01/06/2015

Jihade TIKA - Morgan LIENARDY Projet Traitements d’images

IMR2

**EIGENFACES**

Reconnaissance faciale

Table des matières

[Introduction 3](#_Toc421830151)

[Étude préliminaire 4](#_Toc421830152)

[Spécification 5](#_Toc421830153)

[Mise en place de la solution 8](#_Toc421830154)

[Résultats de la reconnaissance faciale 10](#_Toc421830155)

# Introduction

Le projet consiste à proposer un système de *reconnaissance automatique de visage(s)*, basée sur l’approche *eigenfaces*.

Le principe de la reconnaissance faciale est très simple : comme son nom l’indique, il s’agit d’identifier une face donnée. Selon les algorithmes et les buts recherchés, cette identification peut prendre plusieurs aspects: il peut s’agir de déterminer à qui appartient un visage, de décider si oui ou non ce visage est reconnu, ou même dans certains cas de déterminer s’il s’agit bien d’un visage.

On cherche à spécifier le travail à réaliser en représentant sous forme de blocs les différentes fonctionnalités à implémenter pour cette reconnaissance de visages.

Dans un premier temps on parlera de l’étude préliminaire quant à *eigenfaces*, puis dans un second temps la schématisation des fonctions de traitements, et finalement des résultats obtenus après la mise en œuvre de la solution.

# Étude préliminaire

La reconnaissance faciale s’inscrit dans le domaine plus vaste de la vision par ordinateur, qui part du constat que le sens le plus utilisé par l’homme est la vue.

Dès lors, ainsi, l’ordinateur peut devenir capable de remplacer ceux de l’homme pour des tâches répétitives telles que la reconnaissance de nombreux visages. Le principe de la reconnaissance faciale est d’identifier une face donnée. L’algorithme des eigenfaces permet de déterminer si le visage est reconnu, pas reconnu ou si ce n’en est pas un.

Nous avons effectué plusieurs recherches sur internet afin de déterminer une méthode précise que nous utiliserons dans notre implémentation pour aboutir à une reconnaissance faciale avec l’approche eigenfaces.

Cette méthode constitue plusieurs étapes résumées par le schéma suivant :

Eigenfaces

Initialisation

BDD

Classification --> Résultat

Insertion d’image

Initialisation

# Spécification

Nous spécifions le système en termes de blocs fonctionnels, à différents niveaux de granularité, afin que chaque solution proposée soit validée antérieurement à toute phase de codage proprement dite.

1) Traitement des données en base

2) Ajout d’un visage

Résultat

3) Classification

Base de données

Moyenne

Covariance

SVD

Eigenface

Weight

Distance euclidienne

**Comparaison par rapport à 2 seuils**

Descripteur

1. Traitement des données en base

Base de données

Moyenne

Covariance

SVD

Descripteur

Nous considérons **M** : les images en base, et **p** : les pixels d’une image.

Base de données

Matrice de base B = [p, M]

Matrice avec M colonnes et p lignes. Correspond aux images et leurs pixels

Moyenne

Vecteur moyenne m = [p, 1]

Plus qu’une colonne avec la moyenne de chaque pixel

Covariance

Matrice de covariance C = [p, p]

Avec C = m.m’ (covariance sous matlab)

SVD

3 Matrices de sortie (U, V, S) et U eigenvector = [M, p]

Ne pas oublier de prendre que les 48 premières images

Descripteur

Projection des images sur les eigenfaces d = [M, 48]

Avec d = (B – m).U

2) Ajout d’un visage

Eigenface

Weight

Distance euclidienne

Nous considérons **v** : un nouveau visage

Eigenface

Uinput eigenvector = [1, p]

On récupère les composantes eigenface du nouveau visage

Weight

W = [1, p]]

Compare our input image with our mean image and multiply their difference with each eigenvector

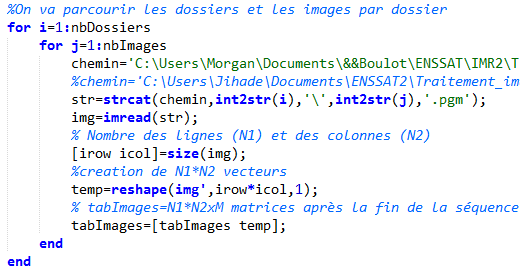
🡪 W = U.(v – m)

Distance euclidienne

Ƹk = | mk - m |²

# Mise en place de la solution

Il faut dans un premier temps récupérer les images de notre base de données :



La variables ‘tabImages’ contient nos ‘nbImages’ images présents dans les ‘nbDossiers.

