Project 1: PCA, Autoencoder and FLD for Analyzing Human Faces

Xiaohan Wang

Part 1: ASM and AAM model for face recognition

PCA: a linear method

1. Compute the mean and first K = 50 eigen-faces for the training images with no landmark alignment and use them to reconstruct the remaining 200 test faces.
   a) Compute the mean face of all training images
   For each image in training set, first, I convert RGB image to HSV image. Then I extract only channel V (128 x 128). Finally, add all training images' channel V and calculate the mean values. Figure 1 shows the mean face. It's very blurred and we cannot tell the character's gender from the image. It's intuitive because there are men and women in the dataset, and the mean face is supposed to have characteristics from both.

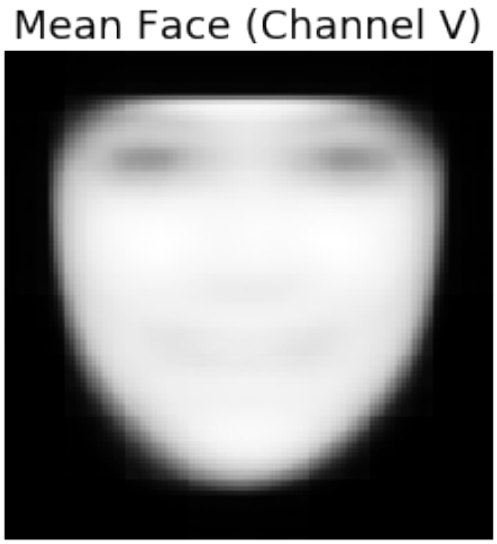


Figure 1. mean face of the training images

   b) Compute eigen-faces
   First, we need to reshape each image as a (128*128) x 1 vector $I_i, i = 1, \dots, 800$.
   Then we can centralize each vector for PCA,

$$\phi_i = I_i - \psi$$

   Next, we can calculate the eigen vectors of the square of covariance matrix

$$C = \sum_{i=1}^{800} \phi_i {\phi_i}^T = AA^T$$

But here AA^T has shape of 16324 x 16324, which is too big to solve. So, we can apply a trick as to do PCA of A^TA. Let μ_i be one eigen value, u_i be one eigen vector of AA^T, and v_i be one eigen vector of A^TA. Then we have

$$A^TAv_i = \mu_i v_i$$
$$AA^TAv_i = \mu_i Av_i$$
$$CAv_i = \mu_i Av_i$$

Let $u_i = Av_i$, then we have

$$Cu_i = \mu_i u_i$$

So A^TA and AA^T have the same eigen value, and $u_i = Av_i$.

In this question, we only extract first 50 eigen-faces, and plot first 10 eigen-faces as below:

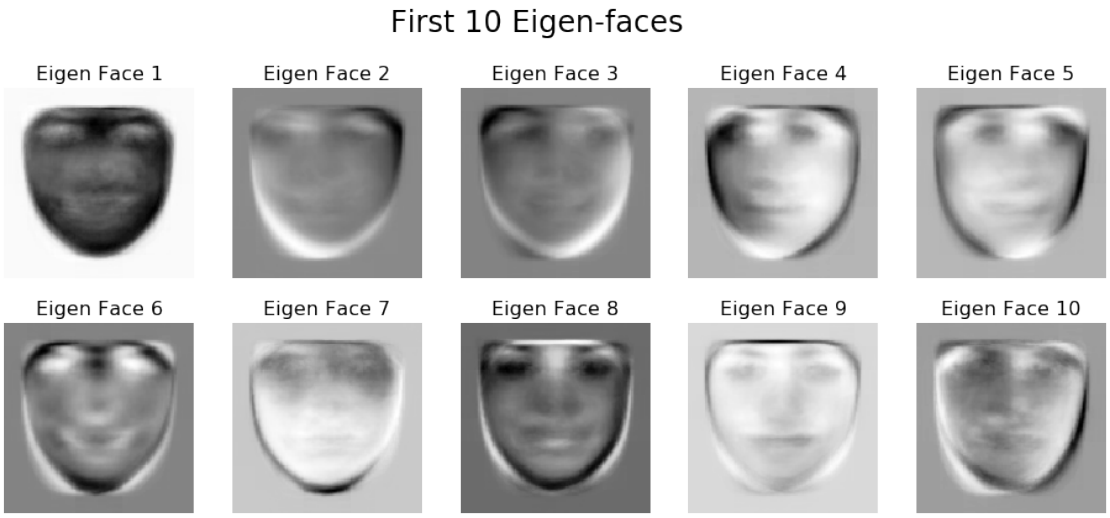


Figure 2. first 10 eigen-faces

c) Reconstruct faces of testing images

First, we also need to convert test images from RGB to HSV and extract channel V. Then centralize test images with mean face of training images. Next, we can project the test images into space of 50 eigen-faces to get coefficient matrix

$$B = Test^T \cdot Eigen$$

Here, $Test$ is 16324 x 200, $Eigen$ is 16324 x 50, B is 200 x 50.

Then, we can reconstruct the images with eigen-faces, coefficient matrix and mean face

$$\mathrm{R} = Eigen \cdot B^T + M$$

Below are first 10 reconstructed faces and the corresponding original faces:

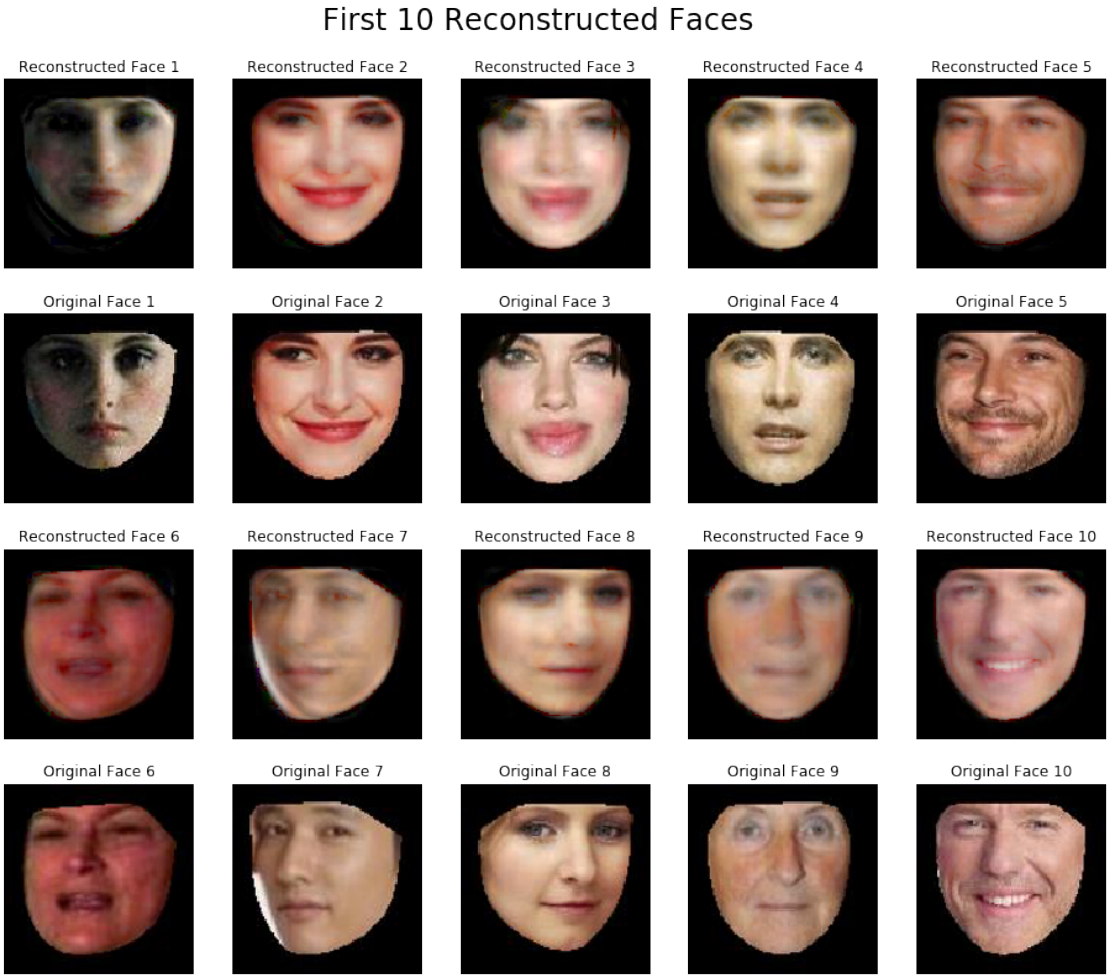


Figure 3. first 10 reconstructed faces

From the results above, we can find that the reconstructed images are similar to the original ones, but more blurred. If we can use more eigen-faces, the reconstruction maybe better.

