Les eigenfaces dans la reconnaissance faciale

Aloïs SIMON

June 12, 2018

Sommaire

Principe

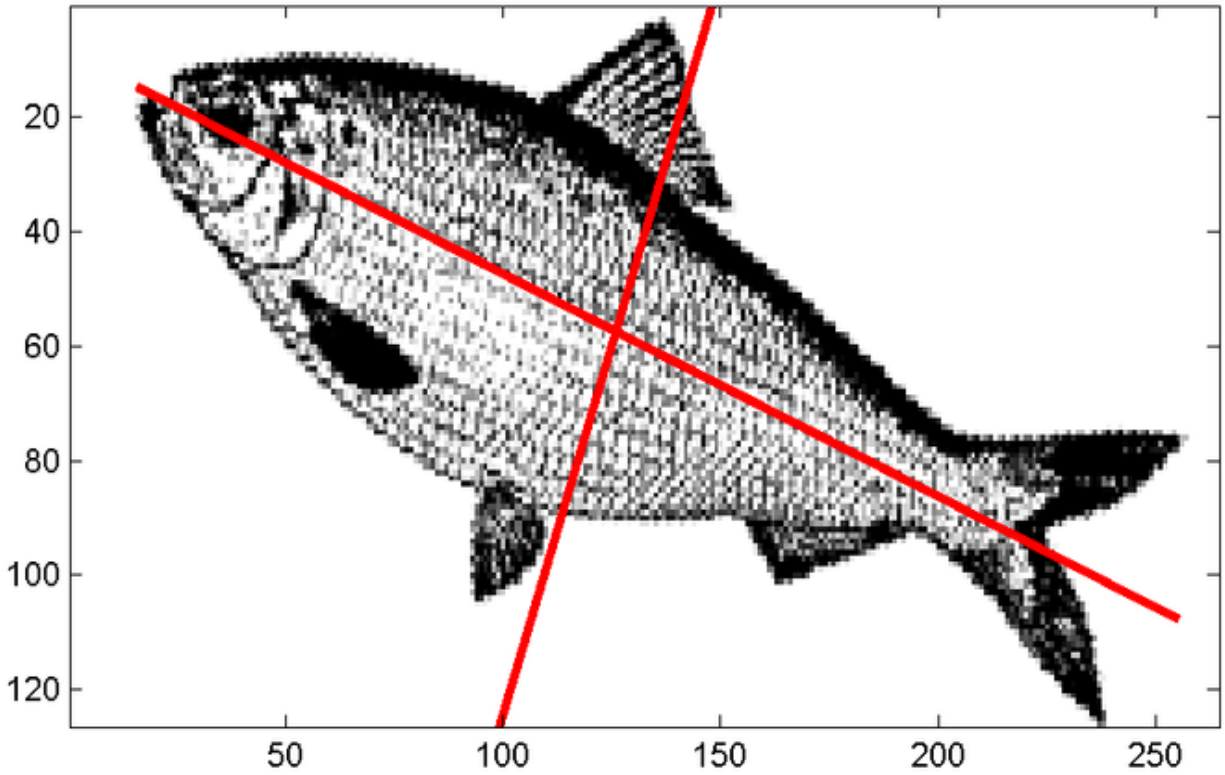
Créer les eigenfaces

1. Créer une collection de M images
2. Calculer les M eigenfaces de la collection d'images et en sélectionner M'
3. Calculer les poids des eigenfaces pour chaque visages

Reconnaître une image

1. Calculer les poids des eigenfaces pour l'image
2. Vérifier que l'image est un visage
3. Vérifier si il est connu ou non
4. (*Optionnel*) Mettre à jour la collection d'images
5. (*Optionnel*) Mettre à jour les eigenfaces

Analyse en composantes principales



Source : https://www.wikipedia.org/

Soit N variables aléatoires $X_1, ..., X_N$ connues à partir d'un échantillon de K réalisations conjointes de ces variables.
On note $\overline{X_1}, ..., \overline{X_N}$ leur moyenne.

$$M = \begin{pmatrix} X_{1,1} & \cdots & X_{1,N} \\ \vdots & \ddots & \vdots \\ X_{K,1} & \cdots & X_{K,N} \end{pmatrix}$$

