CS 6643 - Computer Vision & Image Analysis

New York University
Tandon School of Engineering

Project 2: Face Recognition

Ayan Agrawal
Net ID: aka398

a) Programming Language used and Instruction on how to compile the program:

Programming Language: Python 2.7

Instruction to compile:

a) Go to command prompt of the desktop

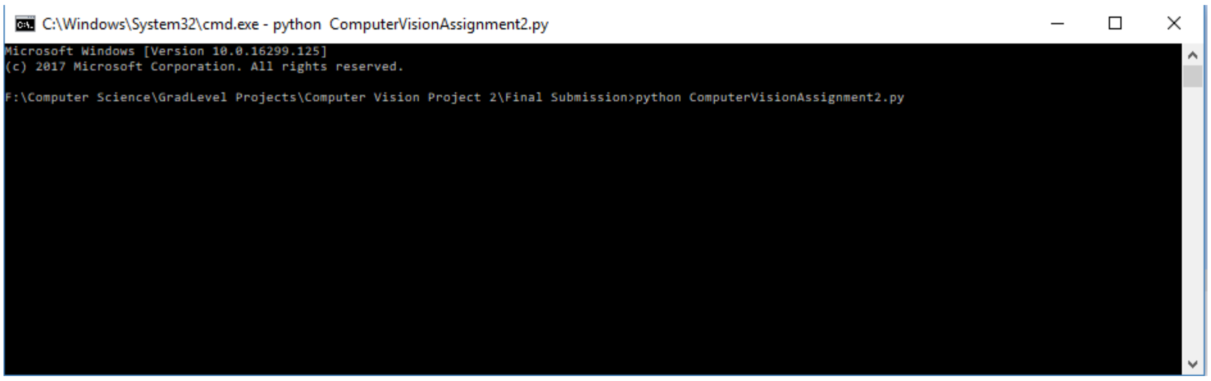
b) Copy and paste the below code in the editor of python and save the file name as file_name.py

c) Go to the path where the file_name.py file is located

d) Write the below commands

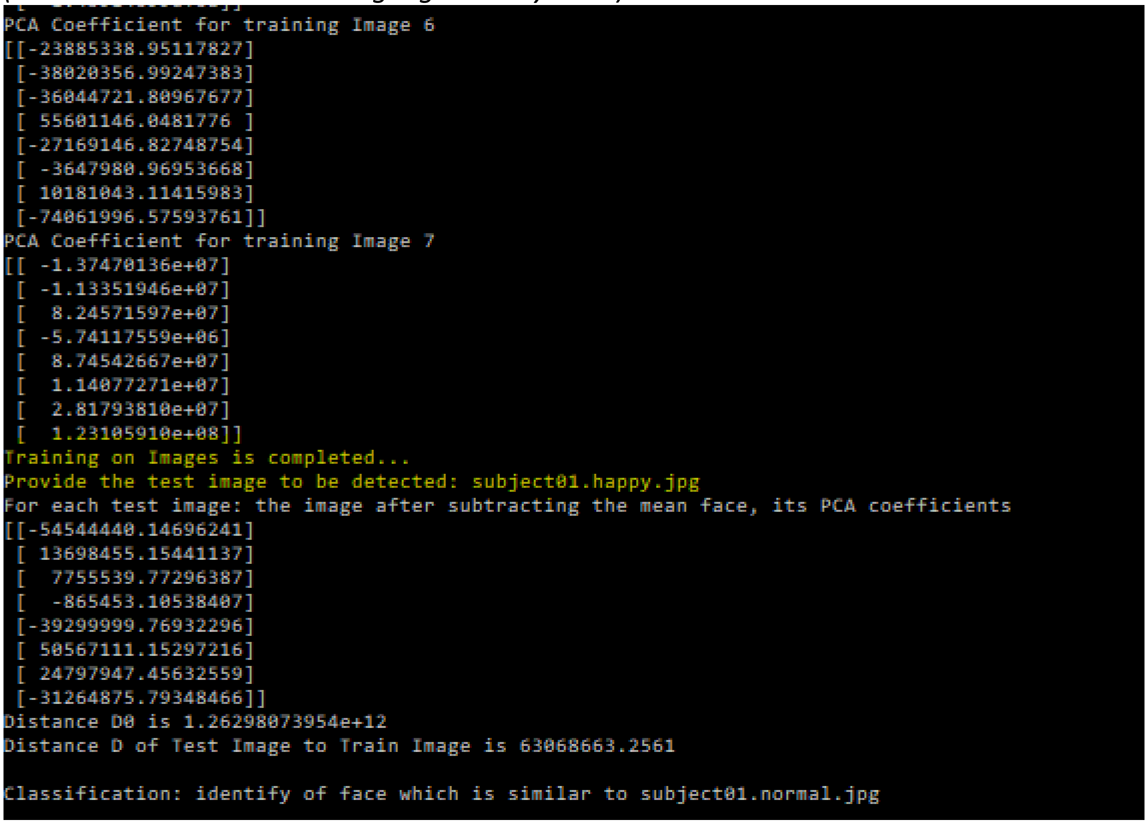
python file_name.py

See the below screenshot for example


C:\Windows\System32\cmd.exe - python ComputerVisionAssignment2.py

```
Microsoft Windows [Version 10.0.16299.125]
(c) 2017 Microsoft Corporation. All rights reserved.

F:\Computer Science\GradLevel Projects\Computer Vision Project 2\Final Submission>python ComputerVisionAssignment2.py
```

e) Later, test image to be provided when prompted after the training on images is done. *(See the below screenshot highlighted in yellow)*



```
PCA Coefficient for training Image 6
[[-23885338.95117827]
 [-38020356.99247383]
 [-36044721.80967677]
 [ 55601146.0481776 ]
 [-27169146.82748754]
 [ -3647980.96953668]
 [ 10181043.11415983]
 [-74061996.57593761]]
PCA Coefficient for training Image 7
[[ -1.37470136e+07]
 [ -1.13351946e+07]
 [  8.24571597e+07]
 [ -5.74117559e+06]
 [  8.74542667e+07]
 [  1.14077271e+07]
 [  2.81793810e+07]
 [  1.23105910e+08]]
Training on Images is completed...
Provide the test image to be detected: subject01.happy.jpg
For each test image: the image after subtracting the mean face, its PCA coefficients
[[-54544440.14696241]
 [ 13698455.15441137]
 [  7755539.77296387]
 [  -865453.10538407]
 [-39299999.76932296]
 [ 50567111.15297216]
 [ 24797947.45632559]
 [-31264875.79348466]]
Distance D0 is 1.26298073954e+12
Distance D of Test Image to Train Image is 63068663.2561

Classification: identify of face which is similar to subject01.normal.jpg
```

Please note: Below libraries are required for the code to be compiled
sys
math
PIL → Image
numpy
matplotlib

