2.1 PCA: a linear method

i) **<u>Aim</u>**: In this experiment, 800 training images in rgb were used to find the eigen vectors and corresponding eigen values. The eigen faces are then used reconstruct the original 200 test images.

<u>Procedure followed, and observations made</u>:

- The rgb images were first converted to hsv and the V channel was extracted.
- The mean of all the images was computed and subtracted from each image. (V channel)
- The top 50 eigen vectors were then chosen for the V channel based on maximum value of the corresponding eigen values.
- 200 test images were used to project into the eigen space using the top 50 eigen vectors using eqn (1).

$$a_k = e^t(X_k - m) \text{---------------------------------} eqn(1)$$

- The test images were reconstructed using eqn(2). The H and S channels from the original image were used to reconstruct the original images.

$$X = m + \sum_{i=1}^{d\text{-}new} a_i * e_i \text{ ---------------------------------} eqn(2)$$

- It was observed that as the eigen-number increased from 1,5,…,50 the total reconstruction error decreased.
- Fig.1.a and Fig.1.b shows the first 10 eigen faces.
- Fig 2. Shows the reconstructed faces and the corresponding original faces.
- Fig 3. Shows Plot of 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

<u>Results:</u>

Fig.1.a First 10- eigen faces (without cmap = ‘gray’):

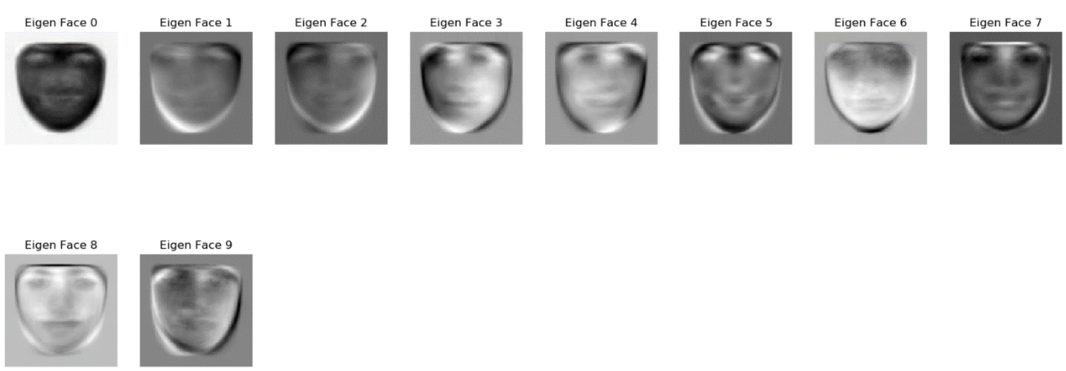


Fig. 1.b (with cmap = ‘gray’):

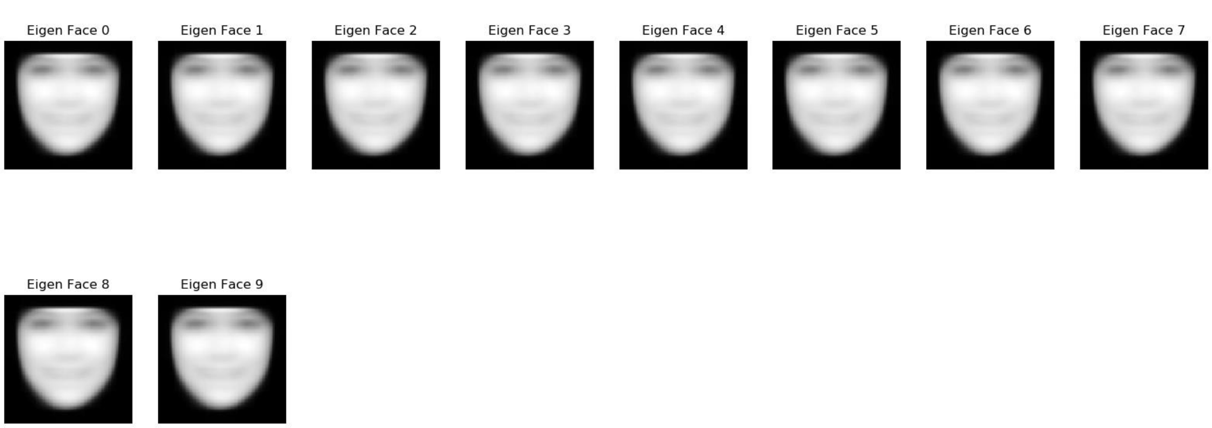


Fig. 2. 10 reconstructed faces and the corresponding original faces (chosen from the test images)

