Design and Evaluate a Face Recognition System

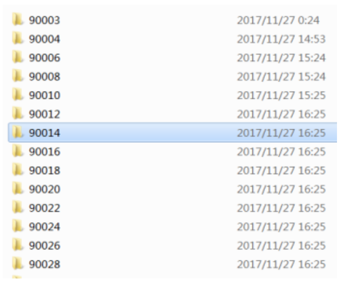
Group members: Ziyang Wu, Xingwei Li, Yuqin Wang

1. Description

In this assignment, we designed and developed a face recognition system by using openCV method. Gallery data and probe data are collected from the face dataset given by instructor. Raw image files are very large and have so many pixels, which is not good for us to process and it may make the runtime of our system very long. Therefore, the first step we did is to pre-process image files before using them in our face recognition system. We used Photoshop to set the size of images to 200*330 pixels.

We used python to develop our system and used MATLAB to plot results. We chose images of 36 people from the dataset to evaluate our system. In our system, we used OpenCV LBPHFaceRecognizer algorithm to train the template of each person, which is our gallery.

We picked ten to twenty front images for each person and put them into 36 folders respectively to train our gallery. Then, we took one front image for each person as probes and put them into each folder we made for each person (these images were our probes and these images were not used when we made our gallery). Using the built-in function predict(), we were able to get confidence values, which indicates the difference between two faces from gallery and probe. These confidence values were from 0 to 200. In fact, the confidence values we got actually means the distance between two faces. That means 0 would be totally identical. So we converted confidence values to the range between 0 and 1 to indicate differences. Finally, we used the data we got to draw the genuine and impostor distribution plots, ROC curves and CMC curves to evaluate our Iris recognition system.



```
90003    2017/11/27 0:24
90004    2017/11/27 14:53
90006    2017/11/27 15:24
90008    2017/11/27 15:24
90010    2017/11/27 15:25
90012    2017/11/27 16:25
90014    2017/11/27 16:25
90016    2017/11/27 16:25
90018    2017/11/27 16:25
90020    2017/11/27 16:25
90022    2017/11/27 16:25
90024    2017/11/27 16:25
90026    2017/11/27 16:25
90028    2017/11/27 16:25
```

Figure 1

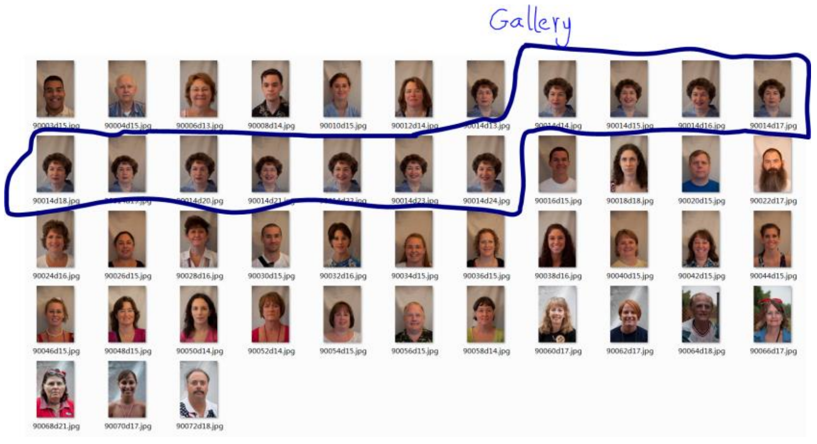
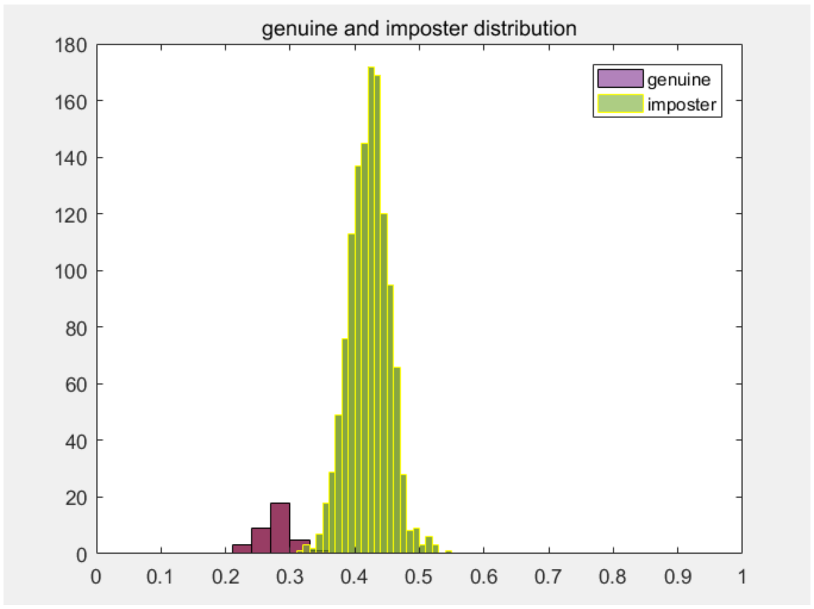


