LEGUEN Yohann **KAMDEM Adrien** **ING1 GMI2**

A l'attention de Zoghlami Naïm

Rapport TP Analyse Numérique noté: Reconnaissance faciale

I) Travail Mathématique Demandé

1)

$$Soit\ A \in M_{n,p}(\mathbb{R})\ et\ x \in \mathbb{R}^p,\ la\ fonction :$$
$$|||A|||_2 = sup_{x \neq 0} \frac{||Ax||_2}{||x||_2} \quad (1)$$
$$est\ la\ norme\ matricielle\ subordonée\ 2$$

2)

$$Soit\ Y \in M_{n,p}(\mathbb{R})\ alors$$
$$Y^tY\ et\ YY^t\ sont\ diagonalisables\ dans\ une\ base\ orthonormée. \quad (2)$$

3)

$$Soit\ Y \in M_{n,p}(\mathbb{R})\ alors\ les\ valeurs\ propres\ de\ YY^t\ et\ Y^tY\ sont\ positives$$

4)

$$On\ a :$$
$$C = Y^tY$$

– *La décomposition en valeur singulière de* Y *nous donne* :

$$Y = U\Sigma V^t$$
$$Y^t = V\Sigma^t U^t$$
avec U et V des matrices orthogonales et Σ une matrice diagonale.

$$- \text{On obtient : } C = V\Sigma^t U^t \, U\Sigma V^t$$

– Comme V et U sont des matrices orhtogonales alors :
$$U^t U = I_n$$
$$V^t V = I_p$$
$$\text{on a alors : } C = V\Sigma^t \Sigma V^t$$

$$- \text{De plus } C \text{ s'écrit sous cette forme :} \qquad (3)$$
$$C = P\,D\,P^t$$

– Avec P une matrice orthogonale et D une matrice diagonale
Par identification, P étant orthogonale, $P = V$,
D étant diagonale, $D = \Sigma^t\Sigma$, *et* $P^t = V^t$.

5)

Les composantes d'un individus normalisé (une ligne de Y) dans cette base sont obtenues à l'aide du produit matriciel $Z = YP$ *avec Y (les individus), P (Vecteurs de la base orthonormée).*
On obtient ainsi les coordonnées des individus dans cette base.

Explications plus explicites :

$$Y = \begin{matrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{matrix} \begin{pmatrix} \text{caractéristique de l'ind } 1 \\ \text{caractéristique de l'ind } 2 \\ \\ \text{caractéristique de l'ind } p \end{pmatrix} \qquad P = \begin{pmatrix} e_1 & e_2 & \dots & e_p \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

$$Z = \begin{matrix} \\ y_1 \\ y_2 \\ \vdots \\ y_p \end{matrix} \begin{pmatrix} e_1\ e_2 \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\ e_p \\ \text{coordonnées ind1 dans la base } e \text{ (composantes 1 à } p \text{ du produit matricielle pour ind 1)} \\ \text{coordonnées ind2 dans la base } e \text{ (composantes 2 à } p \text{ du produit matricielle pour ind 2)} \\ \vdots \\ \text{coordonnées indp dans la base } e \text{ (composantes 1 à } p \text{ du produit matricielle pour ind } p) \end{pmatrix}$$

II) Travail de codage demandé

Objectif du TP:

Nos disposons de deux bases de données BDD1 et BDD2 contenant un grand nombre d'images représentant des célébrités au format 64*64 (pixels). L'objectif de ce Tp noté est de développer une solution sur scilab permettant la reconnaissance faciale de ce format d'image.