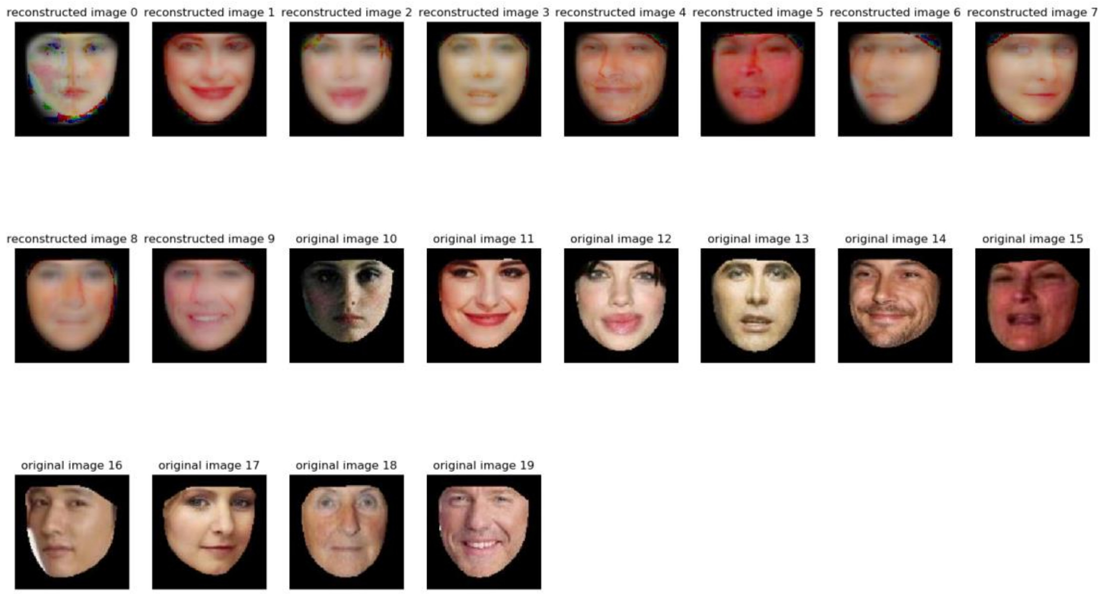
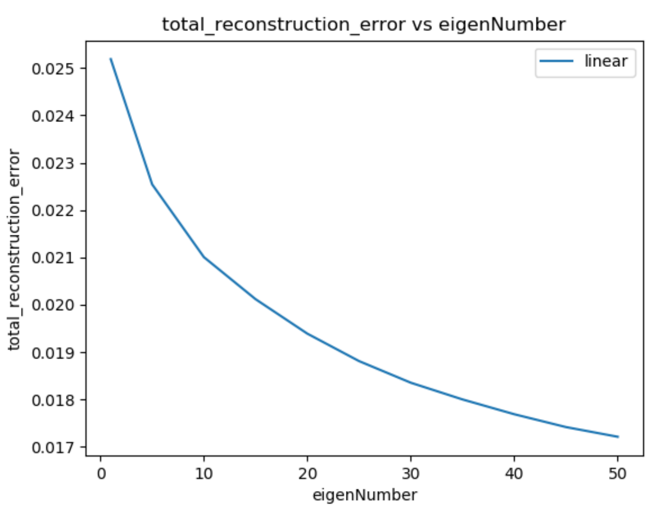


Fig.3 Plot of the total reconstruction error (squared intensity difference between the reconstructed images and their original ones) per pixel (i.e. normalize the error by the pixel number, and average over the testing images) over the number of eigen-faces K = 1; 5; 10; 15; :::; 50.



Code Output:

```
reading input train images
Calculating the mean of the input training images and subtracting from each image..

reading the test images..
projecting the test images into the eigen-space
eigen-faces K =10, total reconstruction error 0.021005951462049935
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  1
shape of eigen vectors extracted (1, 128, 128)
```

```
reading the test images..
projecting the test images into the eigen-space
eigen-faces K =1, total reconstruction error 0.025188793474599708
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  5
shape of eigen vectors extracted (5, 128, 128)

reading the test images..
projecting the test images into the eigen-space
eigen-faces K =5, total reconstruction error 0.022542886864518 47
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  10

shape of eigen vectors extracted (10, 128, 128)
reading the test images..
projecting the test images into the eigen-space
eigen-faces K =10, total reconstruction error 0.02100594501083773

reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  15
shape of eigen vectors extracted (15, 128, 128)

reading the test images..
projecting the test images into the eigen-space
eigen-faces K =15, total reconstruction error 0.02011656993467465
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  20
shape of eigen vectors extracted (20, 128, 128)

reading the test images..
projecting the test images into the eigen-space
eigen-faces K =20, total reconstruction error 0.01938908849004186
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  25

shape of eigen vectors extracted (25, 128, 128)
reading the test images..
projecting the test images into the eigen-space
eigen-faces K =25, total reconstruction error 0.018810046233946737

reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  30
shape of eigen vectors extracted (30, 128, 128)

reading the test images..
projecting the test images into the eigen-space
eigen-faces K =30, total reconstruction error 0.0183512965762059
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  35
shape of eigen vectors extracted (35, 128, 128)

reading the test images..
projecting the test images into the eigen-space
eigen-faces K =35, total reconstruction error 0.018003650688489733
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  40
shape of eigen vectors extracted (40, 128, 128)

reading the test images..
projecting the test images into the eigen-space
```

```
eigen-faces K =40, total reconstruction error 0.017693890105931177
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  45
shape of eigen vectors extracted (45, 128, 128)

reading the test images..
projecting the test images into the eigen-space
eigen-faces K =45, total reconstruction error 0.01741864077467016
reading input train images
Calculating the mean of the input images and subtracting from each image..
Computing the eigen-vectors for eigen number:  50
shape of eigen vectors extracted (50, 128, 128)

TotalError: [0.02518879 0.02254289 0.02100595 0.02011657 0.01938909 0.01881005
 0.0183513  0.01800365 0.01769389 0.01741864 0.01720931]
```

***ii)* Aim**: In this experiment, 800 training landmarks were used to find the eigen vectors and corresponding eigen values of the landmarks (68*2) . The eigen landmarks are then used reconstruct the original 200 test landmarks.

