Contents

List of Figures

List of Tables

Chapter 1

State of the art

1.1 Facial Recognition

The challenge of face recognition can be formulated as followed : with one or several images of a face, the goal would be to find or check the identity of a person by comparing his face to all the face images stored in a database. By the way this skill remains the most acceptable because it more suits with what human beings use in visual interaction; and compared to other methods, the face recognition seems more advantageous, in fact it is a non-intrusive method, in other words it does not require the cooperation of the subject, and a moreover the sensors used are cheaper.

1.1.1 Facial recognition process

Any facial recognition process must take into consideration several factors that contribute to the complexity of its task, because a face is a dynamic entity which constantly changes under the influence of several factors. A facial recognition system is generally divided into the following steps (see the figure):

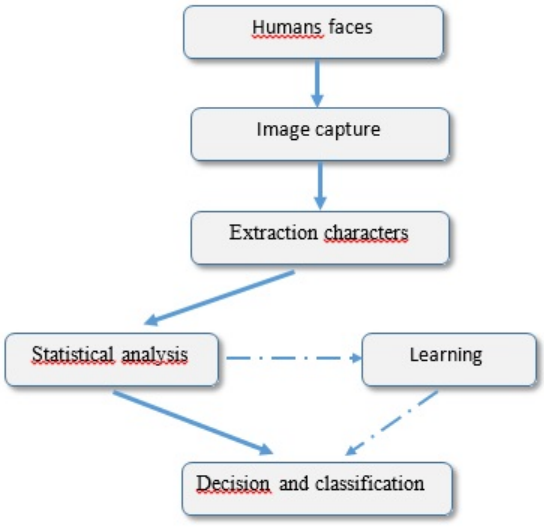

Figure 1 : Facial recognition system Process

Figure 1.1: **Facial recognition system process**

Facial recognition is facing the following problems:

- Change of pose ;
- Light Variations ;
- Variations of expression, age ;
- Partial occultation of the face (concealing).

These variations are the most difficult because the variations in the appearance of a person face according to different pose or light conditions are often far more important than the variation between face images of two different individuals acquired under the same conditions. This explains why pictures should be taken in specific conditions so that facial recognition can be efficient.

1.1.2 The methods used for face recognition

Facial recognition methods can be classified into two broad categories: local and global methods. Amongst those methods, main ones will be presented thereafter.

1.1.2.1 Global methods

Global methods are based on well known techniques of statistical analysis. In these methods, face images (which can be shown as matrices of pixel values) are used as input of the recognition algorithm and are generally transformed into vectors, which are easier to handle. The main advantage of global methods is that they are relatively quick to set up in. However, they are very sensitive to variations of illumination, pose and facial expression. The main existing methods are:

- The Principal Component Analysis (PCA) : EigenFaces
- The LDA (Linear Discriminant Analysis) Algorithm : FisherFaces

1.1.2.2 Local methods (Geometric)

The local methods include transformations applying to specific areas of the image, usually around characteristic points (corners of the eyes, mouth, nose, ...). Therefore, they require a priori knowledge on images. These methods are more difficult to implement but are more robust to the problems due to variations of illumination, pose and facial expression. The main existing methods are:

- EBGM (Elastic Bunch Graph Matching);
- Modular Eigenface;
- Hidden Markov Method.

But in fact, our aim on this project will be obviously to use both main global methods.

Both methods that we will present are using a common training algorithm steps that are :

- Preprocessing of training image set
- Normalization and estimation of mean image
- Use of PCA/ LDA

PCA/ LDA are statistical tools used to implement facial recognition method. For instance the use of PCA is divided into two steps :

- The determination of the input image weight from projecting input image into the face space and by multiplying the resulted vector to eigenfaces of the database.
- A Comparison of results with metrics such as euclidian distance.

1.2 Eigenfaces

1.2.1 Presentation of Eigenfaces

The Eigenface approach began with a search for a low-dimensional representation of face images. Sirovich and Kirby in 1987 showed that Principal Component Analysis could be used on a collection of face images to form a set of basis features. These basis images, known as Eigenpictures, could be linearly combined to reconstruct images in the original training set.The approach of using eigenfaces for recognition was developed by Sirovich and Kirby and used by Matthew Turk and Alex Pentland in face classification. Eigenfaces is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis set.

1.2.2 Procedure

A set of eigenfaces can be generated by performing a mathematical process called Principal Component Analysis (PCA) on a large set of images depicting different human faces. Informally, eigenfaces can be considered as a set of "standardized face ingredients", derived from statistical analysis of many pictures of faces. Any human face can be considered to be a combination of these standard faces.

To create a set of eigenfaces, one must:

- Prepare a training set of face images. The pictures constituting the training set should have been taken under the same lighting conditions, and must be normalized to have the eyes and mouths aligned across all images. They must also be all resampled to a common pixel resolution (r c). Each image is treated as one vector, simply by concatenating the rows of pixels in the original image, resulting in a single row with r c elements. For this implementation, it is assumed that all images of the training set are stored in a single matrix T, where each column of the matrix is an image.
- Calculate the average image by adding each columns of the matrix T and dividing the previous obtained vector by the number of input images.
- Subtract the mean from matrix T to obtain matrix A (where each element represents the luminance variance of each pixel). Once the average image a is calculated and it is then subtracted from each original image in T.
- Calculate the covariance matrix S.
- Calculate the eigenvectors and eigenvalues of the covariance matrix S. Each eigenvector has the same dimensionality (number of components) as the original images, and thus can itself be seen as an image. The eigenvectors of this covariance matrix are therefore called eigenfaces. They are the directions in which the images

differ from the mean image. Sort the eigenvalues in descending order and arrange eigenvectors accordingly.