Les outils développés pour résoudre le problème:

Nous avons d'abord selectionné une base d'apprentissage dans la BDD1 (les 100 premières images de la BDD1) nommée Z (matrice des 100 images). Cette base d'apprentissage est la base sur laquelle nous allons nous appuyer pour reconnaitre les visages. Avant d'utiliser cette base d'apprentissage qui pour le moment est constitué de 100 images de taille 64*64(pixels), nous allons:

- Mettre toute les images de cette base dans une matrice Y. Chaque ligne correpondra à une image. Une image de 64*64 se transformera en image de 1*4096 *(Fonction ligne)*. D'ou la matrice Y aura une taille de 100*4096 *(Fonction construct)*. Cette matrice Y crée, il faudra par la suite la normaliser pour que aucune valeur ne fausse les résultats *(Fonction normalisation).*

- Par la suite notre objectif va être de réduire le nombre de dimension de la matrice Y en ayant une perte d'information minimale. Pour ce faire nous utiliserons la décomposition en valeurs singulières *(Fonction valeurSingu)* qui est liée à l'ACP comme vu précédemment pour réduire la dimension de cette matrice *(Fonction reductionDim)*.

Fonction ligne:

Avec le module SIVP et la fonction imread, nous transformons une image en matrice 64x64.
Pour la suite du TP (cf fonction construc) nous avons besoin de transformer cette matrice carrée en matrice ligne de 1x4096. Voici un exemple, pour une matrice de taille 5x5 :

```
-->A=rand(5,5)
 A  =

    0.2113249    0.6283918    0.5608486    0.2320748    0.3076091
    0.7560439    0.8497452    0.6623569    0.2312237    0.9329616
    0.0002211    0.6857310    0.7263507    0.2164633    0.2146008
    0.3303271    0.8782165    0.1985144    0.8833888    0.312642
    0.6653811    0.0683740    0.5442573    0.6525135    0.3616361

-->X=ligne(A)
 X  =


         column 1 to 6

    0.2113249    0.6283918    0.5608486    0.2320748    0.3076091    0.7560439

         column  7 to 12

    0.8497452    0.6623569    0.2312237    0.9329616    0.0002211    0.6857310

         column 13 to 18

    0.7263507    0.2164633    0.2146008    0.3303271    0.8782165    0.1985144

         column 19 to 24

    0.8833888    0.312642    0.6653811    0.0683740    0.5442573    0.6525135

         column 25

    0.3616361

-->size(X)
 ans  =

    1.    25.
```

Figure 1: Mise en ligne d'une Matrice carrée

Fonction construc:

Cette fonction va nous permettre de construire la Matrice contenant les images de la BDD1.
Dans un premier temps, nous allons ajouter les 100 premières images de BDD1.Pour cela, nous
lisons chaque image que nous plaçons dans une matrice A (donc de taille 64x64), nous faisons la fonction ligne sur l'image A que nous placons sur la première ligne de X, on repète l'opération pour les 99 prochaines images.

```
-->X=construc();

-->size(X)
 ans  =

    100.    4096.
```

Figure 2: Construction de la matrice X de taille 100x4096 (Matrice des images de la BDD1)

Fonction meanAndStdev:

Comme vu dans la partie Mathématique, nous voulons travailler avec la Matrice C, qui est la matrice de corrélation. Pour obtenir cette matrice, on doit tout d'abord obtenir la matrice Y, qui correspond à la normalisation de la matrice des images X. Pour cela, nous passons par une fonction intermerdiaire, qui va construire une matrice de taille 2x4096, la première ligne correspondant à la moyenne des colonnes de X et la deuxième ligne étant l'écart-type des colonnes de X. Nous sauvegardons ce résultat dans la matrice M , que nous réutiliserons par la suite.

Voici le résultat pour une matrice de taille 3x3 :

```
-->X=[[1,2,3];[2,3,4];[3,4,5]]
 X  =

    1.    2.    3.
    2.    3.    4.
    3.    4.    5.

-->M=meanAndStdev(X)
 M  =

    2.    3.    4.
    1.    1.    1.

-->
```

Figure 3: Matrice des moyennes et Ecart-type de X de taille 3x3

```
->M=meanAndStdev(X);

->size(M)
ans  =

    2.    4096.
```

Figure 4: Matrice des moyennes et Ecart-type de X des 100 images (100x4096)

Fonction normalisation:

Comme son nom l'indique, la fonction normalisation va permettre de construire la matrice Y, qui est la matrice normaiser de X: a chaque terme de la matrice, on soustrait la moyenne et on divise par l'Ecart-type associés à la colonne en question.

```
-->X=[[1,2,3];[2,3,4];[3,4,5]]
 X  =

    1.    2.    3.
    2.    3.    4.
    3.    4.    5.

-->M=meanAndStdev(X)
 M  =

    2.    3.    4.
    1.    1.    1.

-->Y=normalisation(X)
 Y  =

  - 1.  - 1.  - 1.
    0.    0.    0.
    1.    1.    1.
```

Figure 5: Normalisation de X

Fonction valeur singu:

Cette fonction va faire une décomposition en valeurs singulieres de Y qui permettra de retrouver comme vu dans la partie mathématique la diagonalisation de la matrice de corrélation C, en identifiant P, la matrice de passage dans la base orthonormée et D (matrice diagonale de la diagonalisation). Cette matrice D composée dans sa diagonale des valeurs propres servira au calcul de l'inertie des vecteurs propres et permettra donc le choix d'un certains nombres de dimensions en fonction du pourcentage d'information que nous voulons conserver après réduction de la dimension. Après réalisation de la SVD sur Y ([U,S,V]=svd(Y) sur scilab), nous obtenons P=V, et D=S'S.

Fonction reductionDim:

La svd effectuée précedemment, nous a permi de diagonaliser la matrice des corrélations, nous allons maintenant réduire le nombre de dimension. Pour cela, il faut suivre les étapes de l'ACP. Il faut tout d'abord trier la matrice D par ordre décroissant des valeurs propres. Cette étape est réalisée directement lors de la svd. Ensuite, on va calculer l'inertie de chaque sous matrice de D, jusqu'a tomber sur un pourcentage que nous avons fixé a 95%.

Avec ce pourcentage fixé a 95%, notre programme nous retourne k=53. Nous réduirons alors la matrice P a uniquement ses 53 premieres colonnes.

```
-1->i=reductionDim(D)
 105503Dimension = 0.950838 i  =

    53.

-1->
```

Figure 6: Réduction de la dimension pour 95% de l'inertie

Il suffiera donc de faire P_k= P(:,1:k); avec k qui correspond au k retourner.

Le problème mis sous forme d'algorithme à l'aide des outils développés précédemment:

Une fois tout les outils développés, nous cherchons à mettre alors le problème sous forme d'algorithme afin de retourner une reconnaissance faciale fonctionelle. Voici ainsi les fonctions que nous développerons dans cette partis pour résoudre le problème de reconnaissance faciale.

Fonction Apprentissage:

Dans un premier temps, il faut pouvoir à l'aide d'une fonction appliquer tout les outils sur notre base d'apprentissage prédéfinit. Nous avons donc appliquer cette fonction à notre base d'apprentissage de 100 images de la BDD1. En effet cette fonction prend en entrée, la matrice des 100 images X , fait un appel à la suite de la fonction *construc, meanAndStdev, normalisation, valeurSingu* ainsi que *reductionDim*.
Cette fonction retourne donc la matrice M des moyennes et écarts-type la matrice P_k , matrice de la base orthonormée, réduite à ses k premieres composantes, et enfin la matrice Z, correspond aux coordonnées de chaque image dans la base P_k.

Fonction Ajout:

La fonction Ajout prend en entrée une nouvelle image ainsi que les matrices M, P_k et Z. A l'aide de cette fonction nous pouvons ajouter des images de taille 1*4096 à une matrice si nous voulons agrandir la base d'apprentissage Z. Nous lisons dans un premier temps l'image que nous plaçons dans une matrice 64x64, nous lui appliquons la fonction ligne pour la

transformer en matrice 1x4096, nous centrons et réduisons cette ligne , que nous mettons dans la matrice Yp et enfin, nous calculons ses coordonnées $Z_p = P_k \times Y_p$, pour ajouter la ligne Z_p à notre matrice Z

Reconnaissance faciale:

Notre fonction reconnaissance_faciale prend en paramètre une nouvelle image (le numéro de l'image) ainsi que les matrices M, P_k et Z. Elle retourne la position de l'image dans la matrice Z, ce qui correspond à l'identité de l'individu.
Tout d'abord, nous avons créé une fonction copieBDD, qui va effectuer les memes instructions que la fonction ajout, mais pour le reste des images de la BDD1 : nous nous retrouvons alors avec une matrice Z de taille 1680x53, décrivant la totalité de la BDD1.
Nous pouvons maintenant passer à la reconnaissance d'une image. Pour cela, on commence par faire les mêmes instructions que dans ajout pour calculer les coordonnées Z_p de l'image en entrée, c'est à dire, la mettre en ligne, la centrer et la réduire.
Ensuite, pour chaque ligne de Z (donc pour chaque image), nous allons calculer la distance de notre image à cette ligne (en utilisant la norme 2) , puis ranger le résultat dans une matrice d, de taille 1x4096. Une fois la matrice d obtenue, nous récupérons la valeur du minimum ainsi que sa position dans la matrice (la position du minimum dans d étant la même que la position de l'image que souhaitons faire correspondre dans Z). Il ne nous reste plus qu'à retourner les coordonnées de l'image, qui correspond donc à la ligne d'indice "position". Mais avant cela, nous instaurons un seuil, arbitrairement fixé à 29: si le minimum de la distance est au dessus de ce seuil, c'est que l'image n'est pas assez proche d'une autre, donc elle n'est pas reconnue, au contraire, si le minimum est en dessous de 29, on retourne la ligne correspond à la reconnaissance, et nous affichons les 2 images, par la fonction imshow du module SIVP.

```
-1->cooImg=reconnaissance_faciale(M,Pk,Z,1323)
 cooImg  =

    1323.
```

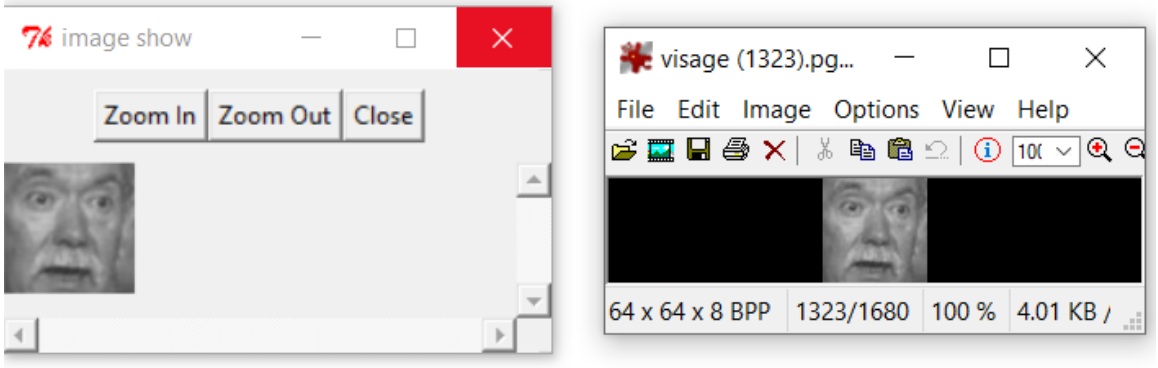

Figure 7: Reconnaissance de l'image visage(1323)

Pour tester la non-reconnaissance d'une image, nous avons enlever la dernière ligne de Z, donc l'image 1680 n'est plus dans la base d'apprentissage. Nous effectuons donc la recherche d'image ,avec le visage(1680) en entrée.

```
-1->Z=Z(1:1589,:);

-1->size(Z)
 ans  =

    1589.    53.

-1->cooImg=reconnaissance_faciale(M,Pk,Z,1680)

 Aucune image n a été reconnue
 cooImg  =

  - 1.
```

Figure 8: Reconnaissance faciale qui n'a pas trouvé d'image correspondante

Conclusion:

La précision de notre algorithme est à discuter.

