EECS-495 PROJECT

Multi-dimensional twin facial recognition system

Haikun Liu	hlg483
Ke Wang	kwp862
Weishen Chu	wci004

Introduction

Twins' distinguishing is more challenging than other problems in face recognition. The motivation of twins' distinguishing is the plethora of twins – among 1000 new births, there will be 33.7 twins. There are different types of twins, Fraternal, "Identical" and "Mirror" Twins. The accuracy of distinguish depends on many factors as Jeffrey R. et al[1]. mentioned. It is easier to distinguish identical twins when images have been acquired on the same day instead of one year apart. Under Ideal conditions, like same day acquisition, studio-like illumination, consistent expression, it is more possible to distinguish identical twins. The Algorithm they used cares little about the age. Even though, the Twin discrimination by face recognition is still having space to improve, which means human still do a much more better job than face matching algorithms[2]. As mentioned by Soma Biswas et al., Human uses skin markings as a major factor to accomplish discriminating Fraternal twins[2]. Multiple methods have been developed to distinguishing twins' face. Vipin Vijayan et al[3]. achieved 3D face of twins. By analyzing iris and fingerprint, Karen Hollingsworth[4] and Anil K. Jain[5]successfully developed methods to distinguishing twins.

Methodology

In order to solve our problem, our group decided to take advantage of Artificial neural networks and Principal Component Analysis to accomplish Twins face recognition.

Artificial neural network

Artificial neural network model is inspired by biological neural network, as shown in figure1 below. Artificial neural network simulates the way human thinking and making decisions. It is generally presented as systems of interconnected "neurons" which exchange messages between each other[6]. The advantage of this network is its ability to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. In our case, the way human recognize twins' faces is quite unknown. That is the reason why Artificial neural network is suitable to address this problem.

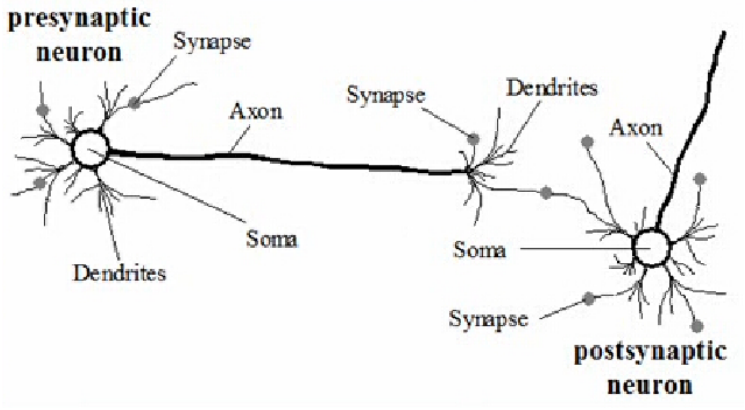

Figure 1: neural system in human brains[7]

The Figure 2 shows the math model of a single neuron. The decision a neuron made (as the output signals part shown in figure) depends on input signals, synaptic weights, summing function, and active function. Among these factors, the set of synaptic weights plays an important pole in making a precise decision. The quality of weight depends on training process. The training process obeys Widrow-Hoff rule. As we increase input signals, the accuracy of synaptic will accordingly increase which leads to a more accurate output signals.

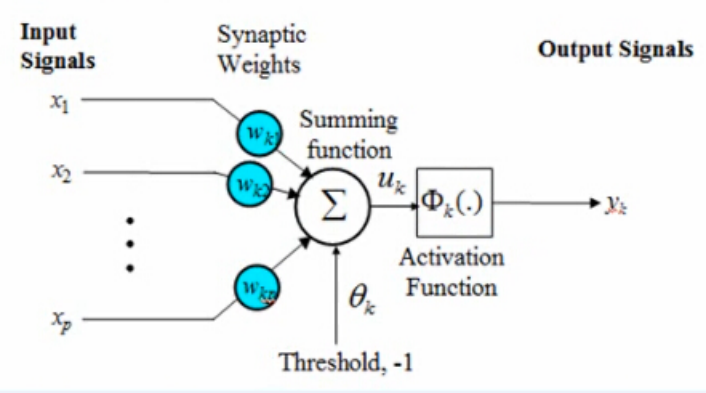

Figure 2: Mathematical model of neural network[7]

After the training process, the artificial neural network will be able to output signals based on the input test signals and trained weights- which is called testing process. From the results of the testing process, we will be able to recognize twins' faces.

