Supervised Remote Lab session: face recognition using eigenfaces

Spring 2020 course: *Imaging for Security Applications*

Starting date: April 3rd, 2020
Send your report by **April 20th, 2020**

Companion skype group chat: https://join.skype.com/eT1OJmPG8eNb

Related documents:

- *Face recognition using Eigenfaces*

J.-L. Dugelay, A.Berthet, M. Jamel Eddine, C. Bisogni, A. Trabelsi & K. Mallat

EURECOM, Sophia Antipolis
e-mail:{**To:** mallat**, Cc:** dugelay}@eurecom.fr

https://my.eurecom.fr/jcms/p0_2020647/en/imsecu

General advices

Before…

- Before starting the Matlab implementation, read all the questions completely. This way you will save time since some tasks will be repeated over different exercises.
- Answer the questions precisely. Remember you should provide your comments and justifications.

After…

- **Send the complete answers by e-mail (in English) in plain text format and attach a zipped folder named "YOUR-SURNAME_Your-first-name" before April 20th.**
- Folder has to contain **one edit file for each exercise**. Please, don't send us images but link the folders containing them used in each edit file (Please, if you download material in a local directory, link anyway the public one)
  (addpath('**\PATH\TO\TpBiometry\public\Matlab'**))
  Replace '\PATH\TO' by the path in wich you extracted TpBiometry.zip
- Everything needed to run the code has to be saved **in the submitted folder**.
- **The code has to run**. If it presents compilation errors the score will be 0.

Additional images and code

For this remote lab., you will need several files (images, Matlab's functions, …); it is recommended that you copy everything in a single directory on your personal computer and work there.
Data can be downloaded using ftp at ftp://ftp.eurecom.fr/incoming/TpBiometry2020.zip.

If you are working under a Linux workstation, you can download all necessary files for the lab here:

- **/PATH/TO/TpBiometry/public/Images**
- **/PATH/TO/TpBiometry/public/Matlab**
-

If you are using a Windows PC, you should access the teaching folder with the following text:
\PATH\TO\TpBiometry\public\.
Please, refer to the next section for more details about the image database and the Matlab's function library.

Introduction to the Lab.

The technology

This lab. will focus on face recognition. We will study the eigenface algorithm introduced by Matthew Turk and Alex Pentland [1]. This is a very popular algorithm for face identification and verification. For instance, this is the technology used by Viisage [3].

The database

We will use the ORL face database [2] for our experiments. It consists of 400 images of 40 individuals (10 faces per individual) where each image is 92x112 pixels. The name of face images is of the form: s*_*.jpg where the first * is a number between #1 and #40 that identifies each person and the second * is a number between #1 and #10 that identifies each variation of the same individual. There can be variations in illumination, pose, scale, facial expressions…

We will split this database into two image sets: set A, which will consist of individuals #1 to #20, and set B, which will consist of individuals #21 to #40. We will subsequently split both sets into training and test sets. The training sets will be used to build the face recognizer. The test sets will be used to assess the performance of the recognizers. Images #1 to #5 will be used as training images and images #6 to #10 will be used as test images. Therefore, if we refer to set **Train A**, we refer to images #1 to #5 of individuals #1 to #20. Hence, all training/test sets will contain a total of 100 images.

Since the image database is fairly large, we suggest you to open them in their original location; if you want to make a local copy, we strongly advise you to work in the temporary directory of your machines and to copy all the files you need there.

Matlab and the functions provided

In order to help you to focus on the main points of this lab., a useful library of Matlab functions is provided. Again, you don't need to copy them in your local space; for example, by adding the following code at the beginning of your Matlab files,

```
addpath \PATH\TO\TpBiometry\public\Matlab
```

you can easily call them just typing their names, as if they were built in Matlab commands.

It may worth to remember that typing

```
help <function>
```

in a Matlab console, it is possible to have a description and some details about that function (built in or provided for the lab). Before asking a for assistance about a function, take a look at its help, please.