d) Calculate total reconstruction error

The total reconstruction error can be calculated as the following equation

$$error = \frac{1}{200}\sum_{i=1}^{200}\frac{1}{128*128}\sum_{j=1}^{128*128}(V_{ij} - \widetilde{V_{ij}})^2$$

Below is the figure of reconstruction error with different number of eigen-faces:

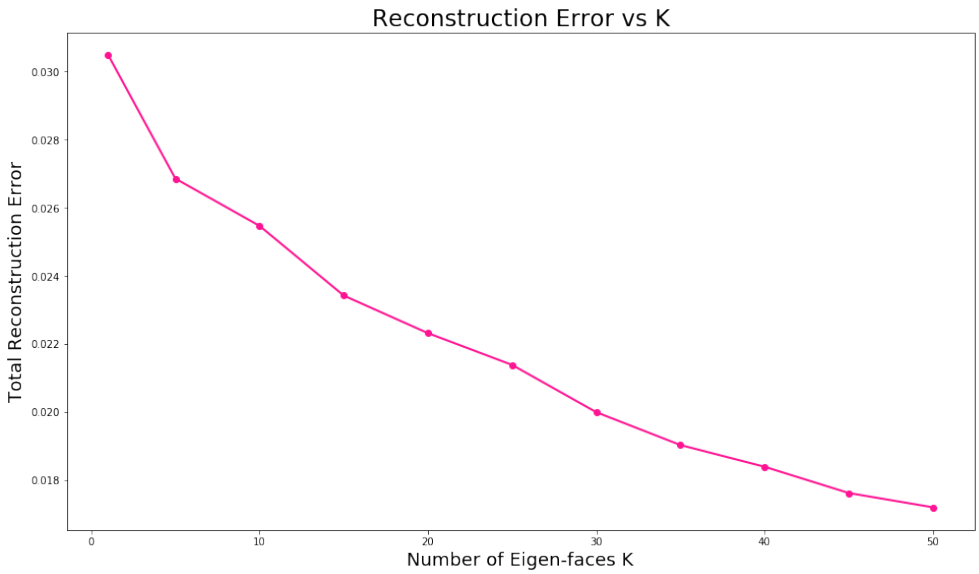


Figure 4. reconstruction error over number of eigen-faces K

From the above figure, we can find that the errors are decreasing with K increases, because with more components (eigen-faces), we can use more information to reconstruct our images, so that with higher accuracies.

2. Compute the mean and first K=50 eigen-warping of the landmarks for the training faces.
   a) Compute mean landmark of all training data

   With the same method as that in last part, we can read landmarks of all training data and add corresponding coordinates together to get the mean landmark. Figure 5 is the mean landmark on mean face. We can find that the landmark almost fits the outline of the mean face.

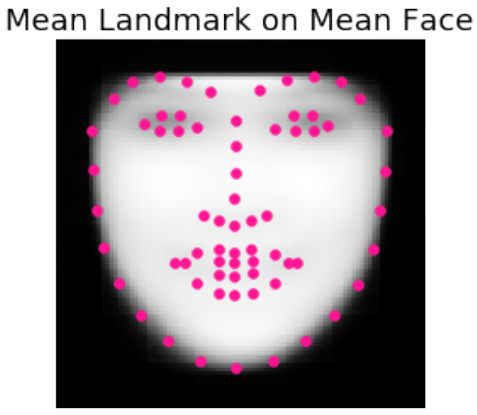


Figure 5. mean landmark on mean face

b) Compute first 10 eigen-warpings

First, we need to centralize the original training landmarks. Then, we can reshape landmarks of the training data into vectors. Next, concatenate those vectors into matrix to do PCA and get eigen-warpings. Below are first 10 eigen-warpings:

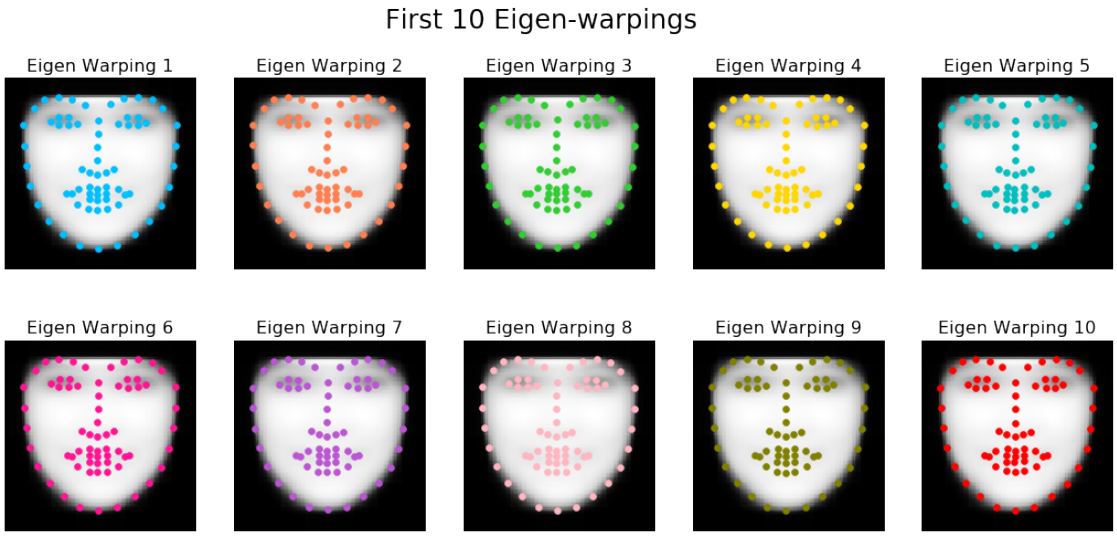


Figure 6. first 10 eigen-warpings

In the above figure, the 10 eigen-warpings are very similar to each other, and we cannot tell the difference. Then I plot them into one figure:

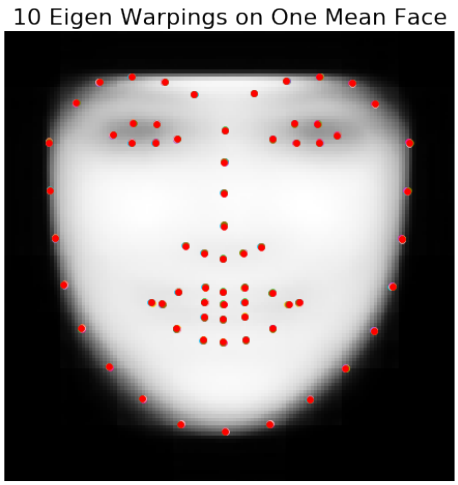


Figure 7. 10 eigen-warpings in one figure

In the above figure, we can see some slight difference among those eigen-warpings. But the discrepancies are very small, and they are almost same.

c) Reconstruct 10 landmarks

With the same strategy in part 1, we can first project centralized test landmarks to eigen-warpings to get the coefficient matrix. Then multiply the coefficient and eigen-warpings and add the mean landmarks to reconstruct the original ones. Below are the first 10 reconstructed landmarks:

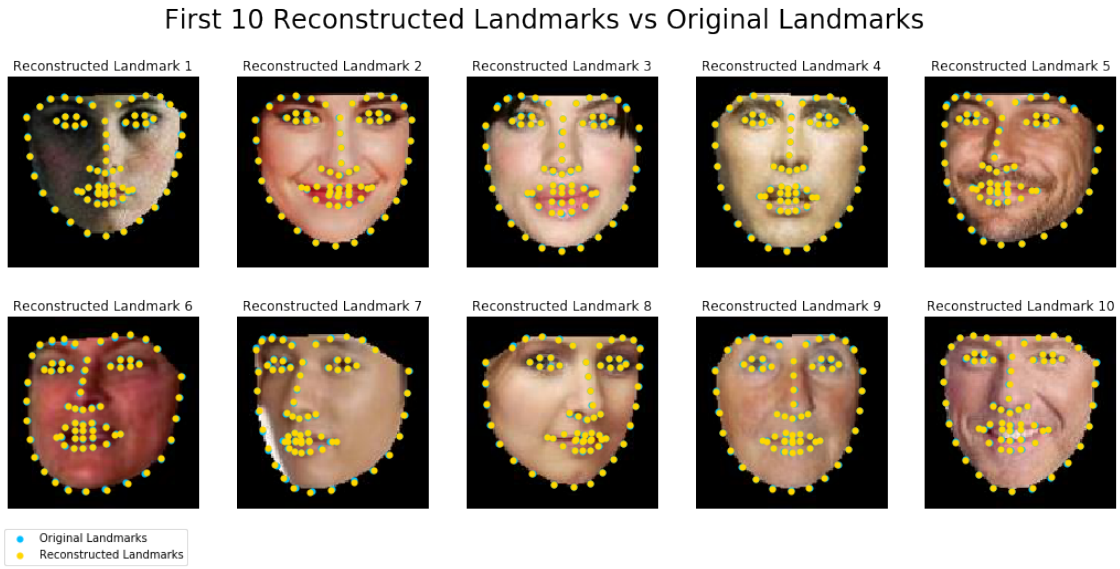


Figure 8. compare first 10 reconstructed landmarks with the original ones

In figure 8, the yellow points are reconstructed landmarks and the blue ones are the original landmarks. The blue ones are almost overlapped by the yellow points, which indicates that the reconstructed landmarks are accurate within the scope.

d) Calculate reconstruction error

We can use the distance between reconstructed point and the original point to evaluate the effect:

$$error = \frac{1}{200}\sum_{i=1}^{200}\frac{1}{68}\sum_{j=1}^{68}\sqrt{(X_{ij}-\widetilde{X_{ij}})^2+(Y_{ij}-\widetilde{Y_{ij}})^2}$$

The reconstruction error line is as below:

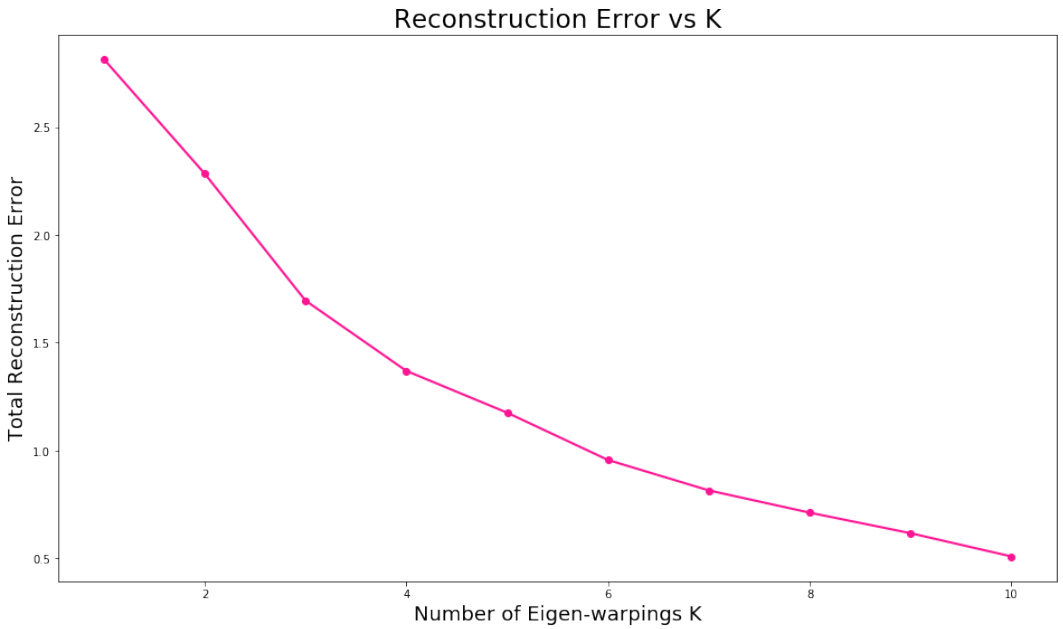


Figure 9. reconstruction error over number of eigen-warpings

From the above figure, we can find that as the number of eigen-warpings increases, the error decreases greatly. The error mainly distributes from 0.5 to 2.7. It indicates that with more eigen-warpings, there will be less loss of the features, so we can get more dimensions or information to reconstruct the landmarks.

3. Reconstruct images based on top 10 eigen-warpings and top 50 eigen-faces
   a) Align training images to mean position

   The first step is to warp each training image from the original landmarks to mean landmarks. Then convert from RGB to HSV format and add together channel V. Then we can calculate the mean aligned face. With warping, the mean face is clearer than that without warping.

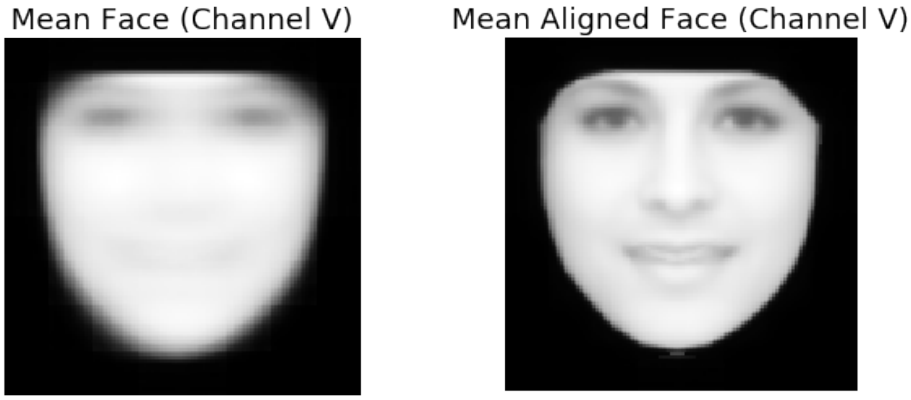


Figure 10. compare mean face (left: without warping, right: with warping)

b) Compute eigen-faces of aligned faces
With the same method in question 1, we can calculate eigen-vectors of aligned faces using PCA. Similarly, the eigen-faces are clearer than those without warping.