- Choose the principal components. The number of principal components k is determined arbitrarily by setting a threshold on the total variance. Total variance v = n*(1+ 2+. . . + n), n= number of data images.
- k is the smallest number satisfies.
- Determinate the input image weight determination from projecting each image.
- Each image is represented by a vector which is used to reconstruct the images. We then save the average image, eigenfaces and the projection (weight) of images.

This ends the training part of the implementation of eigenfces and shows the skills used.

EigenFaces logigram The flowchart we have to use is divided into two basic parts: the learning phase and the identification phase where the Euclidean distance is used to calculate the difference between the weight of the image to be identified and the database images, then the program displays the nearest.

But retain before these two major steps,we have pretreatments and it's the phase which is carried out :

- The selection of the learning base ;
- Reading images ;
- The conversion of grayscale images ;
- Resizing images ;
- And finally the application of histogram equalization.

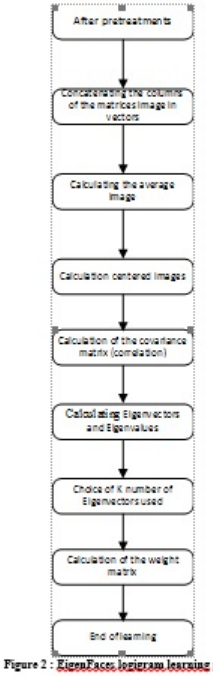

Figure 2 : EigenFaces logigram learning phase

Figure 1.2: **EigenFaces logigram learning phase**

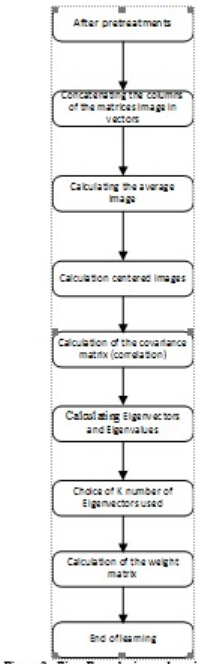

Figure 2 : EigenFaces logigram learning phase

Figure 1.3: **EigenFaces logigram identification phase**

1.2.3 Benefits and deficiencies

1.2.3.1 Benefits

Eigenface provides an easy and cheap way to realize face recognition in that:

- Its training process is completely automatic and easy to code.
- Eigenface adequately reduces statistical complexity in face image representation.
- Once eigenfaces of a database are calculated, face recognition can be achieved in real time.
- Eigenface can handle large databases.

1.2.3.2 Deficiencies

However, the deficiencies of the eigenface method are also obvious:

- Very sensitive to lighting, scale and translation; requires a highly controlled environment.
- Eigenface has difficulty capturing expression changes.
-
-

The most significant eigenfaces are mainly about illumination encoding and don't provide useful information regarding the actual face.

1.3 Fisherfaces

1.3.1 •

1.3.1.1 •

- Very sensitive to lighting, scale and translation; requires a highly controlled environment.
- Eigenface has difficulty capturing expression changes.
-
-

On souhaite dans cette partie transmettre un fichier binaire quelconque sur la chaîne de transmission.

Soit un fichier texte avec le nom "Thomas Le Bris" dedans, son code ASCII en décimal sera :

84 104 111 109 97 32 76 101 32 66 114 105 115

En ouvrant le fichier dans Matlab à l'aide de la fonction précédente nous obtenons bien le code prévu :

```
A =

    84
   104
   111
   109
    97
   115
    32
    76
   101
    32
    66
   114
   105
   115
```

Pour obtenir le code ASCII il nous suffit d'écrire Master 1 RTMA dans le fichier texte puis de l'ouvrir dans Matlab :

```
A =

    77
    97
   115
   116
   101
   114
    32
    49
    32
    82
    84
    77
    65
```

ADS nécessitant un fichier texte ne comportant que des '0' et '1' formatés en une seul colonne Nous allons utiliser Matlab afin d'effectuer la conversion d'un fichier texte en binaire puis son formatage approprié.

cf fichier joint conv2ADS.m

ADS nous renvoyant en sortie un fichier formaté de la même façon il nous réutiliserons Matlab afin de reconvertir le fichier traité par la chaîne de transmission dans son format original.

cf fichier joint ADS2ASCII.m

En transmettant le fichier dans des conditions favorables nous obtenons bien un fichier identique mais en augmentant le bruit des erreurs apparaissent sur le fichier en sortie :

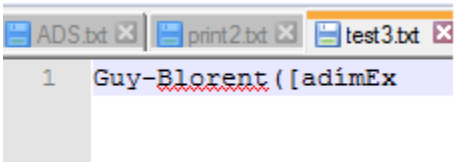

Figure 1.4: TEB = 0.0057

Jusqu'à devenir complètement différent du fichier original avec un TEB égal à 0.5 :

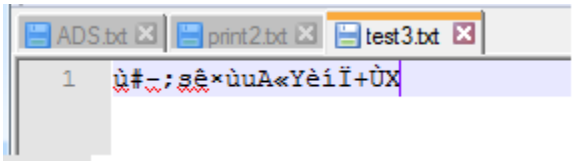

Figure 1.5: TEB = 0.5

Cela est dû au fait que le bruit transforme des 0 en 1 et des 1 en 0 ce qui fausse donc la conversion lors du retour en ASCII.