Figure 2

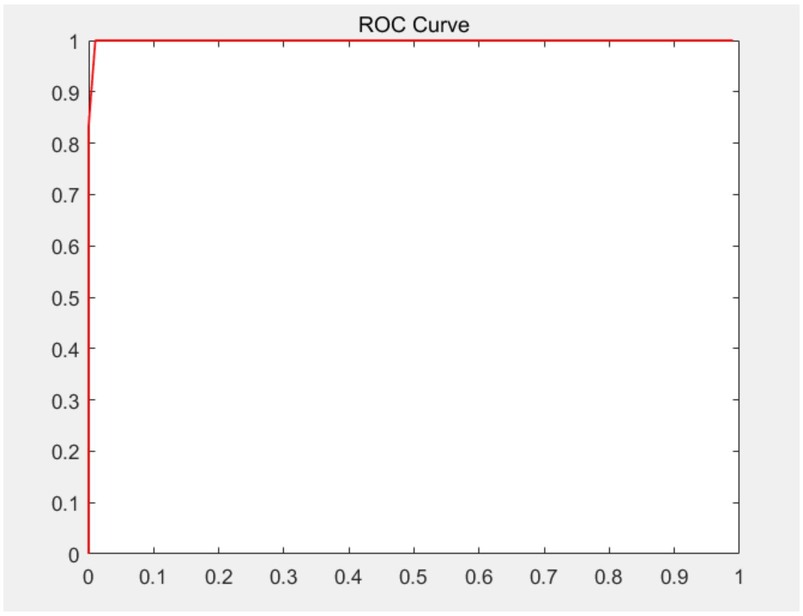
Figure 1 shows the folders used in our system. Figure 2 shows one folder which contains gallery for one person (number 90014) marking by the blue circle and the probes of the 36 persons (photos outside of the blue circle).

