QEA Module 2: Faces!

Minju Kang, Prava Dhulipalla

I. Abstract

Two facial algorithms were implemented using Matlab for the purpose of detecting the faces of Qualitative Engineering Analysis' students and instructors. The two algorithms used were Eigenfaces and Fisherfaces. The former was used because it was generally fairly quick and accurate. Fisherfaces was a natural extension from Eigenfaces, and also more accurate (but less fast). Eigenfaces, in comparison to Fisherfaces, preserves its variance while Fisherfaces discriminates two faces better and also excludes the light intensity of the image. Our findings were that Fisherfaces were more accurate than Eigenfaces, although not significantly, and took slightly more time to run in MATLAB.

II. Introduction

Facial recognition is a tool that is already used in daily life. Security system uses facial recognition to recognize the person in surveillance camera. More commonly, Facebook uses facial recognition to suggest who to tag in the picture. Facial recognition usually first detects faces and then matches detected faces with faces in databases. For the purposes of the MATLAB implementation, there was no code to specifically determine if a face was in fact a face (the data base of training images and testing images used were all faces). This technology is not perfect; changes in viewpoint, expression, lighting, age, and occlusion decrease the accuracy of the facial recognition. Still, facial recognition algorithms are pretty much reliable and there are several algorithms for facial recognition such as ”Eigenfaces”, ”Fisherfaces”, ”Face Features”, ”AAM”, ”Neural Network”, etc. This report was written with the intent to aid with the comprehension of linear algebra terms and concepts behind the methodology of facial recognition and the principle concepts of how facial recognition algorithms work. The MATLAB software makes use of two distinct algorithms - the first being Eigenfaces, and the second being Fisherfaces. Eigenfaces was selected for its ability to be both fairly quick to compile and fairly accurate. It makes use of dimensionality reduction so that an image can be represented by less than a hundred numbers rather than above sixty-five thousand pixels. It also minimized variation within images. The Fisherfaces algorithm implemented also made use of dimensionality reduction and minimized variation within images, but also maximized variation between images. Both Eigenfaces and Fisherfaces will be discussed further in detail later in this report (see section three of the report: *Algorithms and Justification*).

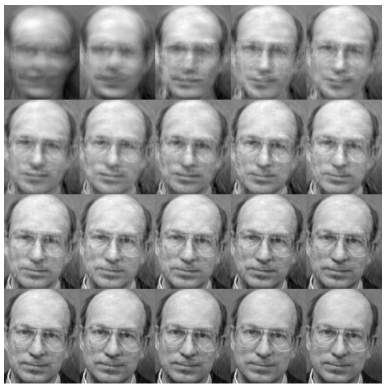


Fig. 1. Examples of facial recognition by using eigenfaces

III. Algorithms and Justification

An image is composed of pixels(a minute area of illumination on a display screen, one of many from which an image is composed). In fact, an image is a massive data set that has each pixel's information and it can be represented as a 3 dimensional matrix like figure 3. Therefore, in another words, an image is a collection of points in high dimensional spaces including 2-dimensional and 3-dimensional. Figure 4 represents the plot of an image that is processed into 2-dimensional coordinate system.

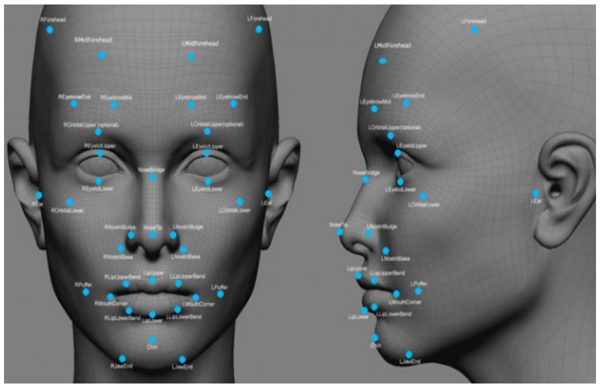


Fig. 2. Examples of facial recognition algorithm that is used by facebook

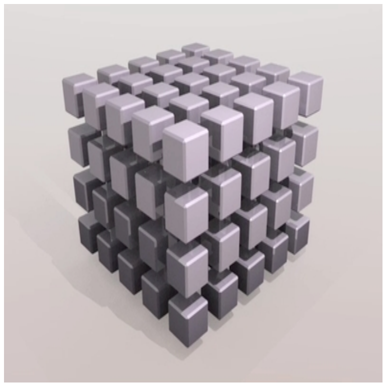


Fig. 3. Representation of Multi Dimensional Matrix(3D)