Source code of the program with inline comments

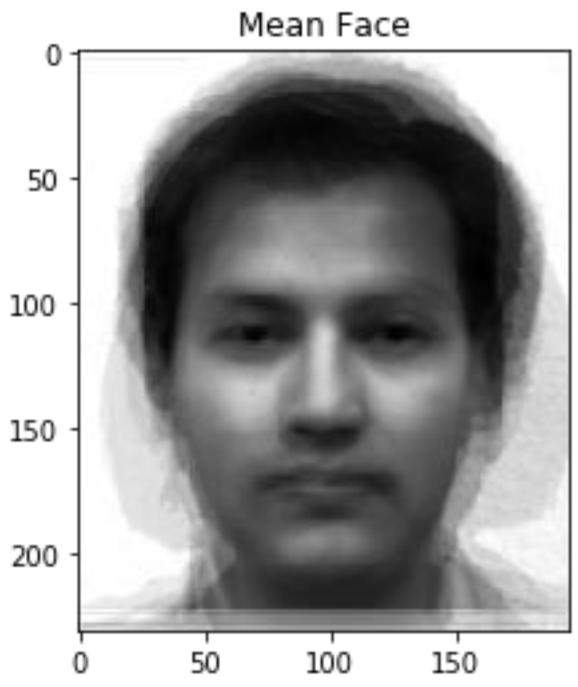
```
import sys
import math
#importing Image library to read and display the image.
from PIL import Image
#Library to show the images
from matplotlib import pyplot as plt
import matplotlib.image as mpimg
#Library for matrix calculations
import numpy as np
#Library to calculate eigen values and eigen vectors
from numpy import linalg as LA

#==================================Implementing EigenFaces Training
Functions=====================================================#
#For each training image, the rows are stacked together to form a column vector Ri
of dimension width*heght
#here all the images are stacked in the list imagesN2vector
def convertToN2vector(width, height):
    imagesN2vector = []
    for images in range(len(Imageobjects)):
        k = 0
        individualImages = np.zeros((width*height,1),dtype = np.int16)
        for i in range(height):
            for j in range(width):
                #storing the pixel value in the form of increasing rows with single
column
                individualImages[k,0] = Imageobjects[images].getpixel((j,i))
                k = k + 1
        #appending single image to stack all the images
        imagesN2vector.append(individualImages)
    return imagesN2vector


#The mean face m(meanFace) is computed by taking the average of the M training face
images
#providing the input as obtained in the above method
def averageFaceVector(imagesN2vector):
    meanFace = np.zeros((width*height,1),dtype = np.int16)
    M = len(imagesN2vector)
    for i in range(len(imagesN2vector[0])):
        sum = 0
        #taking the sum of all pixel in one row and dividing by the number of
images
        for images in range(len(imagesN2vector)):
            sum = sum + imagesN2vector[images][i][0]
        sum = sum/M
        meanFace[i][0] = sum
    return meanFace

#subtracting the mean face m from each training face
#providing the input as obtained in the above method to subtract
def subtractMeanFace(imagesN2vector, meanFace):
    subtractmeanface = []
    for images in range(len(Imageobjects)):
        #using the subtract function of the numpy library
        individualImages = np.subtract(imagesN2vector[images],meanFace)
        #appending the individual subtracted image stacked in one list
        subtractmeanface.append(individualImages)
    return subtractmeanface

#All training faces into a single matrix A of dimension [width*height,
no_of_test_images]
#providing the input as obtained in the above method
def matrixA(R):
    A = np.zeros((width*height,len(R)),dtype = np.int16)
    for i in range(width*height):
        for j in range(len(R)):
            A[i,j] = R[j][i][0]
    return A

#since calculation of eigen values and eigen vectors from co-variance matrix will
require large computational effort as matrix will be large
#Implementing the alternate method to calculate the eigenvalues
#AT is the transpose of matrix A obtained above
#taking the dot product of AT and A to get the matrix L
# calculating the eigenvalue and eigen vector
# w is the eigenvalue, V is the eigenvector
def alternateToCovariance(A):
    AT = np.transpose(A)
    L = np.dot(AT, A)
    w, V = LA.eig(L)
    return V

#Eigen vectors of C can be found by U = AV
# U is the eigenspace, face spave or eigenfaces
def covariance(A, V):
    U = np.dot(A, V)
    return U

#printing all the eigen faces based on the dataset of the images provided.
# For 8 trained image we will get 8 eigenFaces
def printEigenFaces(U):
    for i in range (len(U[0])):
        plt.title('Eigen face'+ str(i))
        plt.imshow((U[:,i].reshape(231,195)),cmap='gray')
        plt.show()

#Each training face can then be projected on the face space
#UT is the transpose of U as obtained above
#projectedfacespace = (UT)(Ri)
def projectedFaceSpace(U, R):
    UT = np.transpose(U)
    rows, column = UT.shape
    projectedfacespace = []
    for images in range(len(Imageobjects)):
        projectedfacespace.append(np.dot(UT, R[images]))
    return projectedfacespace

#printing all the PCA coefficients from the projected face space obtained in above
method
#8 training image will have 8 set of PCA coeffients
def printPCACoefficients(projectedfacespace):
    for i in range(len(projectedfacespace)):
        print "PCA Coefficient for training Image", i
        print projectedfacespace[i]


#===================================Implementing EigenFaces Recognition
Functions=====================================================#
#reading the test image of which the face needs to be recognized
```

```python
#the function will take single test image at a time
def getTestImage(width, height):
    Testimage = np.zeros((width*height,1),dtype = np.int16)
    k = 0
    for i in range(height):
        for j in range(width):
            Testimage[k,0] = test_image_object.getpixel((j,i))
            k = k + 1
    return Testimage

#subtracting the mean face m from each test face
#Mean face m was obtained in the above training methods
#providing the input as obtained in the above method to subtract
def subtractTestFace(testimage, meanFace):
    subtracttestface = np.subtract(testimage,meanFace)
    return subtracttestface

#computing its projection onto the face space
#UT is the transpose of U as obtained above in the training method
#projectionface = (UT)(I)
#I is the subtracted image as obtained above after subtracting
def projectiononFace(U, I):
    projectionface = np.dot(np.transpose(U), I)
    return projectionface

#Reconstruct input face image from the eigenfaces
#reconstructedimage = (U)(projectionface)
#where U is the eigenface and the projectionFace of the test image is obtained
above
def reconstruct(U, projectionface):
    reconstructedimage = np.dot(U, projectionface)
    return reconstructedimage

#Computing the distance between the input face image and the reconstruction of the
image
#Subtracting the value pixel by pixel and then passing the complete vector to get
the euclidean distance
def findEuclideanDistance(reconstructedimage, I):
    subtractedform = np.subtract(reconstructedimage, I)
    subtractedform = LA.norm(subtractedform)
    return subtractedform

#Compute distance between input face image and training images in the face space
#projectionface is the projected test face
#projectedfacespace[i] is the individual traing images
#Test image is subtracted from all the training images then the image with the
minimum distance is taken with the image in the dataset
def computeDistance(projectionface, projectedfacespace):
    Di = []
    for i in range(len(projectedfacespace)):
        di = LA.norm(np.subtract(projectionface, projectedfacespace[i]))
        Di.append(di)
    return Di

def displayFinalResult(testpath,imagePath):
    plt.title('Input Test Image')
    plt.imshow(mpimg.imread(testpath),cmap='gray')
    plt.figure()
    plt.imshow(mpimg.imread(imagePath),cmap='gray')
    plt.title('Resulting image')
    plt.show()

#==================================================================================
================================================
#EigenFaces Training
#Training Dataset
#TrainingImages list contains the set images as training dataset
#change the path in the 'imagePath' variable where all the images are stored
```

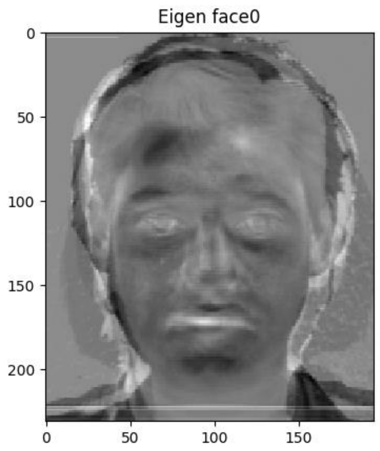
```python
#change to the path of images stored on the machine
storedimagepath = "Dataset\\"
TrainingImages = ['subject01.normal.jpg', 'subject02.normal.jpg',
'subject03.normal.jpg', 'subject07.normal.jpg', 'subject10.normal.jpg',
'subject11.normal.jpg', 'subject14.normal.jpg', 'subject15.normal.jpg']
Imageobjects = []
for i in range(len(TrainingImages)):
    imagePath = storedimagepath + TrainingImages[i]
    image_object = Image.open(imagePath)
    Imageobjects.append(image_object)
#All the images are of same size
#Calculating the width and height
width, height = Imageobjects[0].size

imagesN2vector = convertToN2vector(width, height)

meanFace = averageFaceVector(imagesN2vector)
plt.title("Mean Face m")
plt.imshow(meanFace.reshape(231,195), cmap = "gray")
plt.show()

R = subtractMeanFace(imagesN2vector, meanFace)

A = matrixA(R)

V = alternateToCovariance(A)

U = covariance(A, V)

printEigenFaces(U)

projectedfacespace = projectedFaceSpace(U, R)

printPCACoefficients(projectedfacespace)

#==================================================================================
=======================================================
#Testing
#EigenFaces Recognition
print "Training on Images is completed..."
testImage = raw_input("Provide the test image to be detected: ")
testimagePath = "Dataset\\" + testImage
test_image_object = Image.open(testimagePath)
width, height = test_image_object.size

testimage = getTestImage(width, height)

I = subtractTestFace(testimage, meanFace)
plt.title("The image after subtracting from the mean face")
plt.imshow(I.reshape(231,195), cmap = "gray")
plt.show()

projectionface = projectiononFace(U, I)
print "For each test image: the image after subtracting the mean face, its PCA
coefficients"
print projectionface

reconstructedimage = reconstruct(U, projectionface)
plt.title("The reconstructed face Image")
plt.imshow(reconstructedimage.reshape(231,195), cmap = "gray")
plt.show()

subtractedform = findEuclideanDistance(reconstructedimage, I)
print "Distance D0 is",subtractedform
#Choosing the threshold
#T0 is used to identify whether the image is face or non-face
T0 = 7000000000000
#T1 is used to identify whether the face is present in the dataset or not
```