Soit $g = (\overline{X_1}, ..., \overline{X_N})$

$$\overline{M} = M - \widetilde{1}g^T = \begin{pmatrix} X_{1,1} - \overline{X_1} & \cdots & X_{1,N} - \overline{X_N} \\ \vdots & \ddots & \vdots \\ X_{K,1} - \overline{X_1} & \cdots & X_{K,N} - \overline{X_N} \end{pmatrix}$$

$$C = 1/K \cdot \overline{M}^T \cdot \overline{M}.$$

La projection de l'échantillon des X sur u s'écrit :

$$\pi_u(M) = M \cdot u$$

$$\pi_u(M)^T \cdot 1/K \cdot \pi_u(M) = u^T \cdot M^T \cdot 1/K \cdot M \cdot u = u^T \cdot C \cdot u$$

$$\pi_u(M)^T \cdot 1/K \cdot \pi_u(M) = u^T P^T \Delta P u = (Pu)^T \Delta P u$$

$$v^T \cdot \Delta \cdot v = \lambda_1$$

Calcul des eigenfaces

Visages

Calcul

Images : $\Gamma_1, ..., \Gamma_M$

Moyenne : $\Psi = \frac{1}{M} \sum_{i=0}^{M} \Gamma_i$

Images centrée : $\Phi_i = \Gamma_i - \Psi$

Soit $A = [\Phi_1, ..., \Phi_M]$, on pose $C = AA^T$.

Moyenne

Optimisation

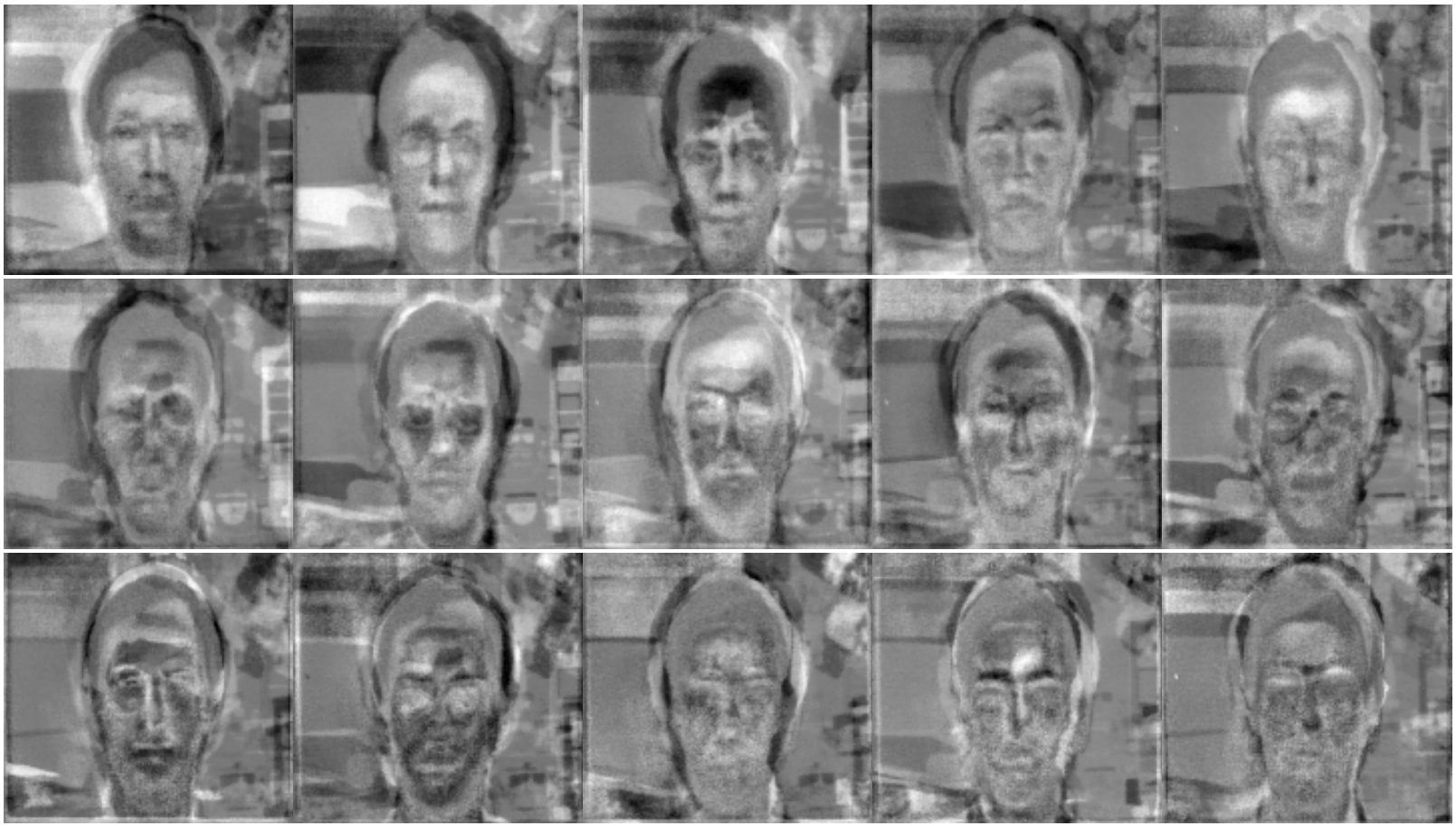
$$A^T A v_k = \mu_k v_k$$

$$A A^T A v_k = \mu_k A v_k$$

$$u_k = \sum_{i=0}^{M} v_{ki} \Phi_i$$

```
def pca(X):
    moyenne = np.mean(X, 0)

    X_centree = X - moyenne

    e_faces, _, _ = np.linalg.svd(X_centree.transpose())

    poids = np.dot(X_centree, e_faces)

    return e_faces.transpose(), poids, moyenne
```