Procedure followed, and observations made:

- The mean of all the landmarks was computed and subtracted from each landmark.
- The top 50 eigen vectors were then chosen for the landmarks based on maximum value of the corresponding eigen values.
- 200 test landmarks were used to project into the eigen space using the top 50 eigen vectors using eqn(1).
- The test landmarks were reconstructed using eqn(2).
- It was observed that as the eigen-number increased from 1,5,…,50 the total reconstruction error decreased.
- The first10 eigen landmarks were overlaid onto the mean face image after adding the mean.
- Fig. 4 shows the first 10 eigen landmarks displayed on the mean face. Fig 5. Shows Plot of the total reconstruction error (squared intensity difference between the reconstructed landmarks and their original ones) per pixel (i.e. normalize the error by the pixel number, and average over the testing landmarks) over the number of landmarks

Results:

Fig. 4. first 10 eigen landmarks displayed on the mean face.

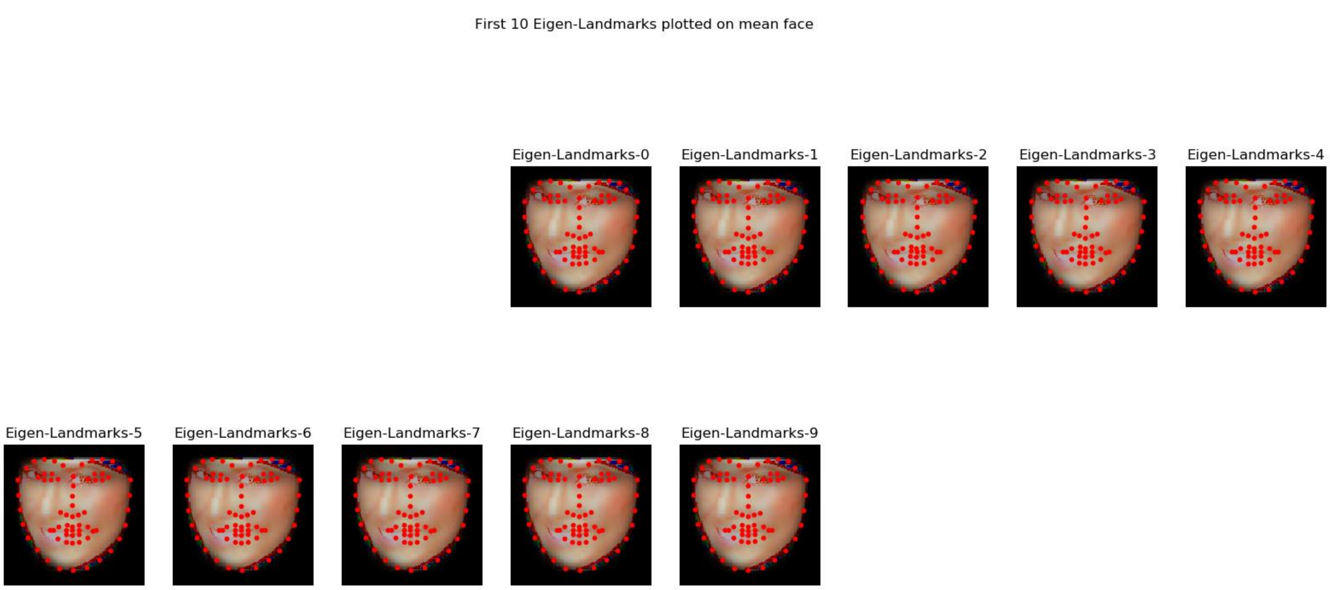
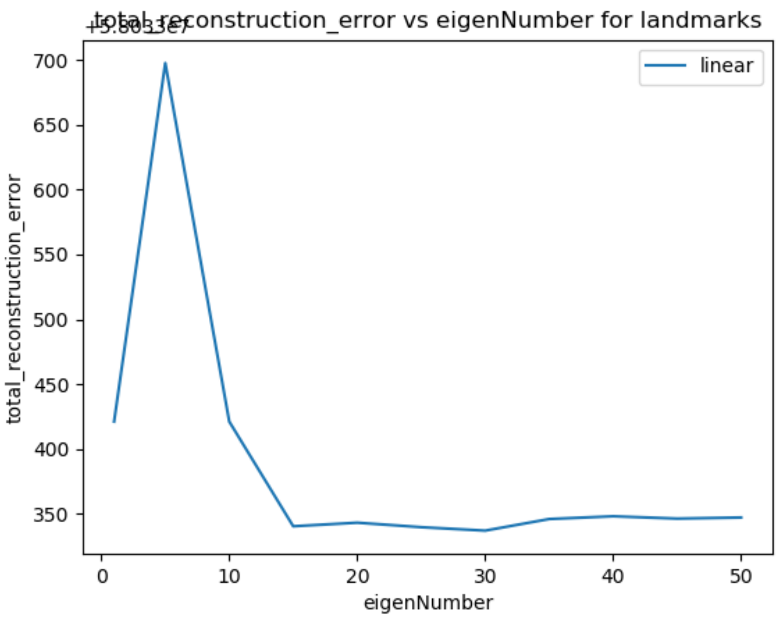


Fig. 5 Plot of reconstruction error (in terms of distance) over
the number of eigen-warpings K = 1; 5; 10; 15; :::; 50 (again, the error is averaged over all the testing landmarks).



