Haonan Zhu

hz1396

N19549395

**Computer Vision Project 2: Face Recognition**

PT576523

How to compile and run the program:

Run Facerecognition.py. Make sure all the training and test images are on the same directory of Facerecognition.py.

(1)

T0: 7 x 1012

T1: 1 x 108

Mean face m:



M eigenfaces:



(2)

**The PCA coefficients :**

subject01.normal.jpg:[-10690469.06587205 44845387.57638889 43194205.74884757

14222155.59591849 -44508657.1889574 -74331087.54911807

-66582627.21369252 76262427.33480631]

subject02.normal.jpg:[ 25753823.85487584 -68513612.4564954 14602302.60837113

-20330165.06339063 1417866.42480582 -32833118.07314866

82351522.69026098 -54146390.88440429]

subject03.normal.jpg:[ -1412746.7156782 -3700741.26924959 52282147.30575235

52284398.57794323 8770899.89913118 50460079.1906319 51601112.39541467

29226393.11718583]

subject07.normal.jpg:[ -6.07226692e+07 -8.15286439e+07 4.78978099e+06 3.57442692e+07

4.50446659e+07 4.69145663e+07 1.13335503e+08 -6.52430239e+07]

subject10.normal.jpg:[ 36281898.80541495 -26409452.49338488 4346080.96456587

29301130.75790551 11700104.64209832 16040013.33178689

-91762000.07091594 -24815408.26342363]

subject11.normal.jpg:[ 4.85791901e+07 2.32378808e+08 -1.20766065e+08 -1.36849597e+08

-1.89070720e+07 -3.45578820e+07 -1.23871293e+08 -9.32339275e+06]

subject14.normal.jpg:[-20552314.41867543 -562539.41618161 30379948.64414814

-26989939.30554907 30286740.85452466 4497422.96389425

-90676891.06181912 38736356.08562103]

subject15.normal.jpg:[ -1.72367135e+07 -9.65092059e+07 -2.88284013e+07 5.26177474e+07

-3.38045485e+07 2.38100059e+07 1.25604673e+08 9.30303930e+06]

(3)

**For test images:**

**The images after subtracting the mean face:**



**PCA coefficients for test images :**

subject01.centerlight.jpg:[ 10395241.06976799 22573856.61180452 538408.38078537

-14967989.40587762 -16146589.88160564 -33516565.55390254

-22154942.79347572 9964932.29084029]

subject01.happy.jpg:[ -7894318.05965359 15860340.390898 36771766.49410542

11622471.56347746 -31140945.8099955 -56872241.88557988

-25179041.35827489 46533113.43020264]

subject01.normal.jpg:[-10690469.06587205 44845387.57638889 43194205.74884757

14222155.59591849 -44508657.1889574 -74331087.54911807

-66582627.21369252 76262427.33480631]

subject02.normal.jpg:[ 25753823.85487584 -68513612.4564954 14602302.60837113

-20330165.06339063 1417866.42480582 -32833118.07314866

82351522.69026098 -54146390.88440429]

subject03.normal.jpg:[ -1412746.7156782 -3700741.26924959 52282147.30575235

52284398.57794323 8770899.89913118 50460079.1906319 51601112.39541467

29226393.11718583]

subject07.centerlight.jpg:[-12504549.94456453 -27044190.35059405 7453398.23834912 -10952672.3891477

38761095.21283457 21187970.44063096 -32893489.18883392

-25363697.19238757]

subject07.happy.jpg:[-20308490.57282989 -72243306.82866921 14193563.86585636

29441193.86798853 19833572.62606781 26733941.49189382

55116395.33983427 -25104961.93489834]

subject07.normal.jpg:[ -6.07226692e+07 -8.15286439e+07 4.78978099e+06 3.57442692e+07

4.50446659e+07 4.69145663e+07 1.13335503e+08 -6.52430239e+07]

subject10.normal.jpg:[ 36281898.80541495 -26409452.49338488 4346080.96456587

29301130.75790551 11700104.64209832 16040013.33178689

-91762000.07091594 -24815408.26342363]

subject11.centerlight.jpg:[ 3.68631502e+07 1.48066564e+08 -9.91102157e+07 -1.17648147e+08

-6.87628641e+06 -2.69434325e+07 -3.48142326e+07 -3.98882029e+07]

subject11.happy.jpg:[ 4.56868926e+07 2.19204196e+08 -1.17389464e+08 -1.39204628e+08

-1.62715545e+07 -3.54897004e+07 -1.08017073e+08 -1.33791650e+07]

subject11.normal.jpg:[ 4.85791901e+07 2.32378808e+08 -1.20766065e+08 -1.36849597e+08