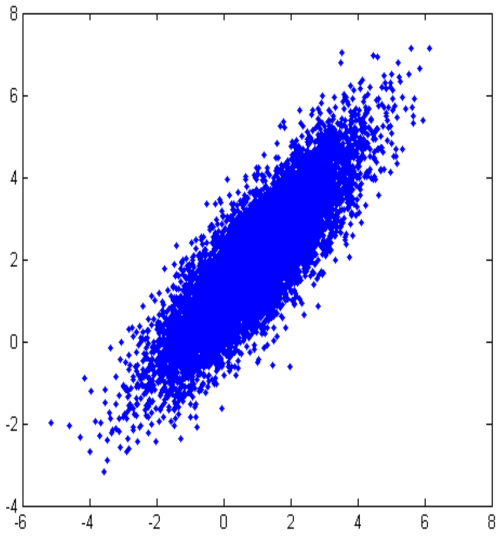


Fig. 4. A Plot of image representation on 2-dimensional coordinate system

A. Eigenfaces

Using the vectorized image (the image that was transformed into two dimensions from three dimensions), the mean of each individual image was found. The respective means were subtracted from each pixel in their respective images in order to find the mean centered data - this is the covariance matrix.

Then, when you take eigenvectors (a special set of vectors associated with a linear system of equations, figure 6) of the covariance matrix, each of the eigenvectors represent the direction of the data set. This process can be done by using Singular Value Decomposition(SVD) which is a factorization of matrix. Single Value Decomposition is a process of decomposing a matrix M with m by n dimension when m is greater than n. Then, M can be written as figure 5 where U is a set of eigenvectors of M transposed * M, S is a set of eigenvalues of M transposed * M, and V is a set of eigenvectors of M * M transposed. All eigenvalues and eigenvectors are from largest to smallest (i.e. first diagonal component of S is a largest eigenvalue of a dataset and first column of U is a corresponding eigenvector).

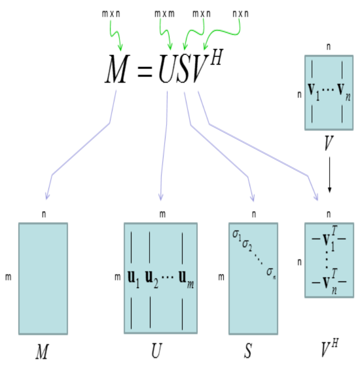


Fig. 5. Singular Value Decomposition of Matrix

After processing SVD to train data set, select desired amount of eigenvectors from U and compare with the test image and this analysis is called Principal Component Analysis, PCA.

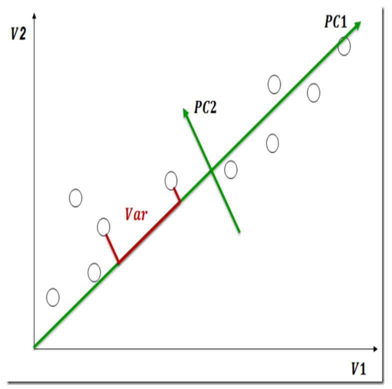


Fig. 6. Eigenvectors on Plotted Data Set. Eigenvectors are showing direction of data set. PC1 and PC2 represent first major eigenvector and second major eigenvector

The blurry face called eigenface can be created when several principal eigenvectors are used instead of all eigenvectors. The software compares the eigenfaces of all faces in train data and an eigenface of test data by measuring the distances between two images and returns the comparison result.

