



## TP-Projet 3 : Application de l'ACP : les «Eigenfaces»

Ce projet s'inspire d'un article intitulé *Eigenfaces for recognition*, écrit par Turk et Pentland et publié dans le *Journal of Cognitive Neuroscience* en 1991.

### Description des données

Vous disposez de  $n$  images de visages d'un ensemble d'individus. Chaque individu est photographié sous le même nombre de postures faciales (gauche, face, trois quart face, etc.). Chacune de ces  $n$  images en niveaux de gris est stockée dans une matrice bidimensionnelle de taille  $480 \times 640$ . Ces  $n$  images constituent les *images d'apprentissage*. En les vectorisant, vous pouvez donc représenter ces images par des vecteurs colonnes de  $\mathbb{R}^p$ , où  $p = 480 \times 640 = 307200$  est le nombre de pixels commun à toutes les images. Alors que dans le TP1, chaque pixel d'une image couleur constitue un point de  $\mathbb{R}^3$ , ici c'est chaque image qui constitue un point d'un espace affine  $\mathbb{R}^p$  de dimension très élevée.

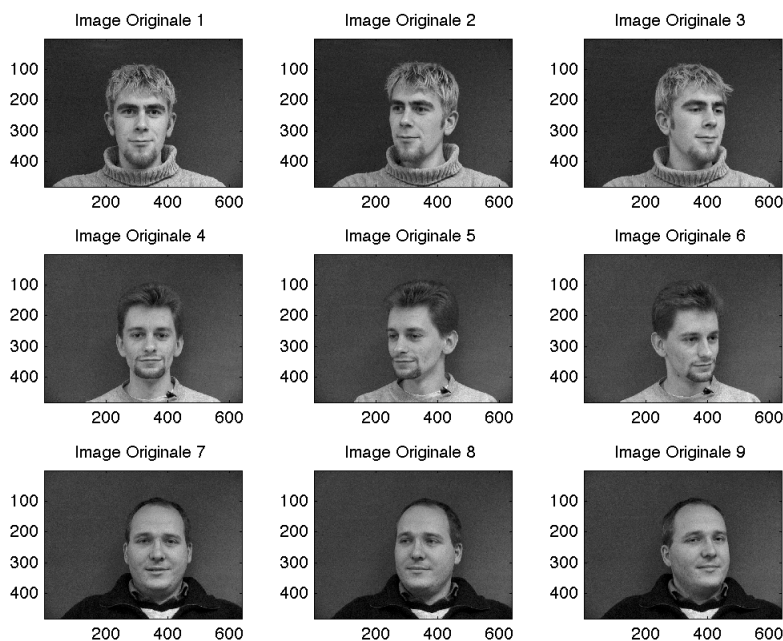


FIGURE 1 – Une base de visages

## Exercice 1 : analyse en composantes principales

La matrice des données  $X$ , de taille  $n \times p$ , contient sur chaque ligne la transposée d'une image vectorisée. Lancez le script `donnees.m` afin de créer cette matrice et de la stocker dans un fichier au format Matlab, de nom `donnees.mat`.

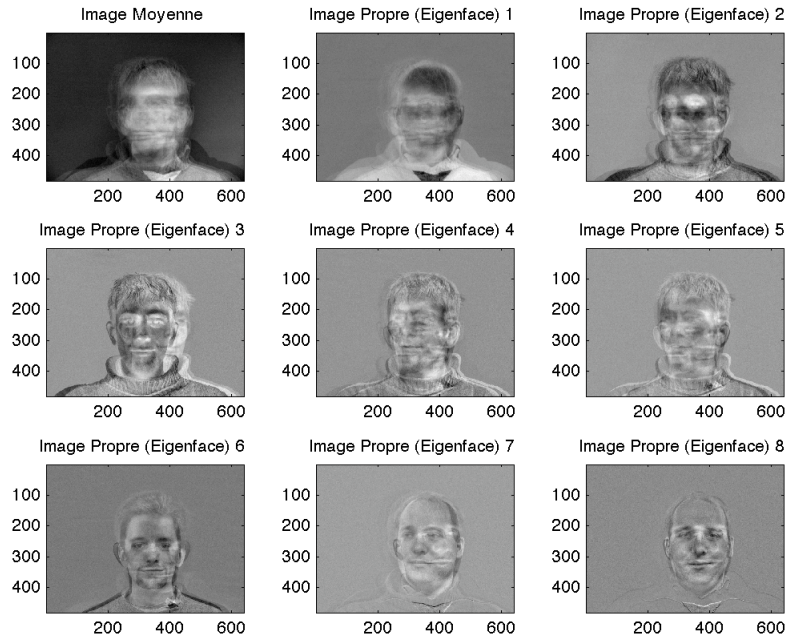


FIGURE 2 – Les «eigenfaces»