Code Output:

```
reading input train landmarks
Computing the mean face for training landmarks..
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =10, total reconstruction error 0.005803342112377515
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =1, total reconstruction error 0.005803342112377515
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =5, total reconstruction error 0.005803369754626289
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =10, total reconstruction error 0.005803342112377515
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
```

```
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =15, total reconstruction error 0.005803334037461352
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =20, total reconstruction error 0.005803334315043105
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =25, total reconstruction error 0.005803333967722455
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =30, total reconstruction error 0.005803333701713771
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =35, total reconstruction error 0.005803334597165946
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =40, total reconstruction error 0.005803334809428563
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =45, total reconstruction error 0.005803334630081076
finding the eigen vectors for the landmarks
Projecting the test landmarks into the eigen space
Reconstructing the test landmarks from the eigen space
reconstructed_image (200, 136)
eigen-Landmarks K =50, total reconstruction error 0.005803334711304555
```

iii) Aim: In this experiment, 800 training images and 800 landmarks were used to find the eigen vectors and corresponding eigen values of both the appearance and geometry of the faces and the original images were reconstructed.

Procedure followed, and observations made:

- Part 2.a and 2.b were repeated to find the eigen vectors of both landmarks and images.
- Each test image was first warped to its mean position, then projected into the eigen appearance space.
- These test images were reconstructed back in two steps. First using the eigen-faces and then using the reconstructed landmarks they were warped back to their original positions using the warp function.
- It was observed that as the eigen-number increased from 1,5,…,50 the total reconstruction error decreased.
- Fig. 6 shows Plot 20 reconstructed faces and their corresponding original faces. Fig 7. Shows Plot of the total reconstruction error.

Results:

Fig. 6. Plot 20 reconstructed faces and their corresponding original faces. Plot the reconstruction errors per pixel against the number of eigen-faces K = 1; 5; 10; 15; :::; 50.

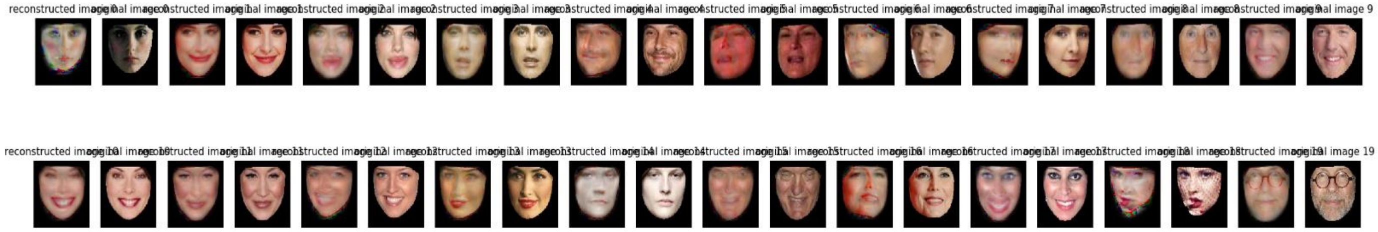
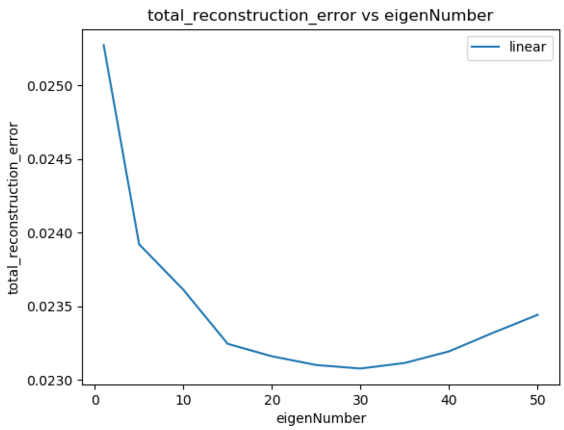


Fig. 7. total reconstruction error. Vs Eigen number



<u>Code Output:</u>

```
Reading training landmarks and finding their mean
mean is (136,)
warping the training images to mean landmark
mean is (16384,)
calculating eigen vectors for training landmarks
Reading Test landmarks, subtracting the mean from them
projecting test landmarks into first ten eigen landmarks
shape of Projected_Test_Landmarks (200, 10)
calculating the reconstructed landmarks
warping the test images to mean landmark
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 50total_reconstruction_error =  0.023429193650066828
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 1total_reconstruction_error =  0.025274582588816372
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 5total_reconstruction_error =  0.023922330785570285
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 10total_reconstruction_error =  0.02361302824245729
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 15total_reconstruction_error =  0.023245964441208265
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
```

```
For eigen no: 20total_reconstruction_error =  0.02316060801018506
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 25total_reconstruction_error =  0.023101988768541647
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 30total_reconstruction_error =  0.023077104708807025
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 35total_reconstruction_error =  0.023114397308209727
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 40total_reconstruction_error =  0.023189163047975213
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 45total_reconstruction_error =  0.023317925311371467
Warping to reconstructed landmarks
calculating eigen vectors for warped training images
Projecting the test images into eigen space and reconstructing
For eigen no: 50total_reconstruction_error =  0.023456692210605153
Warping to reconstructed landmarks
```