-1.89070720e+07 -3.45578820e+07 -1.23871293e+08 -9.32339275e+06]

subject12.normal.jpg:[ 8840455.07550788 31122071.39159385 -5023029.00121815

-35142429.16457748 -4298626.12517298 -37855596.78187285

-72137476.30381912 7658834.13183002]

subject14.happy.jpg:[-25914742.59180762 2221187.72189186 36794643.7026297 -10993621.1389828

21250348.14377664 -3659809.14594334 -67738937.06887874

37812262.56159454]

subject14.normal.jpg:[-20552314.41867543 -562539.41618161 30379948.64414814

-26989939.30554907 30286740.85452466 4497422.96389425

-90676891.06181912 38736356.08562103]

subject14.sad.jpg:[-18059452.52374255 -7837221.94670912 33993071.7370602 -18497714.71341622

32223066.56033503 7688918.7253991 -67530700.31883344

23944303.19854085]

subject15.normal.jpg:[ -1.72367135e+07 -9.65092059e+07 -2.88284013e+07 5.26177474e+07

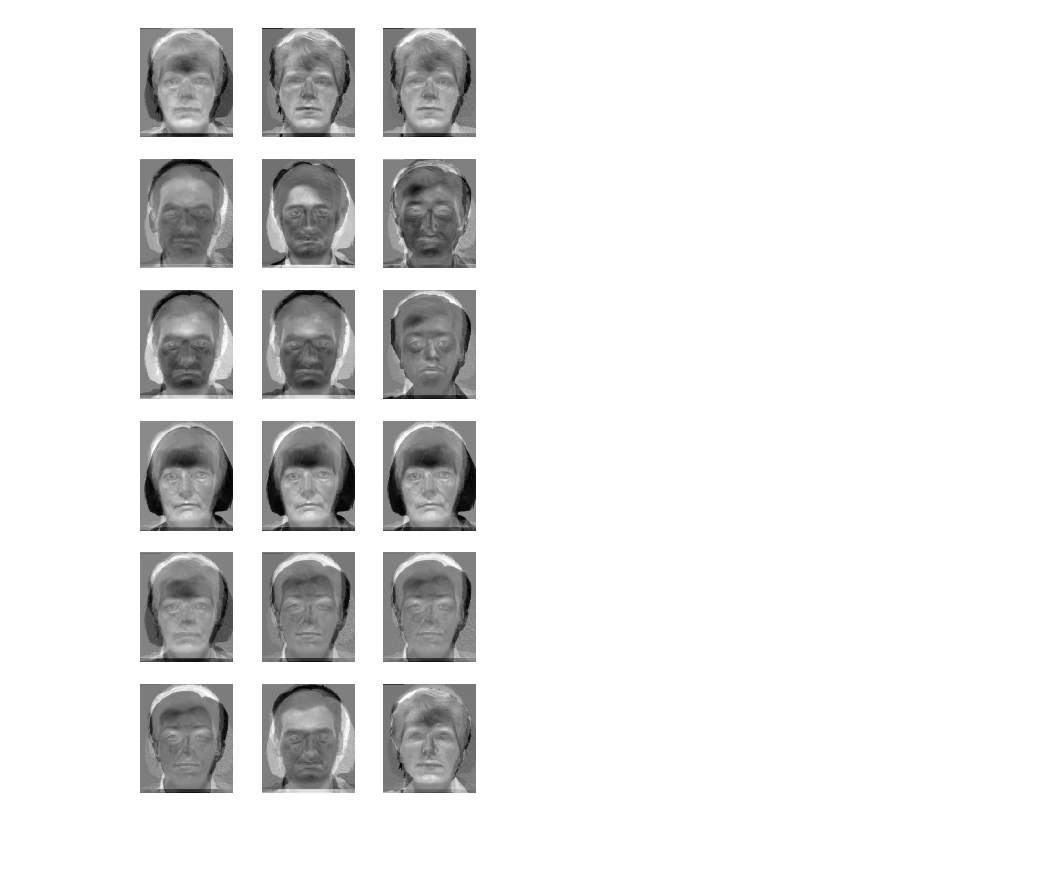
-3.38045485e+07 2.38100059e+07 1.25604673e+08 9.30303930e+06]

apple1\_gray.jpg:[-35643753.06585287 30095795.12025792 17096119.84756729

-5696473.61273158 -15120545.12480078 -40888787.83343802

-124018.31332567 34965383.44272368]