Complétez le script `exercice_1.m`, qui vise à calculer les axes principaux des images d'apprentissage à partir des vecteurs propres associés aux  $n - 1$  valeurs propres non nulles de la matrice de variance/covariance  $\Sigma$  des données. Ces axes principaux sont appelés *eigenfaces* par Turk et Pentland, par contraction des mots anglais *eigenvectors* et *faces*.

## Exercice 2 : projection des images sur les *eigenfaces*

Une fois connues les  $n - 1$  *eigenfaces*, on peut calculer les composantes principales. Complétez le script `exercice_2.m`, de manière à afficher les images d'apprentissage reconstruites à l'aide des  $q$  premières *eigenfaces* et des  $q$  premières composantes principales, pour  $q \in [0, n - 1]$ .

**Attention :** n'oubliez pas d'ajouter l'individu moyen.

Ce script doit également afficher l'évolution, en fonction de  $q$ , de la racine carrée de l'erreur quadratique moyenne (*Root Mean Square Error*, ou RMSE) entre les images originales et les images ainsi reconstruites.

## Exercice 3 : application à la reconnaissance de visages

Le script `clusters.m` calcule les composantes principales des  $n$  images d'apprentissage, puis affiche sous la forme d'un nuage de  $n$  points de  $\mathbb{R}^2$  leurs deux premières composantes principales. Chaque couleur correspond à un même individu de la *base d'apprentissage*. Ce nuage fait apparaître des groupes de points (ou *clusters*) de couleur uniforme, ce qui montre que chaque *cluster* correspond aux différentes postures d'un même individu. Il semble donc possible d'utiliser les *eigenfaces* pour la reconnaissance de visages (comme l'indique le titre de l'article ayant inspiré ce TP : *Eigenfaces for recognition*), en calculant les deux premières composantes principales d'une image, dite *image de test*, n'appartenant pas forcément à la base d'apprentissage, et en cherchant de quelle image d'apprentissage cette image est la plus proche, donc à quel individu elle correspond.

$$\mathbf{x} \in \mathbb{R}^{307200} \rightarrow \boxed{\text{Projection Base ACP}} \rightarrow \phi(\mathbf{x}) \in \mathbb{R}^{q \ll 307200} \rightarrow \boxed{\text{Prédicteur } h} \rightarrow h(\mathbf{x})$$

Le script `exercice_3.m` tire aléatoirement (à l'aide de la fonction `randi` de Matlab) une image de test, parmi les 37 personnes et les six postures faciales disponibles dans la base de données.

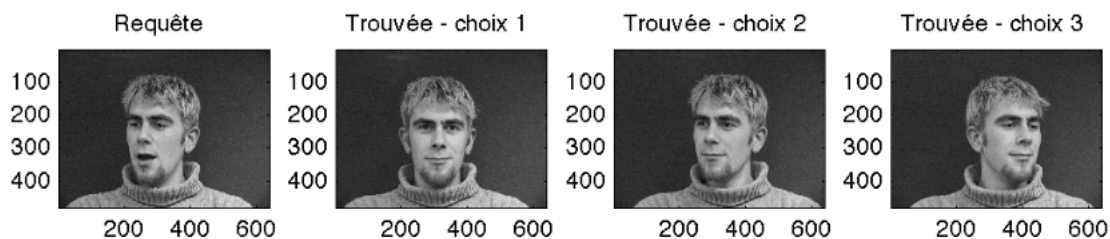


FIGURE 3 – Résultat d'une requête sur une base de visages.

## Questions sur la reconnaissance de visages

### 1. Evaluation de la reconnaissance :

- En vous inspirant du TP4 d'Analyse de Données, configurez un classifieur (type de classifieur 1ppv, 3ppv ou autre..) en complétant le script `exercice_3.m`.
- A partir des résultats obtenus de votre classifieur et des labels des images tests, construisez la matrice de confusion afin d'évaluer la qualité de votre classifieur. Comment optimiser votre classifieur ?  
(on pourra notamment regarder le choix du nombre de composantes principales)

### 2. Discussion :

- Compte tenu de la quantité de données fournie (nombre et taille des images), quelle est votre préconisation (algorithmique et informatique) pour le calcul des couples propres utiles à cette application ?
- Faut-il utiliser l'implantation cholesky (via `eig`) ou les algorithmes "subspace iteration" codés en FORTRAN ?