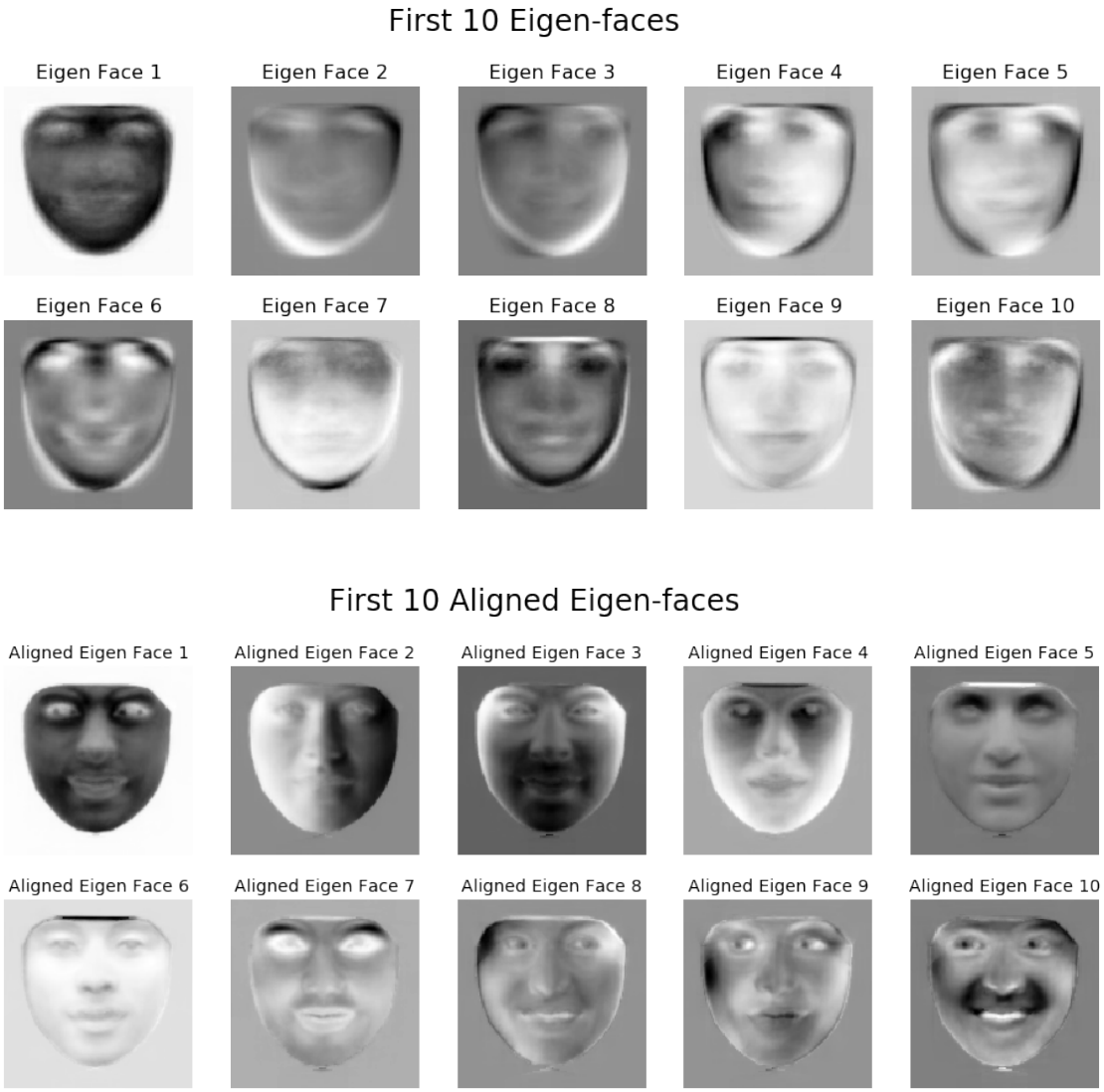
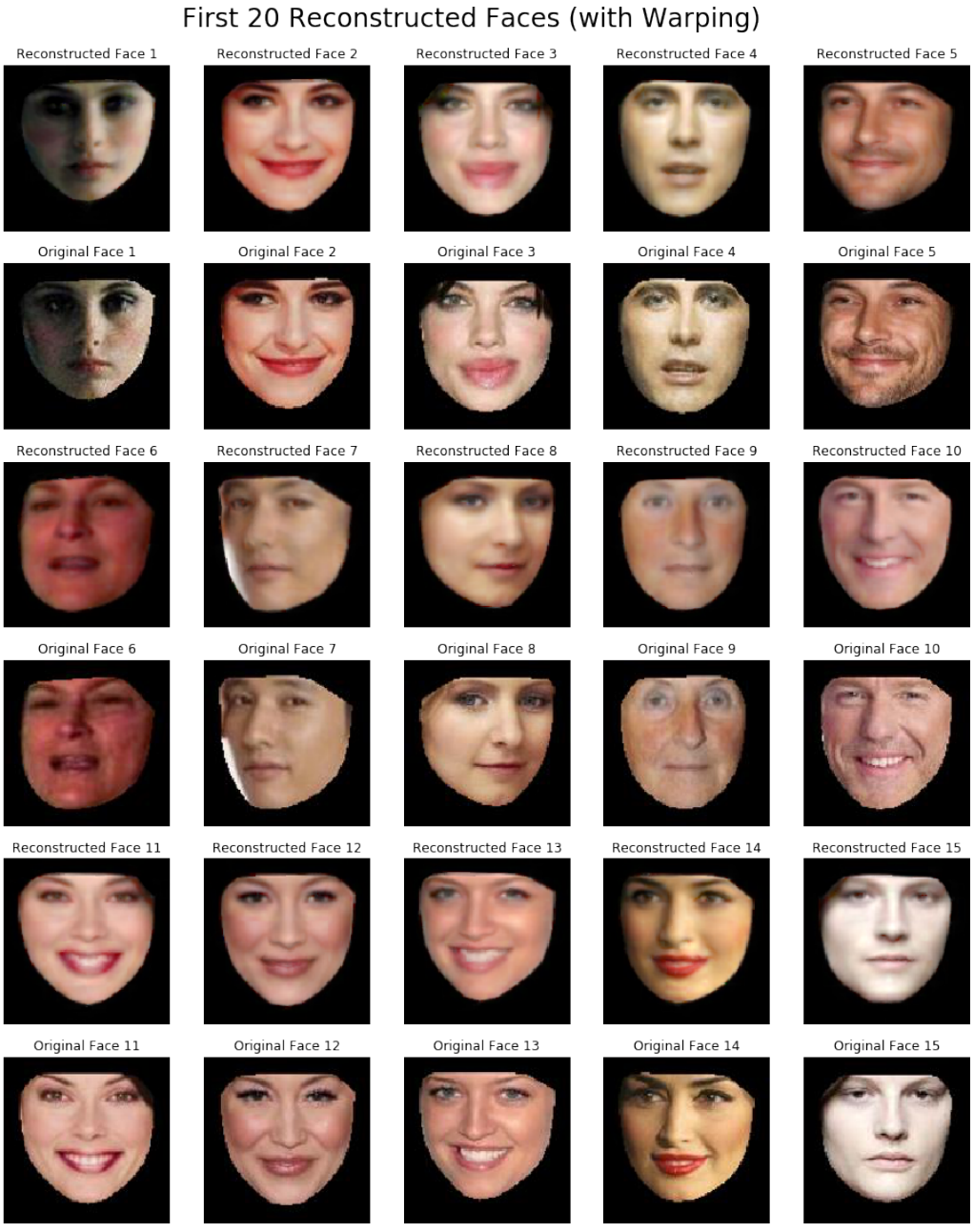


Figure 11. compare eigen-faces (up: without warping, down: with warping)

c) Reconstruct landmarks of all test landmarks. This step is the same as question 2.
d) Warp all test images to mean position
e) Reconstruct test faces at mean position
We can use the same strategy as the previous part that to project the warped test images to eigen-faces to get coefficient matrix. Then multiply coefficient matrix with eigen-faces, and add the mean aligned face to get the reconstructed test faces.

f) Warp reconstructed test faces to reconstructed landmarks

The last step is to warp the reconstructed test faces back to the reconstructed landmarks. Then plot the first 20 reconstructed faces. They are also clearer than those without warping.



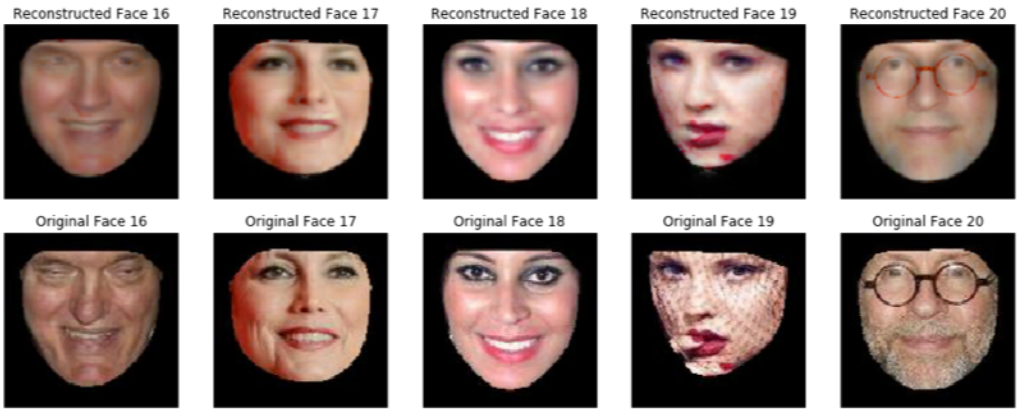


Figure 12. first 20 reconstructed faces with warping operation

g) Calculate reconstruction error

The total reconstruction error over different number of eigen-faces K can be plotted as below:

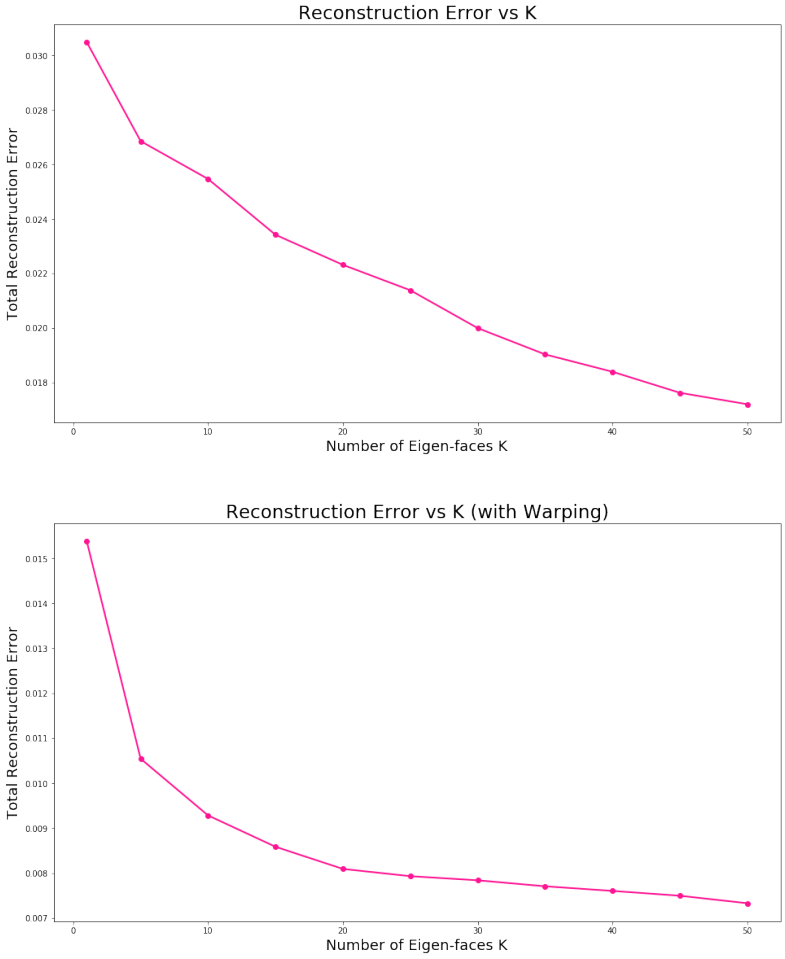


Figure 13. compare reconstruction error (up: without warping, down: with warping)

From the above figures, we can find that with warping operation, the total reconstruction error decreases clearly. It indicates that it's meaningful and sensible to first warp the images to the same landmarks, then process them with PCA. Finally, to warp back to their own landmarks.

4. Synthesize random faces by randomly sampling of the landmarks and appearance
   First, we can get two bases of aligned faces and landmarks. Then, we will randomly generate some coefficients by normal distribution (multiplying the square root of eigen values). Next, we can linearly calculate the reconstructed faces at the mean position. Finally, to warp the faces from mean landmarks to their own randomly generated landmarks. Below are the 50 randomly synthesized faces:

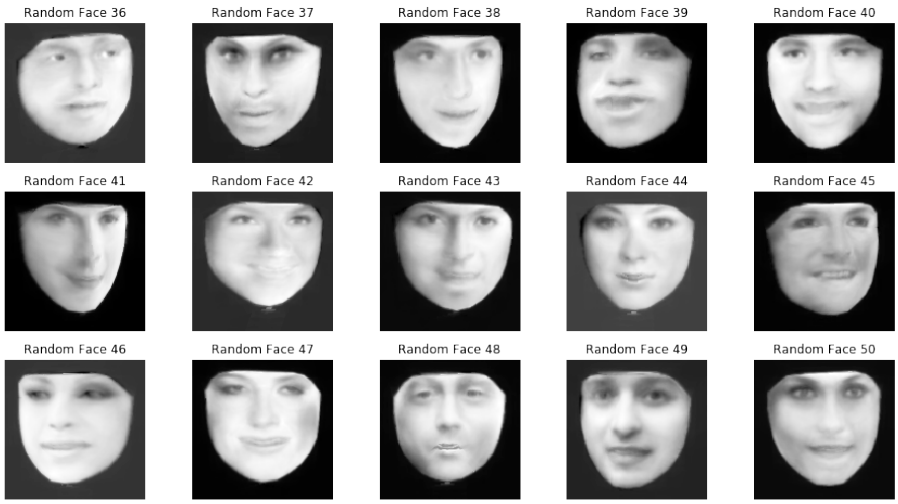


Figure 14. 50 randomly synthesized faces

Autoencoder: a non-linear method

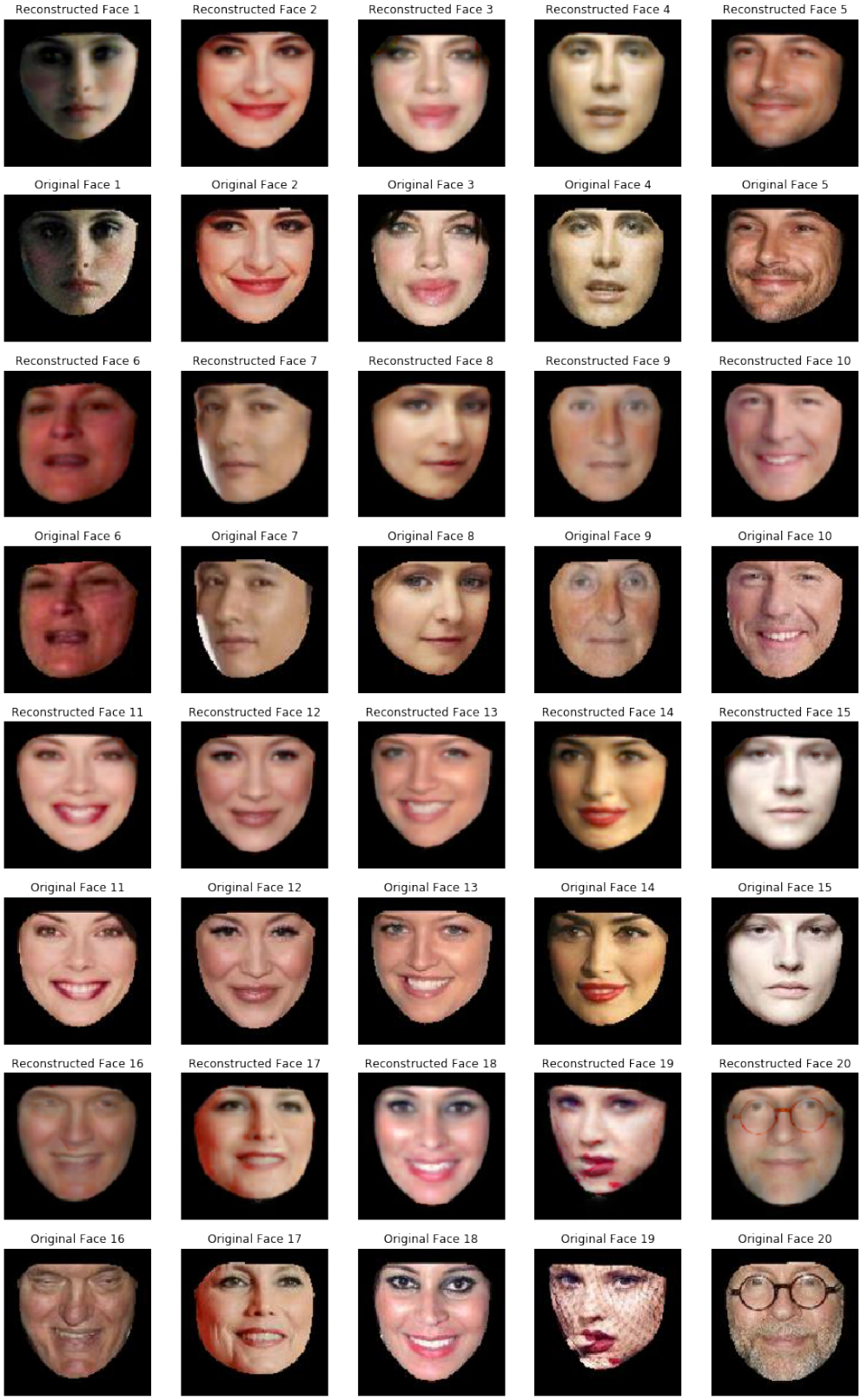
1. Re-perform experiment 3 in 2.1 by replacing PCA with Autoencoder

   We can perform the reconstruction by following steps:

   a) Read original faces and landmarks
   b) Calculate mean landmark of training landmarks
   c) Warp training faces and testing faces to mean positions, and then create DataLoader and Autoencoder
   d) Train appearance model
   e) Train landmark model
   f) Reconstruct faces at mean position by testing appearance model
   g) Reconstruct landmarks by testing landmark model
   h) Warp back reconstructed faces from mean landmark to corresponding reconstructed landmarks

   I use the same structure of neural network in the description, and set epochs = 350, batch_size = 100, appear_learning_rate = 5e-4, landmark_learning_rate = 4e-4. Below are the reconstructed faces by autoencoder:

First 20 Reconstructed Faces (with Warping)



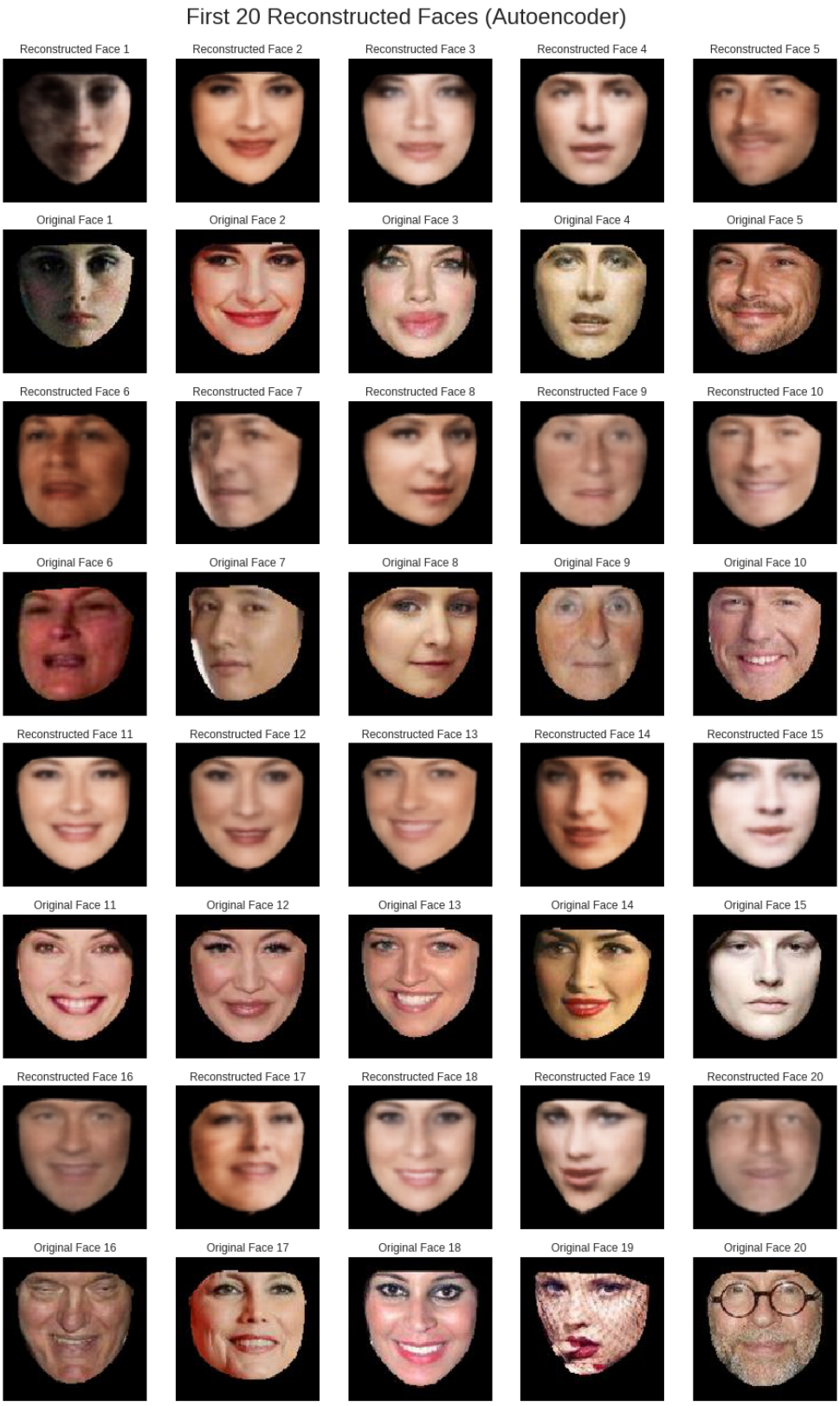


Figure 15. compare reconstructed faces (up: by PCA, down: by Autoencoder)

From the above two figures, in general, the results basically meet our expectation that the reconstructed faces are closed to correspondingly original faces. By compare Autoencoder with PCA, we can find that the faces reconstructed with PCA are more similar to the original faces than those with Autoencoder. In the second figure (Autoencoder), some of the faces are either blur or more like a mean face lacking in some specific characteristics of specific person. But for PCA, the reconstructed faces recovered most of features except that the skins are blurred a bit. If we can use other more complex neural network, maybe we will get better reconstructed faces than PCA does.

2. Interpolation of appearance and landmarks
   a) Interpolation of appearance

      First, we can train all the training data using Autoencoder, and get the encoder matrix after fully connected layer (800 x 50). Then we calculate the variance of each feature and choose the max 4 dimensions. For each dimension, we do 10 interpolation between the min and max values of it, and apply it on image 2. Next, we can pass this new (800 x 50) matrix to Autoencoder to do decoder operation. Below are the interpolations of max 4 dimensions: (In the warp operation, we can directly use the original landmark of image 2)

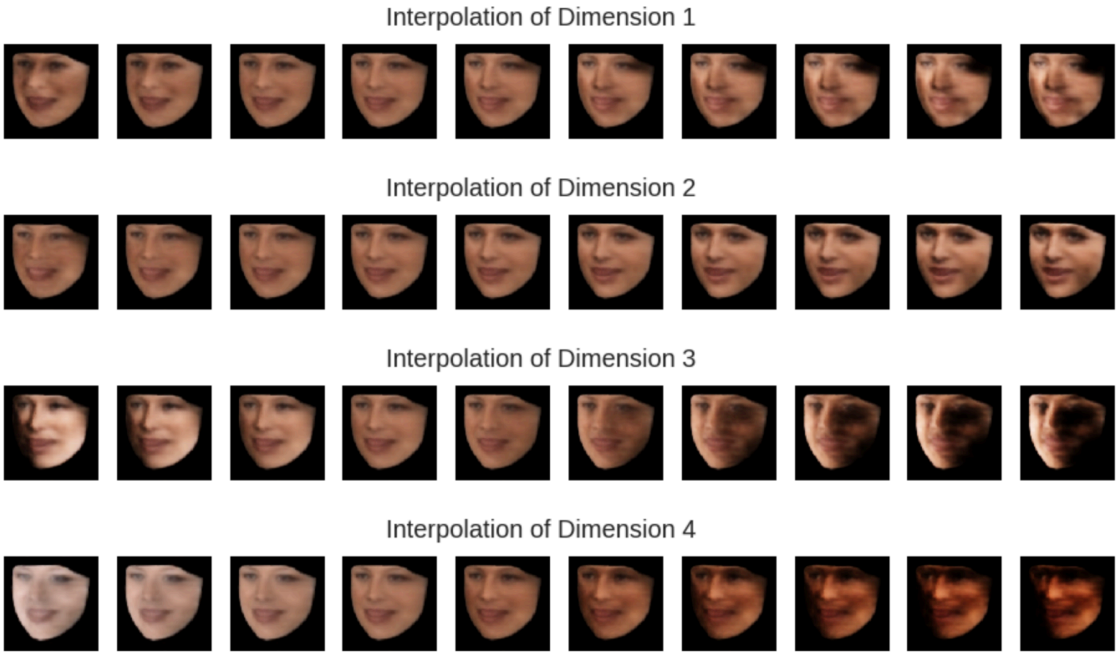


Figure 16. interpolation of appearance

From the above figure, we can find that four dimensions represent 4 different features. For example, the first one represents the shadow on the face. The third one represents the shadow on the right face. The last one represents the skin color.

b) Interpolation of landmarks

For this part, we can use the same workflow as the previous one. Then we can plot interpolated landmarks on original image 2. Below are the interpolations:

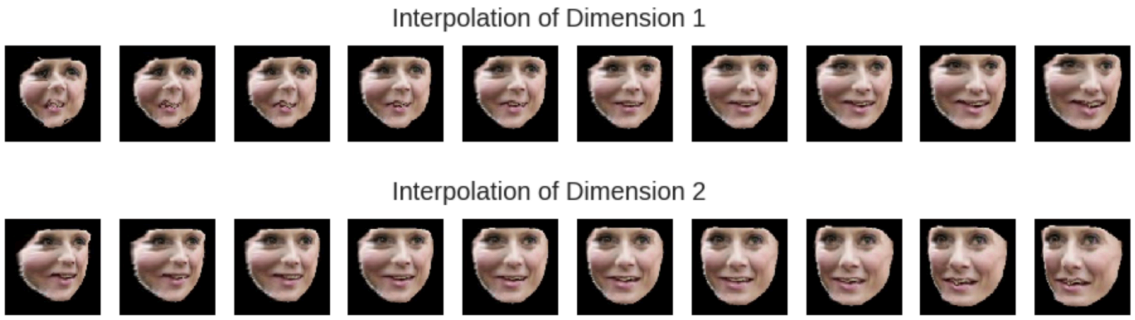


Figure 17. interpolation of landmarks

In the above figure, the first dimension represents the reshape of part of the face. While the second dimension is more like to rotate the whole face.

