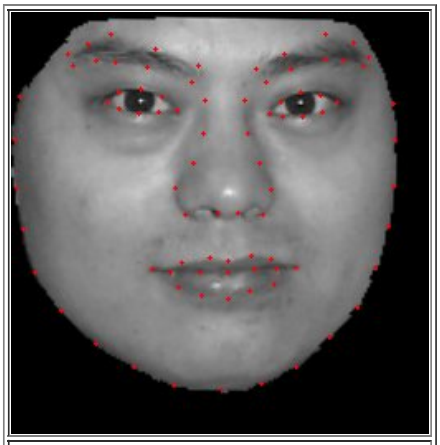
Project 1: PCA and FLD for Analyzing Huamn Faces

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.

We provide a dataset of 177 faces in this project and each has 256 x 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 a number of landmarks which are identified manually for your convenience and they correspond across the dataset. In the second dataset, there are 87 points per image. As we know that 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.



face with 87 landmarks

Part 1: Active Appearance Model (AAM) for face reconstruction and synthesis.

The experiment include the following steps.

we divide the 177 faces into two parts: the first 150 faces are put in a training set, and the remianing 27 faces are put in a test set. In the following steps, you 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. [note that in the dataset Face 103 is removed as it is a duplicate]

(1). 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.

(2). Compute the mean and first k-eigen-warpping of the landmarks for the training faces. Here warping means displacement of points on the face images. Display the first 5 eigen warppings (you need to add the mean to make it meaningful), and use them to reconstruct the landmarks for the test faces. Plot the reconstruction error (in terms of distance) over the number of eigen-warppings k (again, the error is averaged over all the testing images).

(3). Combine the two steps above. Our 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-warpings, you get the reconstructed landmarks. (here you lose a bit of geometric precision of reconstruction); (ii) warp the face image to the mean position and then project to the top k (say k=10) eigen-faces, you get the reconstructed image at mean position (here you further lose a bit of appearance accuracy). (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 (here you have loss in both geometry and appearance). (iv) Plot the reconstruction errors per pixel against the number of eigen-faces k.

(4). 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. (As we discussed in class, each axis has its own unit, that is, the square-root of its eigen-value).

Part 2: Fisher Linear Discriminant (FLD) for gender discrimination.

We have divided the 177 faces into male (88) [in the male face folder, Face 57 is removed as a duplicate] and female faces (85), plus 4 unknown (which is hard even for human eyes to tell based on the faces alone). Then you choose 10 male and 10 female as the testing set. The remaining faces (except the 4 unknown) will be used as training sets.

(5) Find the FLD or Fisher face that distingushes 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.

(6) 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.

[The within-class scatter matrix is again very high dimensional, you can either use the trick that we used in (1) for computing its eigen-values and eigen-vectors, or you may compute the Fisher faces over the reduced dimensions in steps (2) and (3): i.e. each face is now reduced to 10-Dimensional geometric vector + 10 dimensional appearance vector. After the Fisher linear Discriminant analysis, we represent each face by as few as 2 dimensions for discriminative purpose and yet, it can tell apart male from female!

Read and pdf file **compute the Fisher face.pdf** for exactly how to do this in matlab].

What to submit on CCLE:

Your submission will be an experimental report, you need to visualize the results in good figures (**your grade will be based on the quality of results and analysis**), such as mean, eigen-vectors, and plots. Print out the result for submission. Don't print the code. Zip all your code and send to the reader in email for checking and comparison.

2. Datasets and code

The dataset is packed in a zip file, with a readme file.

Here is a piece of the matlab code for warping the images: WarpImage.m

A student in the class has made a video animation for some of the eigen-vectors which represent interesting axes of changes. Click here to see.

A reference for comparing the eigen-face and Fisher face is also attached.