CS-663 Assignment 4 Q5

Soham Naha (193079003)
Akshay Bajpai (193079002)
Mohit Agarwala (19307R004)

November 6, 2020

5

Display in your report the reconstruction of any one face image from the ORL database using $k \in \{2, 10, 20, 50, 75, 100, 125, 150, 175\}$ values. Plot the 25 eigenvectors (eigenfaces) corresponding to the 25 largest eigenvalues using the subplot or subimage commands in MATLAB. [10 points]

The same approach as in Q4 of the assignment was mostly used in this question with some code modifications for reconstruction and saving eigen vectors as images.

Code for the question

```
import numpy as np
import cv2
import os
import matplotlib.pyplot as plt
import sys

def find_mean(array):
    """Find the mean of train data
    :param array : the train image matrix
    :output mean : the mean image vector
    """
    mean = np.mean(array,axis=0)
    return mean

def im2double(im):
    """Normalizes the image to be in range 0-1
    :param im : input image
    :output out : min-max normalized output
    """
    min_val = np.min(im)
    max_val = np.max(im)
    out = (im.astype('float') - min_val) / (max_val - min_val)

    return out

def read_train_images_ORL(file_path,subjects,images_per_subject):
    """Reads the ORL train image data and returns a stacked array
    :param file_path: the folder path of ORL dataset
    :param subjects: the number of subjects under training consideration
    :param images_per_subject: the number of images to consider for each
                               test subject
    :output image_array: stacked array output
    """
    image_array = []
    for i in range(1,subjects+1):
        folder_path = os.path.join(file_path,"s"+str(i))
        #print(folder_path)
        for j in range(1,images_per_subject+1):
            filepath = os.path.join(folder_path,str(j)+".pgm")
```

```
            im = cv2.imread(filepath,0)
            image_array.append(im2double(im).ravel())

    return np.array(image_array)

def subtract_mean(mean,train):
    """Subtracts the mean of the training data from both the
    train and the test stacked array
    :param mean: the mean of the training data
    :param train: the train images stack array
    :output train_mean: mean subtracted train stack
    """
    train_mean = train-mean
    return train_mean

def normalize_vecs(array):
    """Normalizes the Unitary vector matrix
    :param array: the unitary matrix of vectors from svd
    :output array: the normalized array
    """
    _,c = array.shape
    for i in range(c):
        array[:,i] = array[:,i]/(np.sqrt(np.sum(np.square(array[:,i]))))

    return array

def calculate_svd(array):
    """Calculates svd of the train stack
    :param array: the train stack
    :output U: the left unitary matrix of the svd output
    """
    U,sigma,V_T = np.linalg.svd(array.T,full_matrices=False)
    U = normalize_vecs(U)
    return U

def calculate_alpha(V,train):
    """Calculates the reconstruction matrix alpha for train and test images
    :param V: the unitary matrix from decomposition of train stack
    :param train: the train stack
    :output alpha_train: reconstruction coefficients for train data
    """
    alpha_train = np.matmul(V.T,train.T)
    return alpha_train

def reconstruct_eigen_faces(ks, alpha, V, train_mean):
    """Reconstructs the eigen faces depending on the list of Ks supplied
    :param ks: list of number of coefficients
    :param alpha: coefficient matrix
    :param V: unitary matrix of train images obtained using svd
    :param train_mean: the mean vector of the train stack
    """
    num_plots= len(ks)
    plt.figure(figsize=(50,20))
    plt.suptitle("Eigen Face Reconstruction for ORL Database based on k",fontsize=50)
    for k in range(num_plots):
        reconstructed = []
        for i in range(ks[k]):
            reconstructed.append(alpha[0,i]*V[:,i])
        reconstructed = np.array(reconstructed)
        reconstructed = np.sum(reconstructed,axis=0) + train_mean
        plt.subplot(2,5,k+1)
        plt.title("k={}".format(ks[k]),fontsize=20)
        plt.imshow(reconstructed.reshape(112,92),cmap="gray")
    plt.savefig("../images/ReconstructionPlot_k.png",bbox_inches="tight")

```

```python
def plot_top_25_eigen_vectors(V):
    """Plot the top 25 eigen vectors
    :param V: unitary vector
    """
    plt.figure(figsize=(30,30))
    plt.suptitle("Eigen Vectors as Images",fontsize=50)
    for i in range(25):
        plt.subplot(5,5,i+1)
        plt.imshow(V[:,i].reshape(112,92),cmap="gray")
        plt.title("Eigen Vector: {}".format(i+1),fontsize=20)
        plt.axis("off")
    plt.savefig("../images/top25eigenVectors.png",bbox_inches="tight")

if __name__=="__main__":
    ORL_path = "../../ORL"
    ORL_k = [2,10,20,50,75,100,125, 150,175]
    train_images = read_train_images_ORL(ORL_path,subjects=32,images_per_subject=6)
    train_mean = find_mean(train_images)
    train_mean_subtracted = subtract_mean(train_mean, train_images)
    unitary_vectors = calculate_svd(train_mean_subtracted)
    alpha_svd_train= calculate_alpha(unitary_vectors, train_mean_subtracted)
    reconstruct_eigen_faces(ORL_k,alpha_svd_train,unitary_vectors, train_mean)
    plot_top_25_eigen_vectors(unitary_vectors)
```

