ABENZOAR Manoah

BACHA Nawfel

HAFDI Ayoub

A l'attention de Mr. BOURHATTAS

TP Reconnaissance faciale

Table des matières

1- Introduction.....3

2- Travail mathématique demandé 4

3- Analyse du travail de codage….....5

A- Explication du choix de la valeur de k……*5*

B- Gain de mémoire obtenu en %.....*5*

C- Efficacité de cette méthode et nécessité du seuil de distance……*6*

1- Introduction

Ce TP a pour but de nous faire découvrir une des premières techniques de reconnaissance faciale, celle utilisant les "eigenfaces". Cette technique de reconnaissance utilise la méthode d'analyse en composantes principales (ACP) oula méthode de décomposition en valeurs singulières (SVD). L'idée centrale est de diminuer la dimension de l'espace de travail pour simplifier les données et leur interprétation. Une fois cette dimension réduite, la comparaison d'un nouveau visage à ceux de la base de données devient moins coûteuse ce qui permet d'espérer un bon taux de réussite. Les "eigenfaces" sont obtenues lors d'une première phase d'apprentissage. Il s'agit tout simplement des composantes principales de la matrice regroupant les images des visages d'une partie de la base de données. Elles vont servir de base pour exprimer chacune des images. En se limitant aux premières composantes (une fraction qui représente un pourcentage suffisant de l'inertie totale) on réduit la dimension de représentation des images. Ces composantes principales peuvent également être obtenues par la décomposition en valeurs singulières d'une matrice étroitement liée à celle de la base de données comme on va le voir dans la partie mathématique.

2- Travail mathématique demandé

1) $$|||A|||_2 = \sqrt{\rho(A^*.A)} = \sqrt{\rho(A.A^*)} = \sigma_1(A)$$

2) On a $C = Y^T Y$ et $C^T = (Y^T Y)^T = Y^T (Y^T)^T = Y^T Y$ est symétrique réelle donc elle est diagonalisable dans une base orthonormée.

3) Les valeurs propres de $C = Y^T Y$ sont positives. En effet, si Xi est un vecteur propre associé à la valeur propre λ.

$X_i^T C X_i = X_i^T Y^T Y X_i$

$= (Y X_i)^T Y X_i$

$= ||Y X_i||_2^2 \geq 0$

et $X_i^T C X_i = X_i^T \lambda X_i = \lambda X_i^T X_i = \lambda ||X_i||_2^2$

Donc : $\lambda ||X_i||_2^2 \geq 0$ or $||X_i||_2^2 \geq 0$ $\Rightarrow \lambda \geq 0$

4) On a $C = Y^T Y$ or elle est diagonalisable.

$C = PDP^{-1}$ d'où $Y^T Y = PDP^{-1}$

De plus $Y = U \sum V^T$

D'où $C^{-1} = (U \sum V^T)^T (U \sum V^T)$

$\Leftrightarrow$ $C^{-1} = V \sum^T U^T U \sum V^T$

$\Leftrightarrow$ $C^{-1} = V \sum^T \sum V^T$

C étant diagonalisable dans une base orthonormée, les colonnes de la matrice de passage P forment cette base orthonormée. Donc P est une matrice orthogonale.

Par identification : $P = V$ et $D = \sum^T \sum$

5) Les composantes d'un individu normalisé dans la base orthonormée sont obtenues en faisant le produit scalaire des lignes de P avec le colonnes de Y, c-à-d YP.

3- Analyse du travail de codage

Nous avons utilisé certaines librairies dans notre code Python, veillez à bien tout installer si jamais des erreurs apparaissent. Pour ce faire, il vous faut installer pip et ensuite taper « pip install (librairie souhaitée) » dans l'invite de commandes.

A- Choix de la valeur de k et gain de mémoire obtenu en %

Pour afficher les mémoires CPU et RAM utilisées, nous avons utilisé le module psutil.

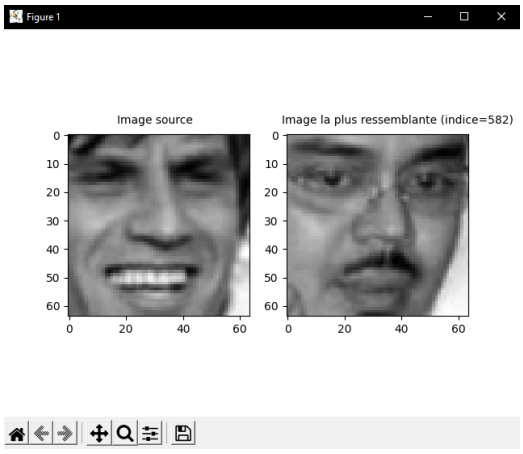
En mettant k à 70, on obtient :

```
Utilisation CPU : 34.4 %
Utilisation RAM : 47.2 %
```

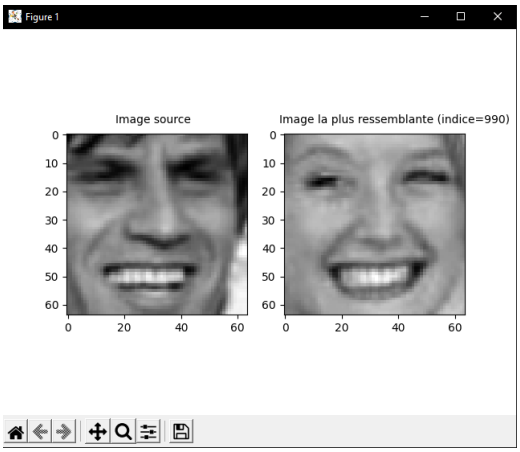
En mettant k à 51, on obtient :

```
Utilisation CPU : 27.3 %
Utilisation RAM : 47.1 %
```

