Face Recognition using eigenfaces

ES-331 Probability and Random Processes

Shubhranshu Singh

16110154

Introduction

The problem of face recognition has been studied intensively. The method of face recognition using eigenfaces gives real time face recognition capability.
The dataset used contains cropped faces but if a dataset without cropped face is given, the first step would be to detect faces using methods like haar cascade and then apply the following algorithm.

Algorithm

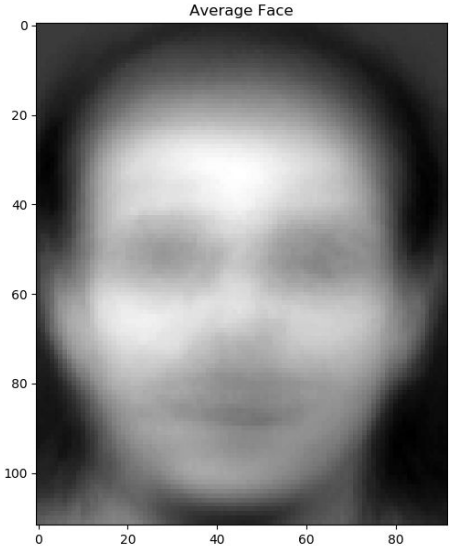
The algorithm for face recognition has been referred from [1], where eigenfaces are used to recognise a given image.

Dataset

The dataset of images of the face is taken from [2]. There are 40 different people and 10 pictures of each person. The size of each image is 112 x 92 pixels (rows x columns) with 256 gray levels per pixel. The images were taken in homogenous dark background with the people in upright and frontal position. There are 40 folders (named s1,s2 … s40) each with 10 images (named 1.pgm,2.pgm … 10.pgm). For training 8 images of each person is taken (total 320 images) and the testing is done on 2 images of each person(total 80 images).Some images are shown below

Preparing the eigen faces

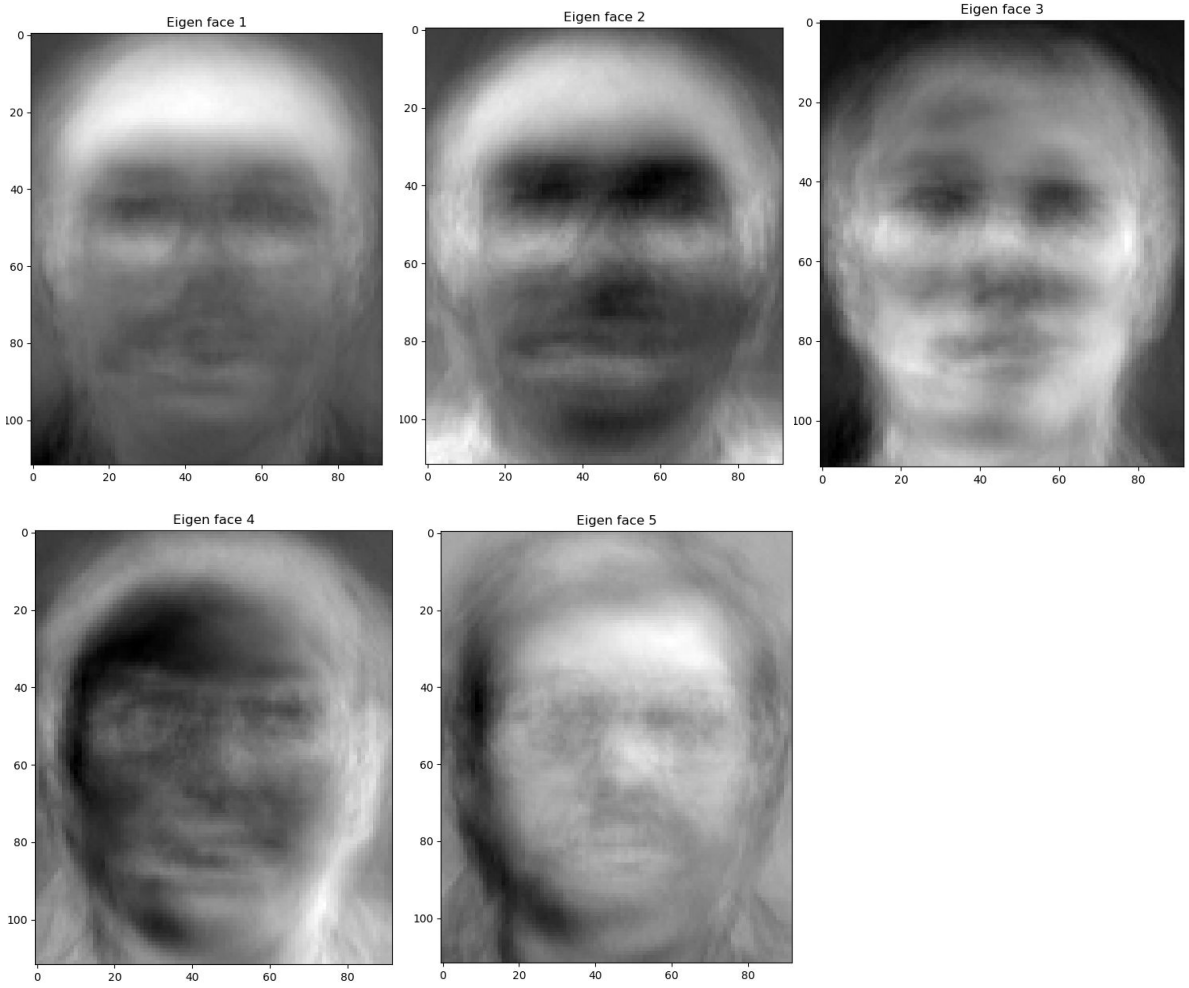
Each image is $\Gamma_i(x,y)\ i\ \varepsilon\ \{1,2,...M\}$ of size $m \times n$ is treated as a $m * n$ dimensional vector. So each image is a point in this high dimensional space. Now the average image is calculated by taking average of all the image vectors. $\Psi_i = \frac{1}{M} \sum_{k=1}^{M} \Gamma_k$. The average image is shown below.