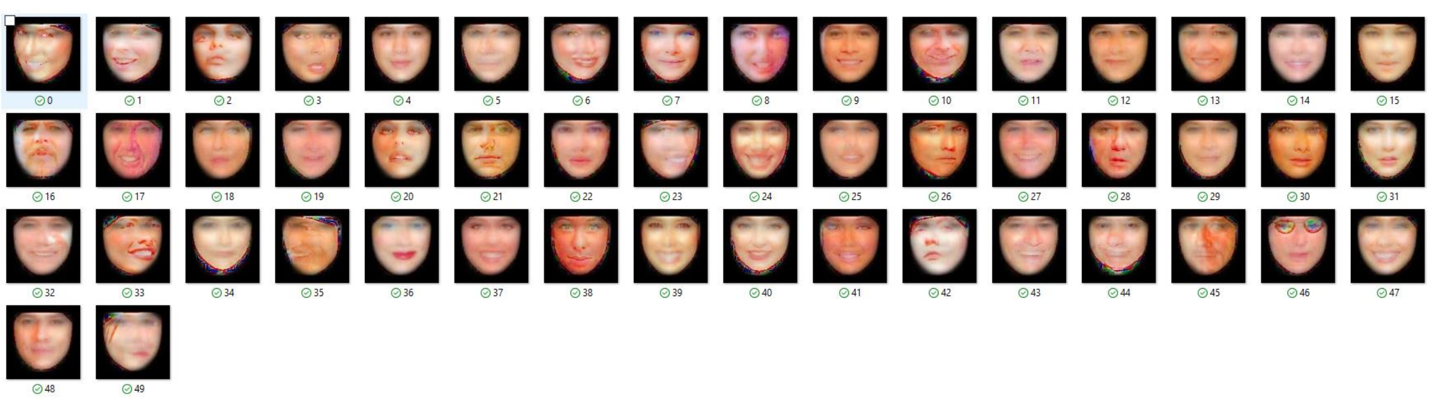
iv)**Aim**: In this experiment, 50 images were synthesized.

Procedure followed, and observations made:

- The random sampling of the latent vectors (coeffecients) of both the appearance and landmarks was done by changing the first 50 eigen vectors of appearance and first 10 eigen vectors of landmarks with a random value that lies in this range (0 , sqrt(lambda$_i$)), where lambda$_i$ is the square-root of the i-th eigen-value.
- The process includes, sampling the 50 eigen vectors of appearance and then warping these to the sampled landmarks.
- Fig. 8 shows the 50 synthesized face images.

Results:

Fig 8. Display 50 synthesized face images.



Part 2.2. Auto-encoder

Aim:

In this experiment, the two-dimensional face images were reconstructed using auto-encoder.

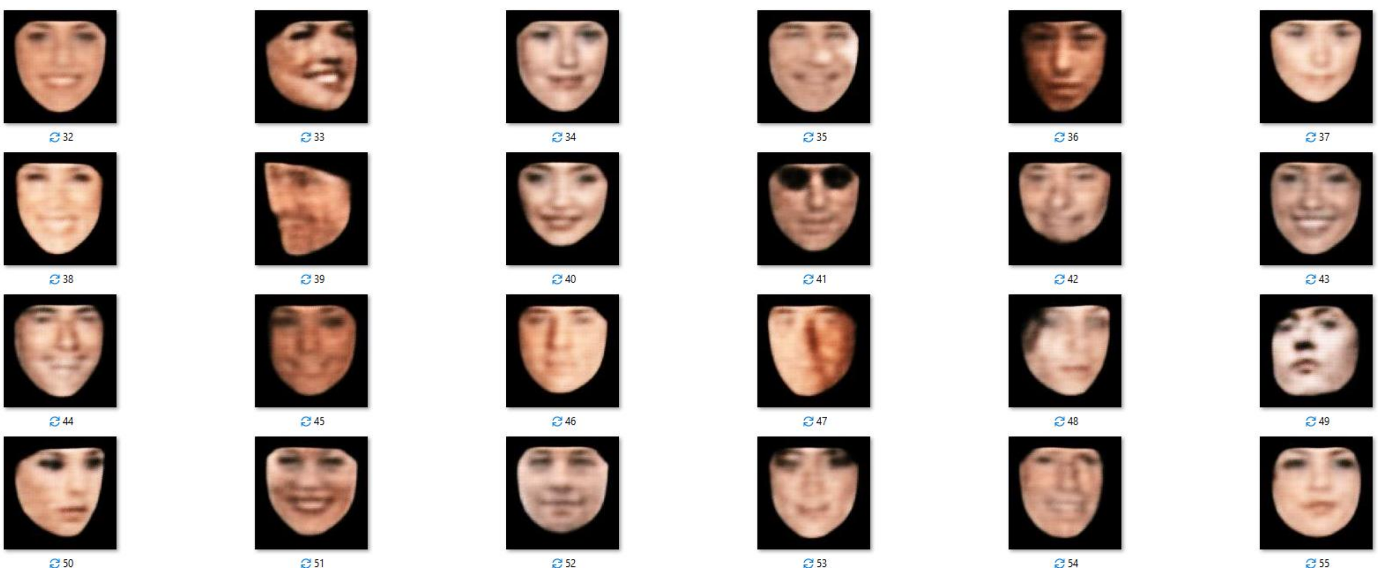
i) and generated by the auto-encoder with convolutional architecture and the landmarks can be reconstructed and generated by the auto-encoder with fully-connected architecture

