Project 1: PCA and FLD for Analyzing Human Faces

Yufei Hu (404944367)

October 25, 2017

1 Objectives

Human face is a very important pattern and has been extensively studied in the past 30 years in vision, graphics, and human computer interaction, for tasks like face recognition, human identification, expression, animation etc. An important step for these tasks is to extract effective representations from the face images. The principal component analysis (PCA) is found to be a good representation for face.

A dataset of 177 faces is provided for this project and each image has 256 × 256 pixels. The face images are pre-processed so that the background and hair are removed and the faces have similar lighting conditions. Each face has several landmarks which are identified manually for convenience and they correspond across the dataset. In the second dataset, there are 87 points per image. These landmarks must be first aligned before we apply the PCA on the intensity. The alignment (geometry or time warping) is quite common in other biology patterns/data, for example, in speech, the speed of utterance is changing over time.

This project will compare two types of representations for dimension reduction: PCA -- a generative method and FLD -- a discriminative method.

2 Active Appearance Model for face reconstruction and synthesis

The 177 faces are divided into two parts: the first 150 faces are put in a training set, and the remaining 27 faces are put in a test set. In the following steps, I always use the training set to calculate the eigen-vectors and eigen-values, and calculate the reconstruction errors for the test images using the eigen-vectors from the training set.

2.1 Q1: Reconstruct faces using eigen-faces

Compute the mean and first k eigen-faces for the training images with no landmark alignment. Display the first K=20 eigen-faces and use them to reconstruct the remaining 27 test faces. Plot the total reconstruction error (squared intensity difference between the reconstructed images and their original ones) per pixel (i.e.

normalize the error by the pixel number, and average over the testing images) over the number of eigen-faces k.

Figure 1: Mean face of training faces without landmark alignment

The mean face is calculated by averaging all the training faces. An interesting phenomenon is that this mean face looks like both a female and a male face.

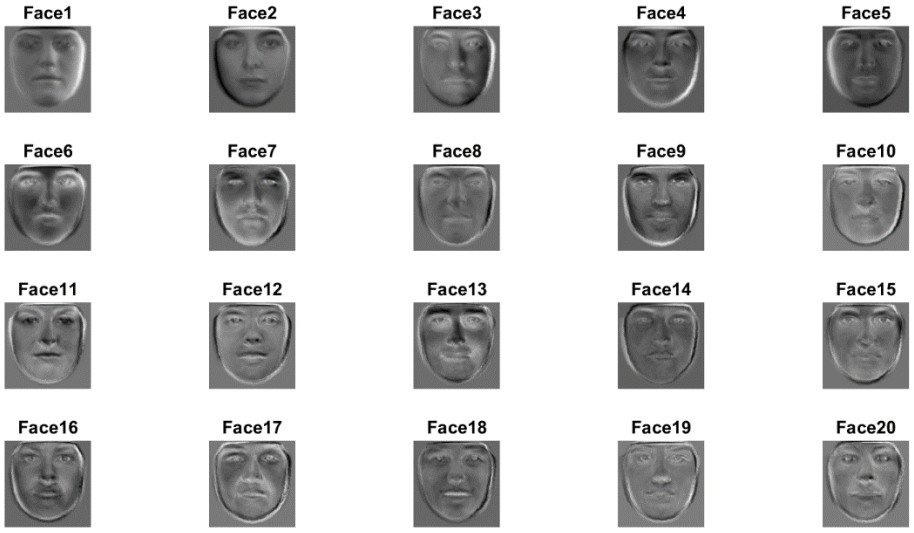

Figure 2: First 20 eigen-faces for the training images with no landmark alignment

To calculate the eigen-faces shown above, I firstly average all the training faces by subtracting the mean face. Then I call a MATLAB function *pca()* to derive all the eigenvectors and eigenvalues. The top 20 eigenvectors are selected and reshaped to be the eigen-faces.

Figure 3: Original and reconstructed testing faces using eigen-faces

Then all 20 eigen-faces are used to reconstruct the remaining 27 testing faces. The projection of testing faces on each eigenface is described as:

$$< I - m,\ e_i > \ = \ \mathrm{b}_i,\ \mathrm{i} \ = \ 1,...,\mathrm{k} \tag{1}$$

where I is training faces, m is the mean face, e_i is the i^{th} eigen-face, and b_i is the calculated projection. Then the following equation is used to reconstruct all the testing faces:

$$\hat{I} = m + \sum_{i=1}^{k} b_i e_i \tag{2}$$

All the reconstructed testing faces are shown in Figure 3 together with the original testing faces so we can have a more straight-forward idea of how different they are. As shown clearly, the reconstructed faces seem to be a little blurred and lack appearance details. This is because we are only using 20 eigen-faces here to do the reconstruction. Some detailed information is lost during reconstruction. However, the general appearance information is reconstructed quite well.