The comparison result of using Eigenfaces(figure 7) for its algorithm shows the test face (the image of face what software uses for facial recognition test) and the train face (the image of face what software uses for data base of people's faces). Then, it shows names that corresponding to each test face and train face. It also shows the accuracy of the software about its data-set. For the easy test data set(data set with faces that are easily recognizable by software), the software's accuracy was 1 which means it is always correct but for the hard test data set(data set with faces that are hardly recognizable by software), the software's accuracy was .825 which means it works for 82-83 faces out of 100 faces. Since the software's accuracy is not 1 for the hard data set, it often recognizes different person as same person like the case of figure 8.

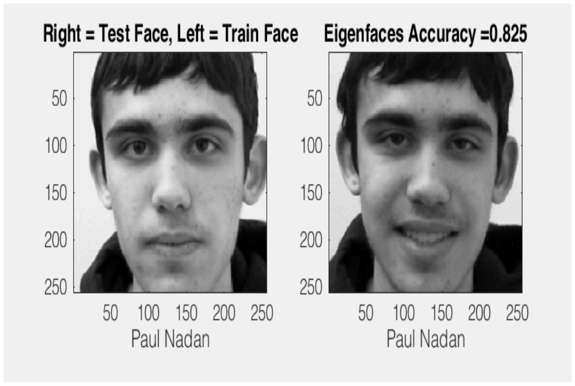

Right = Test Face, Left = Train Face
Eigenfaces Accuracy =0.825
50
100
150
200
250
50 100 150 200 250
Paul Nadan
Paul Nadan

Fig. 7. Result of Eigenface Detection

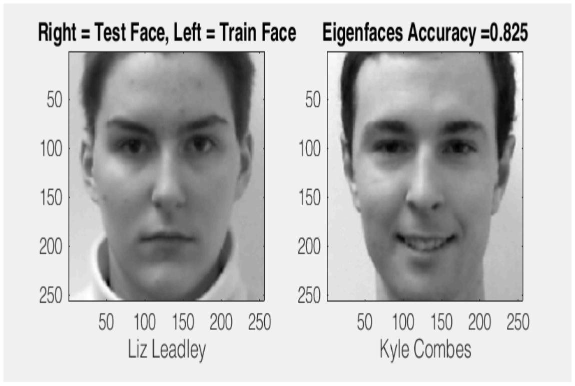

Right = Test Face, Left = Train Face
Eigenfaces Accuracy =0.825
50
100
150
200
250
50 100 150 200 250
Liz Leadley
Kyle Combes

Fig. 8. Result of Failed Eigenface Detection

B. Fisherfaces

Another algorithm, Fisherface (figure 9), uses Linear Discriminant Analysis(LCD), also known as Fisher's Linear Discriminant (FLD) instead of Principal Component Analysis (PCA). The difference between FLD and PCA is that FLD preserves discrimination of the data while PCA preserves the maximum variance. FLD find a projection of the reshape the scattered data set to make the data more easier to classify. The method selects 2 classes, which are between-class and within-class and FLD compares the ratio between two classes(figure 10).

Mathematically, scatter matrix is defined as:

$$S_i = \sum_{x_{k\,i}} (x_k - \mu_i)(x_k - \mu_i)^T$$

where S_i represents scattered matrix, and μ_i represents the mean value of the row of data matrix. This equation represent the matrix where each elements are mean centered (mean of the rows are subtracted) and then squared by multiplying transposed matrix. If the original matrix is

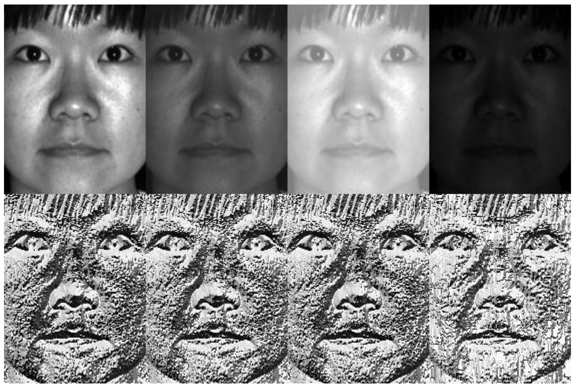


Fig. 9. Examples of Fisher faces

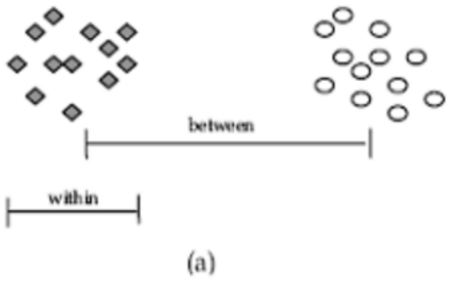


Fig. 10. Visualization of Class Separation

Then, between-class scatter matrix is mathematically defined as:

$$S_B = \sum_{x_{k\,i}}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

where N represents the number of rows, represents the mean value of the image and i represents the mean value of the row.

Within-class scatter matrix is defined as :

$$S_w = \sum_{i=1}^{c} S_i$$

where S_i represents the scatter matrix. By summing all elements within the scatter matrix, the within-class of the data set is calculated. Then, calculate the projection of each classes onto images. Projection of each classes are represented in figure 11. The distance of the projected line is called weight and represented as ω.

determined

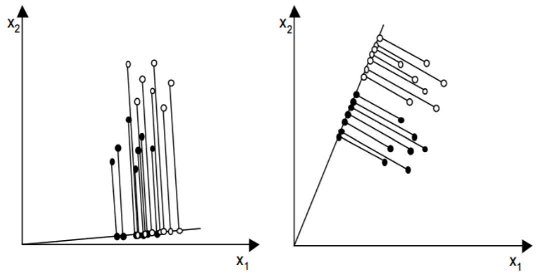


Fig. 11. Visualization of projection onto each data point

Projection line of class onto an image is defined as:

$$y_k = W^T x_k$$

Then, projected between class and within class are mathematically defined as:

$$\bar{S}_B = W^T S_B W$$

$$\bar{S}_W = W^T S_W W$$

and each of them represents the distance to projection line, y_k.

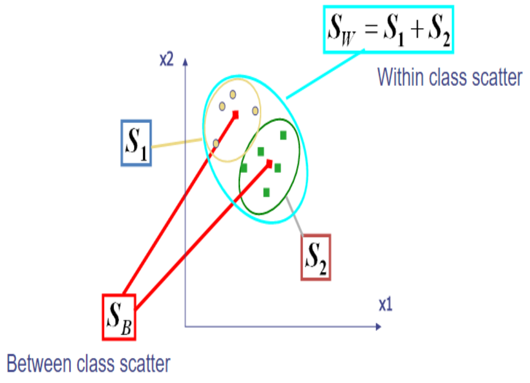


Fig. 12. Representation of projection of data. yk is represented as a red line

Then, the maximum value of the ratio of projected between class and projected within class has been calculated to find the optimal weight, represented as W_opt. W_opt is mathematically defined as:

$$W_{opt} = argmax\frac{\bar{S}_b}{\bar{S}_w}$$

Result can be determined by returning who has the maximum value of the ratio.

For eigenfaces MATLAB implementation, Principal Component Analysis has been used to compare train and test faces. First, both high dimensional train and test image data are vectorized into two dimensional. Then, covariance matrix has been calculated by subtracting mean from data and dividing mean subtracted data by square root of its length. Then, Singular Value Decomposition has conducted and certain number of eigenvectors of train (50 in this case) has been computed. Then, weight(projection of eigenvectors onto an image) of each faces has been calculated by multiplying 50 largest eigenvectors transposed by reshaped train images and test images. Then, distance between test image and train images has been calculated by using equation:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

While calculating the difference between weight of the test data and the train data, because there is only a single column of weight for the test data while train data has columns about its numbers of images, another matrix of train data's weight has been constructed by duplicating its column by number of images in train data set. Then, minimum distance has been computed since minimum distance between train data set and test data set means they are the most identical images. Then, index of the train data set has been computed by using MATLAB's built-in function and the image with corresponding index is shown as a result of facial recognition. In addition to displaying both recognized train image and test image, it also displays the name corresponding to both images and time elapsed for calculation. Accuracy is calculated by using MATLAB logic for name. The software runs for entire test data set and if both train name and test name match, it adds 1 to its accuracy value and if they do not match, it adds 0 to its accuracy value. After summing up all of data set's accuracy value, summed accuracy value is divided by number of images in test data set to display its accuracy rate. The software for eigenfaces showed 1 for accuracy of easy data set(easily recognizable face images) and showed .825 for accuracy of hard data set(face data set that is more difficult to recognize).