In Matlab, it is possible to save workspace variables by typing

```
save('<FileName>', '<Variable1>', '<Variable2>');
```

and to load them using

```
load('<FileName>', '<Variable1>', '<Variable2>');
```

Finally, at the end of this document you will find the list of Matlab functions provided for the lab.

Exercise 1: building the eigenspace

In this introductory exercise, we will build our first eigenspace and compute the location of faces in this space.

Part A: computing the eigenspace A

First, use the set **Train A** to train the eigenspace that we will call **Space A**; use routines **loadImagesInDirectory** and **buildSpace** that are provided. Since building the eigenspace is computationally intensive, we strongly advise you to save **Space A**.

What is the maximum size of the eigenspace, i.e. how many directions of variability can we estimate? The eigenvalues that are computed while building the eigenspace can be interpreted as the amount of information that is carried out by each eigenvector. Why is it so?

Part B: plotting the cumulative sum of eigenvalues

Draw the cumulated sum of eigenvalues; you can use the Matlab's *cumsum()* function.

Please, comment the curve.

Part C: approximating s1_1.jpg

Using routine **approximateImage** try to rebuild the first training face of the first person (image **… train_A\s1_1.jpg**) with a varying number of eigenfaces.

Can you rebuild it perfectly? Why?

Part D: projecting and plotting in face space

In this part, you are enrolling training images of Train A into Space A. Enrolling an image into a space means that you are projecting this image into this space. Using routines **projectImages** (i.e. to project training images of Train A into Space A) and **plotFirst3Coordinates**, plot the coordinates of the five training faces of the first five training individuals (i.e. 25 points) in the space spanned by the first 3 eigenfaces (E1, E2 and E3). Again, we suggest you to save the projected images of **Train A** onto **Space A**.

Please, comment.

Exercise 2: identification

In this exercise, we will obtain our first identification and cumulative identification results. While the traditional identification only answers to the question “Which is the individual with the best rank?”, the cumulative identification answers to the question “Which are the N individuals associated with the best ranks?”.

Part A: projecting and plotting Test A

Project test images of **Test A** into **Space A**. Draw the location of the five test points of the first five individuals in the space spanned by the first 3 eigenvectors of **Space A**.

Compare with the points obtained with the training data. Please, comment.

Part B: approximating s1_6.jpg

Using routine **approximateImage** try to rebuild the first test face of the first person (image **… test_A\s1_6.jpg**) with a varying number of eigenfaces.

Can you re-build it perfectly? Why?

Part C: computing the identification rates (first face)

You will now compute the identification rates by comparing the test identities (i.e. the projected images of Test A in Space A) against the enrolled identities (i.e. the projected images of Train A in Space A). Using routine **identify**, compute the identification rate for a varying number of eigenfaces (i.e., *FirstNEigenfaces* is the varying parameter). In this question, you will use as reference point (or model) of each individual his/her first training face (you can load the previous projected images of **Train A**).

What are the best identification rate and the optimal number of eigenfaces? Draw the identification rate as a function of the number of eigenfaces and comment on this curve.

Part D: more identification rates (mean face)

Repeat the previous question, using this time the average of the five training points as reference point of each individual (i.e. the mean of the five projected images of Train A of each individual).

Please, comment.

Although identification is a very popular measure of performance, the cumulative identification can be very useful too. Give examples of applications where the cumulative identification rate might prove useful.

Part E: drawing identification rates as a function of N-Best

In all the following experiments, the reference point of the individuals will always be the average of his/her five training faces, as you did in *IV.D*. Once more, we invite you to save the reference points for further usage.

Project the images of **Test A** into **Space A**, as before. For a fixed number of eigenfaces (e.g. 5), compute the N-Best identification rates, for various N, using the routine **identify** and draw the identification rate as a function of parameter N.

Comment on this curve.

Exercise 3: verification

In this exercise, we will obtain our first verification results; verification answers to the question “Is he/she the person he/she claims to be?”.