**The reconstructed face image IR**



**Distances di for i = 0 to M:**

i = 0, di =[ 1.11586822e+08 1.55242389e+08 1.46676511e+08 2.29591658e+08

1.19707328e+08 2.92870267e+08 1.07945628e+08 2.14023418e+08]

i = 1, di =[ 6.30678486e+07 1.81936137e+08 1.47059539e+08 2.49271519e+08

1.47572654e+08 3.32324142e+08 1.17694790e+08 2.22118700e+08]

i = 2, di =[ 0.00000000e+00 2.43328948e+08 1.96380903e+08 3.09073374e+08

1.76627570e+08 3.18169128e+08 1.33487089e+08 2.79018319e+08]

i = 3, di =[ 2.43328948e+08 0.00000000e+00 1.62849038e+08 1.42247445e+08

1.95249044e+08 4.09770238e+08 2.18741654e+08 1.42074948e+08]

i = 4, di =[ 1.96380903e+08 1.62849038e+08 0.00000000e+00 1.61764992e+08

1.71550250e+08 4.05241748e+08 1.73347239e+08 1.54399493e+08]

i = 5, di =[ 1.86019216e+08 1.47462237e+08 1.36206360e+08 1.76510231e+08

9.07477021e+07 3.43932045e+08 9.66798449e+07 2.04454021e+08]

i = 6, di =[ 2.33055760e+08 1.00426751e+08 1.03330781e+08 8.88316865e+07

1.64784912e+08 4.25382239e+08 1.85795025e+08 1.09655903e+08]

i = 7, di =[ 3.09073374e+08 1.42247445e+08 1.61764992e+08 0.00000000e+00

2.41359064e+08 4.75562142e+08 2.59247727e+08 1.26447655e+08]

**Classification result:**

Format: image to predict: classification result(number), classification result(original image name)

subject01.centerlight.jpg: unknown face

subject01.happy.jpg: 0, subject01.normal.jpg

subject01.normal.jpg: 0, subject01.normal.jpg

subject02.normal.jpg: 1, subject02.normal.jpg

subject03.normal.jpg: 2, subject03.normal.jpg

subject07.centerlight.jpg: 4, subject10.normal.jpg

subject07.happy.jpg: 3, subject07.normal.jpg

subject07.normal.jpg: 3, subject07.normal.jpg

subject10.normal.jpg: 4, subject10.normal.jpg

subject11.centerlight.jpg: unknown face

subject11.happy.jpg: 5, subject11.normal.jpg

subject11.normal.jpg: 5, subject11.normal.jpg

subject12.normal.jpg: 6, subject14.normal.jpg

subject14.happy.jpg: 6, subject14.normal.jpg

subject14.normal.jpg: 6, subject14.normal.jpg

subject14.sad.jpg: 6, subject14.normal.jpg

subject15.normal.jpg: 7, subject15.normal.jpg

apple1\_gray.jpg: unknown face

**Code**

**import** numpy **as** np

**import** cv2

**import** matplotlib **as** mpl

mpl.use('TkAgg')

**import** matplotlib.pyplot **as** plt

# Load image examples into matrix.

**def** loadExamples(input\_names):

R = []

**for** input\_name **in** input\_names:

img = cv2.imread(input\_name, 0).reshape((1, -1))[0].tolist()

R.append(img)

R = np.array(R)

**return** R

# Subtract means from images.

**def** subtract(mean, R):

A = R - mean.transpose()

**return** A

# Calculate eigenfaces of images.

**def** calculateEigen(A):

L = np.dot(A, np.transpose(A))

**print**(L)

eigen\_values, eigen\_vectors = np.linalg.eig(L)

**print**("eval:", eigen\_values)

**print**("evec:", eigen\_vectors)

eigenfaces = np.dot(eigen\_vectors, A)

**print**("eigen faces:", eigenfaces)

**return** eigenfaces

# Transfer images to eigenfaces.

**def** transferToEigenfaces(A, eigenfaces):

**return** np.dot(A, np.transpose(eigenfaces))

# Classify the images.

**def** recognize(A, U, omega\_i):

omega\_I = np.dot(A, np.transpose(U))

**print**("PCA coefficients for test images:")

printPCACoefficients(testing\_examples\_names, N, omega\_I)

**print**("omega\_I:", omega\_I)

reconstructed\_images = np.dot(omega\_I, U)

normalized\_reconstructed\_images = normalize(reconstructed\_images)

pltMultipleImg(normalized\_reconstructed\_images, N, h, w, 6) # plot reconstructed images

**print**("reconstructed\_images:", reconstructed\_images)

distance\_para\_img\_reconstruction = reconstructed\_images - A

**print**("distance reconstruction:", distance\_para\_img\_reconstruction)

**print**("distance parameters:", distance\_para\_img\_reconstruction.shape)

d0 = np.sqrt(np.sum(np.square(distance\_para\_img\_reconstruction), axis=1))

**print**("d0:", d0)

di = []