Principal Component Analysis

The input parts of artificial neural network require a matrix form. In order to simplify the input process, we conduct Principal Component Analysis (PCA) before inputting image data. PCA can be accomplished by following steps:

- Getting raw data
- Subtracting the mean
- Calculating the covariance matrix
- Calculating the eigenvectors and eigenvalues of the covariance matrix
- Choosing component and forming a feature vector
- Deriving the new dataset

The details of Principal Component Analysis for input data will be shown in the following Process and Results part.

Multi-dimensional twin facial recognition system

In order to accomplish face twin facial recognition, we constructed a method combining Artificial neural networks and Principal Component Analysis, named Multi-dimensional twins face recognition system. The process is shown in Figure 3. Before input data into Artificial neural networks, we divided image of face into two part- front face and side face, and then trained the network separately. We picked five images of the same person to accomplish training. After training process, a test image was inputted into the network. Both of the front face and back face part would output three most similar face. Then a face appeared in both parts (for example face #2) would be selected as the result. Both input and output data were compressed by and Principal Component Analysis. The best solution of the network would appear when the testing image is quite similar with or nearly the same as the training images.

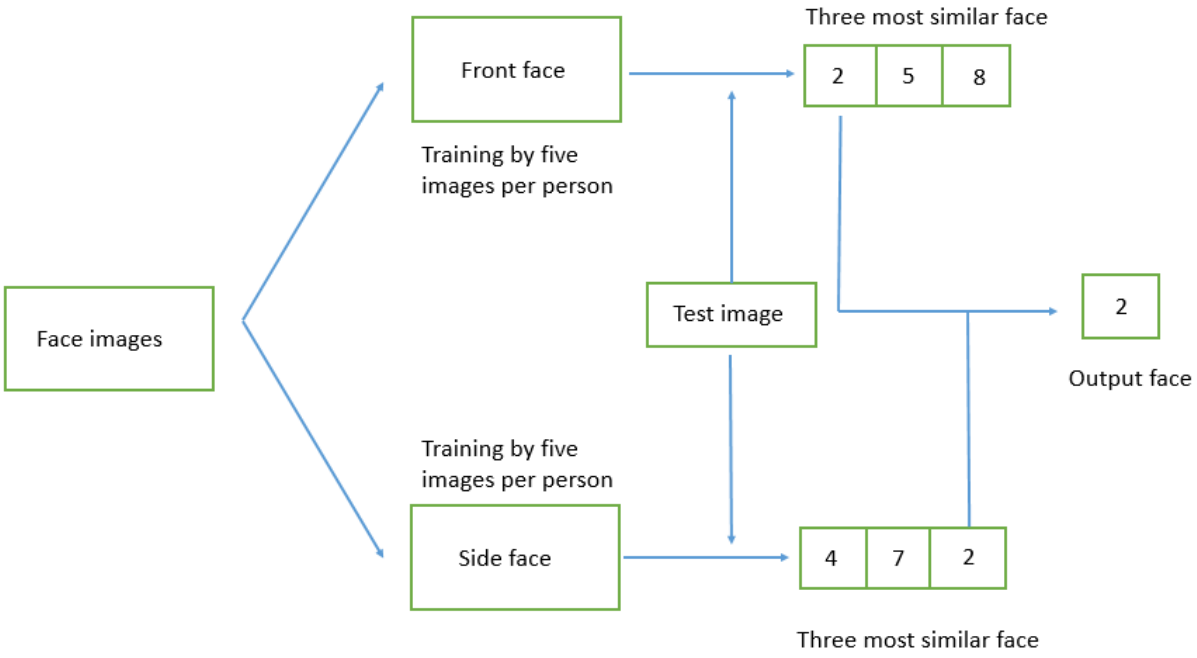

Figure 3: Process flow of multi-dimensional twins face recognition system

Results

In this part we demonstrate one test of our trained multi-dimensional twin facial recognition system. More details of the code can be found in the appendix part.