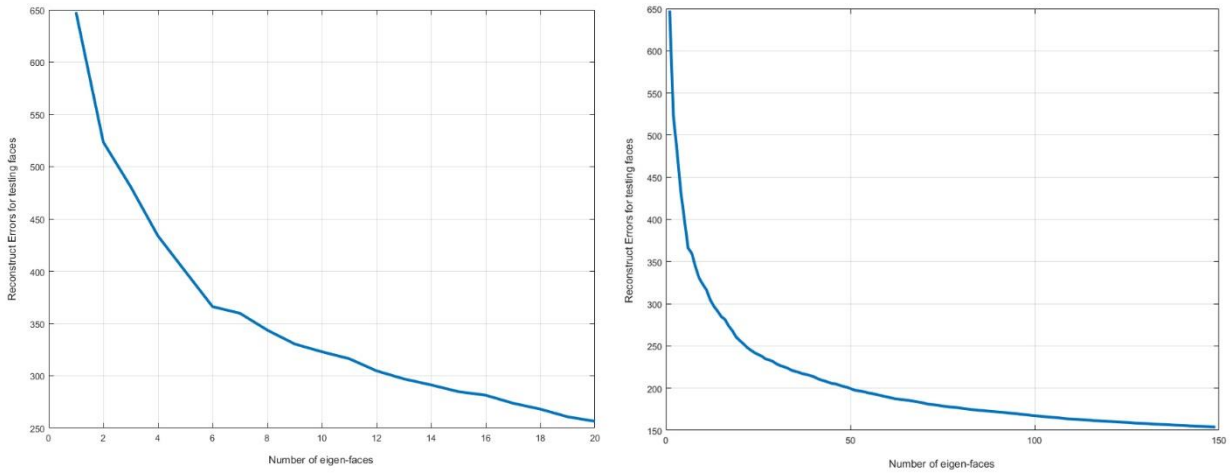

Figure 4: Reconstruction errors

Then reconstruction errors are calculated based on the equation below:

$$error = \frac{1}{n}\sum_{k=1}^{n}[\frac{1}{N^2}\sum_{i=1}^{N^2}(\hat{x}_{ik} - x_{ik})^2] \quad (3)$$

where n is the total number of testing images, N is the image width/height, and x_{ik} is the i^{th} pixel in k^{th} image.

As shown clearly in the figure above, the error decreases when the number of eigen-faces used for reconstruction increases. The reason is quite straight-forward as more eigen-faces are used, more detailed information is captured during reconstruction. But even when all eigen-faces are used, the error can still not be zero as the eigen-faces are only derived from training faces so these testing faces could be totally new to those eigen-faces, thus error is introduced inevitably.

2.2 Q2: Reconstruct landmarks using eigen-warping

Compute the mean and first k-eigen-warping of the landmarks for the training faces. Here warping means displacement of points on the face images. Display the first 5 eigen warping and use them to reconstruct the landmarks for the testing faces. Plot the reconstruction error (in terms of distance) over the number of eigen-warping k.

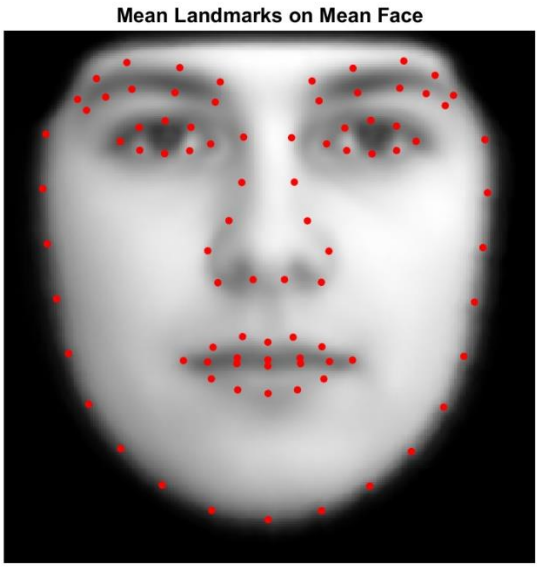

Figure 5: Mean landmarks on mean face

Like Question 1, the mean landmark is calculated by averaging all the training landmarks.