**for** cur\_omega\_i **in** omega\_i:

cur\_distance\_para\_face\_space = omega\_I - cur\_omega\_i

cur\_distance\_face\_space = np.sqrt(np.sum(np.square(cur\_distance\_para\_face\_space), axis=1)).tolist()

di.append(cur\_distance\_face\_space)

di = np.array(di).transpose()

dj = np.amin(di, axis=1) # dj = min{di}

di = np.array(di)

printDistanceDi(M, di)

predict\_sort\_number = np.argsort(di, axis=1)[:, 0]

true\_sort\_number = np.array([0, 0, 0, 1, 2, 3, 3, 3, 4, 5, 5, 5, -1, 6, 6, 6, 7, -1])

**print**("predict sort number:", predict\_sort\_number)

**print**("true sort number:", true\_sort\_number)

**print**("dj:", dj)

printClassificationResult(training\_examples\_names, testing\_examples\_names, d0, dj, predict\_sort\_number)

# Plot single image.

**def** pltImg(img):

img = np.array(img, dtype=np.uint8)

plt.imshow(img, cmap=plt.cm.gray)

plt.axis('off')

# plt.show()

# Plot multiple images.

**def** pltMultipleImg(img, nplt, h, w, plot\_row\_nums):

**for** i **in** range(nplt):

plt.subplot(plot\_row\_nums, nplt / plot\_row\_nums, i + 1)

pltImg(img[i, :].reshape([h, w]))

plt.show()

# Normalize the images to range (0, 255).

**def** normalize(img):

image = img

minval = image.min()

maxval = image.max()

newimg = (img - minval) \* (255.0/(maxval-minval))

**return** newimg

# Print PCA coefficients.

**def** printPCACoefficients(examples\_names, example\_nums, omega\_i):

**for** i **in** range(example\_nums):

**print**(examples\_names[i] + ":" + str(omega\_i[i, :]))

# Print distance between input face and training images in the face space di.

**def** printDistanceDi(M, di):

**for** i **in** range(M):

**print**("i = " + str(i) + ", di =" + str(di[i, :]))

# Print classification result.

**def** printClassificationResult(training\_examples\_names, testing\_examples\_names, d0, dj, predict\_sort\_number):

N = len(testing\_examples\_names)

**print**("Classification result:")

**print**("image to predict: classification result(number), classification result(original image name)")

**for** i **in** range(N):

**if** d0[i] > 7***e***+12:

result = 'non-face'

**elif** dj[i] > 1***e***+8:

result = 'unknown face'

**else**:

result = str(predict\_sort\_number[i]) + ', ' + str(training\_examples\_names[predict\_sort\_number[i]])

**print**(testing\_examples\_names[i] + ": " + result)

training\_examples\_names = ['subject01.normal.jpg', 'subject02.normal.jpg', 'subject03.normal.jpg',

'subject07.normal.jpg', 'subject10.normal.jpg', 'subject11.normal.jpg',

'subject14.normal.jpg', 'subject15.normal.jpg']

testing\_examples\_names = ['subject01.centerlight.jpg', 'subject01.happy.jpg', 'subject01.normal.jpg',

'subject02.normal.jpg', 'subject03.normal.jpg', 'subject07.centerlight.jpg',

'subject07.happy.jpg', 'subject07.normal.jpg', 'subject10.normal.jpg',

'subject11.centerlight.jpg', 'subject11.happy.jpg', 'subject11.normal.jpg',

'subject12.normal.jpg', 'subject14.happy.jpg', 'subject14.normal.jpg', 'subject14.sad.jpg',

'subject15.normal.jpg', 'apple1\_gray.jpg']

M = 8 # number of training examples

N = 18 # number of testing examples

h = 231 # picture height pixels

w = 195 # picture width pixels

R\_training = loadExamples(training\_examples\_names) # training examples' pixel grayscales

m = np.mean(R\_training, axis=0) # mean value of training examples

**print**("shape of mean:", m.shape)

pltImg(m.reshape([h, w])) # print mean face image

A\_training = subtract(m, R\_training) # training face subtracted from the mean face m

**print**("R training:", R\_training)

**print**("A training:", A\_training)

U = calculateEigen(A\_training) # eigenfaces

normalized\_U = normalize(U)

pltMultipleImg(normalized\_U, M, h, w, 2) # plot eigenfaces

omega\_i = transferToEigenfaces(A\_training, U) # training examples in eigenfaces

**print**("print training examples PCA coefficients")

printPCACoefficients(training\_examples\_names, M, omega\_i)

R\_testing = loadExamples(testing\_examples\_names)

A\_testing = subtract(m, R\_testing)

normalized\_A\_testing = normalize(A\_testing)

pltMultipleImg(normalized\_A\_testing, N, h, w, 6) # plot image after subtracting the mean face

recognize(A\_testing, U, omega\_i)