Procedure followed, and observations made:

- The images were first reduced to dimensions of 50 from 128*128 by using a convolutional network (implemented using tf.nn.conv2d). The output from the convolutional network was then passed into a de-convolutional architecture. (implemented using tf.nn.conv2d_Transpose). The de-convolutional network reconstructs back the original image.
- In case of landmarks, they are reduced to 10 dimensions from 68*2 using a fully-connected network and then back to original dimensions using second a fully-connected network.
- The test images were warped to the mean, sent through the learned conv-deconv networks.
- The test landmarks were sent through the learned conv-deconv networks.
- Finally, the test images are warped back from mean to their original landmarks that were reconstructed in the previous step.
- Fig. 9 shows the 10 reconstructed face images.

Results:

Fig. 9. 20 reconstructed face images.



ii) **Aim**: In this experiment, we perform interpolation on the first 4 dimensions of the latent variables of appearance and the first 2 dimensions of the latent variables of landmarks.

Procedure followed, and observations made:

Appearance

- The training images were used to learn conv-deconv network. Using the learned network, the output from the encoder, which is of dimension 1*1*50 is sorted according to variance values.
- The top four latent variables are picked from this sorted vector and for each of the four dimensions, we show interpolation by picking a random value in the range (maxOfDimension - minOfDimension) and increasing it ten

times by a step size = (maxOfDimension - minOfDimension)/10, thereby producing 10 interpolation results for each of the four dimension (40 images) in total.

- Fig. 10 shows the random images for each appearance dimension.

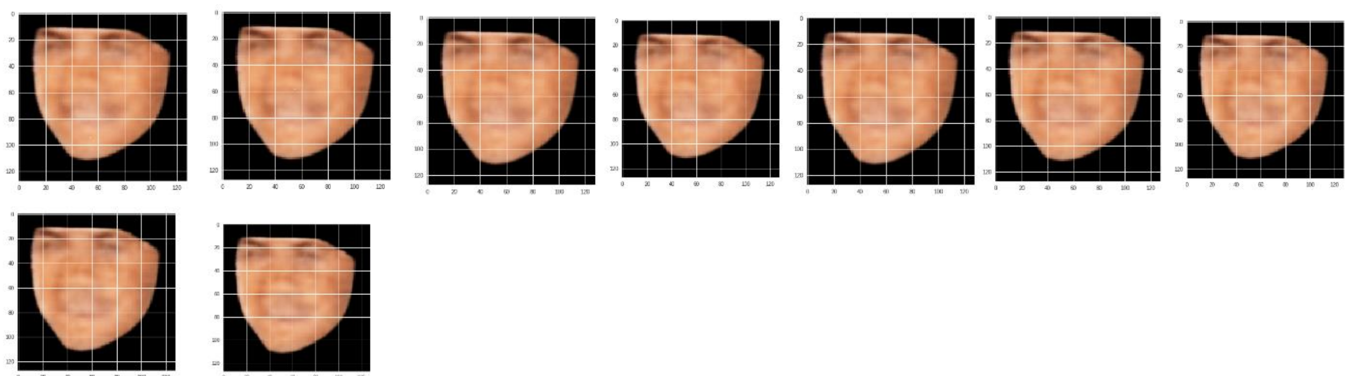
Landmarks

- The training landmarks were used to learn an fully connected encoder-decoder network. Using the learned network, the output from the encoder, which is of dimension 1*1*10 is sorted according to variance values.
- The top four latent variables are picked from this sorted vector and for each of the 2 dimensions, we show interpolation by picking a random value in the range (maxOfDimension - minOfDimension) and increasing it ten times by a step size = (maxOfDimension - minOfDimension)/10, thereby producing 10 interpolation results for each of the 2 dimension in total.
- We take any image and then warp it to the interpolated landmarks from the above step.
- Fig. 11 shows the random images for each landmarks dimension.