3. **Question supplémentaire :** En travaillant sur tout ou partie de la Base d'Apprentissage, discutez de la pertinence (spectrale) de l'introduction de la couleur dans la reconnaissance des visages.

Lancez le script `donneesCouleur.m` afin de créer cette matrice et de la stocker dans un fichier au format Matlab, de nom `donneesCouleur.mat`.

## Annexe : Différences importantes avec le TP1 d'analyse de données

Dans le TP1, la matrice des données centrées  $X_c$  est de taille  $p \times 3$ , où  $p$  désigne le nombre de pixels de l'image RVB. Or, le rang d'une matrice est inférieur à sa plus petite dimension. Comme  $p \gg 3$ , cela signifie que  $\text{rg}(X_c) \leq 3$ . Pour une image naturelle, le coefficient de corrélation linéaire entre les 3 canaux RVB n'est jamais exactement égal à  $\pm 1$ . Cela signifie que les 3 colonnes de  $X_c$  sont linéairement indépendantes, donc que  $\text{rg}(X_c) = 3$  (on dit que  $X_c$  est *de rang maximal*). Une conséquence de ce résultat est que la matrice de variance/covariance  $\Sigma = X_c^\top X_c / p$  est également de rang 3. Comme elle est de taille  $3 \times 3$ , cette matrice est inversible.

Dans ce TP, la matrice  $X_c$  des données centrées, obtenue en retranchant à chaque ligne de  $X$  l'*individu moyen*  $\bar{X}$  (égal à la moyenne des lignes de  $X$ ), est de taille  $n \times p$ , où  $n$  désigne le nombre d'images et  $p$  le nombre de pixels commun à toutes ces images. Comme  $p \gg n$ , on en déduit que  $\text{rg}(X_c) \leq n$ . Pour que cette matrice soit de rang maximal, il faudrait que ses  $n$  lignes soient linéairement indépendantes. Or, leur somme est égale au vecteur nul, puisque  $\bar{X}$  est égal à la moyenne des  $n$  lignes de  $X$ . Pour des images naturelles, on en déduit que  $\text{rg}(X_c) = n - 1$ . La matrice de variance/covariance  $\Sigma = X_c^\top X_c / n$  est donc elle aussi de rang  $n - 1$ . Comme elle est de taille  $p \times p$ , et que  $p \gg n$ , cette matrice n'est pas inversible. En l'occurrence, le noyau de  $\Sigma$  est de dimension  $p - n + 1$ .

Une autre différence avec le TP1 vient de ce que, dans ce TP, la fonction `eig` ne peut pas être directement appliquée à  $\Sigma$ . En effet, sa taille  $p \times p$  est gigantesque ( $p = 307200$ ). Or, pour une matrice  $M$  quelconque,  $M^\top M$  et  $M M^\top$  ont les mêmes valeurs propres *non nulles*. On peut donc appliquer la fonction `eig` à  $\Sigma_2 = X_c X_c^\top / n$ , de taille  $n \times n$  beaucoup plus petite, pour calculer les valeurs propres non nulles de  $\Sigma$ .

Si  $Y$  est un vecteur propre de  $\Sigma_2$  associé à une des  $n - 1$  valeurs propres  $\lambda$  non nulles, alors par définition  $(X_c X_c^\top / n) Y = \lambda Y$ , d'où  $(X_c^\top X_c / n) X_c^\top Y = \lambda X_c^\top Y$ . D'autre part,  $X_c^\top Y$  est un vecteur non nul : sinon, cela impliquerait que le vecteur  $\lambda Y$  est nul, ce qui est impossible puisque  $\lambda \neq 0$  par hypothèse et que, étant un vecteur propre,  $Y$  ne peut pas être un vecteur nul. Par conséquent,  $X_c^\top Y$  est un vecteur propre de  $\Sigma = X_c^\top X_c / n$  associé à la valeur propre  $\lambda$ . Il est facile de montrer que les  $n - 1$  vecteurs  $X_c^\top Y$  sont orthogonaux deux à deux. En les normalisant, on obtient donc une base orthonormée  $\mathcal{B}$  de  $\text{Im}(\Sigma)$ .