# Projet Calcul Scientifique/Analyse de données

# - Rapport 1er rendu -

Travail réalisé par Foucher Nathan et Guidez Martin 1-SN-C

Département Sciences du Numérique (SN)

Mars 2022

# Table des matières

# Première partie
# Introduction

L'objectif du projet est de réussir à reformer un visage masqué en utilisant une base de donnée préalable de visages entier. Pour la 1er partie du projet nous allons tout d'abords utiliser l'ACP (d'après le travail de M. Turk et A. Pentland intitulé «Eigenfaces for Recognition»). Les données étants de très grandes tailles (16 images en niveaux de gris), l'ACP va permettre de réduire la dimension des images et de créer un système de reconnaissance grâce aux composantes principales pour retrouver le visage entier.

Nous devrions avoir 16 images de femmes et hommes non masqués et une autre base de données similaire avec des visage masqués. Pour le TP1 nous partirons de seulement 4 individus (2 femmes et 2 hommes) pour générer la base de données.

## Deuxième partie

# Les «Eigenfaces» non masqués

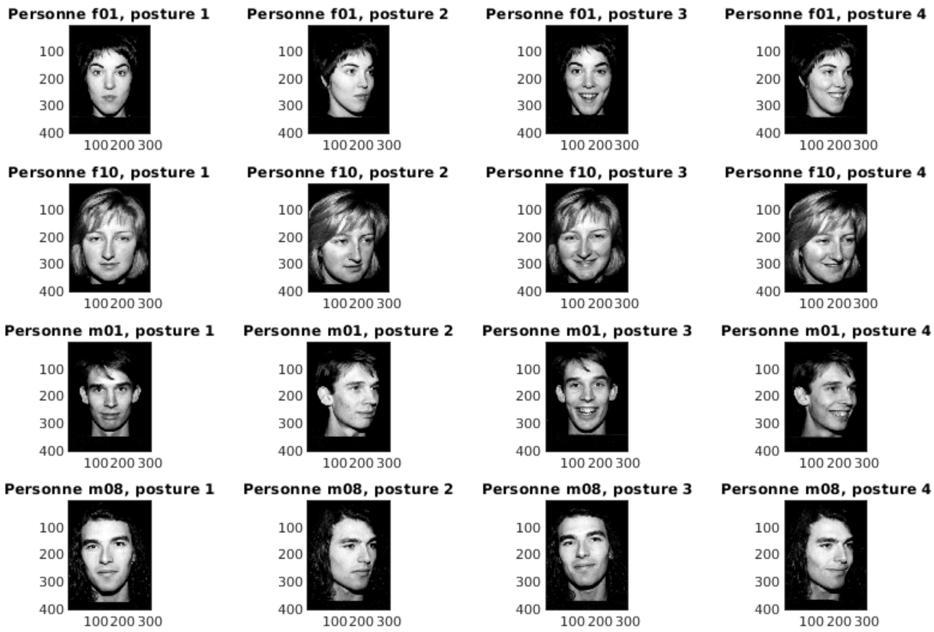


FIGURE 1 – images en entrée

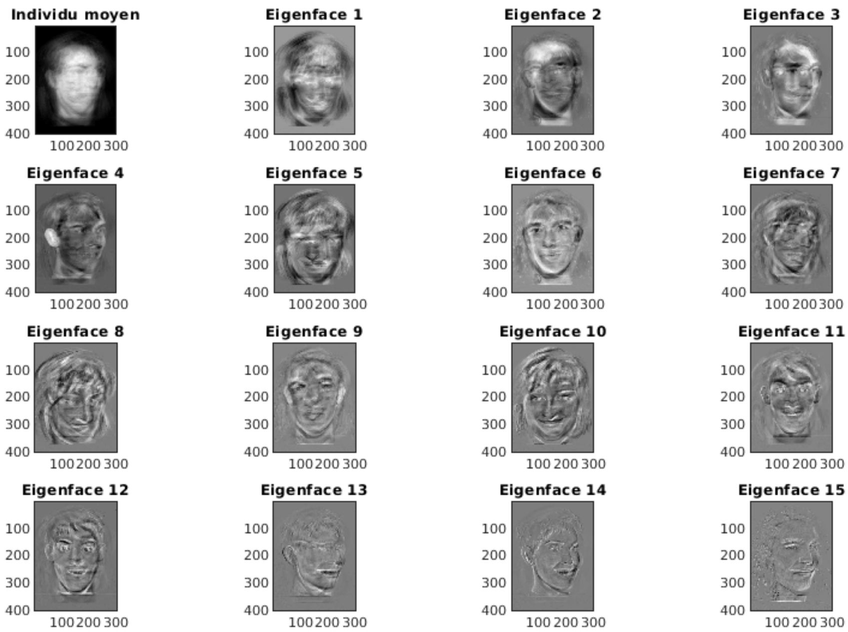


FIGURE 2 – images en sortie après codage

On remarque bien après exécution que l'ACP ne garde que les axes principaux des images.