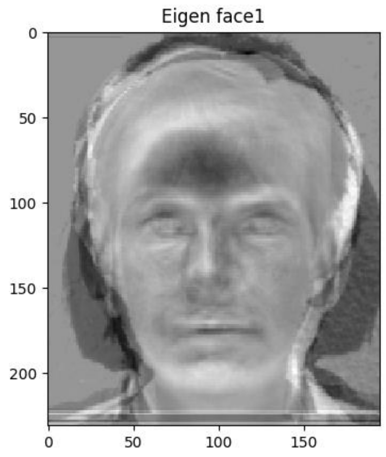
```
T1 = 89000000
if (subtractedform > T0):
    print ""
    print "Classification: non-face"

else:
    Di = computeDistance(projectionface, projectedfacespace)
    print "Distance D of Test Image to Train Image is",min(Di)
    if (min(Di) > T1):
        print ""
        print "Classification: unknown face"
    else:
        print ""
        print "Classification: identify of face which is similar to",
TrainingImages[Di.index(min(Di))]

        displayFinalResult(testimagePath,storedimagepath +
TrainingImages[Di.index(min(Di))])
```

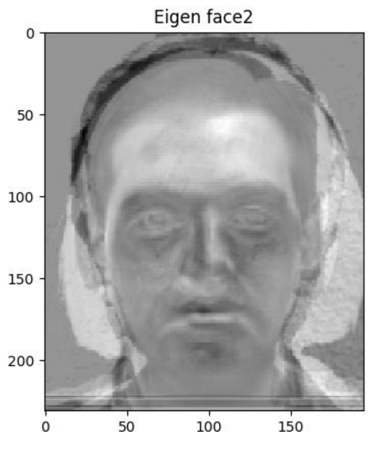
1) Manually chosen Threshold values **T_0 = 7000000000000** **T_1 = 89000000**
The mean Face *m* is

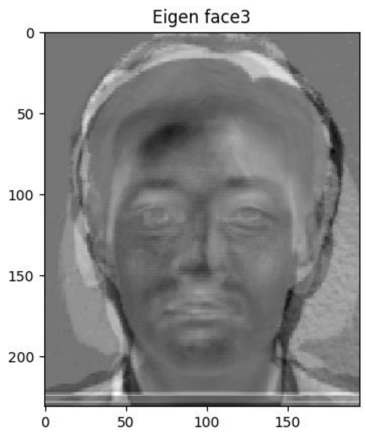


The M Eigenfaces are

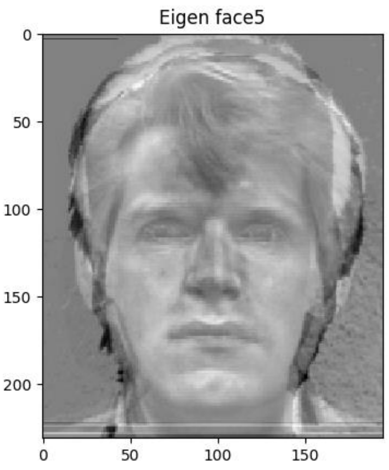


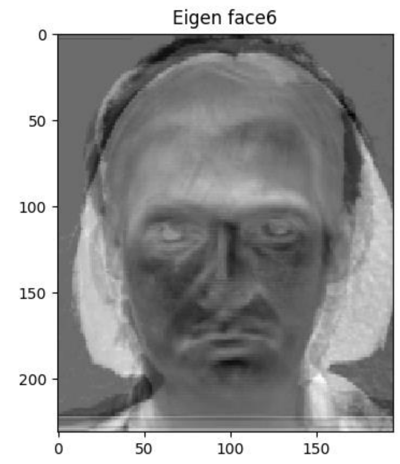


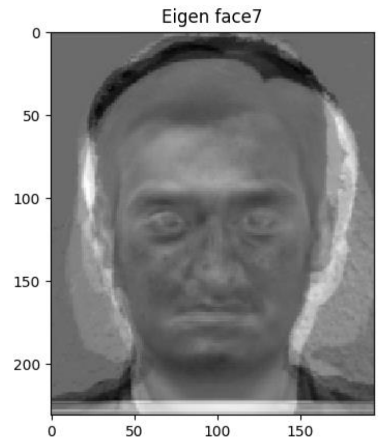



Eigen face3
0
50
100
150
200
0
50
100
150

Eigen face4
0
50
100
150
200
0
50
100
150


Eigen face5
0
50
100
150
200
0
50
100
150





(2) The PCA coefficients (Ωi) for each training image.

PCA Coefficient for training Image 'subject01.normal.jpg'
[[-80205224.60550584], [10921231.32780848], [-7900044.86273242], [6733589.57055455],
[-73825931.56067842], [78334122.09522335], [11902241.54677157], [-60917507.72030157]]

PCA Coefficient for training Image 'subject02.normal.jpg'
[[27892934.57546353], [44730062.0064938], [69311634.75480063], [-34802096.75608037],
[60037790.27052878], [-34354328.33880106], [52984127.12661389], [19034103.74077592]]

PCA Coefficient for training Image 'subject03.normal.jpg'
[[-28668031.43676752], [-57753655.62062956], [-5769809.31555467], [-31032457.53040214],
[-6704369.90149155], [-11911976.36457956], [45679017.82804587], [65466025.71603002]]

PCA Coefficient for training Image 'subject07.normal.jpg'
[[5.79712078e+07], [-2.73065746e+06], [1.92375614e+07], [2.19613691e+07],
[7.65241976e+07], [-4.51439070e+07], [7.99699093e+07], [1.20183374e+08]]

PCA Coefficient for training Image 'subject10.normal.jpg'
[[40076541.46590143], [-50802807.97120582], [34765863.02261727], [34687717.42100515]
[-55313829.4070871], [-15704513.25544917], [-17965697.71993631], [-45287637.82963333]]

PCA Coefficient for training Image 'subject11.normal.jpg'
[[1.62112834e+07], [1.03868223e+08], [-1.54883456e+08], [-4.69933026e+07]
[-6.21221660e+07], [2.41960251e+07], [-2.08372249e+08], [-1.45914099e+08]]

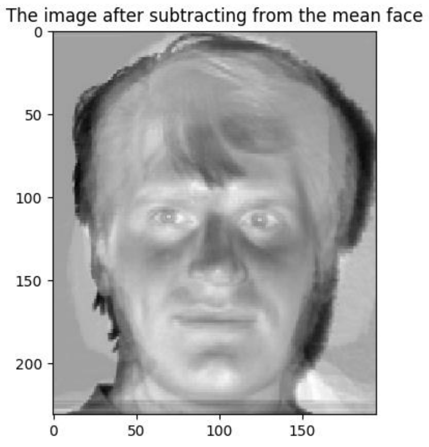
PCA Coefficient for training Image 'subject14.normal.jpg'
[[-23885338.95117828], [-38020356.99247386], [-36044721.80967672], [55601146.04817761]
[-27169146.82748754], [-3647980.96953667], [10181043.11415982], [-74061996.57593761]]