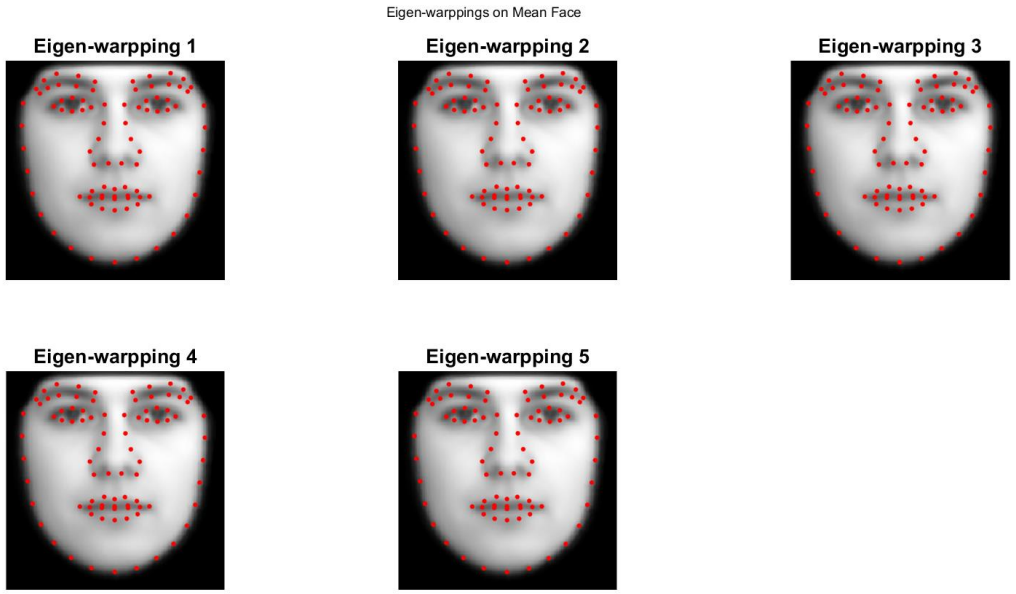

Figure 6: Top 5 eigen-warping on the mean face

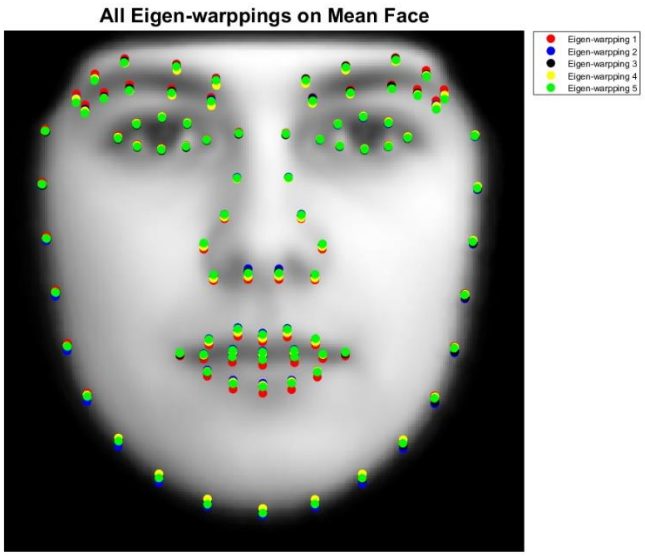

Figure 7: All 5 eigen-warping on one single mean face

Shown in Figure 6, I have plotted all the top 5 eigen-warping on one mean face. However, the difference between these 5 eigen-warping is so small that we can hardly detect the difference by our eyes. Then I multiply all eigen-warping with the square root of their own eigenvalue to differentiate and these new eigen-warping are shown in Figure 7 on one single mean face. However, the difference is still very small as these 5 eigen-warping are almost overlapping with each other.

Figure 8: Reconstructed landmarks shown on the original testing faces

Then all 5 eigen-warping are used to reconstruct the remaining 27 testing landmarks. The projection of testing landmarks on each eigen-warping is described as:

$$< X - m_L,\ e_{Li} > \ = \ \mathrm{b}_{Li},\ \mathrm{i} \ = \ 1,...,\mathrm{k} \tag{4}$$

where X is training landmarks, m_L is the mean landmark, e_{Li} is the i^{th} eigen-warping, and b_{Li} is the calculated projection. Then the following equation is used to reconstruct all the testing landmarks:

$$\hat{X} = m_L + \sum_{i=1}^{k} b_{Li} e_{Li} \tag{5}$$

This entire process is quite close to the one in Question 1. All the reconstructed testing landmarks are shown in Figure 8 together with the original testing landmarks. The reconstructed landmarks almost overlap with the original landmarks, indicating that the top 5 eigen-warping has provided enough information for the reconstruction.