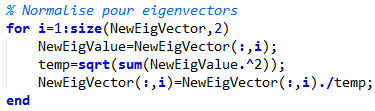
Ensuite on normalise nos images avec la moyenne et lla deviation standard :



On calcule nos eigenVecteurs et eigenValeurs à partir de la matrice (ici L)



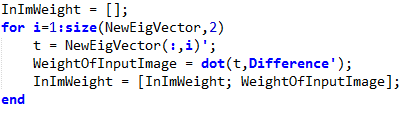
On normalise les eigenVecteurs



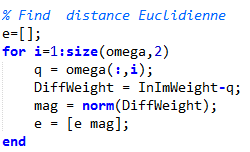
Ensuite, il ne reste qu’à insérer une ou plusieurs images de test :



Calcul du poids avec les nouvelles eigenVecteurs :



La distance euclidienne :



# Résultats de la reconnaissance faciale

Pour nos tests nous avons pris une base d’apprentissage qui représente le 1/10 de la base de données qui nous a été fournie avec l’énoncé du projet.

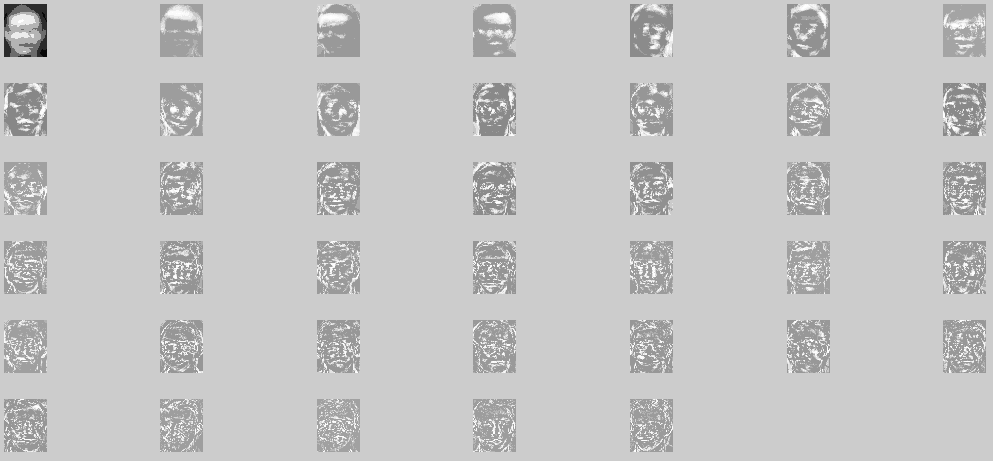
Voici notre base d’apprentissage normalisée :



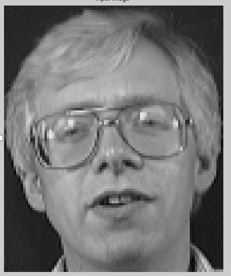
L’image moyenne :



Les visages propres sont présentés ci-dessous.