2. Results

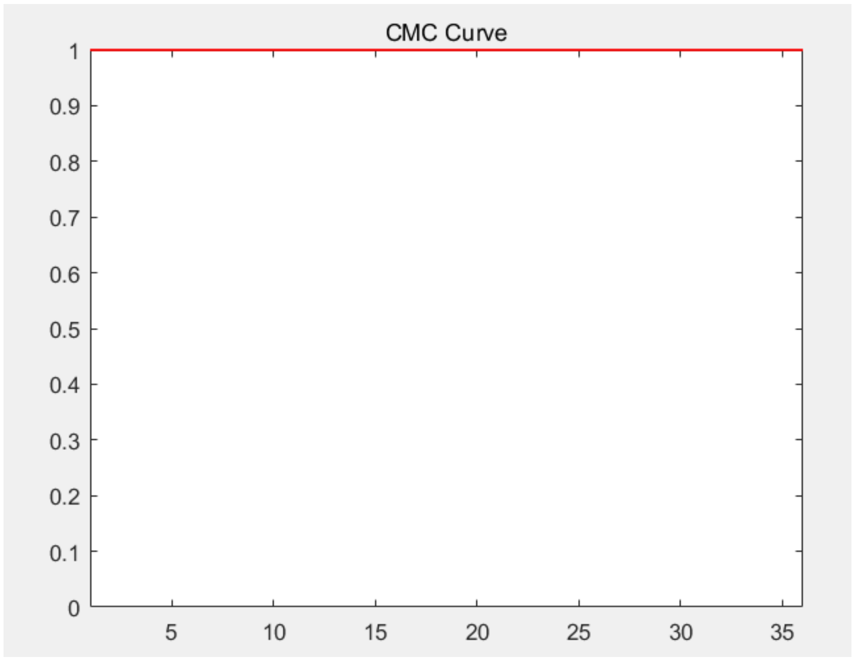
(1) Genuine and Imposter Distribution Plots



(2) ROC Curve



(3) CMC Curve



(4) Findings

According to the genuine and imposter distribution plot we shown above, it is obvious that the peak of genuine plot is separate with the peak of imposter plot. Also, we can see that the ROC curve and CMC curve of our face recognition system is almost ideal. That means our results are good and our face recognition system can recognize or verify people's identities by using their pictures. We can draw a conclusion that our face recognition system is accurate in most cases.

3. Source Code

(1) DrawPlots.m

```
 load('data.mat');

%-----------genuine and imposter curve---------------
 figure(1)
 h1 = histogram(conf_self/200);
 hold on;
 h2 = histogram(conf_oth/200);
 axis([0 1 0 180]);
 set(h2,'EdgeColor','yellow');
 title('genuine and imposter distribution');
 legend([h1,h2],'genuine','imposter');

 %----------------ROC curve-----------------------
TMR=zeros(10);
```

```
 FMR=zeros(10);
  for j=1:20
      threshold = 0+j/20;
      L1=length(find(conf_self/200<threshold));
      TMR(j) = L1/length(conf_self);
      L2=length(find(conf_oth/200<threshold));
      FMR(j) = L2/length(conf_oth);
  end
  figure(2)
  ROC = plot(FMR,TMR,'r');
  title('ROC Curve');
  set(ROC,'LineWidth',1);
 %------------------------CMC curve------------------
load('CMCrank.mat');
count = 0;
L = length(rank);
y2=zeros(1,L);

for i=1:L
    for k=1:L
       if rank(i)<=k
          count= count+1;
       end
    end
    y2(i) = count/L;
    count = 0;
end
 figure(3)
  CMC = plot(1:L,y2,'r');
  axis([1 36 0 1]);
  title('CMC Curve');
  set(CMC,'LineWidth',1);
```