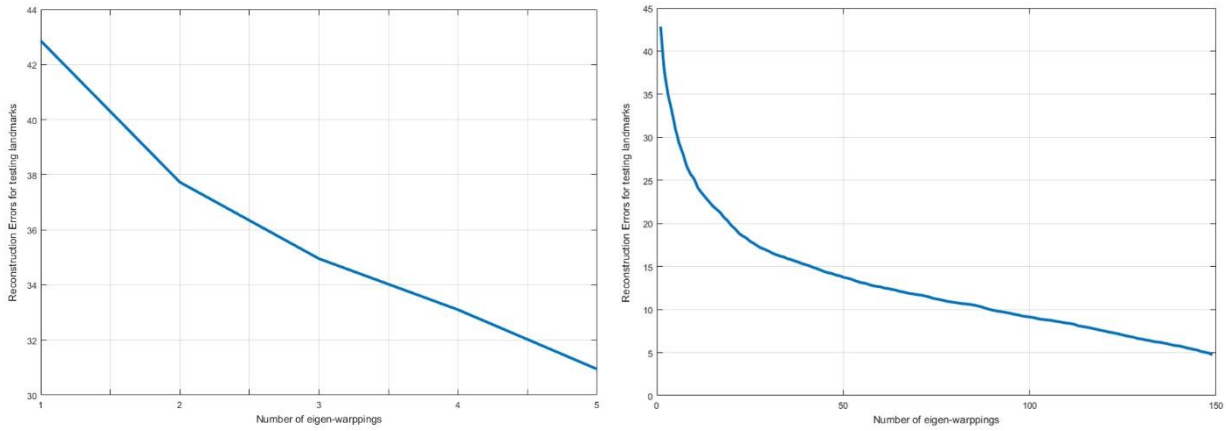

Figure 9: Reconstruction errors

Then reconstruction errors are calculated based on the equation below:

$$error = \frac{1}{n} \sum_{k=1}^{n} \sqrt{\sum_{i=1}^{N} (\hat{x}_{Lik} - x_{Lik})^2 + \sum_{i=1}^{N} (\hat{y}_{Lik} - y_{Lik})^2} \tag{6}$$

where n is the total number of testing landmarks, N is the number of landmarks, and x_{Lik} and y_{Lik} are the i^{th} landmark position in k^{th} testing image.

As shown in the figure above, the error decreases when the number of eigen-warping used for reconstruction increases. The reason is the same to that in Question 1.

2.3 Q3: Reconstruct faces with landmark alignment

Combine the two steps above. The objective is to reconstruct images based on top 10 eigen-vectors for the warping and then top k (say 10) eigen-vectors for the appearance, in the context of compressing the face images and communicate through a network with small number of bits. For the training images, we first align the images by warping their landmarks into the mean position (interpolation between landmarks is needed), and then compute the eigen-faces (appearance) from these aligned images. For each testing face: (i) project its landmarks to the top 10 eigen-warping and the reconstructed landmarks are derived; (ii) warp the face image to the mean position and then project to the top k (say k=10) eigen-faces, the reconstructed image at mean position is derived; (iii) Warp the reconstructed faces in step (ii) to the positions reconstructed in step (i). Note that this new image is constructed from 20 numbers. Then compare the reconstructed faces against the original testing images. (iv) Plot the reconstruction errors per pixel against the number of eigen-faces k.

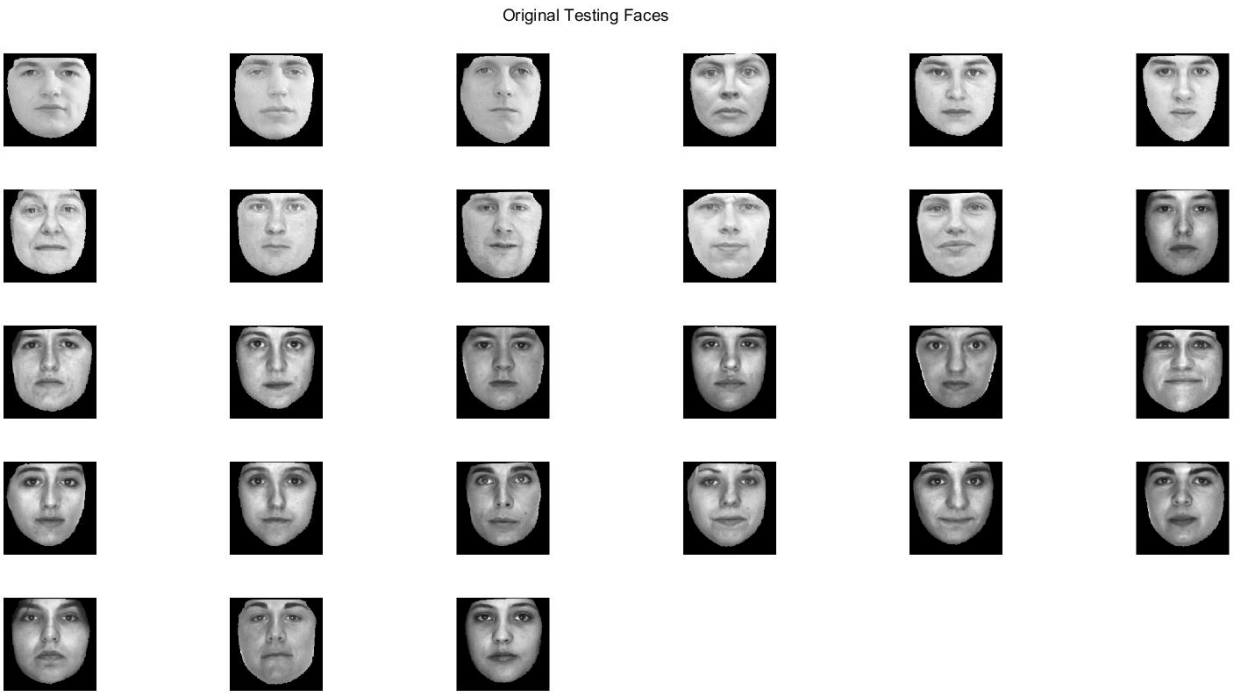

Figure 10: Original and reconstructed testing faces using eigen-faces and eigen-warping