PCA Coefficient for training Image 'subject15.normal.jpg'
[[-1.37470136e+07], [-1.13351946e+07], [8.24571597e+07], [-5.74117559e+06]
[8.74542667e+07], [1.14077271e+07], [2.81793810e+07], [1.23105910e+08]]

(3) For each test image:

1) For Image: subject01.centerlight.jpg

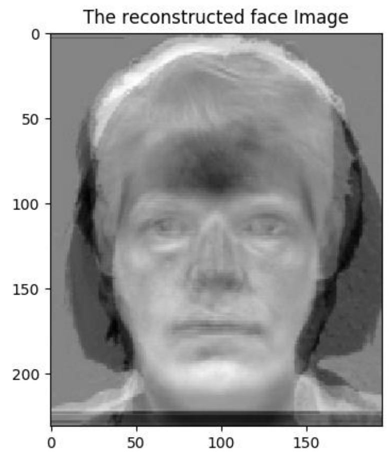
The image after subtracting the mean face (I)



PCA coefficients

```
[[-14872799.77606913]
 [ 22294172.21058068]
 [ -1576774.69698971]
 [ -8284186.99806547]
 [-18069392.98878718]
 [ 19700950.31492519]
 [-11467257.15441374]
 [-34471886.86632913]]
```

The reconstructed face image (IR)

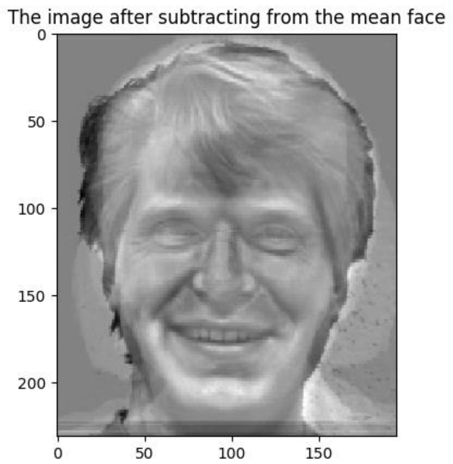


Distance D0 is 917157501536.0
Distance D of Test Image to Train Image is 107946501.821

Classification: unknown face

2) For Image: subject01.happy.jpg

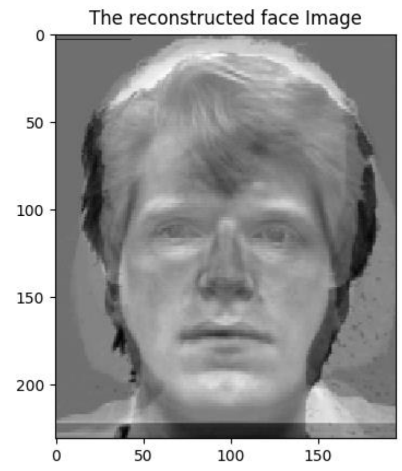
The image after subtracting the mean face (I)



PCA coefficients

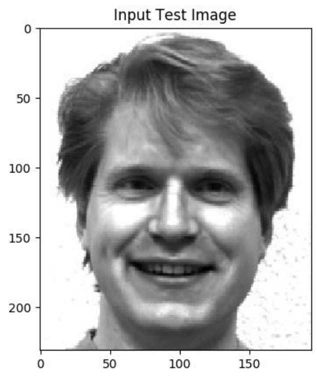
```
[[-54544440.14696241]
 [ 13698455.15441137]
 [  7755539.77296387]
 [  -865453.10538407]
 [-39299999.76932296]
 [ 50567111.15297216]
 [ 24797947.45632559]
 [-31264875.79348466]]
```

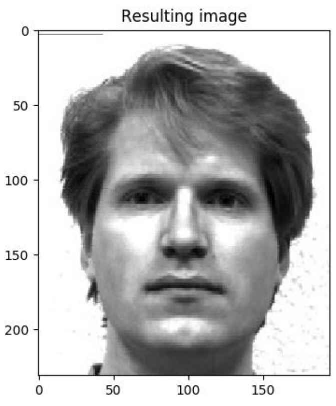
The reconstructed face image (IR)



Distance D0 is 1.26298073954e+12
Distance D of Test Image to Train Image is 63068663.2561

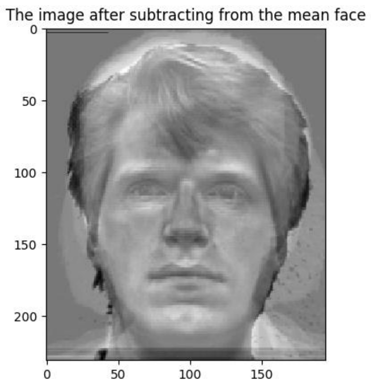
Classification: identify of face which is similar to subject01.normal.jpg





3) For Image: subject01.normal.jpg

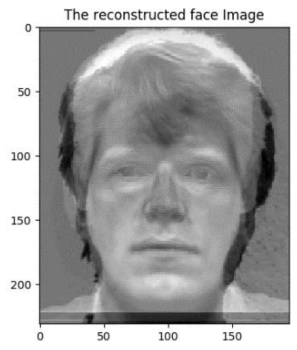
The image after subtracting the mean face (I)



PCA coefficients

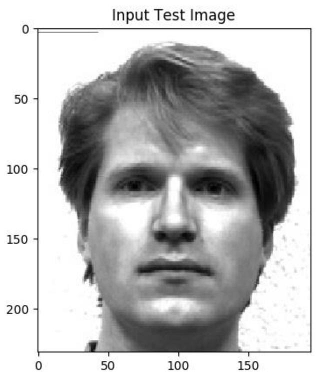
```
[[-80205224.60550581]
 [ 10921231.32780861]
 [ -7900044.86273236]
 [  6733589.57055458]
 [-73825931.56067851]
 [ 78334122.09522331]
 [ 11902241.54677152]
 [-60917507.72030156]]
```

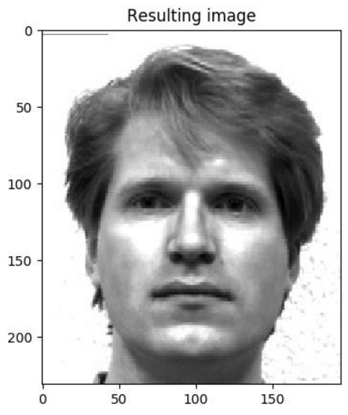
The reconstructed face image (IR)



Distance D0 is 2.24581189888e+12
Distance D of Test Image to Train Image is 0.0

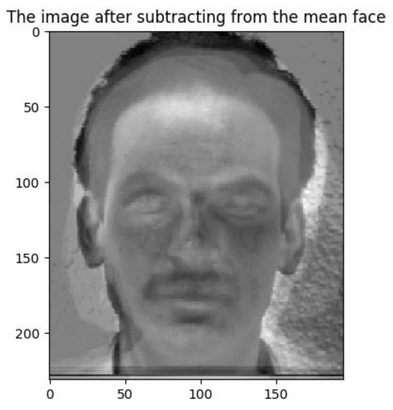
Classification: Identify of face which is similar to subject01.normal.jpg





4) For Image: subject02.normal.jpg

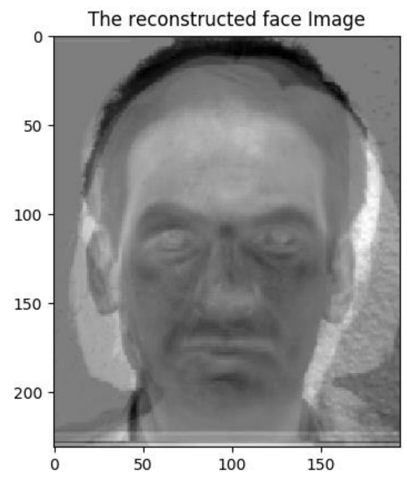
The image after subtracting the mean face (I)