Part A: computing client and impostor scores

Project the images of **Test A** into **Space A** or load a saved copy. For a fixed number of eigenfaces (e.g. 5), compute the client and impostor scores using routine **verify**. Plot on the same figure the histograms of client and impostor scores.
In order to better interpret the results, you should rescale the histograms and plot their coordinates as follows:

```
[YClients XClients] = hist(-log(DistancesClients), 10);
[YImpostors XImpostors] = hist(-log(DistancesImpostors), 10);
figure;
hold on;
plot(XClients, YClients, 'b', XImpostors, (YImpostors / 19), 'r');
```

Please, comment.

Part B: plotting the Receiver Operating Characteristic (ROC) curve

Using routine **computeVerificationRates** plot the Receiver Operating Characteristic (ROC) with the same fixed number of eigenfaces.

Please, comment. For what kind of applications would you use this technology?

Part C: drawing the Equal Error Rate (EER) curve

Now, using routine **computeEER**, compute the Equal Error Rate (EER) for various numbers of eigenfaces.

Draw the EER as a function of the number of eigenfaces and comment on this curve.

Exercise 4: mismatch between the eigenspace and test individuals

In this exercise, we will try to evaluate the influence of a mismatch when the individuals used to build the eigenspace are different from the test individuals.

Part A: computing identification rates for set B

Enroll images of set **Train B** into **Space A** (i.e. Project images of set **Train B** into **Space A**) and project images of set **Test B** into **Space A**. Draw the location of the 5 faces of the first five individuals of **Test B** in the space spanned by the first 3 eigenfaces of **Space A**. Compute the identification rate, for a varying number of eigenfaces.

What are the best identification rate and the optimal number of eigenfaces?

Part B: computing the eigenspace B

Now enroll the images of set **Train B** into **Space B** (i.e. Project the images of set **Train B** into **Space B**) and project images of set **Test B** into **Space B**. **Space B** is computed using the images of set **Train B**. Draw the location of the 5 faces of the first five individuals of **Test B** in the space spanned by the first 3 eigenfaces of **Space B**. Compute the identification rate for a varying number of eigenfaces.

What are the best identification rate and the optimal number of eigenfaces? Compare results obtained in the two latter questions. Are they surprising? Is there a significant difference in performance? Compare advantages and disadvantages of using the same individuals for both building the eigenspace and testing.

Annex: Matlab routines for the Lab

The routines that you will be using are available in the following directory: **\PATH\TO\ TpBiometry\public\Matlab**.

approximateImage()

FigureHandle = approximateImage(FileName, Means, Space, FirstEigenfaces)

It approximates the image contained in 'FileName' using the first '*FirstEigenfaces*' eigenfaces. 'Means', the average of all the faces used to train the space, and 'Space', the eigenspace, must be provided.
The original face image and its approximation are displayed. FileName contains the relative path (to the current one) and the image filename.

Usage example:
approximateImage('\PATH\TOTpBiometry\public\Images\train_A\s11_5.jpg', Means, Space, 88);

buildSpace()

[Means, Space, Eigenvalues] = buildSpace(ImgsVectors)

It computes the average 'Means' of all vectors in 'ImgsVectors' and the eigenvalues 'Eigenvalues' and eigenspace 'Space' generated by these vectors; the method used is Singular Value Decomposition (SVD).
Eigenvectors are automatically sorted by their contribution, i.e. eigenvectors which correspond to the largest eigenvalues are ranked first in 'Space'.

Usage example:
[Means, Space, Eigenvalues] = buildSpace(ImgsVectors);

computeDistanceFromSpace()

Distance = computeDistanceFromSpace(ImgsVectors, Means, Space, FirstEigenfaces)

It computes the distance of one or many images contained in 'ImgsVectors' to the projected ones in the space spanned by the first '*FirstEigenfaces*' vectors of 'Space'; 'Means', the average of all the faces used to train the eigenspace, must be provided. 'Distance', is the Euclidean distance in the face space.

Usage example:
Distance = computeDistanceFromSpace(ImgsVectors, Means, Space, 5);

computeEER()

EqualErrorRate = computeEER(DistancesClients, DistancesImpostors)

It computes the Equal Error Rate (EER) based on the client scores 'DistancesClients' and impostor scores 'DistancesImpostors'.