The entire process of calculating eigen-faces and eigen-warping and reconstruction are the same to that in Question 1 and 2. Both the original and reconstructed testing faces are shown in the figure above. By comparing to those reconstructed faces in Question 1, it is clear to see that these faces are more accurate in their geometric details. This is done by using eigen-warping for reconstructing the geometric information. Still it is amazing to see that the original image could be compressed to a representation containing merely 20 numbers.

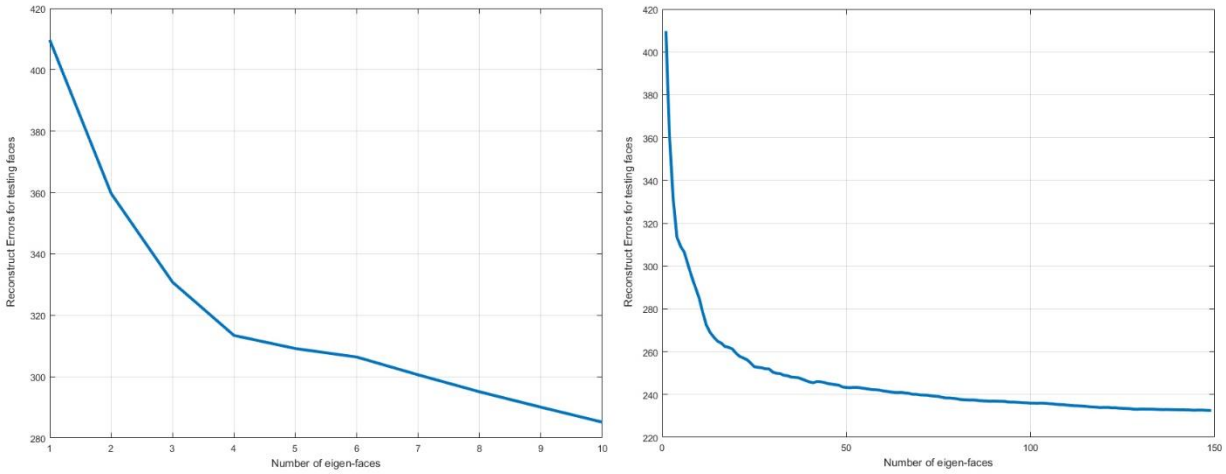

Figure 11: Reconstruction errors

Similarly, the errors are derived based on equation 3 above after aligning the reconstructed images using eigen-faces to the reconstructed landmarks using eigen-warping. By comparing Figure 4 with Figure 11, it is seen that the error decreases a lot by almost 30% overall at the beginning. The reason is due to the extra step of using eigen-warping for reconstructing the landmarks.

2.4 Q4: Synthesize random faces by randomly sampling landmarks and appearance

Synthesize random faces by a random sampling of the landmarks (based on the top 10 eigen-values and eigen-vectors in the wrapping analysis) and a random sampling of the appearance (based on the top 10 eigen-values and eigen-vectors in the intensity analysis). Display 20 synthesized face images.

