MASTER ISI :

FACIALE RECOGNITION

TP N°5

Réalisé par :

ESSADEQ Ayoub

RAHHAOUI ABDESSAMAD

2023-2024

Tache 1: Lecture de la base Olivetti

Importation de la base de données Olivetti Faces.

Obtention des données d'images des visages et des informations associées.

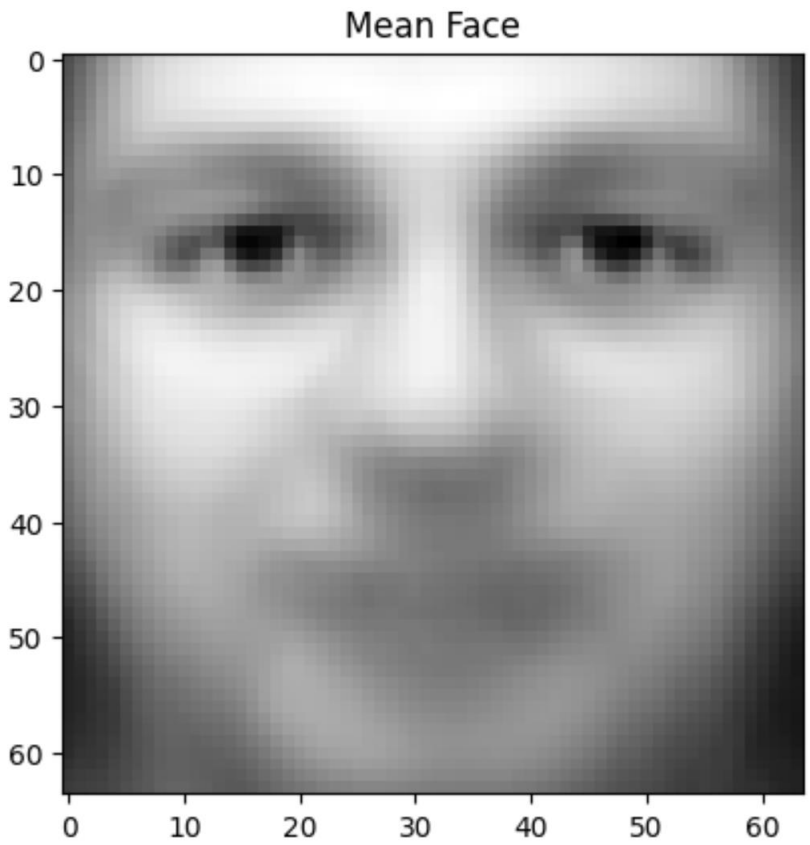
```
from sklearn.datasets import fetch_olivetti_faces
import matplotlib.pyplot as plt

# Lecture de la base Olivetti
olivetti_faces = fetch_olivetti_faces(shuffle=True, random_state=42)
faces_data = olivetti_faces.data
faces_images = olivetti_faces.images

# Affichage de quelques images de la base
fig, axes = plt.subplots(1, 5, figsize=(12, 3),
                         subplot_kw={'xticks':[], 'yticks':[]})
for i, ax in enumerate(axes.flat):
    ax.imshow(faces_images[i], cmap='gray')
    ax.set_title(f"Person {i // 5 + 1}")
plt.show()
```

Tache 2: Calcul et affichage du visage Moyen

Calcul du visage moyen en prenant la moyenne pixel par pixel de toutes les images.

Affichage du visage moyen pour visualiser la moyenne des traits faciaux.

```
import numpy as np

# Calcul et affichage du visage Moyen
mean_face = np.mean(faces_data, axis=0)
plt.imshow(mean_face.reshape((64, 64)), cmap='gray')
plt.title('Mean Face')
plt.show()
```



Tache 3: Retrancher le visage Moyen de la matrice des faces

Normalisation des images en retranchant le visage moyen, ce qui élimine les variations dues à l'éclairage et d'autres facteurs.

```
# Retrancher le visage Moyen de la matrice des faces
centered_faces = faces_data - mean_face
```

Tache 4: Appliquer la méthode PCA

Utilisation de l'analyse en composantes principales (PCA) pour extraire les composantes principales des visages normalisés.

