ES-331 Probability and Random Processes

# Real Time Face Recognition System

## Assignment-1

### Introduction

Human faces are complex and multidimensional, thus detecting and recognizing them is a difficult task. Face recognition can help in various tasks such as security, forensic investigation, image and video processing, etc. Recognizing faces using eigenfaces is an efficient method since using PCA reduces the dimensionality of the problem while keeping most of the information. In the Turk and Pentland paper, they have assumed that face recognition is a 2D problem since faces are mostly upright.

### The method used in Turk and Pentland Paper

In this paper they use Principal Component Analysis to find the eigenfaces and then encode the faces using those eigen faces. While recognizing an image they encode that image to and compare it with stored encodings. PCA helps in reducing the dimensionality of the encoding.

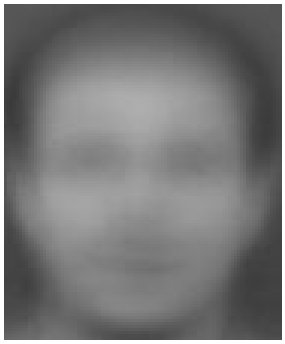
Dataset: This database of faces consists of ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details(glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

In addition to the above dataset, I downloaded images from Google images of 5 non face images, 5 face images of persons not in the ORL dataset and 5 images of people in a larger frame. These images are used only for testing purposes.

We divide the data set into two parts: training data and test data. Training data contains randomly chosen 8 images of each person. Test data contains rest of the images.

## Training Algorithm

- There are M training images. We read these images and represent each NxK image as an NKx1 vector. All images are centered in the ORL dataset and are of the same size.
- Then we calculate the mean of these images and subtract the mean face from each image. The resultant images are stored in a vector $A = [\phi_1, \phi_2, ..., \phi_M]$



- These images are then used to calculate eigenfaces using PCA. In principal we need a set of M orthonormal u vectors such that, such that $\lambda_k$ is maximum for a given $u_k$, where $\lambda_k = \frac{1}{M} (u^T{}_k \phi_n)^2$. Here $\lambda_k$ and $u_k$ are respectively the eigenvalues and eigenvectors of covariance matrix $C = AA^T$.
- Dimensions of $C$ are NK x NK. For normal image sizes, computing NK eigenvectors and eigenvalues is a very difficult task. Since number of images M<<NK, there are only M significant eigenvectors and eigenvalues. We can calculate them using the matrix $L = A^T A$. $A(A^T A)v = \mu Av \Rightarrow AA^T(Av) = \mu(Av)$ where $v$ is eigenvector of $A^T A$. Thus we calculate eigenvectors of $L$ and pre multiply with $A$ and normalize them to get M eigenvectors of $C$. We sort eigenvectors in decreasing order of eigenvalues and take top k eigenvectors. These eigenvectors are the eigenfaces and form a basis for the face space. Top 12 eigenfaces obtained are as follows:

- For each image in the training set we calculate the weights such that, $\phi_i = \Sigma w_{ij} u_j$ where $w_i$ is the weight associated with the *i* th image and $u_j$ are the eigenfaces.

## Checking whether an image is near face space

For checking this we reconstruct face images in the test data and images of unknown person and take the euclidean distance between original and reconstructed image. We add 100 to max distance obtained since the test images are too close to the eigenfaces and a new image of a person in database may have higher distance. The number 100 is chosen heuristically. The received number serves as a threshold for the face space.

## Testing algorithm

- We read the test images and subtract the mean face vector from each of them. Then we calculate the weights for each image. There are four possibilities for an image:
    - It is near face space but not near a person class
    - It is near face space and near a person class
    - It is not near face space and not near a person class
    - It is not near face space but near a person class
- We reconstruct the image using eigenfaces and check whether it is near face space. To check we check whether the distance between the original and reconstructed image is less than the threshold for face space. If the image does not belong to the

face space, then it is classified as 'nf' i.e. a non face image. The last two cases are taken care by this.

- If the image is near face space then we calculate the minimum euclidean distance between the weights of test image and the weights of the training images. If that distance is less than a chosen threshold, then this image is classified as the image of the person with which it had the minimum distance. Otherwise, it is classified as 'unk', i.e. image of an unknown person.

## Real time face detection

Earlier we fed images that were centered and had dimensions that of the training images. In real conditions, the images will be obtained from a video input and will not be centered. So we preprocess the image to obtain images of faces that are centered and have dimensions NxK.

- Since background is a problem we remove the white gaussian noise from the image.
- Then we use the Haar cascade method to detect the face.
- Then we crop the detected face and resize it to the required dimensions.

We then use this image to recognize whether it is of some person in the database using the method described above.

## Issues:

- Sometimes background noise can be classified as a face by the haar cascade algorithm. To eliminate that we have used a gaussian filter.
- Accuracy of the model depends on orientation of the faces. If orientation and face size is similar to the size of the training images then the accuracy is generally good.

## References:


- http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/mcl/1992_turk_eigenfaces_for_recognition.pdf
- https://ieeexplore.ieee.org/document/139758