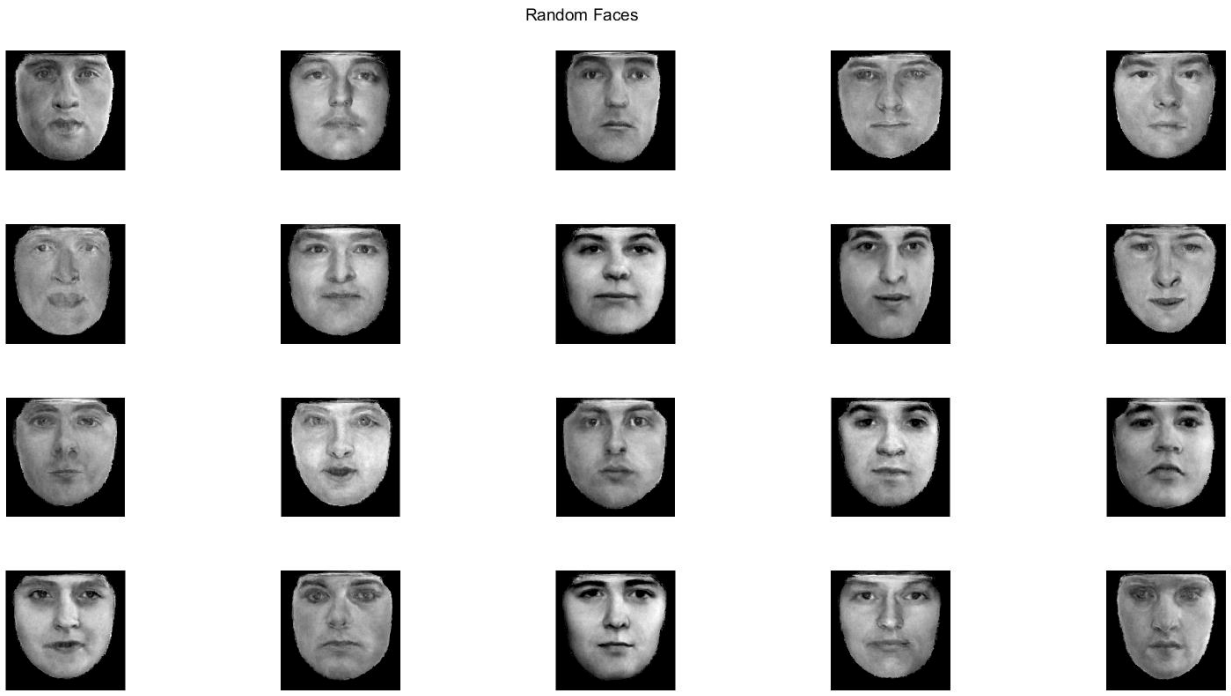

Figure 12: Synthesized random faces

By following the same process described above, eigen-faces and eigen-warping are derived. Then I randomly generate some numbers following normal distribution with zero mean and a variance of the square root of corresponding eigen-values. The results are then shown above. It is interesting to see that some faces are actually quite close to a real face we can imagine.

3 Fisher Linear Discriminant for gender discrimination

We have divided the 177 faces into male (88) and female faces (85), plus 4 unknown (which is hard even for human eyes to tell based on the faces alone). Then 10 male and 10 female images are chosen as the testing set. The remaining faces (except the 4 unknown) will be used as training sets.

3.1 Q5: Use FLD to distinguish male from female

Find the FLD or Fisher face that distinguishes male from female using the training sets, and then test it on the 20 testing faces and see how much percentage is right. This Fisher face mixes both geometry and appearance difference between male and female.

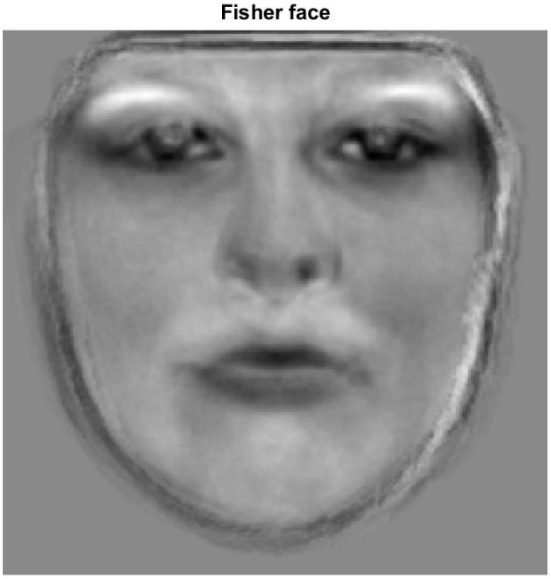

Figure 13: Fisher face for distinguishing male from female faces

The equation below is used for calculating the Fisher face for classification:

$$S_w w = m_F - m_M \tag{7}$$

where S_w is the with-in class scatter matrix defined as:

$$S_w = \sum_{x \in F} (x - m_F)(x - m_F)^T + \sum_{x \in M} (x - m_M)(x - m_M)^T \tag{8}$$

and m_F and m_M are the mean face of female faces and male faces respectively. w in equation 7 is exactly the Fisher face we want to derive.

However, noticing the fact the dimensionality of S_w is as big as $N^2 \times N^2$, it is impractical to directly calculate w as the computer will throw out a memory warning. My method of deriving the w is firstly

reducing the dimension of original scatter matrix by applying a PCA. Particularly in this case, I chose to reduce the dimension to 20. Then using the projected points on eigen-faces, I derive a new Scatter matrix based on these projection points with a dimension of 20 × 20. A w in the new dimension is calculated. Then I multiply this w with 20 eigen-faces I picked previously to finally get the w I want. Though this w has some information loss problem due to PCA, later you will see it still performs quite well when doing classification.