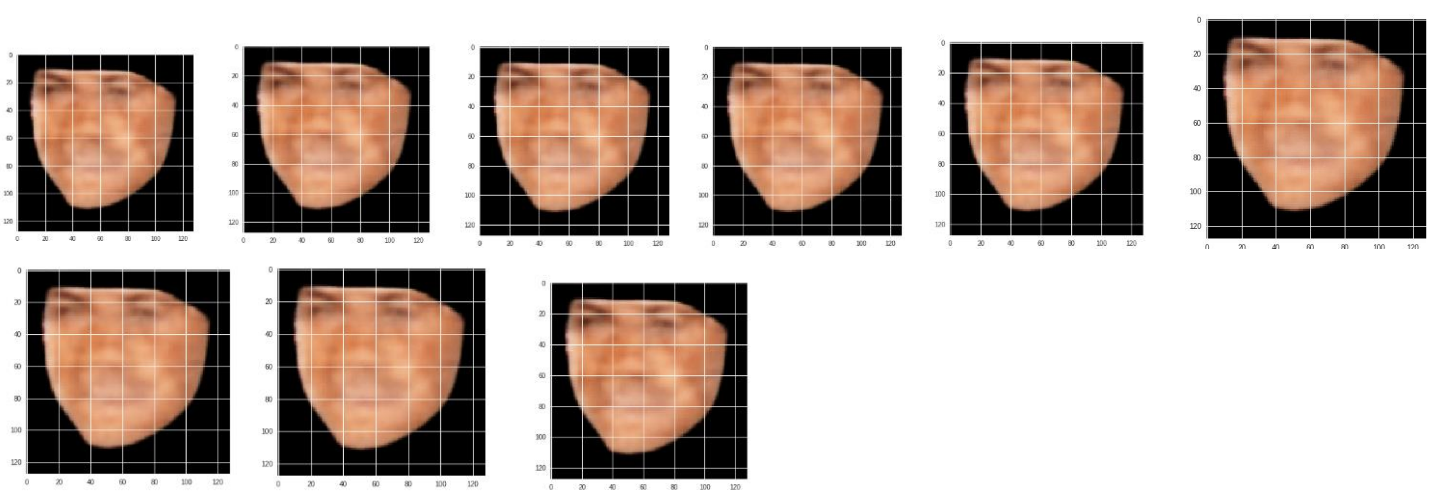
<u>Results:</u>

<u>Fig. 10</u> the random images for each appearance dimension.

Ten random images from Dimension 1:

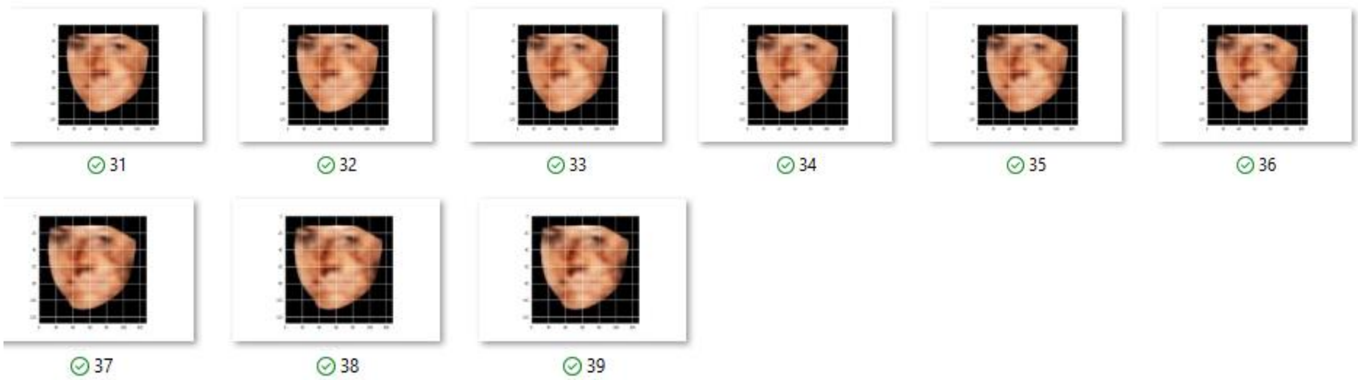


Ten random images from Dimension 2:



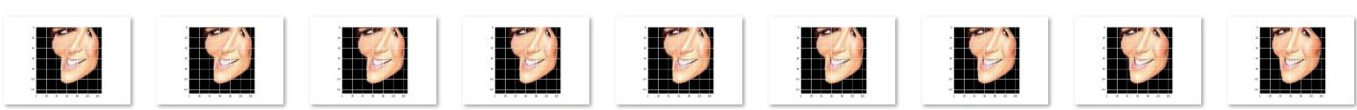
Ten random images from Dimension 3:

Ten random images from Dimension 4:

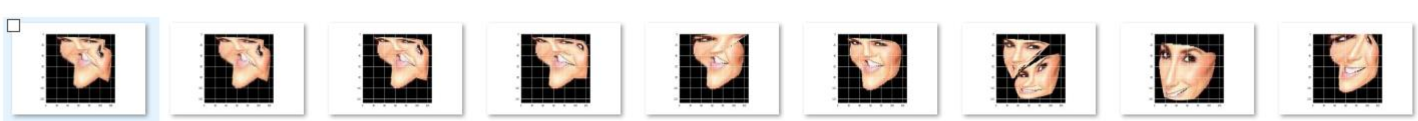


<u>Fig. 11</u> Random images for each landmarks dimension.

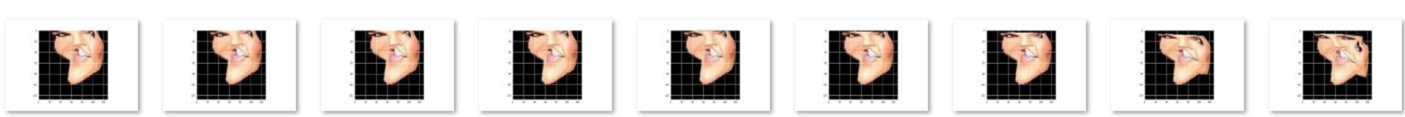
Ten random images from Dimension 1:



Ten random images from Dimension 2:



Ten random images from Dimension 3:



Ten random images from Dimension 4:

Part 2.3: Fisher faces for gender discrimination.

i)

Aim: In this experiment, the fisher faces that distinguishes male from female was found using the training sets. It was then tested on the 200 testing faces. The Fisher face here mixes both geometry and appearance difference between male and female.

