calculated and listed below**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number** | **20** | **40** | **60** | **80** | **100** |
| **PSNR** | **12.6106** | **13.1313** | **13.7595** | **14.2406** | **13.9365** |

**The PSNR values are all very low, even when all eigenfaces are used, indicating very low fidelity representations to the original picture, which is also consistent with the low similarity of figure2 to the original.**

**This suggests that the 100 eigenfaces given for this experiment are not sufficient to describe all the features of the images well. However, in figure2, to the naked eye, the synthesized image does manage to preserves to a level of extent detail of the original image, although a portion of greyscale details is lost.**

**Task4**

**(1)**

**In the case of using the same eigenfaces set, the faces should be identified by calculating weights and comparing the Euclidean distance between the weights of the target image and the weights in the database, as the lab instruction indicated.**

**The essential code is listed below**

%extract the weighting data

weight = employees\_DB(:,i).weights;

%calculate the Euclidean distance

d = (weights\_of\_face - weight(1:100)).^2;

Euclidean\_distance = sqrt(sum(d(:)));

%use comparator to help findout the smallest Euclidean distance

%along with assoicated id

if Euclidean\_distance < comparator

comparator = Euclidean\_distance;

ID = employees\_DB(:,i).id;

**Note that a large-valued comparator was used to initial the comparison.**

**(2)**

**The ID is 81**

****

****

Figure 3-displayed id

**(3)**

**This seems that the face of the employee with id 71 is the closest to the original image apart from the face of the employee with id 81, with the employee with id 37 being the third closest.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **number** | **20** | **40** | **60** | **80** | **100** |
| **id** | **37** | **71** | **81** | **81** | **81** |

**(4)**

**For noise, as** Ding, Y., Cheng, Y., Cheng, X. et **concluded [1], two methods could be used to improve the noise immunity of the algorithm, one is to transfer the image signals to another domain where they can be more easily separated from the noise, and the other is to capture the image statistics directly in the image domain. For the algorithm constructed for this experiment, the first method is more applicable.**

**J. Portilla, V. Strela, M. J. Wainwright and E. P. Simoncelli further explained [2] that it is possible to decompose the conditions on how structural vectors transform into Gaussian noise in ways such as the Gaussian Scale Mixtures, and then restrict them, forming a denoising component that could help the eigenface-based image recognition algorithm.**

**In addition, Khaung Tin, Dr.Hlaing Htake pointed out [3] that pre applying method such as Median filtering could reduce “salt and pepper" noise and is “more effective than convolution when the goal is to simultaneously reduce noise and preserve edges”.**

**Moreover, if the input image does not belong to a face in the database, then the lower algorithm will obtain feature projection matches with very low absolute values of covariance. This can be handled more effectively if a minimum recognition confirmation value is set to filter out the poor matches, in the case where the face is not in the database.**

****SUMMARY****

In this experiment students learned how to use MATLAB to perform image transform. Students mastered advance image processing technics including calculating image matrix vectorization. Students gained insight into important concepts such as eigenfaces and principal components analysis. Students also develop skills on coding and mathematical to progress in the field of image processing.

Reference list

[1] Ding, Y., Cheng, Y., Cheng, X. et al. Noise-resistant network: a deep-learning method for face recognition under noise. J Image Video Proc. 2017, 43 (2017). <https://doi.org/10.1186/s13640-017-0188-z>

[2] J. Portilla, V. Strela, M. J. Wainwright and E. P. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," in IEEE Transactions on Image Processing, vol. 12, no. 11, pp. 1338-1351, Nov. 2003, doi: 10.1109/TIP.2003.818640.

[3] Khaung Tin, Dr.Hlaing Htake. (2013). How Old Are You?: Age Prediction using Eigen Face. Conference: International Conference on Science and Engineering (ICSE 2013).<https://www.researchgate.net/publication/263547734_How_Old_Are_You_Age_Prediction_using_Eigen_Face>

APPENDIX

**Images**

**Figure1, task2.jpg-** **data sequence of weights generated for task 2.2**

**Figure2-** **face generated using Weights\_of\_face**

**Figure3- employee id required to find**

**im.bmp-** **synthesize the image of the face using all eigenfaces**

**im20.bmp, im40.bmp, im60.bmp, im80.bmp-** **synthesize the image of the face using 20,40,60,80 eigenfaces**

**Scripts**

**Scripts are provided with MATLAB functions corresponded to the tasks’ requirements.**

**.mat files**

**weights\_of\_faces.mat-** **weighting parameters for the image find\_id.jpg**