ECEN 662 Final Project

Qiang Zhang UIN:927000888

1. Problem description:

An important application of face recognition is to assist law enforcement. Automatic retrieval of photos of suspects from the police mug shot database can help the police narrow down potential suspects quickly. However, in most cases, the photo image of a suspect is not available. The best substitute is often a sketch drawing based on the recollection of an eyewitness.[1]

In my final project, I'll set up an 88 photos-sketches library.[2] The purpose is to select a sketch randomly and then recognize its respective photo.

Fig. 1 Example of photos and sketches

2. My research

(1) Preprocessing

First, all the photos and sketches are loaded into workspace. Then tailor these variables by reshaping them. Also, we need to convert RGB into gray for these colorful photos. In this project, the pixel after process is 483*414.

For convenience, I read all the photos and sketches and then save these variables as '*input.mat*' after preprocess. In the code, I'll not show the details of preprocess which are quite easy.

(2) Algorithm

a) Scale-invariant feature transform (SIFT)

The scale-invariant feature transform is an algorithm in computer vision to detect and describe local features in images.(from *Wiki*) It can extract the extrema as the features of an image.

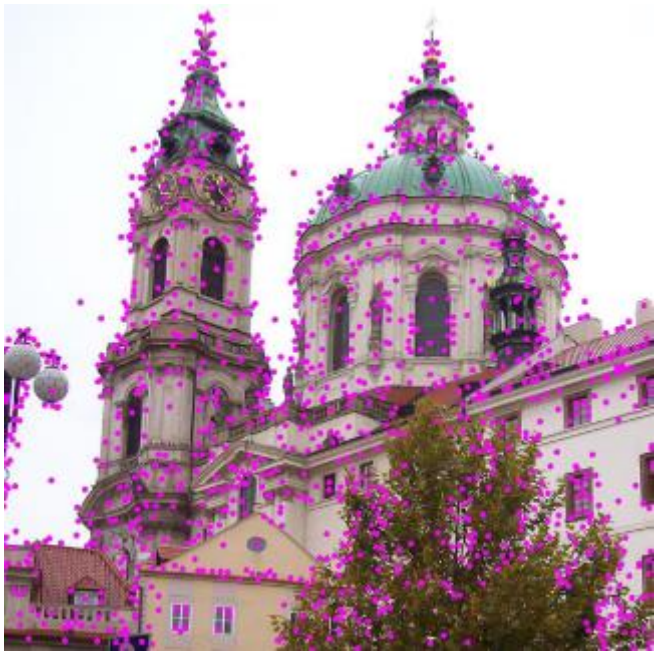

Fig. 2 Example of extrema points in SIFT (from *Wiki*)

But for this project, if we try to compare the extrema points of an photo and a sketch, the fact is a sketch always much more points than an photo. So it's almost impossible to make an estimation by SIFT. While, the first step of SIFT, difference of Gaussian(doG), is still effective.

We can see the photo and sketch which belong to one person seem similar after the process of doG.

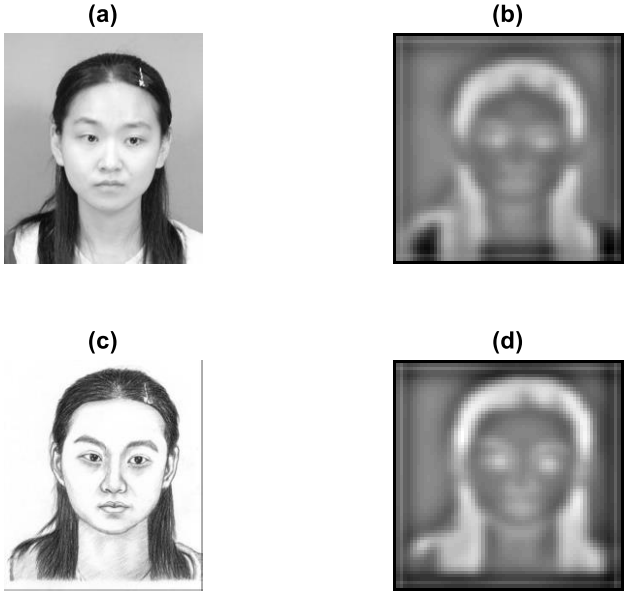

Fig. 3 (a) photo (b) photo after doG (c) sketch (d) sketch after doG

After doG, each image has a feature vector. Then we use these feature vectors to compare and estimate the correct photo by a sketch.

b) PCA

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

Briefly, I want to use PCA to simplify the feature vectors. In my simulation, the length of input vectors is 4356 and output is 100. This is enough because the 100^{th} eigenvalue is 2.8583e-13.

Fig. 4 The first 9^{th} eigenfaces

Then we can project 'the long features' to 'the short features', which are used to make the decision of our estimation.

3. Simulation

In the '*Code*' folder, you need only open '*main.m*' in matlab and run it.

'*input.mat*': data after preprocessing

'*displayData.m*': show the image after doG

'*feature_doG.m*': function of doG

'*featureNormalize.m*': normalize features before PCA

'*pca.m*': function of PCA

'*projectData.m*': project original features to principal features

One-time result:

Fig. 5 A random input sketch

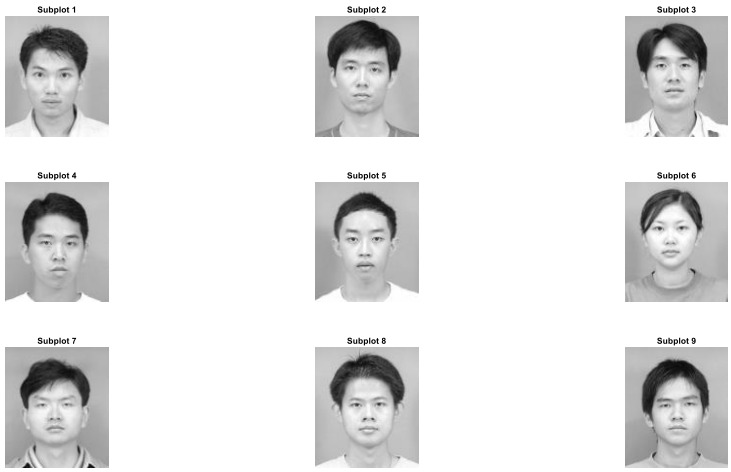

Fig. 6 Photos of the most 9th possible persons

From Fig. 5&6, we can easily find that No.4 is the correct one. This algorithm is the best one among these algorithms I've tried. I can't assure that No.1 is the correct one, while the correct one must be in the top 9. You can run this program for several times, and the top 9 always work.

4. Conclusion

Obviously, this algorithm use in this project is immature. Sometimes, it works well and the No.1 is just the correct one. But, occasionally its effect is quite poor. For one reason, I think it should be the criteria that makes the decision. I just compare the Euclidian distance between input feature and features in library. Sometimes, this method may not make sense because the shift of pixel will affect the result even if one pixel-shift.

So in my opinion, the future step of this research is to find the appropriate comparing method for features. Intuitively, this should be a feasible way and may work well.

Actually, I've tried several algorithms for this project, like Haar Classifiers, LBP and directly using PCA. But none of these works well. The combo of PCA and doG seems the best I've ever try.[3]

Reference:

1. X. Wang and X. Tang, "Face Photo-Sketch Synthesis and Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 31, 2009.
2. Dataset used in this project comes from *http://mmlab.ie.cuhk.edu.hk/archive/facesketch.html*.
3. *http://www.ehu.eus/ccwintco/uploads/e/eb/PFC-IonMarques.pdf*