As shown in Figure 4., the first photo in the testing photo. After inputting this photo into our system, the system can find out three most similar photos, as shown in last three photos in Figure 4.

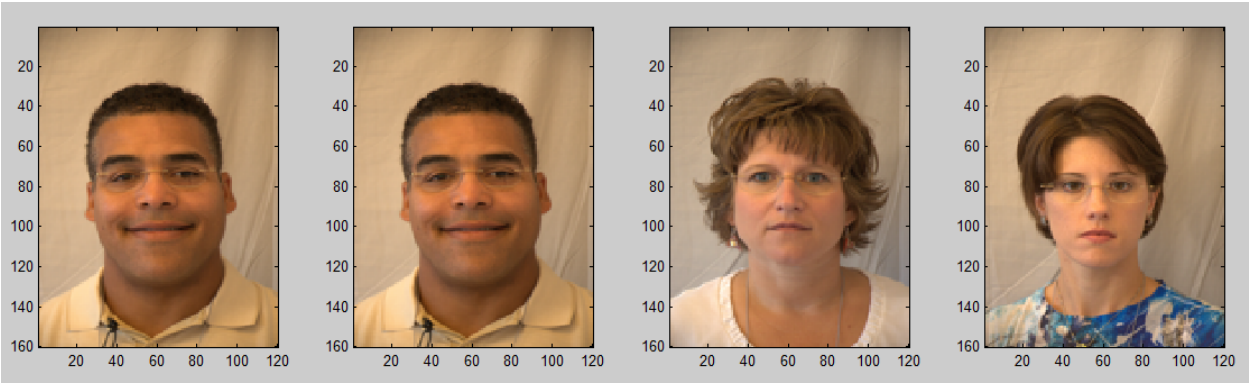

Figure 4: Front part test

We can then get the similar results from the side part of the system. As shown in Figure 5, the first photo is the testing photo, and the following three are three most similar photos found by our system.

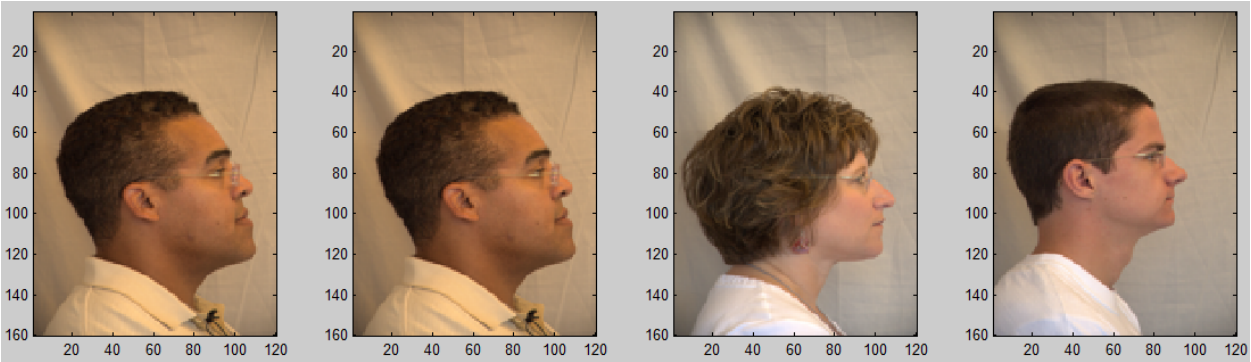

Figure 5: Front part test

After getting two sets of results from both front and side part, the system then found out the photo appeared in both sets. The right photo is the one picked up by the system and the left photo in Figure 6 is the testing photo. This result demonstrate that our system can successfully recognize twins' facial images.