PCA coefficients

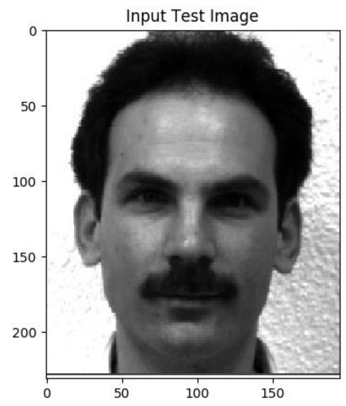
```
[[ 27892934.57546354]
 [ 44730062.00649374]
 [ 69311634.75480068]
 [-34802096.75608035]
 [ 60037790.2705288 ]
 [-34354328.33880103]
 [ 52984127.12661389]
 [ 19034103.74077592]]
```

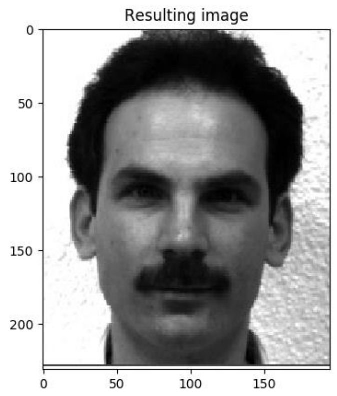
The reconstructed face image (IR)



Distance D0 is 2.04860054474e+12
Distance D of Test Image to Train Image is 0.0

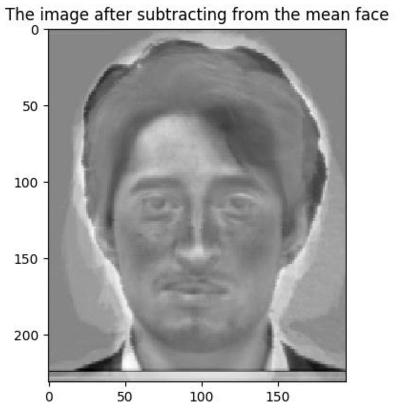
Classification: Identify of face which is similar to subject02.normal.jpg





5) For Image: subject03.normal.jpg

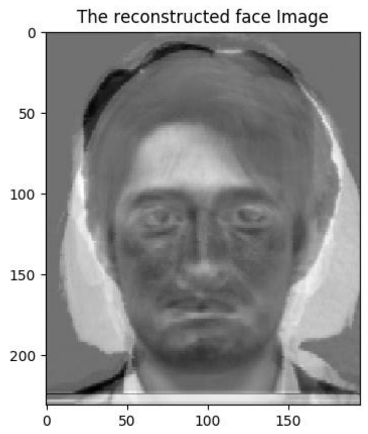
The image after subtracting the mean face (I)



PCA coefficients

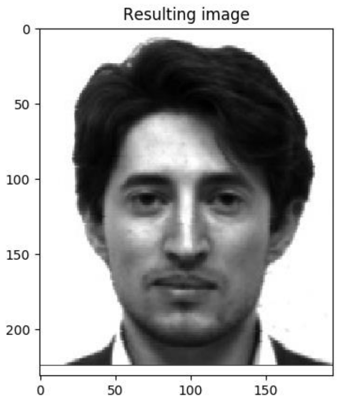
```
[[-28668031.43676755]
 [-57753655.62062956]
 [ -5769809.31555461]
 [-31032457.53040218]
 [ -6704369.90149159]
 [-11911976.36457955]
 [ 45679017.82804589]
 [ 65466025.71603002]]
```

The reconstructed face image (IR)



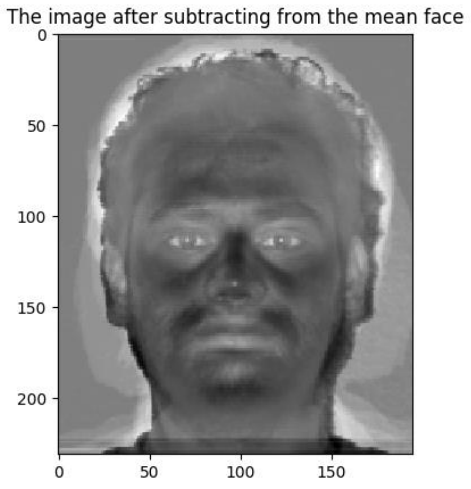
Distance D0 is 1.62824684849e+12
Distance D of Test Image to Train Image is 0.0

Classification: Identify of face which is similar to subject03.normal.jpg



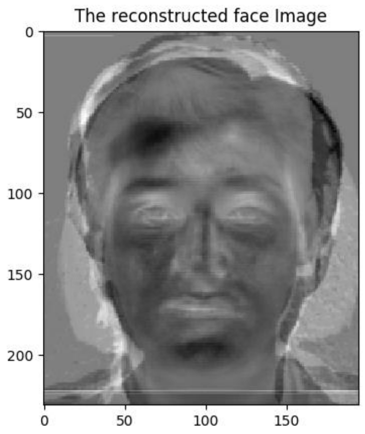
6) For Image: subject07.centerlight

The image after subtracting the mean face (I)



PCA coefficients
```
[[ 32786080.66342004]
 [-20850065.70159742]
 [ -7764624.32620306]
 [ 38308921.84201576]
 [  3025138.90665058]
 [-32061098.96800448]
 [ 15473938.21380893]
 [-19330412.34346486]]
```

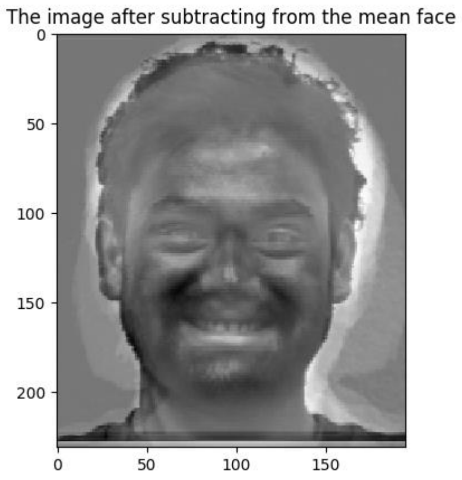
The reconstructed face image (IR)



Distance D0 is 783149411393.0
Distance D of Test Image to Train Image is 90748184.5007

Classification: unknown face

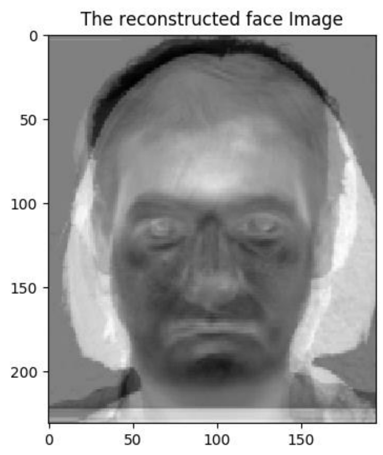
7) For Image: subject07.happy
The image after subtracting the mean face (I)



PCA coefficients

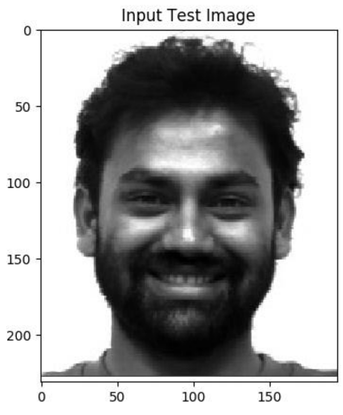
```
[[ 22694887.72016218]
 [-21292171.28866842]
 [ 37807675.25791071]
 [ 16129797.15767356]
 [ 41269293.45986101]
 [-26048131.31434928]
 [ 53249561.49343546]
 [ 60167647.89946216]]
```

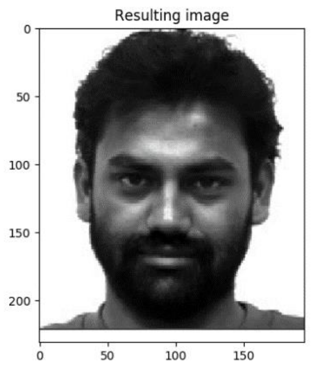
<u>The reconstructed face image (IR)</u>



<u>Distance D0</u> <u>is 2.12385368524e+12</u>
<u>Distance D</u> <u>of Test Image to Train Image is 88832574.2068</u>