Usage example:
EqualErrorRate = computeEER(DistancesClients, DistancesImpostors);

computeVerificationRates()

[FalseRejectionRates, FalseAcceptanceRates] = computeVerificationRates (DistancesClients, DistancesImpostors)

It computes the False Rejection Rate (FRR) and the False Acceptance Rate (FAR) based on the client scores 'DistancesClients' and impostor scores 'DistancesImpostors'.

Usage example:
[FalseRejectionRates, FalseAcceptanceRates] = computeVerificationRates(DistancesClients, DistancesImpostors);

identify()

IdentificationRate = identify(Models, Test, FirstEigenfaces, NBest)

It evaluates the percentage of identification.
'*FirstEigenfaces*' is the number of eigenfaces used for the identification; 'NBest' is the number of candidates that are retained for each test image (N-best). 'Models' and 'Test' contain respectively the coordinates of training models and test **projected** images. We assume that 'Models' contains ONE model (or point) per individual. 'Test' may contain many points per individual but individuals should be sorted in the same manner in 'Models' and 'Test', and all the images of one individual should be contiguous in 'Test'.

Usage example:
Score = identify(TrainAModels, TestA, 5, 1);

loadImage()

Image = loadImage(FileName, Path)

It loads an image, automatically resizing it to 56 * 46 pixels; the output will be a vector of size 1 * 2,576.
'FileName' is the image FileName and 'Path' (optional) is the relative path (to the current one) for selecting the directory.

Usage example:
Image = loadImage('baboon.pgm', '\PATH\TO\TpBiometry\public\Images\detection\');
Alternative usage example:
Image = loadImage('\PATH\TO\TpBiometry\public\Images\detection\baboon.pgm');

loadImagesInDirectory()

Images = loadImagesInDirectory(Path)

It loads all images present in a directory, automatically resizing them to 56 * 46 pixels; if N is the number of pictures in the directory, then the output will be a matrix of size N * 2,576.
'Path' is the relative path (to the current one) for selecting the directory.

Usage example:
Images = loadImagesInDirectory('\PATH\TO\TpBiometry\public\Images\train_A\');

plotFirst3Coordinates()

FigureHandle = plotFirst3Coordinates(Locations, NbrIndividuals, NbrFaces)

If 'Locations' contains the coordinates of the **projected** faces of various individuals, then this function plots the first 3 coordinates of the first 'NbrFaces' faces of the first 'NbrIndividuals' individuals. Of course, the coordinates of different faces of the same individual should be contiguous in 'Locations'. No more than 5 individuals are allowed.

Usage example:
plotFirst3Coordinates(Locations, 4, 3);

projectImages()

Locations = projectImages(ImgsVectors, Means, Space)

It projects vectors 'ImgsVectors' into the space spanned by vectors 'Space'. 'Means', the average of all the faces used to train the space, must be provided.

Usage example:
Locations = projectImages(ImgsVectors, Means, Space);

verify()

[DistancesClients, DistancesImpostors] = verify(Models, Test, FirstEigenfaces)

It computes scores for all clients and impostors.
'*FirstEigenfaces*' is the number of eigenfaces used for the identification. 'Models' and 'Test' contain respectively the coordinates of training models and test **projected** images. The same restrictions on 'Models' and 'Test' as for identify apply to verify.
Client (resp. impostor) scores are returned in 'DistancesClients' (resp. 'DistancesImpostors'); more precisely, these scores are the Euclidean distances of vectors in face space.

Usage example:
[DistancesClients, DistancesImpostors] = verify(Models, Test, 5);

References

[1] Matthew A. Turk and Alex P. Pentland, *Face Recognition Using Eigenfaces*, Proceedings of the IEEE conference on Computer Vision and Pattern recognition, Maui, Hawaii, 1991.
[2] Fernando A. Samaria, *Face Recognition Using Hidden Markov Models*, Trinity College, Cambridge University, 1994.
[3] Viisage, http://www.viisage.com