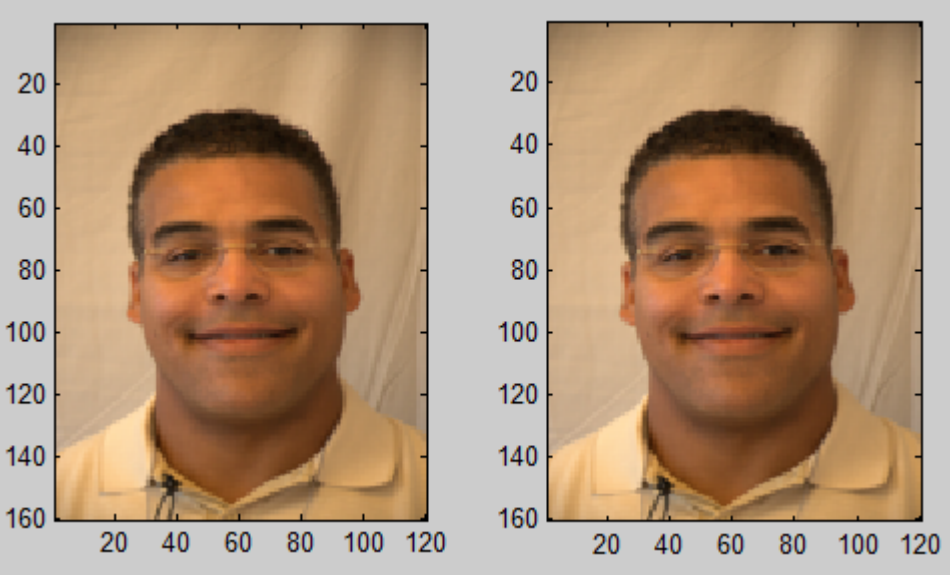

Figure 6. Final results of testing

Future works

As the limited time, we are not able to train a huge number of images. By increasing the database set in future, the network will output more accurate results. Another limit of our network is that it only containing front face and side face, even though it is called 'multi-dimensional'. A promising way is to take video of the face from side to front. By doing so, we will realize a true multi-dimensional, so that the accuracy of network will be promoted.

We will plot ROC of the network in future due the time limitation such that we do not not finish the whole project. In addition, Ada-boost may be used to improve our network. By doing so, we may be able to transfer a sort of weak recognizers to a strong and sound recognizer.

Conclusions

Twins' distinguishing is more challenging than other problems in face recognition. Multiple researches have been accomplished focusing on that. We make a combination of Artificial Neural Network and Principal Component Analysis to construct a new method named multi-dimensional Twin facial recognition system. After dividing data provided into two parts, front and side, Artificial neural networks output three most similar faces. The face appears in both parts is selected to be the final results.

Reference

1. Paone, Jeffrey R., et al. "Double trouble: Differentiating identical twins by face recognition." Information Forensics and Security, IEEE Transactions on 9.2 (2014): 285-295.

2. Biswas, Santosh, Kevin W. Bowyer, and Patrick J. Flynn. "A study of face recognition of identical twins by humans." Information Forensics and Security (WIFS), 2011 IEEE International Workshop on. IEEE, 2011.

3. Vijayan, Vipin, et al. "Twins 3D face recognition challenge." Biometrics (IJCB), 2011 International Joint Conference on. IEEE, 2011.

4. Hollingsworth, Karen, et al. "Genetically identical irises have texture similarity that is not detected by iris biometrics." Computer Vision and Image Understanding 115.11 (2011): 1493-1502.

5. Jain, Anil K., Salil Prabhakar, and Sharath Pankanti. "On the similarity of identical twin fingerprints." Pattern Recognition 35.11 (2002): 2653-2663.

6. https://en.wikipedia.org/wiki/Artificial_neural_network

7. GE, Lei, and Ai-qing HUO. "Application research of Widrow-Hoff neural network learning rule [J]." Electronic Design Engineering 6 (2009): 010.