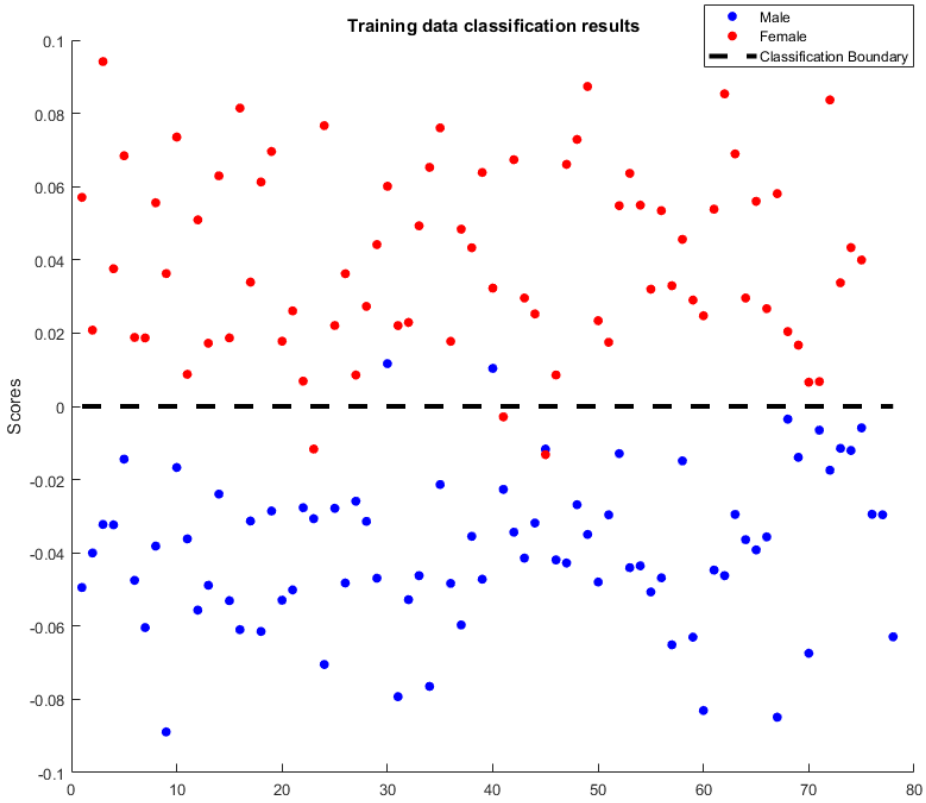

Figure 14: Classifying training data using Fisher face

During testing, the following equation is used for classifying faces to be male or female:

$$w^T x + w_0 = \begin{cases} < 0 & \text{classify as male} \\ \geq 0 & \text{classify as female} \end{cases} \quad (9)$$

As illustrated in the figure above clearly, when I choose w_0 to be 0, the classification result is already quite good. After running some experiments, I found setting the w_0 to be 0 is the optimal case for classification. The training accuracy is 96.7%.

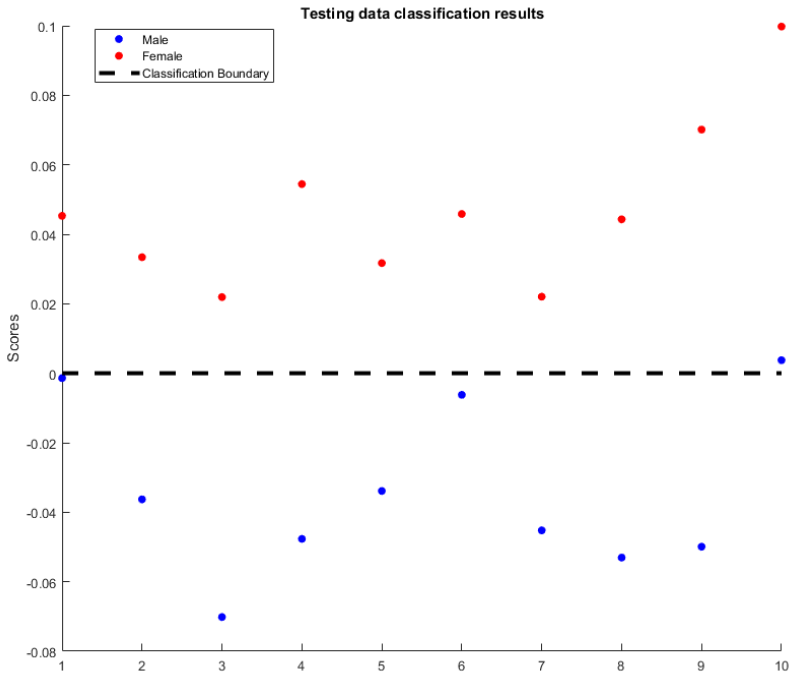

Figure 15: Classifying training data using Fisher face

Then I classify all the testing faces and the accuracy is as high as 95%. Only one male face is classified wrongly. I found this particular male face is even hard for me to tell if it is a male face or female face, so it is quite reasonable for the classifier to make this only mistake.