Part 2: Fisher faces for gender discrimination

1. Find the Fisher face that distinguishes male from female using the training sets, and then test it on the 200 testing faces and report the error rate.

   The whole dataset has high dimension that makes calculation difficult. So, we can use PCA to extract principal features and reduce dimensions. Then we can train FLD model. The test dataset is 108 female images and 92 male images. Below are error rate over increasing of number of principal components in PCA, and some figures of Fisher face:

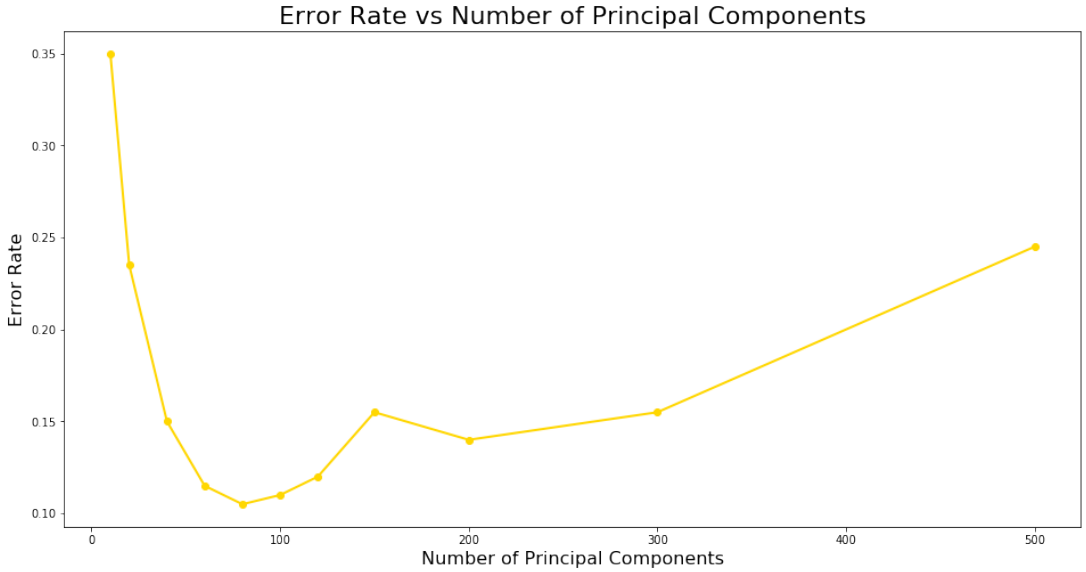
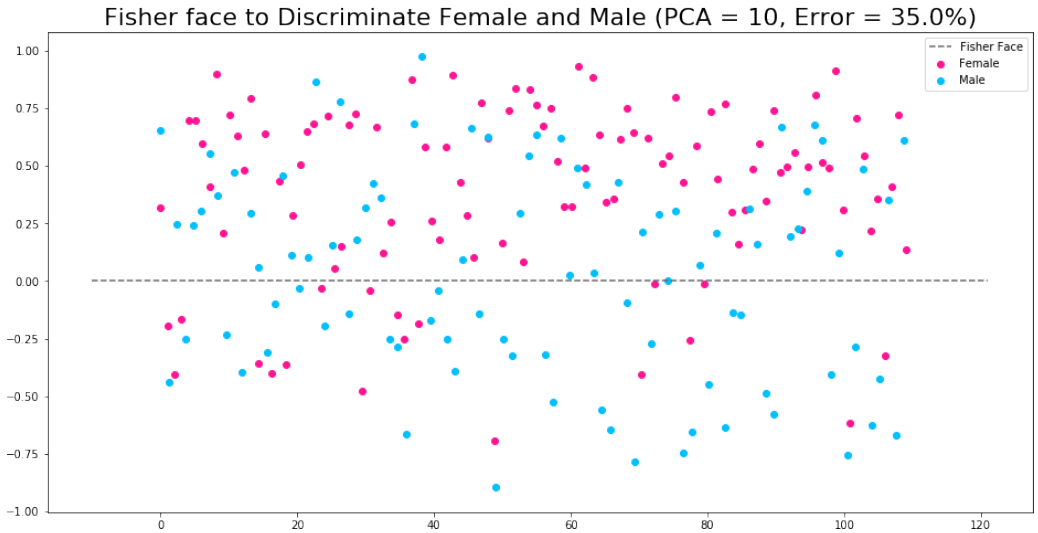
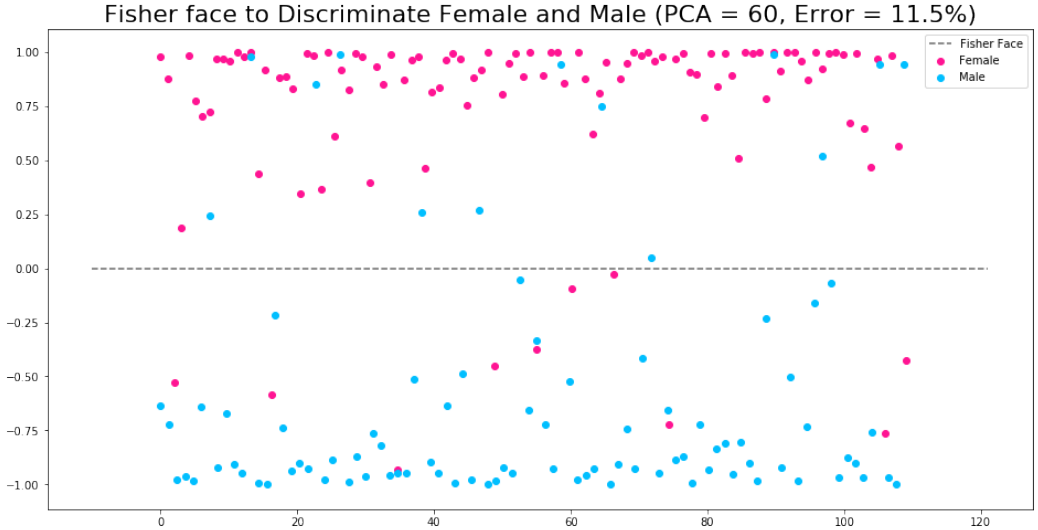


Figure 18. error rate over number of principal components


Fisher face to Discriminate Female and Male (PCA = 10, Error = 35.0%)
Fisher Face
Female
Male


Fisher face to Discriminate Female and Male (PCA = 60, Error = 11.5%)
Fisher Face
Female
Male

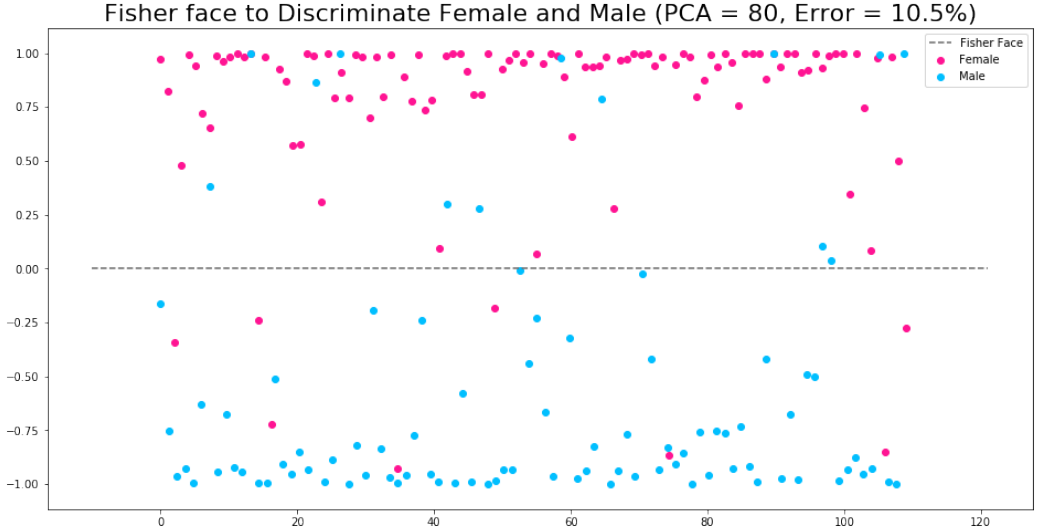

Fisher face to Discriminate Female and Male (PCA = 80, Error = 10.5%)
Fisher Face
Female
Male

Figure 19. dataset and Fisher face with different number of principal components

From the above figures, we can find that with the number of principal components increases, the error rate decreases first and then increases. The lowest error rate happens at number of principal components = 80. At this time, we can clearly see that female and male data distributes in two separate sides of the Fisher face, with only a few of mis-classifying points.