(2) LBPHFaceRec.py

```
import cv2, os
import numpy as np
import scipy.io as scio
from PIL import Image

# For face detection we will use the Haar Cascade provided by OpenCV.
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)
```

```
# For face recognition we will the the LBPH Face Recognizer
recognizer = cv2.face.LBPHFaceRecognizer_create()

#gallery[i] contains photos of one person as enrolled data and photos of probes
gallery =
['90003d','90004d','90006d','90008d','90010d','90012d','90014d','90016d','90018d','90020d','90
022d','90024d','90026d','90028d','90030d','90032d','90034d','90036d','90038d','90040d','90042
d','90044d','90046d','90048d','90050d','90052d','90054d','90056d','90058d','90060d','90062d','9
0064d','90066d','90068d','90070d','90072d']
#probe list, each one is a photo of one person which is not included in the galleries.
probe =
['90003d15','90004d15','90006d13','90008d14','90010d15','90012d14','90014d15','90016d15','9
0018d18','90020d15','90022d17','90024d16','90026d15','90028d16','90030d15','90032d16','900
34d15','90036d15','90038d16','90040d15','90042d15','90044d15','90046d15','90048d15','90050
d14','90052d14','90054d15','90056d15','90058d14','90060d17','90062d17','90064d18','90066d1
7','90068d21','90070d17','90072d18']
#confidence of correct recognization and incorrect recognization
#confidence is ranging from 0 to 200, 0 means perfect matching
conf_oth = []
conf_self = []

list4rank = []
CMCrank = []

def get_images_and_labels(path,i):

   image_paths = [os.path.join(path, f) for f in os.listdir(path) if f.startswith(gallery[i]) and not
f.startswith(probe[i])]
   # images will contains face images
   images = []
   # labels will contains the label that is assigned to the image (unique for each person)
   labels = []
   for image_path in image_paths:
      # Read the image and convert to grayscale
      image_pil = Image.open(image_path).convert('L')
      # Convert the image format into numpy array
      image = np.array(image_pil, 'uint8')
      # Get the label of the image
      nbr = int(os.path.split(image_path)[1].split("d")[0])
      # Detect the face in the image
      faces = faceCascade.detectMultiScale(image)
      # If face is detected, append the face to images and the label to labels
      for (x, y, w, h) in faces:
         images.append(image[y: y + h, x: x + w])
```

```
            labels.append(nbr)
            cv2.imshow("Adding faces to traning set...", image[y: y + h, x: x + w])
            cv2.waitKey(50)
    # return the images list and labels list
    return images, labels

rootpath = 'E:/Courses/495Biometrics/HW3/LBPHFaceRec'
# Path to the pre-processed database
paths = [os.path.join(rootpath, f) for f in os.listdir(rootpath) if f.startswith('900')]

i = 0
#number of images in probe
num = 36

for path in paths:
    # Call the get_images_and_labels function and get the face images and the
    # corresponding labels

    images, labels = get_images_and_labels(path,i)
    i += 1
    cv2.destroyAllWindows()

    # Perform the tranining
    recognizer.train(images, np.array(labels))

    #index is reffering to photos of the probe, here we have 36 persons as our probe
    for index in range(0,num):
        print (index)
        # Append the probe images into image_paths
        image_paths = [os.path.join(path, f) for f in os.listdir(path) if f.startswith(probe[index])]
        for image_path in image_paths:
            predict_image_pil = Image.open(image_path).convert('L')
            predict_image = np.array(predict_image_pil, 'uint8')
            faces = faceCascade.detectMultiScale(predict_image)
            for (x, y, w, h) in faces:
                nbr_predicted, conf = recognizer.predict(predict_image[y: y + h, x: x + w])
                nbr_actual = int(os.path.split(image_path)[1].split("d")[0])
                if nbr_actual == nbr_predicted:
                    print ("{} is Correctly Recognized as {} with confidence {}".format(nbr_actual,
nbr_predicted, conf))
                    conf_self.append(conf)
                else:
                    print ("{} is Incorrect Recognized as {} and conf is {}".format(nbr_actual,
nbr_predicted, conf))
```

```
                    conf_oth.append(conf)
                cv2.imshow("Recognizing Face", predict_image[y: y + h, x: x + w])
                cv2.waitKey(100)
        #write confidence into .mat file
        dataNew = 'data.mat'
        scio.savemat(dataNew, {'conf_self':conf_self,'conf_oth':conf_oth})
#Calculate CMC
for a in range(0,len(conf_self)):
    list4rank.append(conf_self[a])
    genuine = conf_self[a]
    for b in range(a*(num-1),(a+1)*(num-1)):
        list4rank.append(conf_oth[b])
    list4rank.sort()
    #Find the position of the genuine match (rank)
    rank = list4rank.index(genuine,0,3)+1
    CMCrank.append(rank)
    del list4rank[:]
#write the ranks into CMCrank.mat file
scio.savemat('CMCrank.mat',{'rank':CMCrank})
```