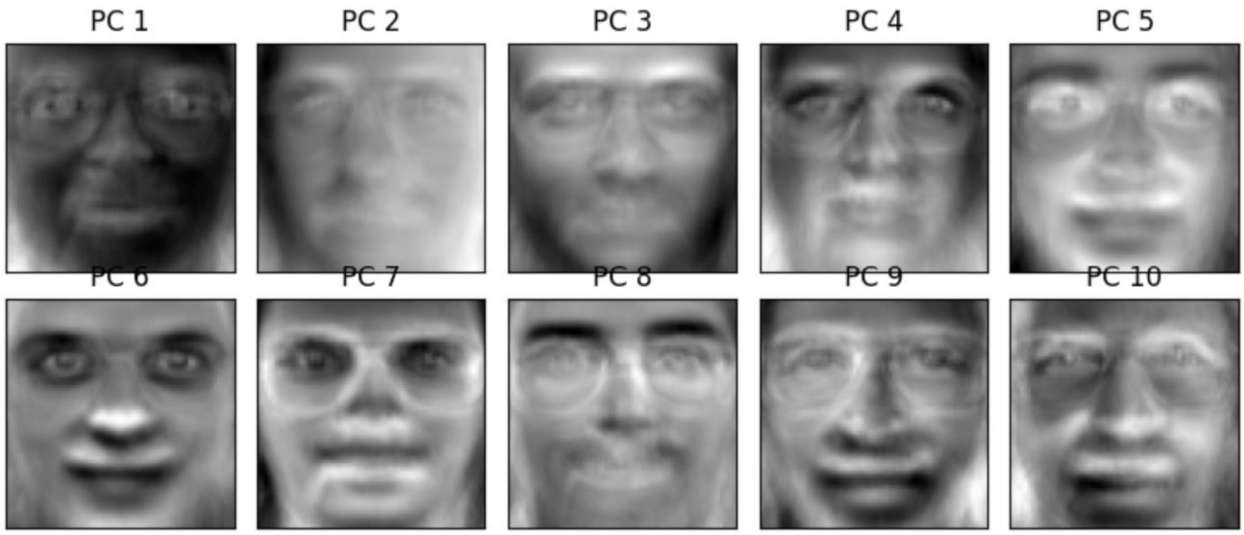
```
from sklearn.decomposition import PCA

# Appliquer la méthode PCA
n_components = 100
pca = PCA(n_components=n_components)
pca.fit(centered_faces)
```

```
PCA
PCA(n_components=100)
```

Tache 5: Afficher les EigenFaces

Affichage des "Eigenfaces", qui sont les vecteurs propres résultant de l'application de la PCA. Ces vecteurs représentent les directions principales de la variation dans les données.

```
# Afficher les EigenFaces
eigenfaces = pca.components_
fig, axes = plt.subplots(2, 5, figsize=(10, 4),
                         subplot_kw={'xticks':[], 'yticks':[]},
                         gridspec_kw=dict(hspace=0.1, wspace=0.1))
for i, ax in enumerate(axes.flat):
    ax.imshow(eigenfaces[i].reshape((64, 64)), cmap='gray')
    ax.set_title(f"PC {i+1}")
plt.show()
```