2. Compute the Fisher face for the key point and the appearance respectively.
   In this part, we first centralize landmarks and warp images to mean position. Then we can use PCA to reduce dimensions and train FLD model to get Fisher face.

First, we compute the Fisher face for landmarks. Figure 20 shows the results. We cannot distinguish female group and male group clearly, and the boundary is very vague.

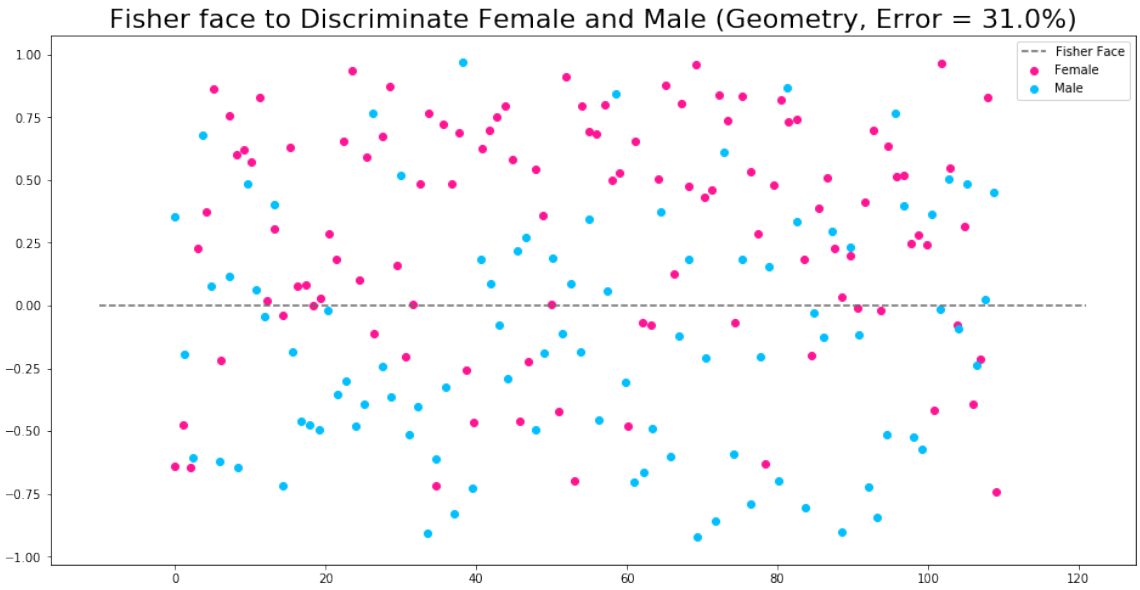


Figure 20. compute the Fisher face for landmarks

Then we compute the Fisher face for the appearance with the same process. This time, there is a clear boundary between female and male data. It indicates that it's easier to tell gender from the whole appearance instead of only the geometry shape.

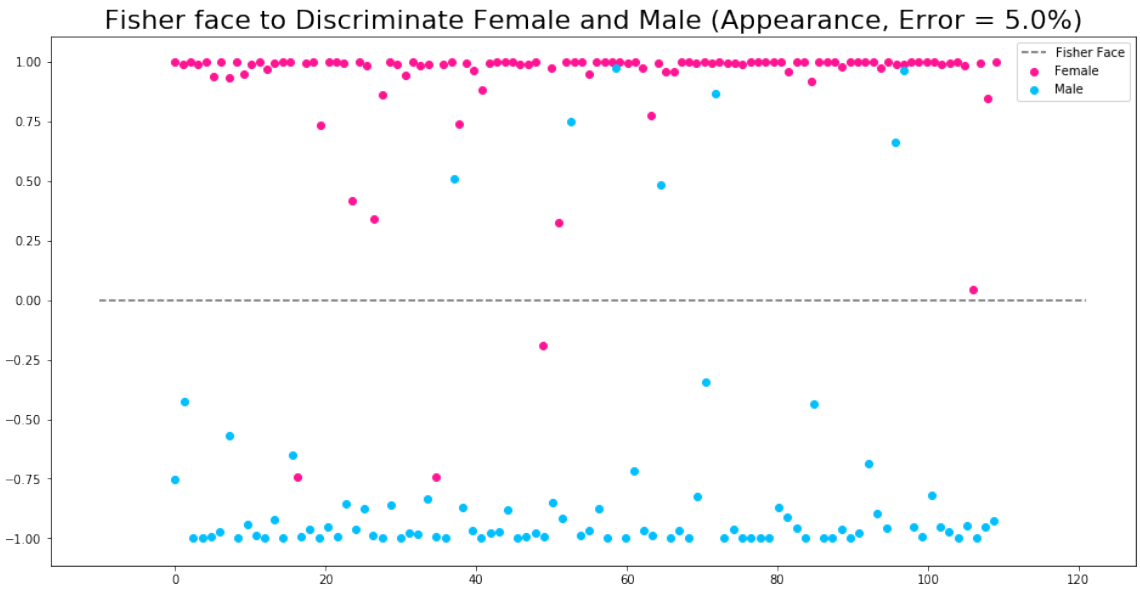


Figure 21. compute the Fisher face for appearance

Last, we can combine those two steps and predict the results from two dimensions, both appearance and geometry shape. Below is the projection figure. Now it's clear enough to distinguish female data from male ones.

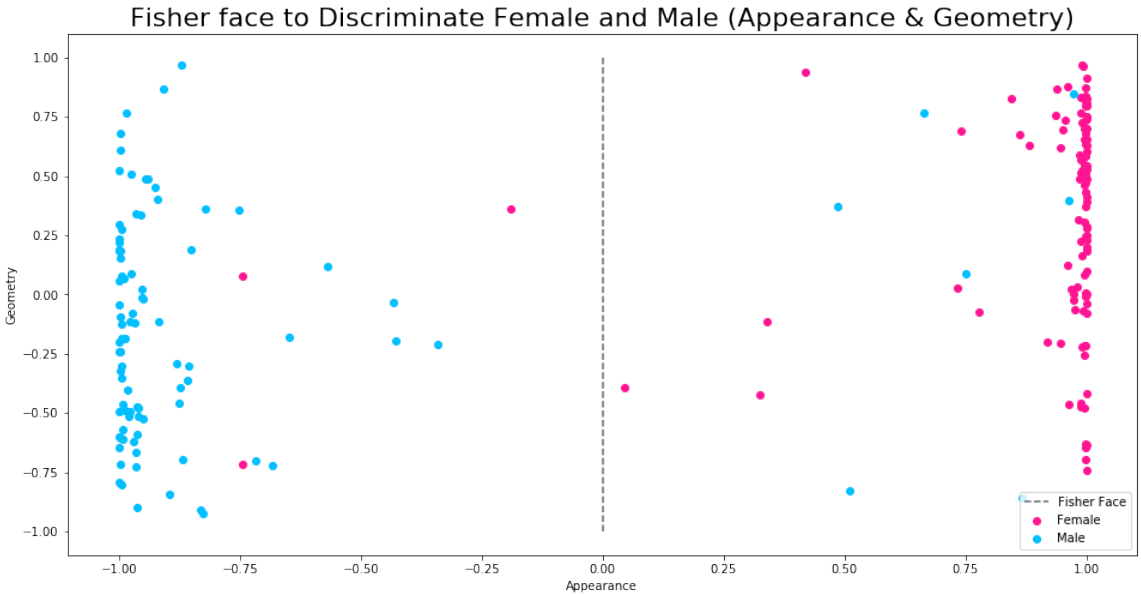


Figure 22. projection all the faces to 2D-feature space