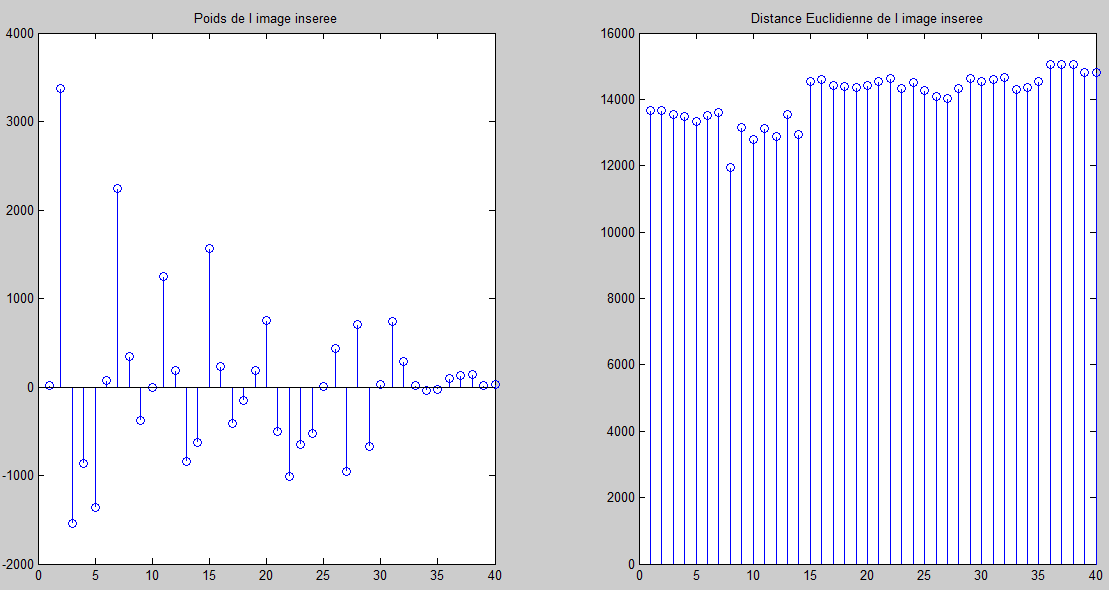
Nous voulons maintenant reconnaitre cette image :

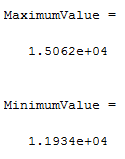


Jusqu’ici nous n’avons pas encore montré le résultat. Pour cela dans la première série d’images, nous avons entré une image connu ci-dessus, et nous avons observé la distance euclidienne. Cette distance montre à quel point l’image d’entrée est proche des images que nous avons eues dans notre base d’apprentissage.

La distance euclidienne maximale pour un visage est d’environ 15000 et la distance euclidienne minimale est d’environ 11000.

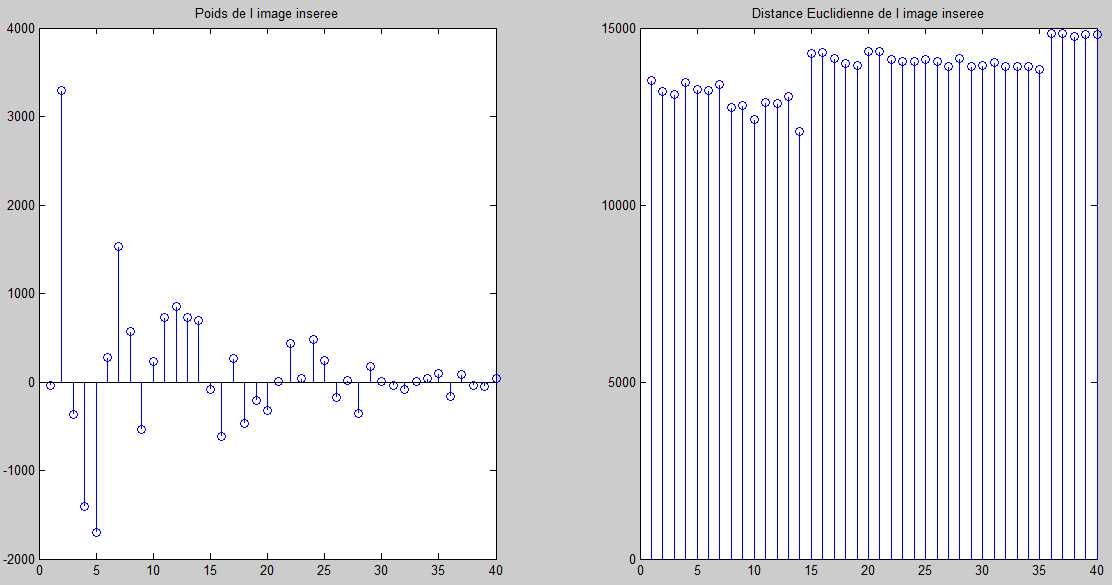
Basé sur ces distances, nous pouvons décider si le visage est un visage connu, un visage inconnu, ou ce n’est pas du tout un visage.