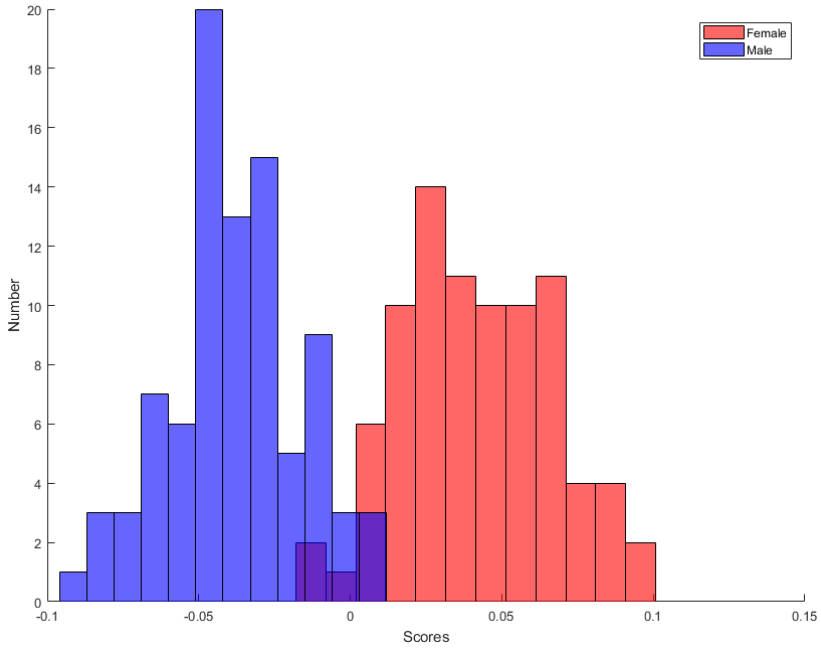

Figure 16: Histogram of classification scores for all the data

Finally, to have a more intuitive look at the classification performance using Fisher face, I have plotted the histogram showing the distribution of the classification scores for all the data. As shown in Figure 16, there is a clear boundary somewhere near the score of 0, which is exactly the classification boundary I have chosen for this case.

3.2 Q6: Project faces to a 2D-feature space

Compute the Fisher face for the key point (geometric shape) and Fisher face for the appearance (after aligning them to the mean position) respectively, and thus each face is projected to a 2D-feature space, and visualize how separable these points are.

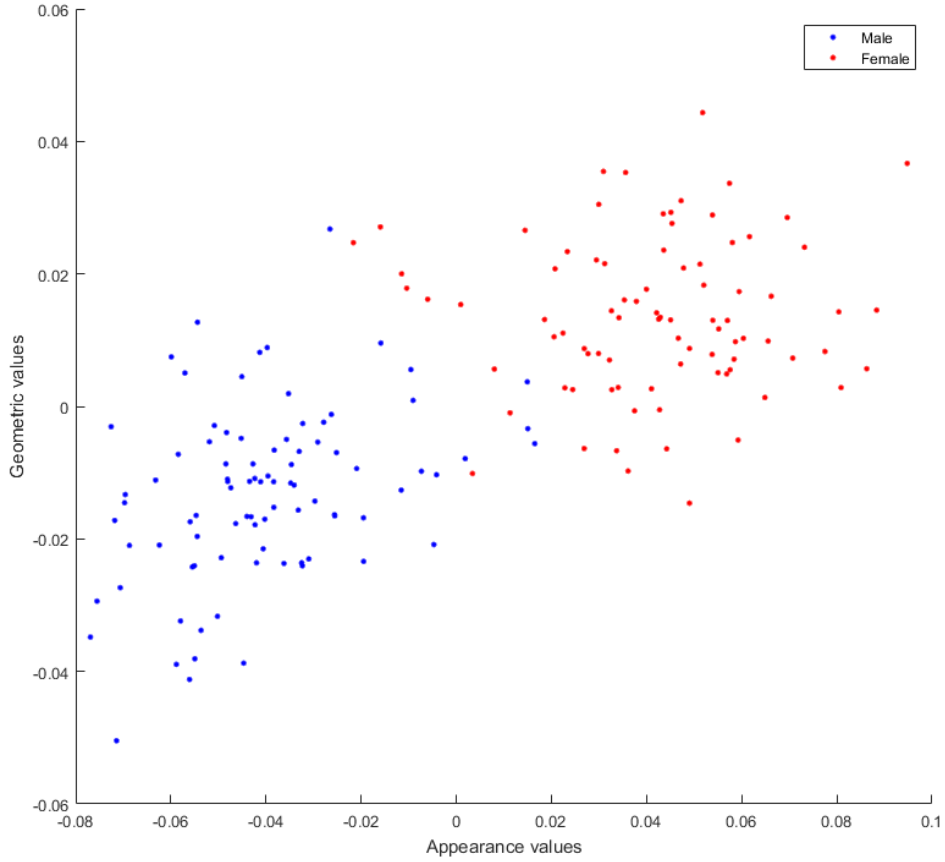

Figure 17: Projection of all faces onto 2D-feature space

Following the same process mentioned above, after deriving 20 eigen-faces and 20 eigen-warping, I derive two Fisher vectors for appearance and geometry respectively. Then I calculate the projection of each face onto these two Fisher vectors. The 2D projection map is shown in the Figure 17. It is seen that the points are well divided, providing a clear classification boundary. Therefore it is reasonable to have such a high classification accuracy on both training and testing dataset.