Procedure followed, and observations made:

- Each training image and training landmark is first reduced in dimensions using the pca process. We get a 50 dimension + 10 dimension = 60-dimension input vector by this process.
- The fisher linear discriminant or "w" is then obtained using eqn (3)

$$W = S_w^{-1} \text{(MeanOfFemale – MeanOfMale)------eqn(3)}$$

- We know project our male and female input vectors into this one-dimensional space defined by fisher linear discriminant.
- The threshold which distinguishes male from female is then found using eqn (4)

$$\text{Threshold} = \text{W(MeanOfFemale – MeanOfMale)/2------ eqn (4)}$$

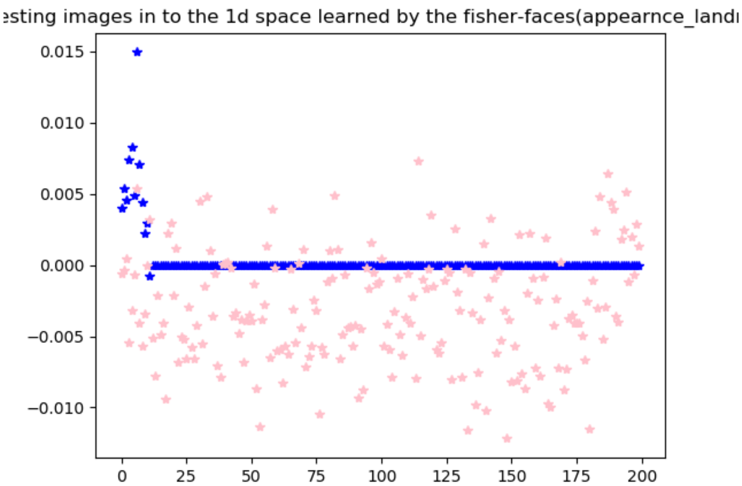
- Fig. 12 shows the plot of male vs female landmark projections and male vs female appearance projections on the training set.
- Fig 13 shows, male vs female 1D projections for landmarks + appearance on the test set.

Results:

Error rate(for geometry and appearance combined) is 0.22

Fig. 12

Fig. 13



ii) **Aim**: In this experiment, the fisher faces that distinguishes male from female was found for (geometric shape) and appearance separately. using the training sets. All the faces were then projected to the 2D-feature space learned by the fisher-faces, and visualized how separable these points are.

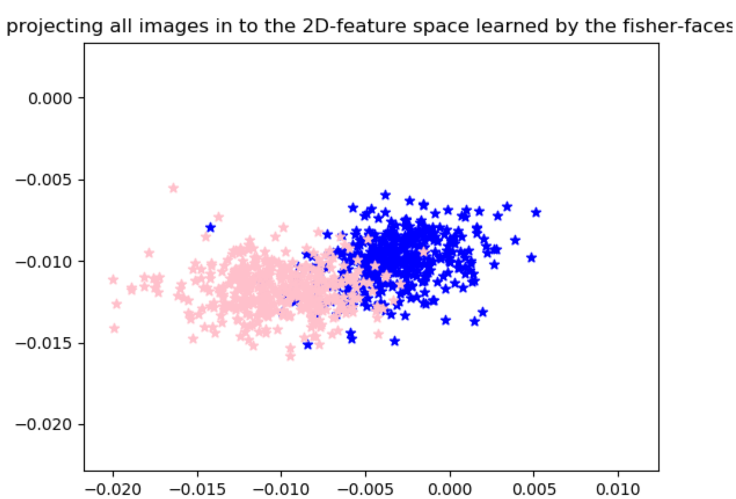
Procedure followed, and observations made:

- Each training image is first reduced in dimensions using the pca process to 50 dimension.
- The fisher linear discriminant or "w_of_ appearance" for appearance is then obtained using the eqn (3)
- Each training landmark is first reduced in dimensions using the pca process to 10 dimension.
- The fisher linear discriminant or "w_of_ landmark " for landmark is then obtained using the eqn (3)
- We know project our male and female input vectors into this 2d-space (w_of_ appearance and landmark).
- The threshold which distinguishes male from female is then found using the eqn (4).
- Fig 14. shows the projection of all the faces to the 2D-feature space learned by the fisher-faces.

Results:

The fisher faces(for geometry and appearance combined) that distinguishes male from female is the point at -0.000330508150811559

Fig. 14



Code Output :

```
C:/Users/vaish/OneDrive/Desktop/PycharmProjects/prml_project1/FISHER-Submit.py

Running part 1: finding fisher for geometry and appearance combined
reading input train images
threshold is -0.0003305081508115559
error rate(for geometry and appearance combined) is 0.325
the fisher faces(for geometry and appearance combined) that distinguishes male from female is the point at -0.0003305081508115559
End of Part 1
Running part 2: finding fisher for geometry and appearance individually
The fisher face for the key point (appearance) is -0.0067671795862068839
reading input train images
The fisher face for the key point (geometry) is -0.0108558115130953l3
End of Part 2
```