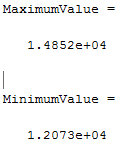
Ces deux images sont prises de notre base d’apprentissage :



Nous remarquons que les distances maximale et minimale sont dans la fourchette établie.

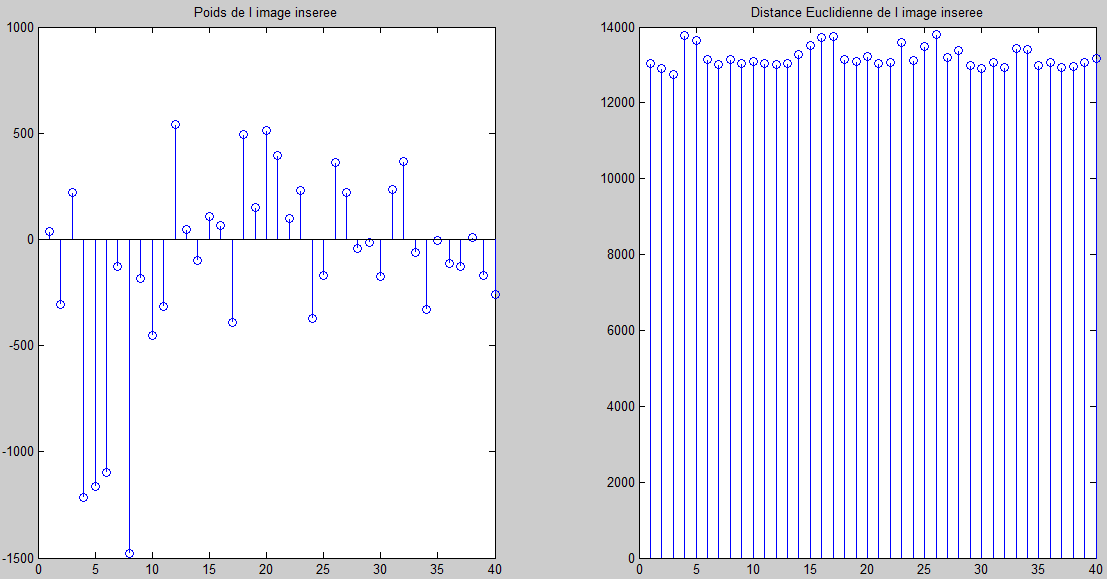
Voici une autre image de notre base d’apprentissage :

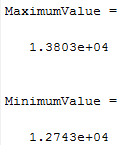




Ici encore une fois, les distances maximale et minimale sont dans la fourchette établie.

