Homework 2 v.1.1: Learning theory II
(BME 580.692, CS 600.462)

Instructor: René Vidal, Phone: 410-516-7306,
E-mail: rvidal@cis.jhu.edu

Due on Thursday September 28th, 2006, beginning of class.

1. Read Chapters 1 and 2 of GPCA book. Go to *http://www.vision.jhu.edu/gpcabook/* and submit all the typos you find as well as suggestions you may have to improve the quality and/or readability of the material. **You will receive credit for each interesting typo or suggestion you submit**.

2. Exercises 2.1 and 2.7 of GPCA book.

3. **Face recognition using PCA.** In this exercise you will use the AT&T database[1] (previously also known as ORL), which contains photos of 40 individuals, with 10 poses for each individual. Divide all the images in two sets: *Set A* (images from individuals 1 to 20) and *Set B* (images from the others individual, 21 to 40). Subdivide *Set A* in two parts: the *Training Set* (poses from 1 to 5) and the *Validation Set* (poses from 6 to 10). Notice also that there are 5 non-face images (accessible as Individual #41, poses from 1 to 5). We will refer to these as *Set C*.

   Download the file *hw2-06-dataset.zip*. This file contains the image database along with the MATLAB function `loadimage.m`. Decompress the file and type `help loadimage` at the MATLAB prompt to see how to use this function). The function operates as follows.

   Function `img=loadimage(individual,pose)`

   Parameters

`individual`	Number of the individual.
`pose`	Number of the pose.

   Returned values

`img`	The pixel image loaded from the database.

   Description

   Read and resize an image from the AT&T database. The database (directory `att_faces`) must be in the same directory as this file.

   This exercise is composed of two parts: in the first part you are asked to implement some MATLAB functions. Then, you will use these functions to do the experiments in the second part. You will receive credit for each one of the functions you write, as well as for your answers to the questions in the experimental part.

[1] AT&T Laboratories Cambridge,
`http://www.cl.cam.ac.uk/Research/DTG/attarchive/facedatabase.html`.

Functions to be implemented

Function `[vectorimg,sz]=image2vector(img)`

Parameters

`img` A pixel image loaded with `loadimage`

Returned values

`vectorimg` The same image as a vector of `double`.

`sz` A vector `[m,n]` containing the original size of the image.

Description

Form a vector from a loaded image by stacking all the pixel values read columnwise (note that you will need also to convert the data from the MATLAB type `uint8` to `double` for further processing).

Function `[img]=vector2image(vectorimg,sz)`

Parameters

`vectorimg` Same as returned by `image2vector`

`sz` Same as returned by `image2vector`

Returned values

`img` The pixel image corresponding to `vectorimg`.

Description

The dual function of `image2vector`.

Function `[meanface,efaces,coeffs,sv]=eigenfaces(facevectors)`

Parameters

`facevectors` The images of all the individuals in the training set, `nposes` for each individual. Each column is a vector image given by `image2vector` from the training set. There should be `nposes` images for each individual and columns corresponding to the same individual should appear in succession.

Returned values

`meanface` Average of the vectors in `facevectors`.

`efaces` The principal components from the vectors in `facevectors` with the average subtracted.

`coeffs` The coefficients obtained by projecting `facevectors` on `efaces`

`sv` The singular values given by the SVD.

Description

Computes the "eigenfaces" (the principal components) `efaces` of the space spanned by the vectors of `facevectors` (with the mean `meanface` subtracted) and, at the same time, the coefficients of the same vectors projected on the eigenfaces, all using the Singular Value Decomposition (SVD). Returns also the vector of the singular values `sv`. Can you notice a difference in between using `svd(A)` and `svd(A,0)`?

Function `[imgrec,dist]=reconstruct(img,meanface,efaces)`

Parameters

`img`	A pixel image (loaded with `loadimage`).
`meanface`	Same as the value returned by `eigenfaces`.
`efaces`	Same as the value returned by `eigenfaces`.

Returned values

`imgrec`	The pixel image corresponding to `img` projected on `efaces`.
`dist`	The distance between `img` and its projection `imgrec`.

Description

Given an image, projects it on the principal components contained in `efaces` and then recovers the image of the projection (all by taking into account `meanface`). Returns also the distance between the given vector and its projection.

Function `plotindividuals(coeffs,nposes)`

Parameters

`coeffs`	The coefficients of all the images in the database projected on the eigenfaces (as returned by `eigenfaces`).
`nposes`	Number of poses for each individual in `coeffs`.

Description

Displays a three-dimensional plot of the points given by the first three coefficients of each projection contained in `coeffs` using a different color/marker for each individual.

Function `membership=kNN(labels, distances, k)`

Parameters

`labels`	set of labels for each neighbour.
`distances`	distance of each neighbour.

Returned values

`membership`	the label assigned by the k-Nearest Neighbours algorithm.

Description

Implements the k-Nearest Neighbours algorithm. By sorting distances, find the labels of the k nearest points among all the neighbours. The membership is the label which appear most often in the k nearest ones. In case of “tie”, the minimum average distance in the k neighbours is considered.

Function `indrec=recognize(img,meanface,efaces,coeffs,nposes,k)`

Parameters

`img`	pixel image with the face of the individuals to be recognized.
`meanface`	Same as the value returned by `eigenfaces`.
`efaces`	Same as the value returned by `eigenfaces`.
`coeffs`	Same as the value returned by `eigenfaces`.
`nposes`	Number of poses for each individual used in the computation of the eigenfaces.
`k`	Number of neighbours for the k-Nearest Neighbours algorithm.

Returned values

`indrec`	The index of the recognized individual in the training set.

Description

Computes the distance between the coefficients of a new image `img` and the coefficients of all the vectors in `coeffs` from the projection onto the subspace spanned by the columns of `efaces`. Then, by knowing the order in which the images have been loaded during the training, use these distances and k-Nearest Neighbours (function `kNN`) to assign `img` to one of the individual in the training set.

Experiments

(a) Using `loadimage` and `imshow`, display some images for some individuals and some poses. What can you note about the images in this database? Under which conditions are the photos taken (e.g. changes in illumination, pose, expression, etc.)? Quantify if it is possible. Are the photos "normalized" in some way (e.g. the nose is always at the same pixel location)?

What is the dimensionality of the vectors returned by `image2vector`?

(b) Load the images of the *Training Set* (the other Sets will be used later) and store their vector representation in a matrix called `facevectors` (loading all the images is slow and can take up to a minute).

(c) Compute the eigenfaces using the function `eigenfaces`. Plot the singular values. How can you estimate the number of eigenfaces to use (i.e. the dimensionality of the "face" subspace)?

(d) Choose three images: one from *Set A, Training Set* (e.g. individual 1, pose 1), one from *Set A, Validation Set* (e.g. individual 2, pose 8) and one from *Set B* (e.g. individual 23, pose 1). Display and compare the images with their corresponding reconstructions using $n = 30, 50$ and 100 components. Please, comment.

(e) Using the function `plotindividuals`, plots the first three coefficients of the vectors `coeffs` returned by `eigenfaces`. What can you notice?

(f) Choose some images from *Set A, Validation Set* and compare them to the images corresponding to the individual returned by `recognize` with $k = 1$ (i.e. Nearest Neighbour algorithm). Using also the results from the part (e), explain why this system can be used for face recognition?