On observe donc un gain de 7,1 % sur l'utilisation du CPU lorsque k = 51.



k=50

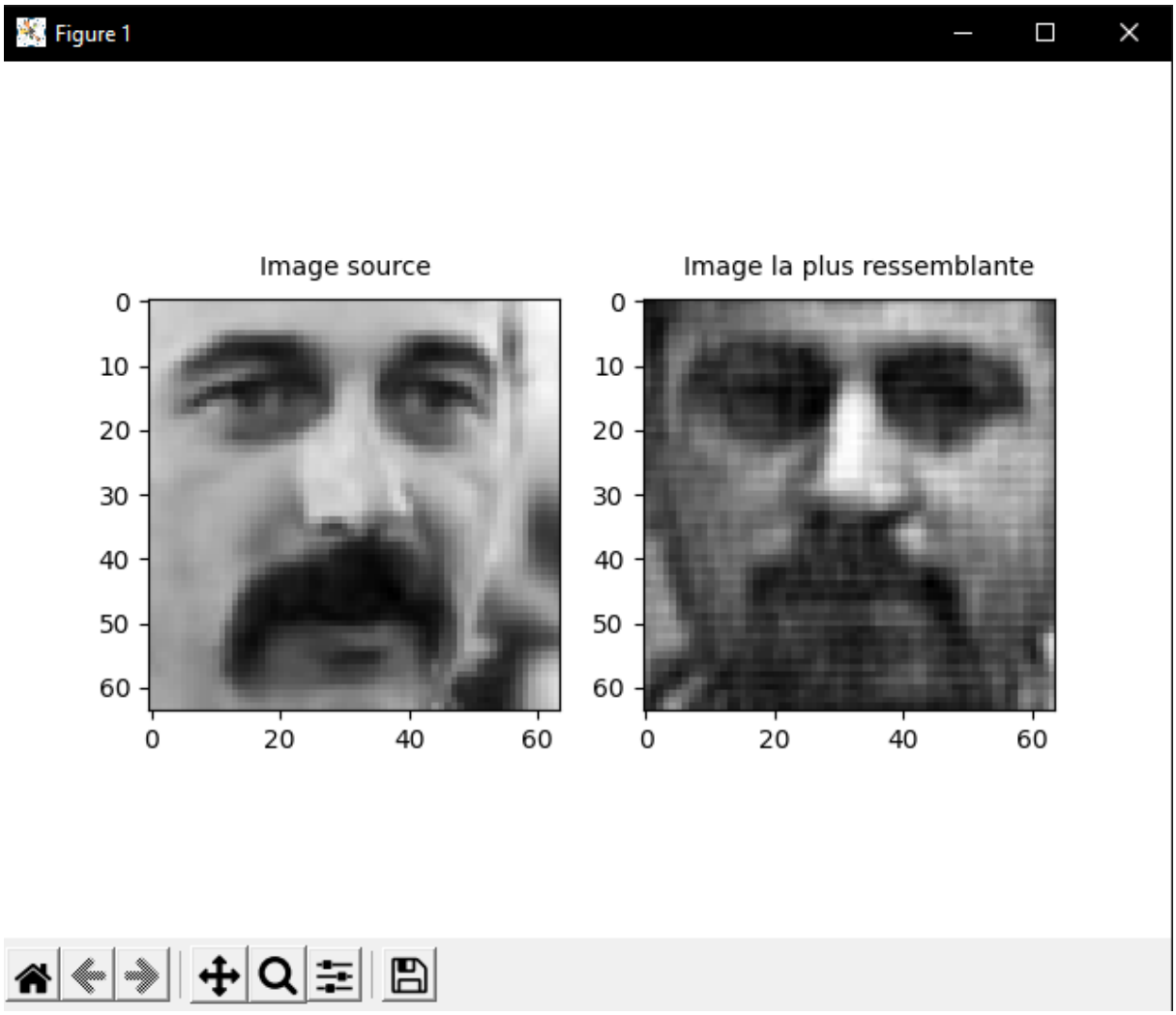


k=51

En mettant k à 50, on obtient un visage qui ne ressemble pas à l'image source, cependant en mettant k à 51, c'est ressemblant. C'est donc pour cela que nous avons choisi cette valeur qui permet d'obtenir une ressemblance tout en utilisant peu de ressources mémoire.

B- Efficacité de cette méthode et nécessité d'un seuil de distance

Globalement, la méthode est assez efficace si l'image source se trouve également dans le dossier d'apprentissage. Cependant, si ce n'est pas le cas (voir exemple ci-dessous), le programme aura tendance à choisir une image ressemblante sur une seule partie du visage. C'est pour cela qu'il est important d'instaurer un seuil de distance au-dessus duquel on peut décider qu'il n'y a pas de reconnaissance. En plaçant un seuil à 1 par exemple, on évite d'avoir des reconnaissances avec des images trop différentes.



C- Analyse des méthodes utilisées

M est la matrice composée de la ligne avec les moyennes et la ligne avec les écarts-types.

Z contient les informations de chaque image (100) et k = 51.

PK est la matrice des eigenfaces.

```
Dimensions des matrices :

M = (2, 4096)
PK = (4096, 51)
Z = (100, 51)
```

Z a changé de dimension car il contient les informations de toutes les images de BDD1.

```
Dimensions des matrices :

Z = (1680, 51)
```

Nous avons pu afficher l'utilisation du CPU et de la RAM à l'aide du module « psutil ».

```
Utilisation CPU : 34.4 %
Utilisation RAM : 47.2 %
```

Pour une analyse plus détaillée du code, veuillez-vous référer au fichier main.py où vous trouverez des commentaires expliquant les différentes étapes des méthodes.