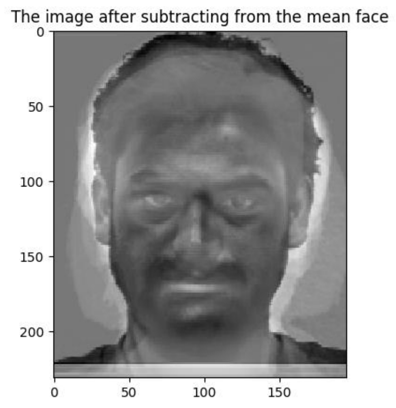
<u>Classification:</u> Identify of face which is similar to subject07.normal.jpg





8) For Image: subject07.normal

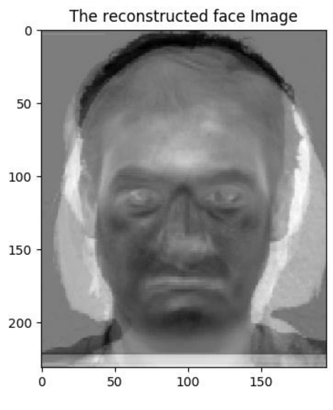
The image after subtracting the mean face (I)



PCA coefficients

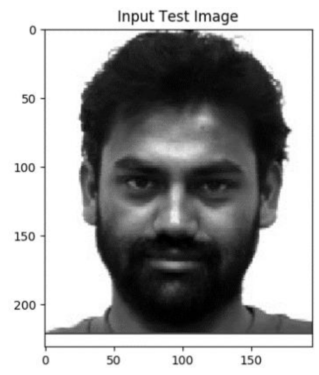
```
[[  5.79712078e+07]
 [ -2.73065746e+06]
 [  1.92375614e+07]
 [  2.19613691e+07]
 [  7.65241976e+07]
 [ -4.51439070e+07]
 [  7.99699093e+07]
 [  1.20183374e+08]]
```

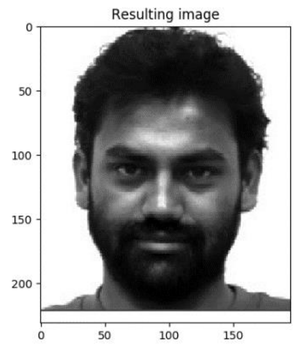
The reconstructed face image (IR)



Distance D0 is 3.38521595036e+12
Distance D of Test Image to Train Image is 0.0

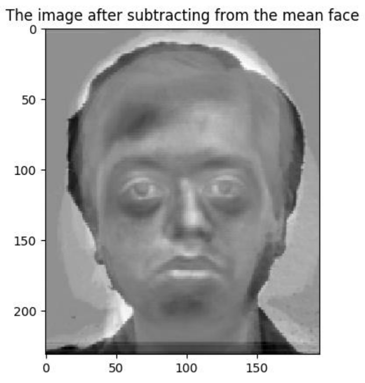
Classification: Identify of face which is similar to subject07.normal.jpg





<u>9) For Image: subject10.normal</u>

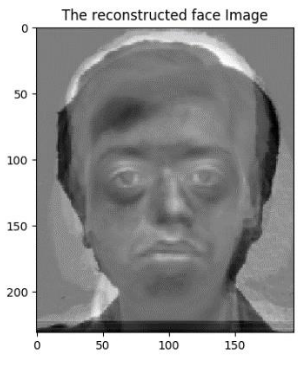
<u>The image after subtracting the mean face (I)</u>



<u>PCA coefficients</u>

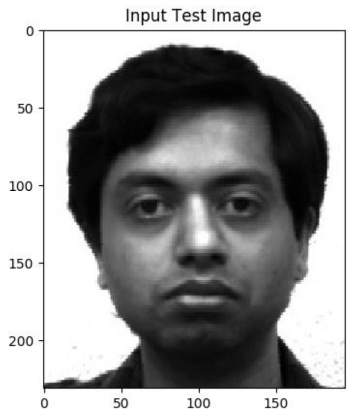
```
[[ 40076541.46590145]
 [-50802807.97120582]
 [ 34765863.02261723]
 [ 34687717.42100519]
 [-55313829.40707707]
 [-15704513.25544923]
 [-17965697.7199363 ]
 [-45287637.82963336]]
```

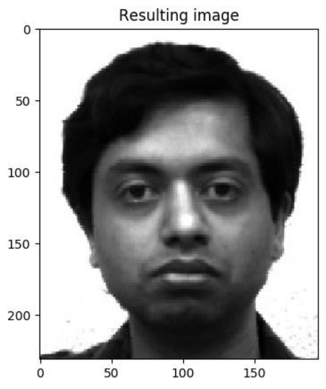
<u>The reconstructed face image (IR)</u>



Distance D0 is 1.35493722207e+12
Distance D of Test Image to Train Image is 0.0

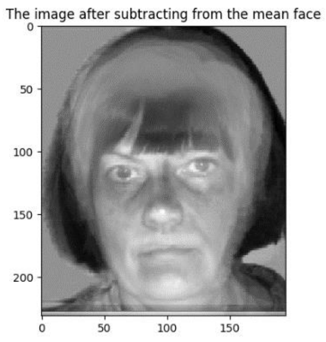
Classification: Identify of face which is similar to subject10.normal.jpg





10) For Image: subject11.centerlight

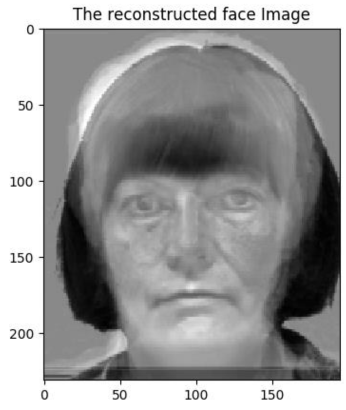
The image after subtracting the mean face (I)



PCA coefficients

```
[[  3.42153564e+07]
 [  1.04208376e+08]
 [ -9.88799316e+07]
 [ -5.21542850e+07]
 [ -7.16501257e+05]
 [ -4.08491357e+06]
 [ -1.38612238e+08]
 [ -8.08906554e+07]]
```

The reconstructed face image (IR)

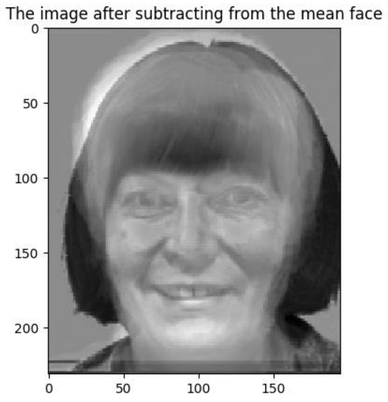


Distance D0 is 4.12146771073e+12
Distance D of Test Image to Train Image is 130966665.208

Classification: unknown face

11) For Image: subject11.happy

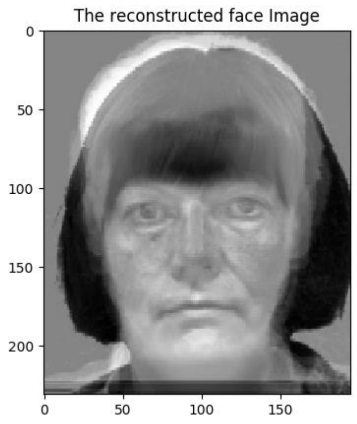
The image after subtracting the mean face (I)



PCA coefficients

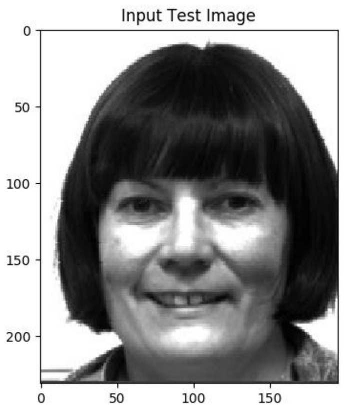
```
[[  1.74586565e+07]
 [  1.07322365e+08]
 [ -1.47788862e+08]
 [ -4.81995240e+07]
 [ -4.86603885e+07]
 [  1.89942320e+07]
 [ -1.96510220e+08]
 [ -1.38150826e+08]]
```