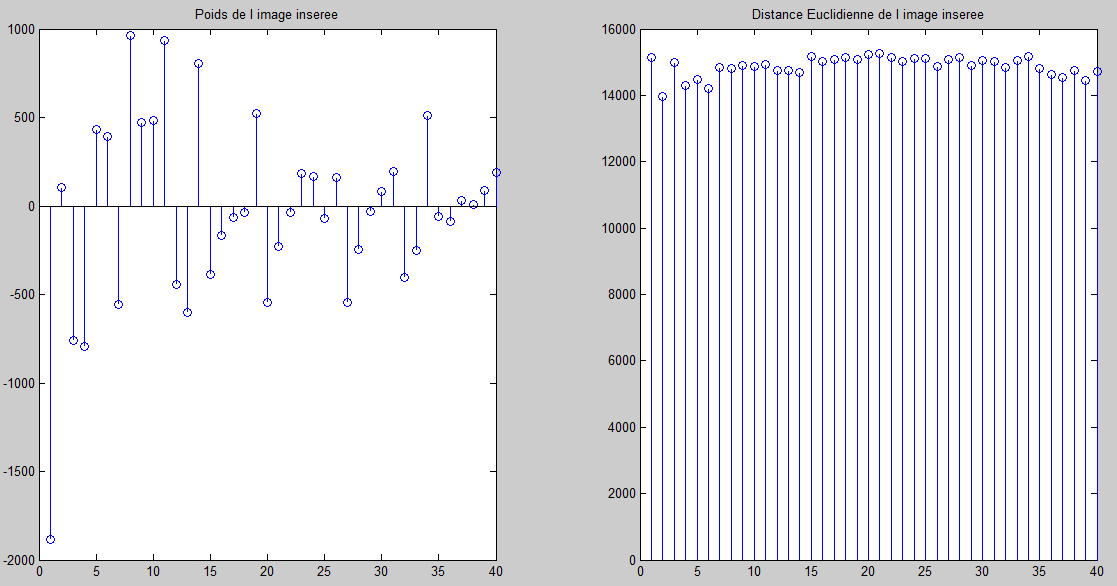
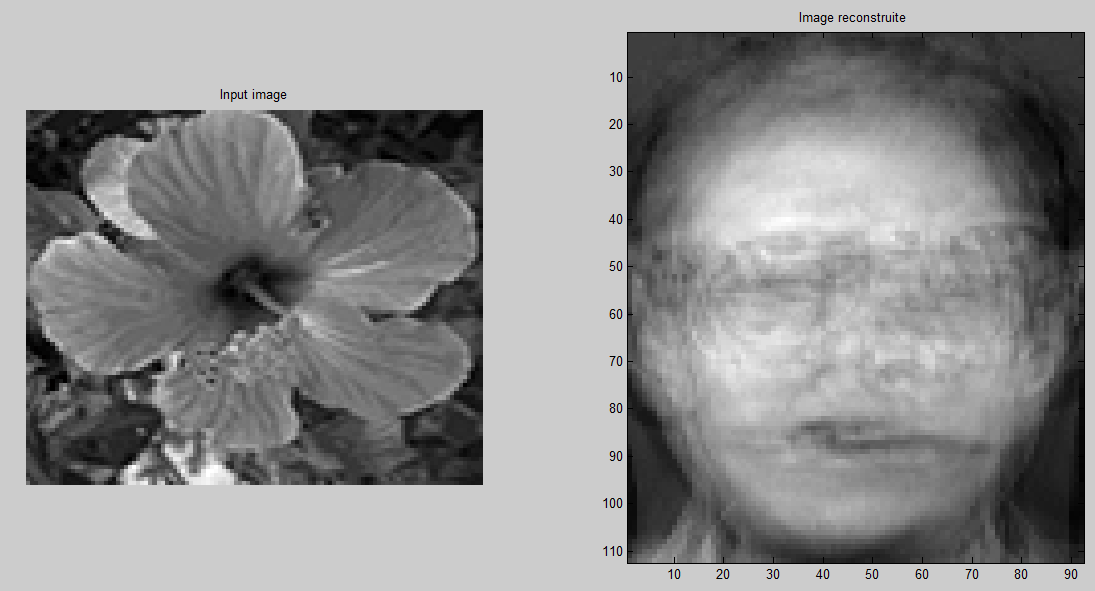
Ensuite, nous avons utilisé un visage non connu en entrée, et nous avons observé les résultats suivant : 

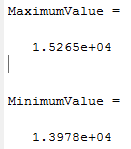




D’une part l’image peut être déterminée comme étant un visage parce que la valeur de la distance maximale se trouve dans l’intervalle établi précédemment. D'une autre part, la valeur minimale est supérieure comme cela a été prévu.

Dans les images suivantes nous avons utilisé en entrée une image qui n’était pas un visage.





Nous pouvons constater que la valeur maximum 15000, et que la valeur minimale est très éloignée de 11000. En conséquence, l’image est classée comme étant autre chose qu’un visage.