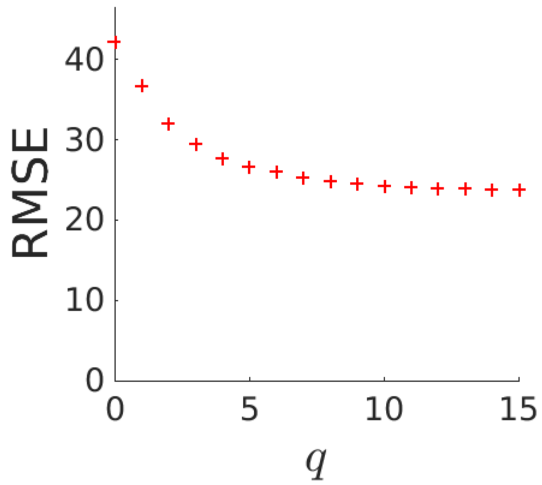


Figure 3 – RMSE en fonction du nombre de composantes principales

D'après le graphique précèdent on remarque bien que plus on utilise les composantes principales calculées précèdement plus l'image reconstruite est fidèle à l'originale. On obtient au final un RMSE inférieur à 30.

# Troisième partie

# Les «Eigenfaces» masqués

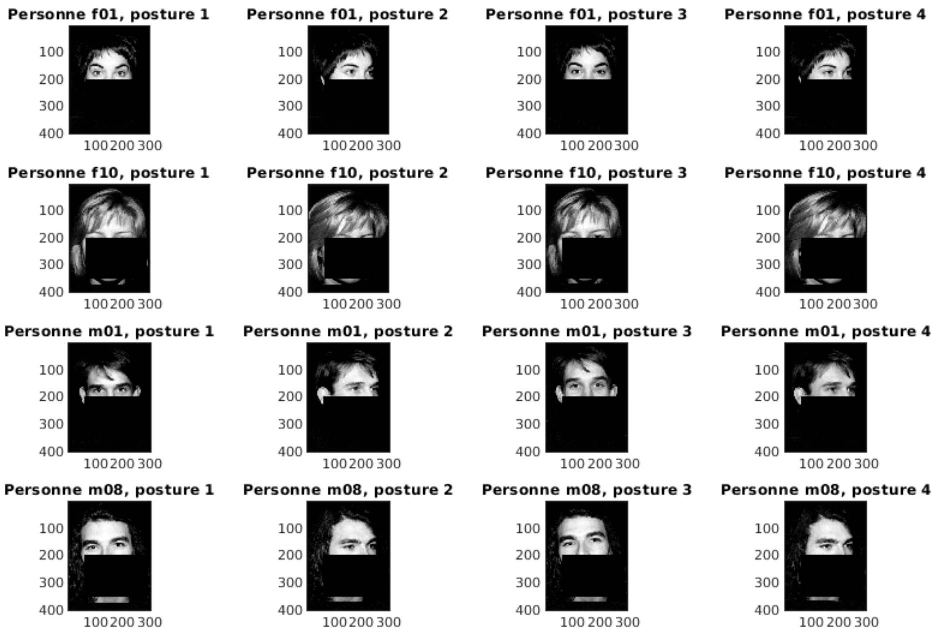


FIGURE 4 – images masquées en entrée
Ici on supprime volontairement des pixels au centre des images pour simuler le port d'un masque.