Exemple



Calculer les poids des visages de la base d'images

Poids des eigenfaces pour le visage k : $\omega_{i,k} = u_k^T \Phi_i$ pour $i \in [\![1, M']\!]$.
On note $\Omega_k = [\omega_{1,k}, ..., \omega_{M',k}]$

Reconstruction

Identifier un visage

Calculer les poids du visage

Soit Γ un nouveau visage
Calcul des poids du nouveau visage : $\omega_k = u_k^T(\Gamma - \Psi)$ pour $k \in [\![1, M']\!]$
$\Omega = [\omega_1, ..., \omega_{M'}]$

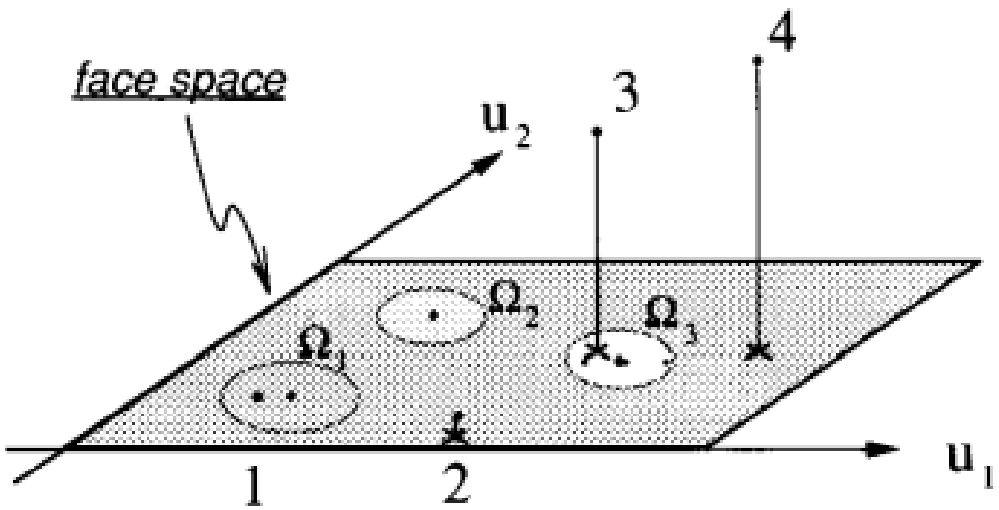
Reconnaître le visage

Soit $\phi = \Gamma - \Psi$ et $\phi_f = \sum_{i=1}^{M'} \omega_i u_i$

$\epsilon = ||\phi - \phi_f||$

$d_k = ||\Gamma - \Gamma_k||$

Reconnaître le visage



Reconnaître le visage

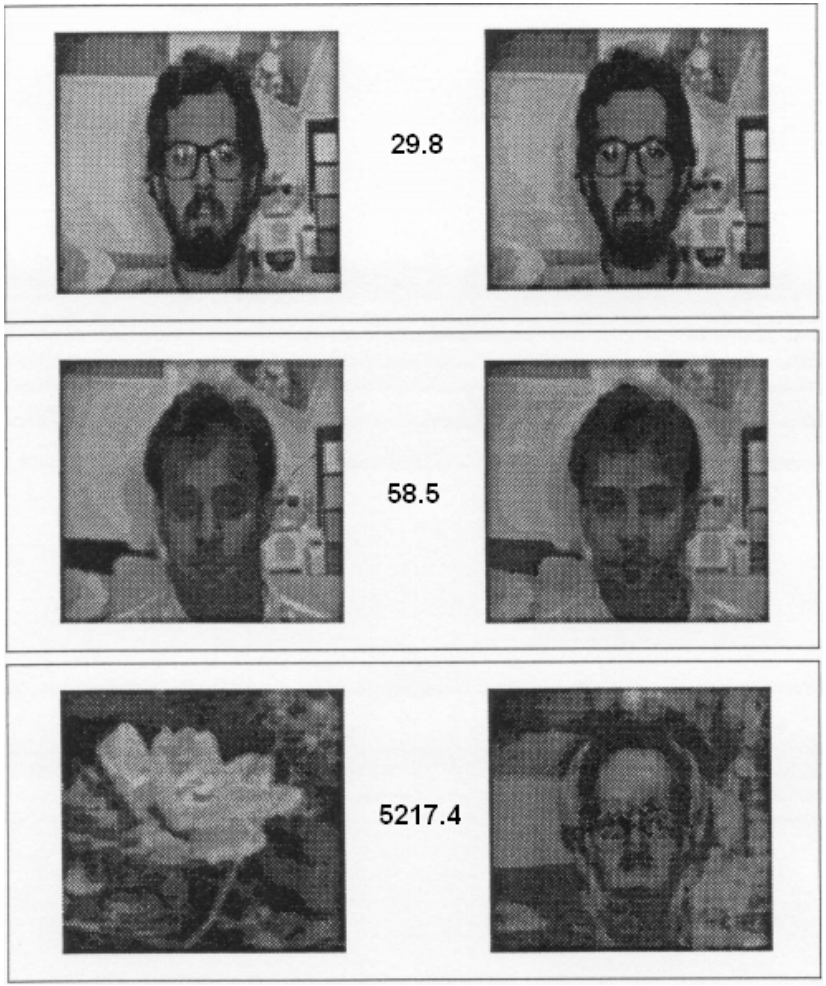


Figure: Source :

Bibliographie / Sitographie et annexes

Bibliographie

1. WIKIPEDIA, Eigenface, https://en.wikipedia.org/w/index.php?title=Eigenface&oldid=819247723
2. M. Turk; A. Pentland, *Face recognition using eigenfaces*, http://www.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf
3. Sirovich; Kirby *Low-dimensional procedure for the characterization of human face*, http://engr.case.edu/merat_francis/EECS%20490%20F04/References/Face%20Recognition/LD%2
4. Delac, K., Grgic, M., Liatsis, P , http://www.vcl.fer.hr/papers_pdf/Appearance-based%20Statistical%20Methods%20for%20Face%20Recognition.pdf
5. M. Turk; A. Pentland, *"Eigenfaces for recognition*, doi:10.1162/jocn.1991.3.1.71

```
from PIL import Image
import numpy as np
import pylab
import os

dossier = os.curdir

lien = 'C:/Users/alois/Google Drive/Documents/Prépa/spé/TIPE/progamme/'
os.chdir(lien)
ext = 'png'

def loadImg(repertoire):
    img_liste = []
    images = os.listdir(repertoire + 'images/')
    nb_img = len(images)

    for i in range(len(images)):
        img_liste.append(np.array(Image.open(repertoire + 'images/' + images[i]).convert("L")))

    return img_liste


def pca(X):
    """ Principal Component Analysis