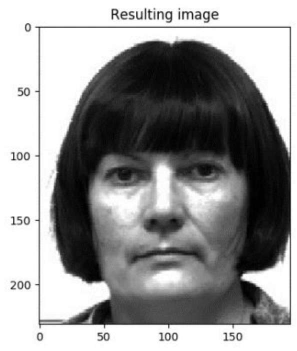
The reconstructed face image (IR)



Distance D0 is 6.06667954242e+12
Distance D of Test Image to Train Image is 21783698.2848

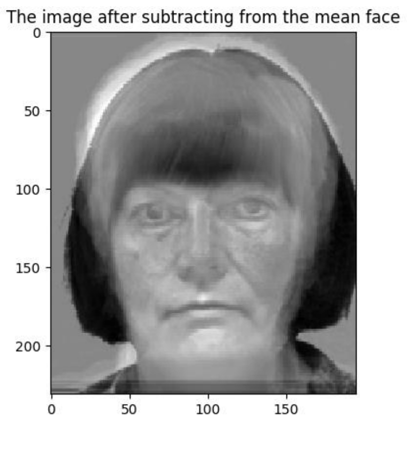
Classification: Identify of face which is similar to subject11.normal.jpg





12) For Image: subject11.normal

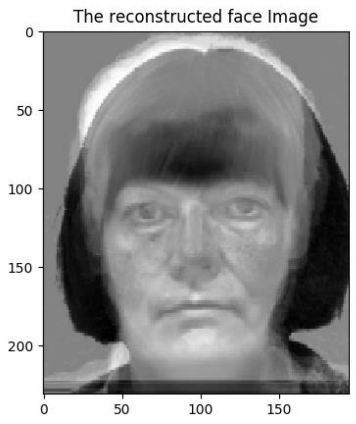
The image after subtracting the mean face (I)



PCA coefficients

```
[[  1.62112834e+07]
 [  1.03868223e+08]
 [ -1.54883456e+08]
 [ -4.69933026e+07]
 [ -6.21221660e+07]
 [  2.41960251e+07]
 [ -2.08372249e+08]
 [ -1.45914099e+08]]
```

The reconstructed face image (IR)

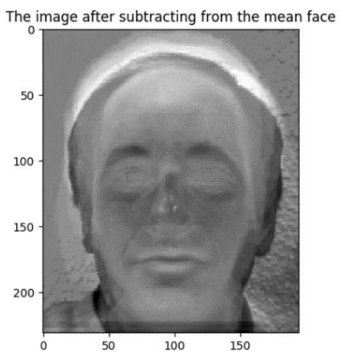


Distance D0 is 6.41360821122e+12
Distance D of Test Image to Train Image is 0.0

Classification: Identify of face which is similar to subject11.normal.jpg

13) For Image: subject12.normal

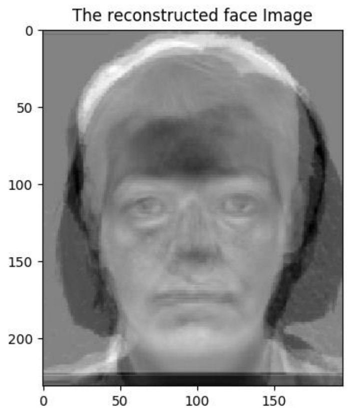
The image after subtracting the mean face (I)



PCA coefficients

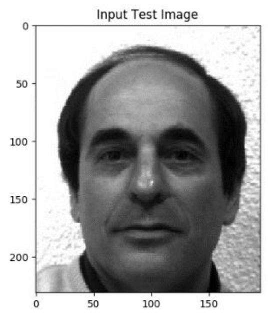
```
[[ -6357288.10402619]
 [ 20931027.62905409]
 [-17170228.37407878]
 [ 16170931.67471084]
 [-33129640.19396954]
 [ 15978828.15499067]
 [-26776279.85087397]
 [-76728642.89871714]]
```

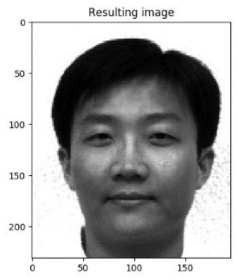
The reconstructed face image (IR)



Distance D0 is 1.7772096763e+12
Distance D of Test Image to Train Image is 86528495.0997

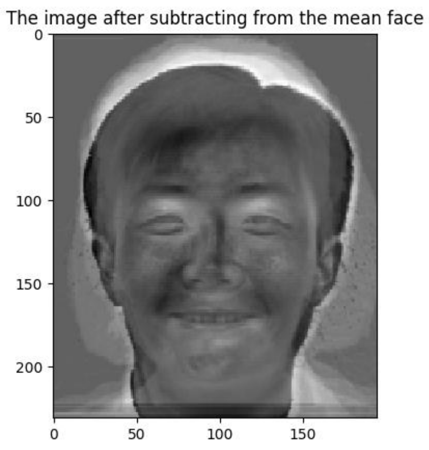
Classification: identify of face which is similar to subject14.normal.jpg





14) For Image: subject14.happy

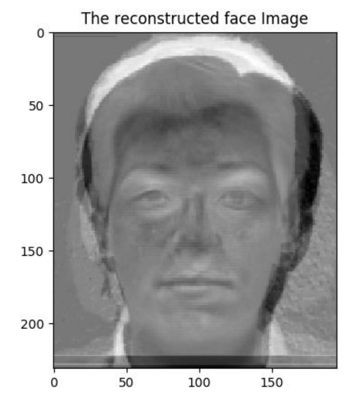
The image after subtracting the mean face (I)



PCA coefficients

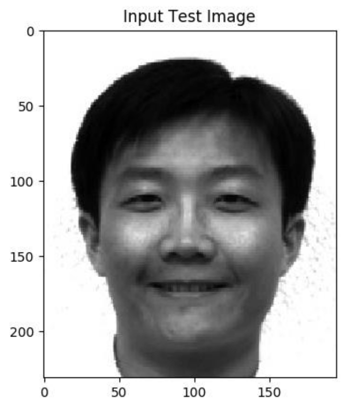
```
[[-28304868.25107065]
 [-29345677.08782681]
 [-30369106.39333002]
 [ 44211266.06091312]
 [-29388758.97947909]
 [  6702082.46481717]
 [ 22706706.22025731]
 [-52080530.81764869]]
```

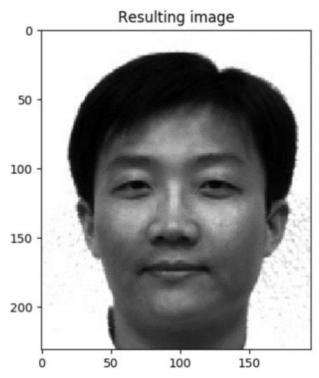
The reconstructed face image (IR)



Distance D0 is 1.32759773169e+12
Distance D of Test Image to Train Image is 31762428.2996

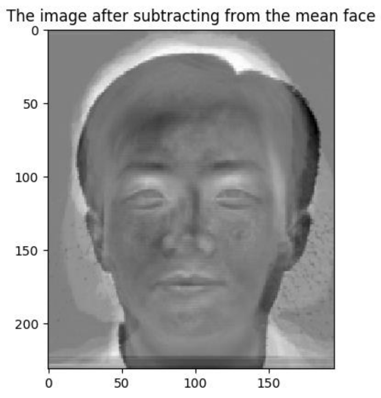
Classification: identify of face which is similar to subject14.normal.jpg





15) For Image: subject14.normal

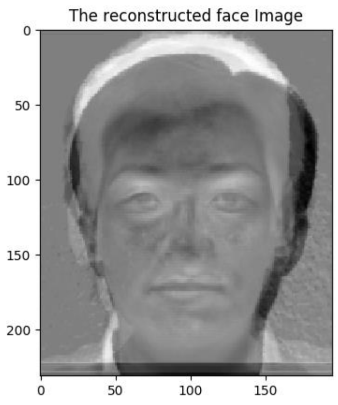
The image after subtracting the mean face (I)



