ECE 515: Image Analysis II Midterm Project

Dr. J. Ben-Arie

The aim of this project is to implement the Karhunen-Loeve (KL) transform for face detection and recognition after the approach of Moghaddam and Pentland[1].

The project can be completely implemented in Matlab. You can choose to implement it in C or any other language, but it is advisable to use Matlab for ease of programming, display and printing of images, available functions for determining eigenvectors and eigenvalues and presentation of concise source code. **Please present concise and well documented source code for full credit. All requirements for the project are bulleted and must be completed for full credit.**

You are provided with 18 training images of different face models 1 through 18 in the files: KL\_norm\_train.dat and KL\_norm\_train2.dat. These are text files which have the data stored in an arrays of size 1600 × 9 and can be directly read into Matlab. Each column is an image which can be *reshaped* into 40 × 40 (N × N) for purposes of display and printing. Each of these images have been normalized to have a zero DC value i.e.

(1)

and unit variance i.e.

(2)

* Display and print these 18 face images. Remember to scale the images appropriately to the dynamic range of the display for correct display and printing
* Find and print the average face image. The average face when ordered as a vector corresponds to (in the lecture).

(3)

M is the number of training images. This represents the ensemble average of all the training faces. Compute the covariance matrix of the data. Find the eigenvalues and eigenvectors for the data covariance matrix. The eigenvectors can be determined from this covariance matrix using the function *eig* in Matlab. This matrix is of size 1600×1600 and hence takes a long time to compute. An alternate computationally efficient method using Singular Valued Decomposition [2] may be used instead. This was explained in class. Verify that the eigenvectors are orthonormal. The eigen images must be considered in decreasing order of eigenvalues.

* Display and print the 9 eigenvectors as eigen images (by reshaping them to size 40 *×* 40 and scaling them to the dynamic range of the display.
* Reconstruct all the 9 training images with 6 eigenvectors.

Next, you are given an image of size 120 × 120 in the file Test\_image.dat. This is formatted as an ASCII matrix of size 120 × 120. This image has a set of faces which are to be detected using the eigen images. The image is not normalized and has a dynamic range between 0 and 255.

• Display and print this image after scaling appropriately.

In processing the test image, you need to scan the image with a 40 × 40 window. The part of the image within the window is first normalized by making it zero mean and unit variance to match the statistics of the test image.

• Is this necessary? Why or why not? Please provide a well-thought out argument based on the methods used for detection and recognition.

You can detect the faces in the image using the following formulae. Let W be the 40 × 40 window being scanned and be the eigen image corresponding to eigenvalue. The projection of the window on an eigen image is given by the inner product

(4)

• Why is it necessary to subtract from W?

is a vector of coefficients in the transformed (KL) space. We can use a subset of these coefficients and the optimal representation capability of the KL to compute the distance of the window to the cluster faces represented by our 18 exemplars. If this distance is minimized, the window W is close to the cluster of faces and is a face with a high likelihood. This distance is given by

(5)

We can estimate this distance using fewer eigenvectors using the formula.

(6)

Where

(7)

and

(8)

The advantages of this method is not really seen in using as few training images as 18, but the computational savings become considerable when you can use about 5 or 6 eigenvectors obtained from 100’s of training images. Use 6 eigenvectors in your approach. You can try all 18 eigenvectors and compare.

You should get an image matrix of the estimated distance which looks like the one shown in Figure 1.

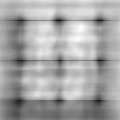


Figure 1: Example distance image. The local minima given by the dark spots indicate the location of the faces.

•Provide the source code for the detection algorithm, i.e. the implementation of the formulae.

• Display and print the distance image.

• Use a non-minimal suppression algorithm to extract the coordinates of the actual face locations. Non-minimal suppression algorithm will be explained in class. Provide source code.

• Provide the coordinates of detected faces.

Next, recognize the faces using the 6 eigen coefficients at the detected face locations. The approach is to compare the eigen coefficients of each training image to the eigen coefficients each face image. The minimum distance between these coefficients indicate the recognized faces.

• Print a 9 × 6 table of the KL coefficients of each of the detected faces.

• Provide the 9 × 9 matrix of these distances of test faces versus training faces in the file: KL\_norm\_train.dat and indicate the recognized face based on this matrix. You can also plot these distances for better visualization.

**References**

[1] Moghaddam Baback, Pentland Alex, “Probabilistic Visual Learning for Object Rep- resentation,” IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, pp.696-710, July 1997.

[2] Anil K. Jain, Fundamentals of Digital Image Processing, pages 176-178, Prentice Hall, New Jersey, 1989.