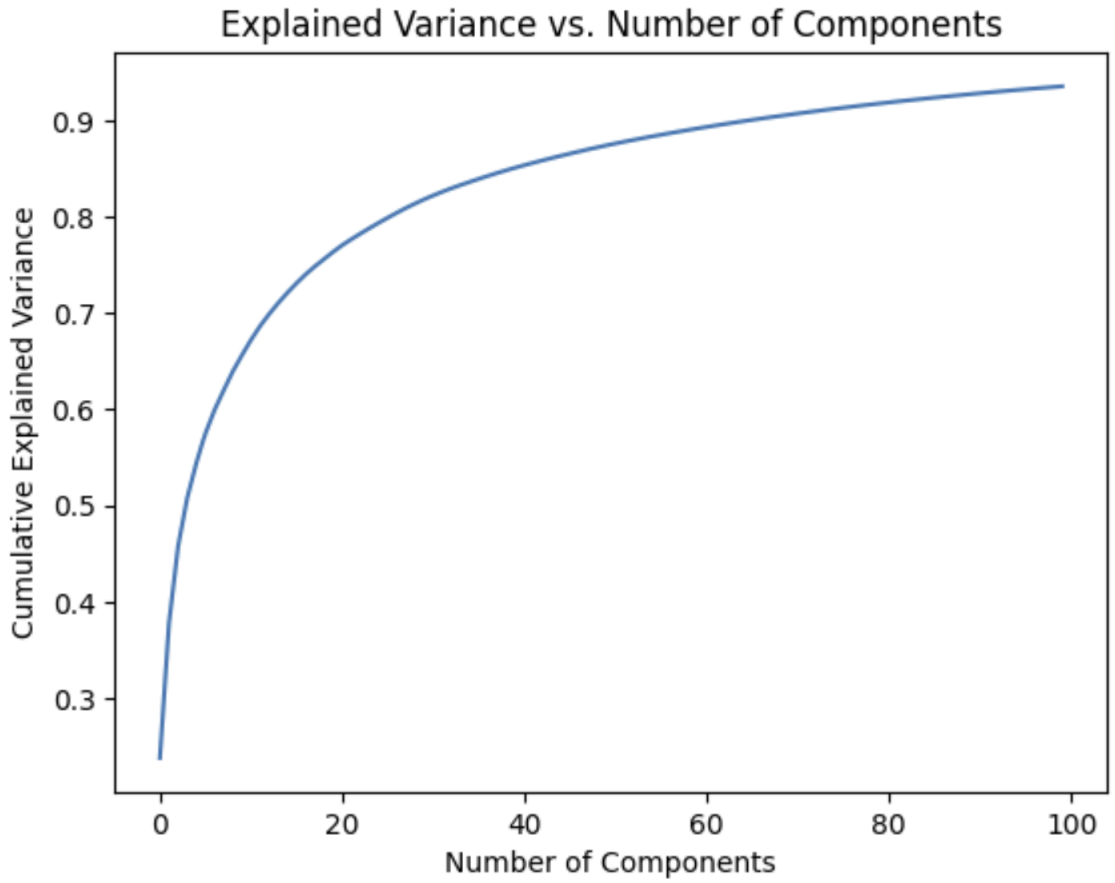


Tache 6: Choix du nombre de composantes à retenir

Exploration de la variance cumulative expliquée pour aider à choisir le nombre optimal de composantes principales à retenir.

```
# Tache 6: Choix du nombre de composantes à retenir
# Ceci peut être basé sur l'exploration de la variance cumulative
expliquée
cumulative_explained_variance = np.cumsum(pca.explained_variance_ratio_)
plt.plot(cumulative_explained_variance)
```

```
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance vs. Number of Components')
plt.show()
```



Tache 7: Calcul des projections de chaque image de la base sur les composantes retenues

Calcul des projections de chaque image sur les composantes principales retenues après PCA.

```
# Tache 7: Calcul des projections de chaque image de la base sur les
composantes retenues
projections = pca.transform(centered_faces)
```

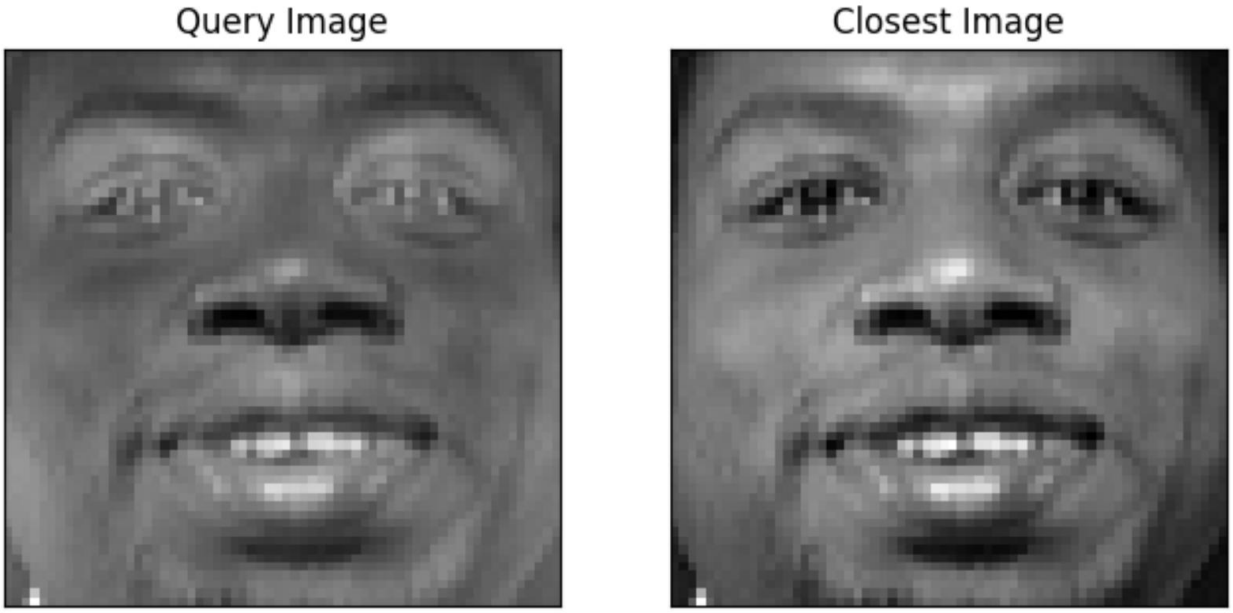
Tache 8: Faire la reconnaissance

Choix aléatoire d'une image pour la reconnaissance.

Calcul des projections de l'image requête et mesure de la distance entre cette projection et celles des images de la base.

Identification de l'image la plus proche en termes de distance pour la reconnaissance.

```
# Tache 8: Faire la reconnaissance
# Choisir une image pour la reconnaissance
random_image_index = np.random.randint(0, len(centered_faces))
query_image = centered_faces[random_image_index]
query_projection = pca.transform([query_image])

# Calculer la distance entre le vecteur de projection de l'image requête
et les vecteurs de projection des images de la base
distances = np.linalg.norm(projections - query_projection, axis=1)

# Identifier l'image la plus proche
closest_image_index = np.argmin(distances)

# Afficher l'image requête et l'image la plus proche
fig, axes = plt.subplots(1, 2, figsize=(8, 4),
                         subplot_kw={'xticks':[], 'yticks':[]})
axes[0].imshow(query_image.reshape((64, 64)), cmap='gray')
axes[0].set_title('Query Image')

axes[1].imshow(faces_data[closest_image_index].reshape((64, 64)),
cmap='gray')
axes[1].set_title('Closest Image')
plt.show()
```



Chaque tâche contribue à la mise en œuvre d'un système de reconnaissance faciale basé sur l'analyse en composantes principales (PCA), depuis la préparation des données jusqu'à la reconnaissance effective.