        :param X : Matrice avec les images d'entrainement applaties

        :return e_faces : eigenfaces
        :return poids : poids associés aux images d'entraînement pour leur reconstruction par les eigenface
        :return moyenne : Moyenne des images
    """

    moyenne = np.mean(X, 0)
```

```
    # Centre les données
    X_centree = X - moyenne

    # Effectue la décomposition en valeurs propres
    e_faces, _, _ = np.linalg.svd(X_centree.transpose())

    # Calcul le poids de chaque eigenface
    poids = np.dot(X_centree, e_faces)

    return e_faces.transpose(), poids, moyenne


def run(eigen_affiche = False, recons_affiche = False):
    img_liste = loadImg(lien)
    m, n = img_liste[0].shape # Récupère la taille des images

    # Crée une matrice contenant les images sous formes de vecteurs
    img_matrice = np.array([img_liste[i].flatten() for i in range(len(img_liste))],'f')

    # Applique le PCA
    V, poids, moyenne_temp = pca(img_matrice)

    # "Reforme" l'image moyenne pour l'afficher
    moyenne = moyenne_temp.reshape(m,n)

    eigenfaces = [V[k].reshape(m,n) for k in range(len(V))]

    if recons_affiche:
        decomposition(img_matrice, V, poids, moyenne_temp, m, n)

    # Affiche les images
```

```
    pylab.figure()
    pylab.gray()
    pylab.imshow(moyenne)

    if eigen_affiche:
        for k in range(len(eigenfaces)):
            pylab.figure()
            pylab.gray()
            pylab.imshow(eigenfaces[k])

    pylab.show()

def decomposition(img_liste, eigenfaces, poids, moyenne, m, n):
    M = len(eigenfaces)
    for j in range(len(img_liste)):
        img_test = img_liste[j]
        poids = []
        img_tmp = img_test - moyenne

        for i in range(M):
            poids.append(np.dot(eigenfaces[i].T, img_tmp))

        recons = np.dot(np.array(poids).reshape(1, 16), eigenfaces)

        r = moyenne.reshape(m, n) + recons.reshape(m, n)

        pylab.figure()
        pylab.gray()
        pylab.imshow(r)
```