Appendix: Matlab Code

```
%%%%%%run.m%%%%%%%
clear,clc;
runPCAtraining
runPCAtesting
%%%%%%runPCAtraining.m%%%%%%
clear;
clc;
display('*****************************************')
display('*****************************************')
display('*****************************************')
display('Train: Strat training front face')
display('Train: Please wait...')
[ MeanFace_front,eigenVector_front,coefficient_front ] = train_front();
display('Train: Front face training complete')
display('*****************************************')
display('*****************************************')
display('*****************************************')
display('---------------------------------------')
display('*****************************************')
display('*****************************************')
display('*****************************************')
display('Train: Strat training side face')
display('Train: Please wait...')
[ MeanFace_side,eigenVector_side,coefficient_side ] = train_side();
display('Train: Side face training complete')
display('*****************************************')
display('*****************************************')
display('*****************************************')
%%%%%%runPCAtesting.m%%%%%%
display('*****************************************')
```

```
display('****************************************')
display('****************************************')
display('Test: Load the address of the test front face...')
str_Load_test_front = 'C:\Users\Ke Wang\Desktop\CodeV2\Testingface\1.jpg';
display('Test: Start testing front face')
display('Test: Please wait...')
[ index_front] =
test_front( str_Load_test_front,MeanFace_front,eigenVector_front,coefficient_front);
display('Test: Front face matching complete')
display('----------------------------------------')
display('Plots: The first face from the left is the test front face followed by three top matched
front faces')
%plots
figure('name','Front face matching');
subplot(1,4,1);
subimage(loadImg(str_Load_test_front));
str_Path_front_matching = 'C:\Users\Ke Wang\Desktop\CodeV2\Trainingface/';
for i=1:3
   str_Load_recon_front = strcat(str_Path_front_matching, num2str(index_front(i)), '.jpg');
   subplot(1,4,i+1);
   subimage(loadImg(str_Load_recon_front));
end
%plots
display('****************************************')
display('****************************************')
display('****************************************')
display('Test: Load the address of the test side face...')
str_Load_test_side = 'C:\Users\Ke Wang\Desktop\CodeV2\Testingside\1.jpg';
display('Test: Start testing side face')
display('Test: Please wait...')
[ index_side] = test_side( str_Load_test_side,MeanFace_side,eigenVector_side,coefficient_side);
```

```
display('Test: Side face matching complete')
display('----------------------------------------')
display('Plots: The first face from the left is the test side face followed by three top matched side faces')
%plots
figure('name','Side face matching');
subplot(1,4,1);
subimage(loadImg(str_Load_test_side));
str_Path_side_matching = 'C:\Users\Ke Wang\Desktop\CodeV2\Trainingside\';
for i=1:3
    str_Load_recon_side = strcat(str_Path_side_matching, num2str(index_side(i)), '.jpg');
    subplot(1,4,i+1);
    subimage(loadImg(str_Load_recon_side));
end
%plots
display('****************************************')
display('****************************************')
display('****************************************')
display('----------------------------------------')
display('Test: Start front + side matching, find the most frequent index')
Idx_set = [index_front index_side];
[Idx_match freq] = mode(Idx_set);
if freq<=1
    errordlg('Can not find match!','Error')
else
    display('Plot: Left is test face, right is matched face')
    %plot
    figure('name','Final matching result');