Par exemple:

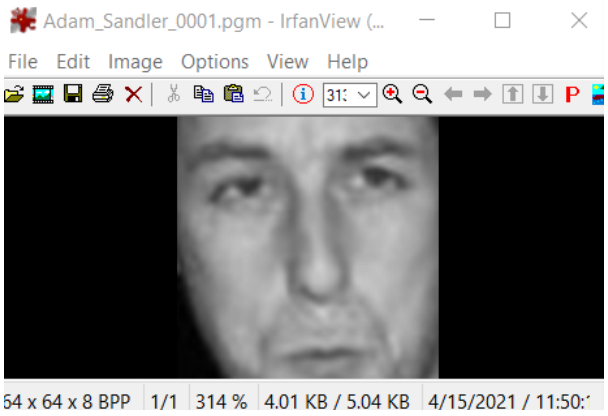

Figure 9: Image de Adam Sandler, provenant de la BDD2

Nous cherchons à faire une reconaissance faciale de cette image dans la base de d'apprentissage. Voila ce que notre algortihme de reconnaissance faciale nous retourne:

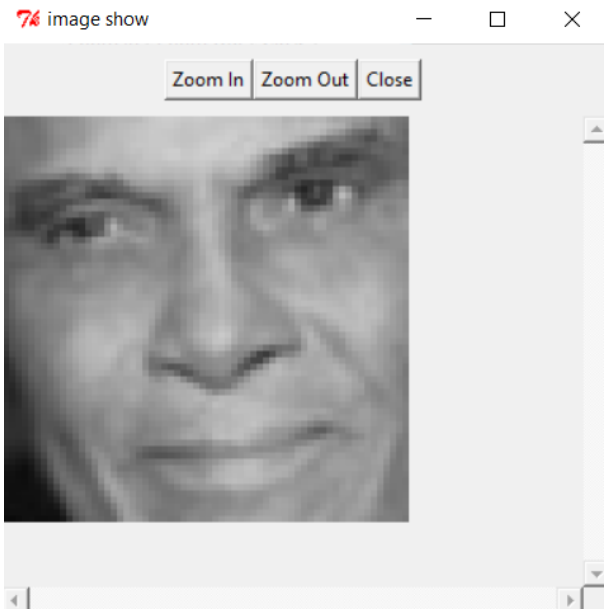

Figure 10: Image reconnue par le programme de reconnaissance faciale

Or il existe dans la base d'apprentissage une image de ADAM SANDLER. La voici:

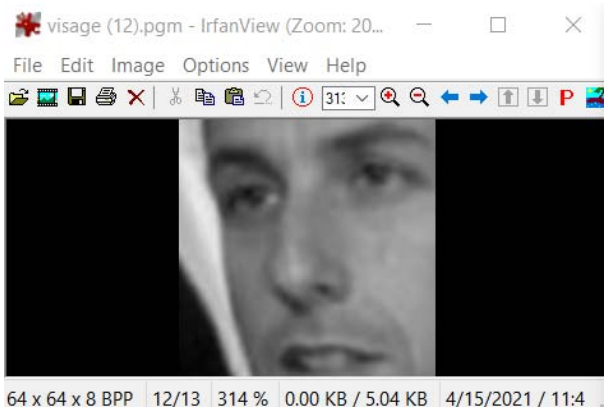

Figure 11: Visage(12) de la BDD1, dont l'identité est Adam Sandler

Explications:

L'image que l'algorithme trouve n'est pas celle d'adam sandler mais lui très proche de part la disposition et la couleur des pixels.

Ces images sont ressemblantes en terme de clarté, les visages sont de face, ont des traits similaires au niveau du nez, des faussettes et de la bouche, les regards sont moins

orientés sur le coté. Tout cela fait que l'image retournée semble plus proche en terme de distance de pixels que l'image devant être reconnue par l'algorithme. Cela pose alors une question sur la précision d'un algorithme de reconnaissance faciale basée sur la distance de pixels.

Propositions d'améliorations:

On pourrait en effet éliminer le problème auquel nous avons fait face avec la reconnaissance faciale du visage de Adam Sandler. En effet nous pourrions identifier la position de certains éléments clés du visage et les comparer entre eux plutot que de comparer les pixels un à un. Ceci rendrait alors plus précis notre algorithme de reconnaissance faciale.