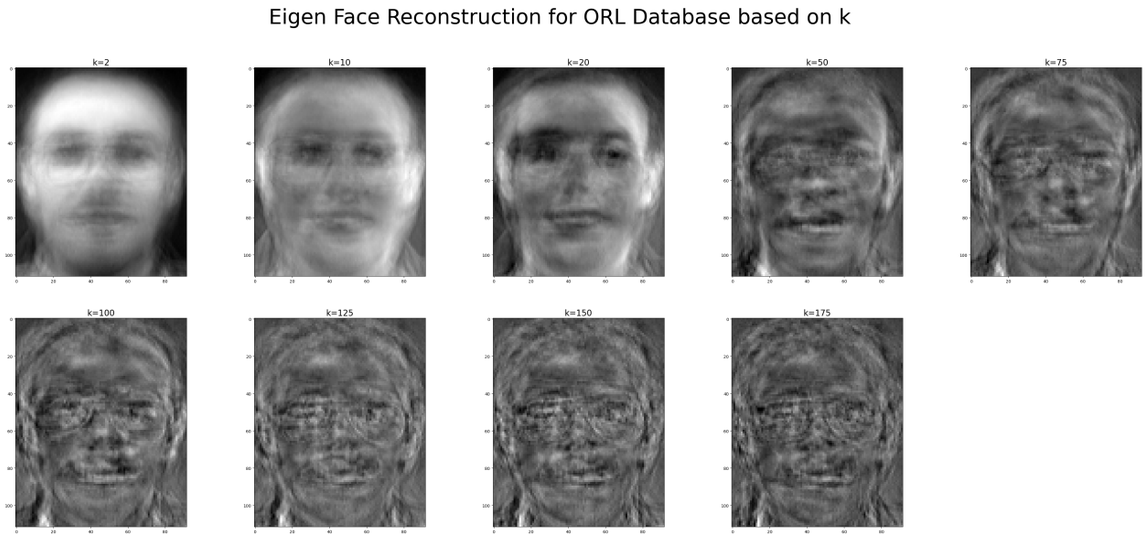

Figure 1: Reconstruction of Eigen Faces vs k

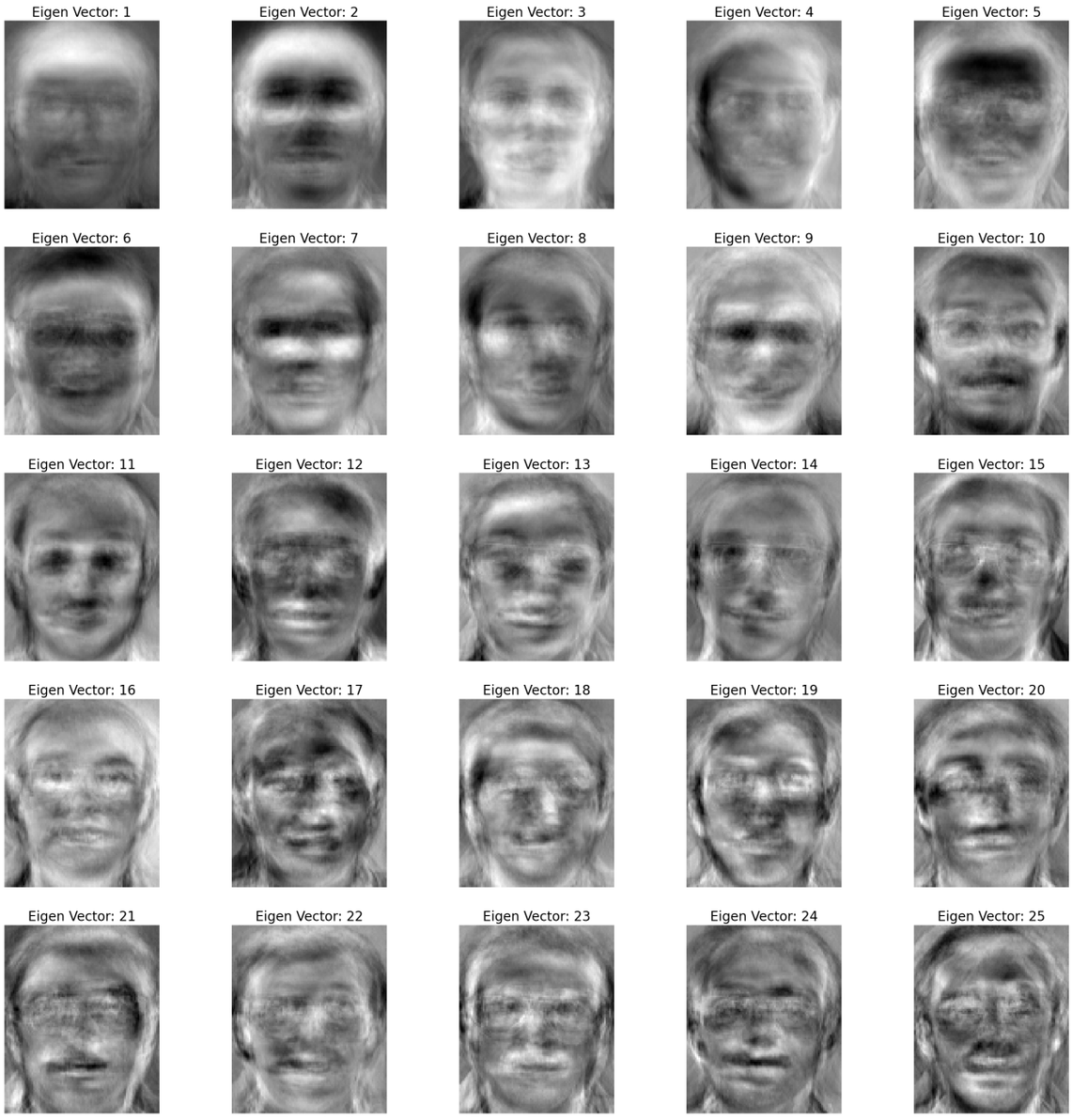

Figure 2: Top 25 Eigen Vectors as Images