For fisherface that also is implemented by using MATLAB, all images are processed by using PCA. Both train images and test images's dimensions are reduced. Then, within-class and between-class scatter matrices for PCA coefficients. Then, optimum weight ratio has been computed by using argmax built-in function of MATLAB. Then, that weight is projected to FLD subspace by using equation of figure**??**. Then, minimum projection distance and its index has been computed and return as a result. Accuracy of the software is computed in the same way as it did in eigenfaces part.

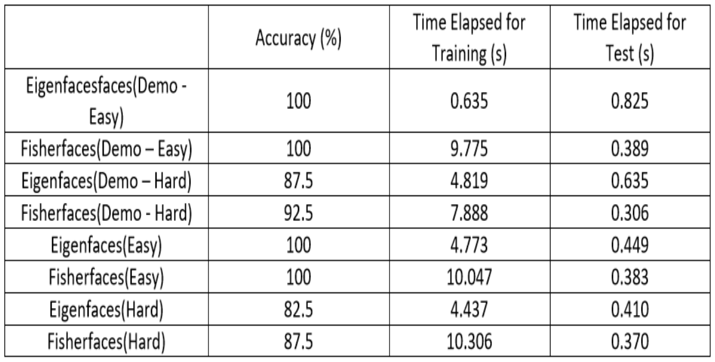
IV. Comparison of Performance

The MATLAB software's accuracy for using eigenfaces is .825 and for using fisherfaces is .875. Because fisherface consider about the image's discriminant, it has higher accuracy rate but takes longer to process. Even though eigenface method has lower accuracy rate than using fisherface, it has its own benefits. Using PCA preserves the most of variance of the images(e.g. eyeware, facial expression, light intensity) with a much more compact representation. It also require lower storage and its speed is faster. However, the matching is irritated a lot by background variation but in MATLAB implementation, background irritation is almost negligible since all test and train data sets have cropped background. Also, the direction of maximum variance might not be good for classification since data are all mixed up in one giant matrix.

Fisherface method separates image more clearly by using FLD therefore it has higher accuracy rate because it preserves the discrimination. However, since it does more processing and stores more variables, it takes longer to process and requires more storage.

V. Conclusions

There are many methods of building facial recognition software and most of them reduces its images dimension by using either Principal Component



	Accuracy (%)	Time Elapsed for Training (s)	Time Elapsed for Test (s)
Eigenfacesfaces(Demo - Easy)	100	0.635	0.825
Fisherfaces(Demo – Easy)	100	9.775	0.389
Eigenfaces(Demo – Hard)	87.5	4.819	0.635
Fisherfaces(Demo - Hard)	92.5	7.888	0.306
Eigenfaces(Easy)	100	4.773	0.449
Fisherfaces(Easy)	100	10.047	0.383
Eigenfaces(Hard)	82.5	4.437	0.410
Fisherfaces(Hard)	87.5	10.306	0.370

Fig. 13. Result of Tests for each algorithm

Analysis or Fisher Linear Discriminants. Each of methods have its pros and cons. Using fisher faces has higher accuracy rate but it requires more time to process and more storage while eigenfaces has lower accuracy rate with lower processing time and required amount of storage. Generally, facial recognition software works better at controlled environment because background image can significantly decrease the accuracy.

VI. Citations

Yang, Shuo, et al. ”From facial parts responses to face detection: A deep learning approach.” *Proceedings of the IEEE International Conference on Computer Vision. 2015. APA*

Turk, Matthew, and Alex Pentland. ”Eigenfaces for recognition.” *Journal of cognitive neuroscience 3.1 (1991): 71-86.*

Zhao, Wenyi, et al. ”Face recognition: A literature survey.” *ACM computing surveys (CSUR) 35.4 (2003): 399-458.*