PCA coefficients

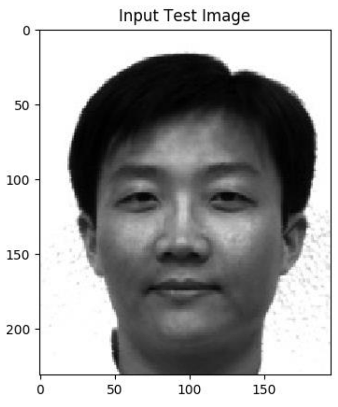
```
[[-23885338.95117827]
 [-38020356.99247383]
 [-36044721.80967677]
 [ 55601146.0481776 ]
 [-27169146.82748754]
 [ -3647980.96953668]
 [ 10181043.11415983]
 [-74061996.57593761]]
```

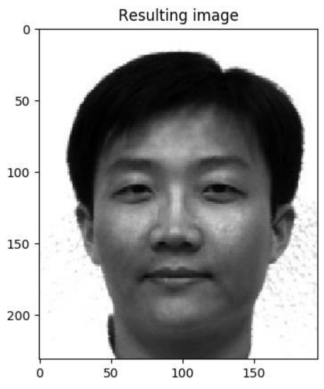
The reconstructed face image (IR)



Distance D0 is 1.64704333779e+12
Distance D of Test Image to Train Image is 0.0

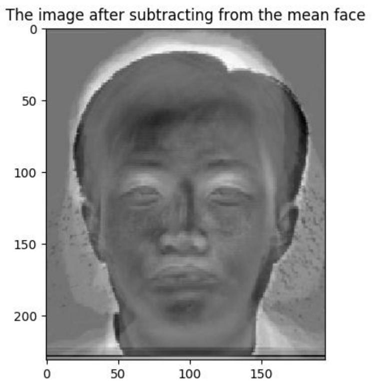
Classification: identify of face which is similar to subject14.normal.jpg





16) For Image: subject14.sad

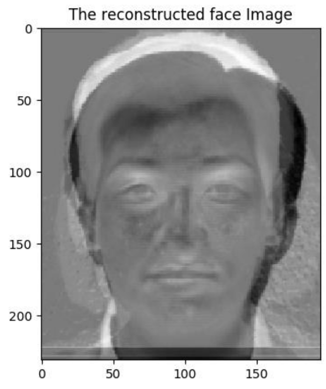
The image after subtracting the mean face (I)



PCA coefficients

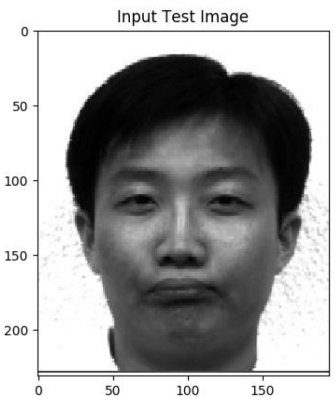
```
[[-13349753.5778023 ]
 [-34508563.909333  ]
 [-27213011.67619629]
 [ 44863129.61847532]
 [-20623594.37715862]
 [-11242382.63339401]
 [ 21272885.23194516]
 [-54758778.2982702 ]]
```

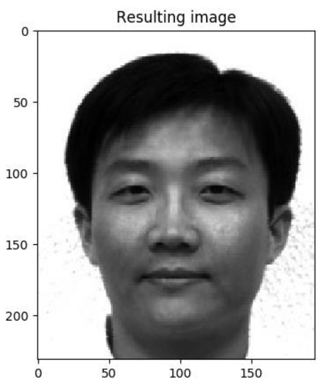
The reconstructed face image (IR)



Distance D0 is 1.21669154957e+12
Distance D of Test Image to Train Image is 30212542.7844

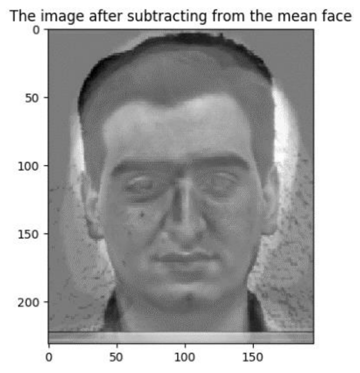
Classification: identify of face which is similar to subject14.normal.jpg





17) For Image: subject15.normal

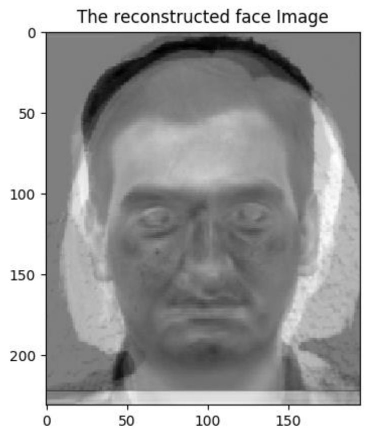
The image after subtracting the mean face (I)



PCA coefficients

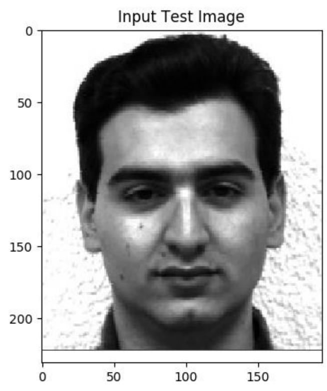
```
[[ -1.37470136e+07]
 [ -1.13351946e+07]
 [  8.24571597e+07]
 [ -5.74117559e+06]
 [  8.74542667e+07]
 [  1.14077271e+07]
 [  2.81793810e+07]
 [  1.23105910e+08]]
```

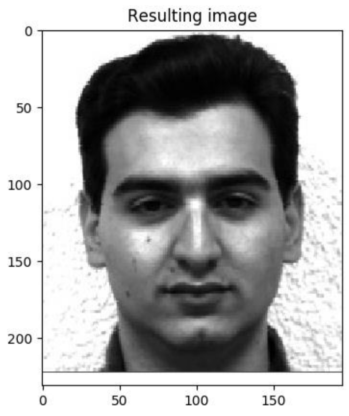
The reconstructed face image (IR)



Distance D0 is 3.24570215127e+12
Distance D of Test Image to Train Image is 0.0

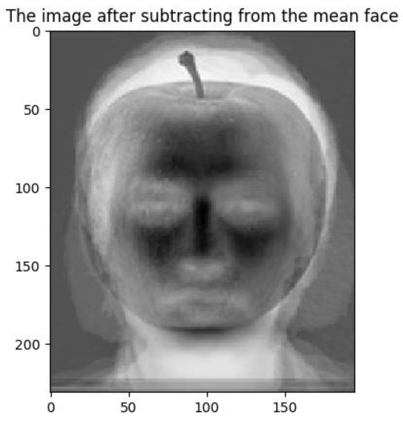
Classification: identify of face which is similar to subject15.normal.jpg





18) For Image: apple1_gray.jpg

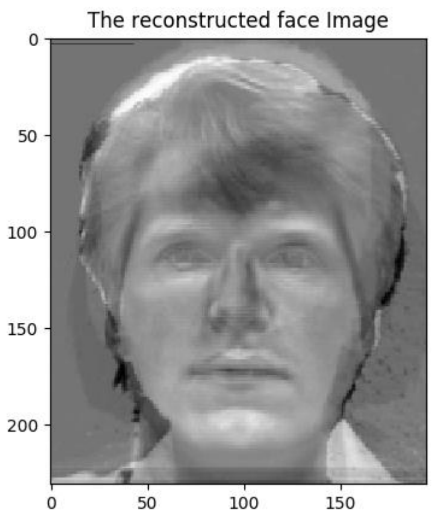
The image after subtracting the mean face (I)



PCA coefficients

```
[[-42627177.01786121]
 [ 29603174.38876884]
 [-26535253.77380313]
 [  4710795.49591369]
 [-12860160.66792136]
 [ 40755115.54007928]
 [ 19468225.85534585]
 [ -8729543.94871198]]
```

The reconstructed face image (IR)



Distance D0 is 917761191498.0
Distance D of Test Image to Train Image is 100111547.783

Classification: unknown face