The average image is then subtracted from each image. $\Phi_i = \Gamma_i - \Psi_i$.
The next step is to perform principal component analysis on the Φ_i vectors but using a covariance matrix to find the eigenvectors would be computationally expensive as covariance matrix would be of the size $m * n \times m * n$ (Size of images is generally quite large). Thus the eigenvectors of a $M \times M$ matrix(L) are calculated as follows which will be the meaningful eigenvectors of the covariance matrix. $L_{ab} = \Phi_a^T \Phi_b$ and its eigenvectors are v_l.
So , in the given dataset L would be of the size 320×320 and the covariance matrix would have been 10304×10304.

Now eigenfaces are calculated using v_l and Φ_i.

$u_l = \sum_{k=1}^{M} v_{lk}\Phi_k \; l \, \varepsilon \, \{1, 2, \ldots, M\}$. Only top M' eigenfaces are retained and it is taken to be 100 for this dataset.The top five eigenfaces are shown below



Calculating weight vector of image

A vector denoting the components of an image onto the eigenfaces is found as follows

$\omega_k = u_k^T(\Gamma - \Psi) \; k \, \varepsilon \, \{1, 2, \ldots, M'\}$. These weights form a vector

$\Omega^T = [\omega_1, \omega_2 \ldots, \omega_{M'}]$. Similarly, this vector is found for all training images and the test image vector is compared to each training image vector. The one with the smallest norm-2 is the closest training image and the test image is labelled same as the matched training image.

Test set accuracy

Out of 80 test images, 77 were correctly recognised giving an accuracy of 96.25 %.

Challenges

Initially I tried to recognise faces on Georgia Tech face database, which did not have cropped faces and I used haar cascade method to detect faces. But the accuracy was less compared to the cropped images dataset that I have used now. This may be because the Georgia Tech face database had people which wore glasses in some of the images and not in the other and also had greater head tilt angle. Also haar cascade method is not 100% accurate to locate a face. Thus I shifted my work to the [2] database.

References

[1] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Maui, HI, USA, 1991, pp. 586-591

[2] Dataset: AT&T Laboratories Cambridge, http://www.cl.cam.ac.uk/Research/DTG/attarchive/pub/data/att_faces.tar.Z

[3] Georgia Tech face database http://www.anefian.com/research/face_reco.htm

[4] Haar cascade https://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html