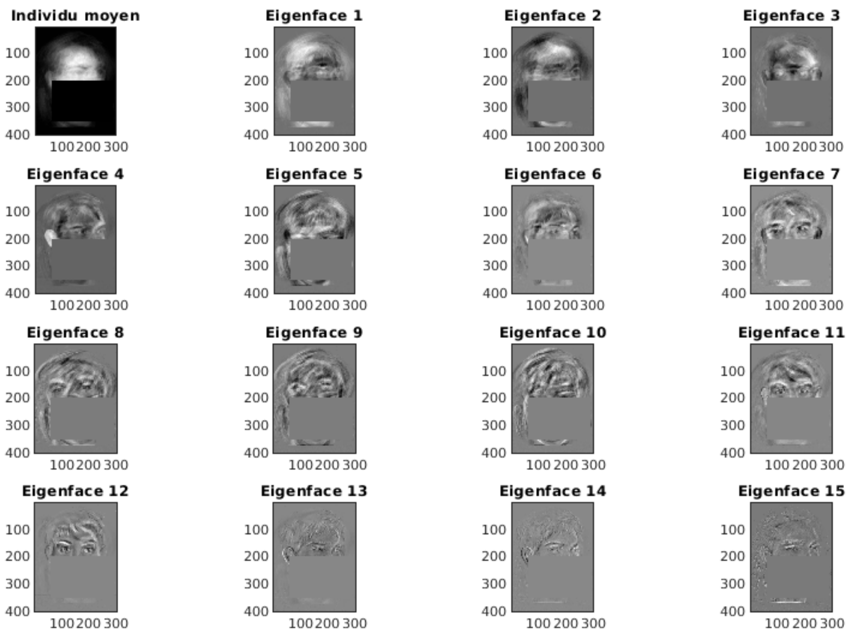


FIGURE 5 – images masquées en sortie après codage

On retrouve un résultat similaire au précèdent mais ici avec evidement des informations manquantes sur les axes principaux.

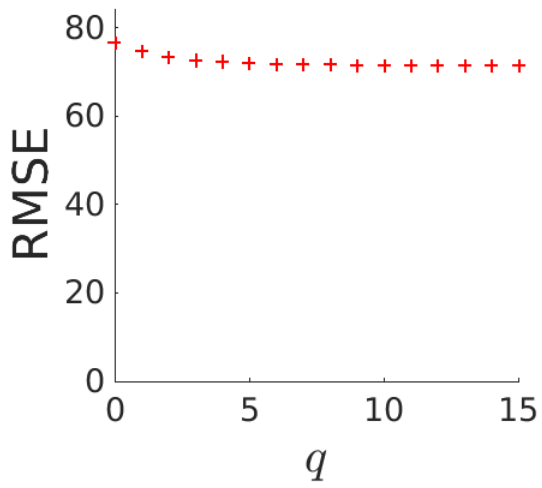


FIGURE 6 – RMSE en fonction du nombre de composantes principales pour images masquées

Ici on voit que le RMSE ne diminue que très peu au fur et à mesure. En effet il manque beaucoup d'informations sur les images d'entrées. Une image reconstruite fidèle à l'originale est plus difficile à obtenir comme on s'y attendait.

## Quatrième partie

# L'ACP et la méthode de la puissance itérée

## 1 Réponses aux questions

### 1.1 Question 4

Soit une matrice rectangulaire $H \in R_pn$. Expliquer pourquoi connaître les éléments propres, i.e les valeurs propres et les vecteurs propres, de $H^tH$ permet de connaître les éléments propres de $HH^t$.

Soit $v$ une valeur propre de $H^tH$ dont un vecteur propre associé est X.

$$H^tHX = vX \iff HH^tHX = HvX \iff HH^tHX = vHX \iff HH^tY = vY \qquad (1)$$

avec $Y = HX \neq 0$

Ainsi v est aussi une valeur propre de $HH^t$ avec pour vecteur propre associé $Y = HX$. Donc on peut connaitre les élèments propre de $HH^t$ en connaissant ceux de $H^tH$

## 1.2 Question 6

Lien avec l'ACP : est-il plus utile en théorie d'utiliser une fonction telle que eig ou la méthode de la puissance itérée pour calculer les éléments propres de $\sum$ si le but est d'effectuer une ACP pour réduire les dimensions d'un espace ? Justifier.
En théorie la fonction eig devrait être plus rapide, vérifions ce résultat.
Après codage des deux autres méthodes utilisation de la fonction cputime de matlab pour comparer les temps d'execution on obtient le résulat suivant :

```
Valeur propre dominante (methode avec la grande matrice) = 9.183e+04
Valeur propre dominante (methode avec la petite matrice) = 9.183e+04
Valeur propre dominante (fonction eig) = 9.183e+04
Temps avec la grande matrice en secondes = 7.100e-01
Temps avec la petite matrice en secondes = 4.000e-02
Temps avec eig en secondes = 5.100e-01
```

FIGURE 7 – Temps d'éxecution selon la méthode

On remarque que les valeurs propres trouvées sont exactement les mêmes, en revanche la fonction eig de matlab est plus rapide.

## 1.3 Question 7

Si l'on choisit d'utiliser la méthode de la puissance itérée avec déflation pour calculer les éléments propres de $\sum$, sur quelle matrice doit-on appliquer la méthode pour minimiser le temps de calcul et la mémoire utilisée ?
On voit que le temps d'execution est presque le même entre les deux méthodes. En revanche la matrice $H^tHX$ étant plus petite, il est plus judicieux d'utiliser la méthode utilisant cette dernière dans un soucis d'économiser la mémoire.