    subplot(1,2,1);
    subimage(loadImg(str_Load_test_front));
    str_Load_recon_final = strcat(str_Path_front_matching, num2str(Idx_match), '.jpg');
```

```
    subplot(1,2,2);
    subimage(loadImg(str_Load_recon_final));
    %plot
end
display('*****************************************')
display('*****************************************')
display('*****************************************')
%%%%%%PCAtesting%%%%%%
function [idx] = PCAtesting( TestingImage, MeanFace,eigenVector,coefficient)
%Demean images
DemeanTestingFace = TestingImage - MeanFace;
coefTesting = DemeanTestingFace'*eigenVector;
[NumOfSamples,LengthOfCoefficient] = size(coefficient);
%compute the differences
for j=1:NumOfSamples
    CoefTestingDifference(j,:) =coefTesting-coefficient(j,:);
    EuclideanDistance(j,:)=sqrt(sum(CoefTestingDifference(j,:).*CoefTestingDifference(j,:)));
end
[EuclideanDistance, index]=sort(EuclideanDistance, 'ascend');
idx=[];
for i=1:3
    idx(i)=index(i);
end
end
%%%%%%train_front.m%%%%%%
function [ MeanFace_front,eigenVector_front,coefficient_front ] = train_front()
NumOfIms = 22;
NumOfEigenVectors = 20;
str_Path = 'C:\Users\Ke Wang\Desktop\CodeV2\Trainingface\';
for i = 1: NumOfIms
    str_Load = strcat(str_Path, num2str(i), '.jpg');
```

```
    Image = loadImg(str_Load);
    grayImage = rgb2gray(Image);
    TrainingImage(:,i) = double(reshape(grayImage, [ ], 1));
end
[ MeanFace_front,eigenVector_front,coefficient_front ] = PCAtraining( TrainingImage,20 );
eigenVector_front=eigenVector_front(:,1:NumOfEigenVectors);
end
%%%%%%test_front.m%%%%%%
function [ index_front ] =
test_front( str_Load_front,MeanFace_front,eigenVector_front,coefficient_front)
Image = loadImg(str_Load_front);
grayImage = rgb2gray(Image);
TestingImage = double(reshape(grayImage, [ ], 1));
[index_front] = PCAtesting( TestingImage,
MeanFace_front,eigenVector_front,coefficient_front);
end
%%%%%%test_side%%%%%%
function [ index_side ] =
test_side( str_Load_side,MeanFace_side,eigenVector_side,coefficient_side)
Image = loadImg(str_Load_side);
grayImage = rgb2gray(Image);
TestingImage = double(reshape(grayImage, [ ], 1));
[index_side] = PCAtesting( TestingImage, MeanFace_side,eigenVector_side,coefficient_side);
end
%%%%%%train_side%%%%%%
function [ MeanFace_side,eigenVector_side,coefficient_side ] = train_side()
NumOfIms = 22;
NumOfEigenVectors = 20;
str_Path = 'C:\Users\Ke Wang\Desktop\CodeV2\Trainingside/';
for i = 1: NumOfIms
    str_Load = strcat(str_Path, num2str(i), '.jpg');
```

```
    Image = loadImg(str_Load);
    grayImage = rgb2gray(Image);
    TrainingImage(:,i) = double(reshape(grayImage, [ ], 1));
end
[ MeanFace_side,eigenVector_side,coefficient_side ] = PCAtraining( TrainingImage,20 );
eigenVector_side=eigenVector_side(:,1:NumOfEigenVectors);
end
%%%%%%PCAtraining.m%%%%%%
function [ MeanFace,eigenVector,coefficient ] =
PCAtraining( TrainingImage,NumOfEigenVectors )
[NumOfPixelPerSample,NumOfSamples] = size(TrainingImage);
MeanFace=0;
DemeanFace=zeros(NumOfPixelPerSample,NumOfSamples);
%meanface
for i = 1: NumOfSamples
    MeanFace = MeanFace+TrainingImage(:,i);
end
MeanFace = MeanFace/NumOfSamples;
%demeanface
for i = 1: NumOfSamples
    DemeanFace(:,i) = TrainingImage(:,i) - MeanFace;
end
%Eigenvectors&Eigenvalues
[vectors,D]=eig(DemeanFace'*DemeanFace);
eigenValue2=eig(DemeanFace'*DemeanFace);
for i=1:NumOfSamples
    eigenVector2(:,i)=(DemeanFace*vectors(:,i))/sqrt(eigenValue2(i));
end
[eigenValue, index]=sort(eigenValue2, 'descend');
eigenVector=eigenVector2(:,index);
for i=1:NumOfSamples
```

```
    for j = 1:NumOfEigenVectors
        coefficient(i,j) = DemeanFace(:,i)'*eigenVector(:,j);
    end
end
end
%%%%%%loadImg.m%%%%%%
function [ Img ] = loadImg( str_Load )
Image = imread(str_Load);
Img = imresize(Image, [160 120]);
end
```