I. INTRODUCTION

Face recognition and facial expression recognition have been paid much attention in the past two to three decades, which play an important role in such areas as access control, human–computer interaction, production control, e-learning, fatigue driving detection, and emotional robot. There are ordinarily seven kinds of facial expression to be classified: anger, disgust, fear, happiness, sadness, surprise and neutral.

However, as there are too many features in face or facial expression, most of these methods make trade-offs like hardware requirements, time to update image database, time for feature extraction, response time, etc. So generally researchers would like to choose a special method that will reduce the amount of calculation and thus make the experiment much more efficient. Principal Component Analysis (PCA) is a main and long term studied method to extract feature sets by making sample projection from higher dimension to lower dimension. Yet, for most occasions computers will run out of memory. Even with enough memory it will take much time to extract all eigenvalues. Therefore, a better method called Fast Principal Component Analysis (FPCA), which will calculate in a much faster way, yet brings the same results as PCA does.

Besides PCA, many researches also proposed new algorithms to improve recognition rate. Nevertheless, not like other researches that focus on improving algorithm in order to bring a better recognition rate, we are more interested in pre-processing of images, since it may also bring some positive effects on the final performance but in a more simple way. Although many researches also include pre-processing in their study, like adjusting face position, converting into grey-level images, or make some image corrections, most of them don’t consider a way that enhancing features of a person. Noticing this ignored aspect, we try to utilize a suitable method in pre-processing. And our method is Fast Fourier transform (FFT). By using FFT, we process original images and combine amplitude spectrum of one image with phase spectrum of another image to enhance features before extracting eigenvectors.

In this paper, a novel approach is proposed in order to improve the recognition rate of facial expression. We firstly discuss some basic algorithms to explain our method, Fast Fourier Transform. And then we evaluate our approach based on self-made image database, which contains 252 images of 7 facial expressions of 3 individuals. Since our experiment is based on color images, we first convert RGB color space into YCbCr color space. After lighting compensation, we extract skin and find skin color block to detect identified face. Finally we present our results and compare to PCA only method.

II. Pre-processing

A. Fourier Transform

To discuss our method before, we firstly make a description about Fourier Transform. Virtually everything in the world can be described via a waveform. The Fourier Transform gives us a unique and powerful way of viewing these waveforms.

If we assume the period is T, Fourier series should be like:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{+\infty}(a_k \cos n\,\omega_0 x + b_k \sin n\,\omega_0 x) \qquad (1)$$

where $\omega_0 = \frac{2\pi}{T} = 2\pi u$, and $u = 1/T$, which is the frequency of $f(x)$.

It has already been proven that the first N terms of Fourier series is the best approximation of anti derivative f(t) in given energy:

$$\lim_{N\to\infty} \int_0^T |f(t) - [\frac{a_0}{2} + \sum_{k=1}^{+\infty}(a_k \cos n\,\omega_0 x + b_k \sin n\,\omega_0 x)]|^2\, dx = 0 \qquad (2)$$

In this paper, because we are dealing with images, so it is obvious for us to know two-dimension Discrete Fourier Transform (DFT):

$$F(u,v) = \sum_{X=0}^{M-1} \sum_{X=0}^{N-1} f(x,y) e^{-i2\pi(\frac{ux}{M}+\frac{vy}{N})} \qquad (3)$$

where u and v are variables in domain. The reason why we need Fast Fourier Transform (FFT) is that for a sequence of N, its definition of DFT transform and inverse transformation is:

$$\begin{cases} F(u) = \sum_{X=0}^{N-1} f(x) W_N^{ux}, u = 0,1,\dots,N-1, W_N = e^{-j\frac{2\pi}{N}} \\ f(x) = \frac{1}{N}\sum_{u=0}^{N-1} F(u) W_N^{-ux}, x = 0,1,\dots,N-1 \end{cases} \qquad (4)$$

As we can see in Eq.4, it is not hard to find that in order to calculate a sequence of N, it should do N^2 plural multiplications and N (N-1) plural additions. Yet, the fundament of DFT is not so complex, W_N has its own periodicity, actually it only has N values. And also, these N values have symmetric relation in some degree. Because of periodicity and symmetry of W_N, we could make a conclusion as follows:

$$W_N^0 = 1, W_N^{\frac{N}{2}} = 1 \tag{5}$$

$$W_N^{N+r} = W_N^r, W_N^{\frac{N}{2}+r} = -W_N^r \tag{6}$$

where Eq.5 represents some special values in matrix W, while Eq.6 explains periodicity and symmetry of matrix W.

By utilizing periodicity of matrix W, some terms in DFT calculation can be combined; by utilizing its symmetry, we just need calculate half of W. Based on these two advantages, we could reduce a lot of operations, which is the basic idea of FFT.

B. Amplitude Spectrum and Phase Spectrum

In this paper, we apply FFT to combine amplitude spectrum of one image with phase spectrum of another image to enhance features of images.

The definition of amplitude spectrum:

$$|F(u,v)| = [Re(u,v)^2 + Im(u,v)^2]^{1/2} \tag{7}$$

Every amplitude spectrum |F(u, v)| of (u, v) in frequency domain could represent the ratio of sine (or cosine) plane wave to superimposition. Thus amplitude spectrum could reflect the frequency information and have a high practical value in filtering.

The definition of phase spectrum:

$$\varphi(u,v) = \arg\tan\frac{Im(u,v)}{Re(u,v)} \tag{8}$$

Though it is not so clear to figure out the information about phase spectrum as amplitude spectrum, it contains a kind of ratio relation between real part and imaginary part.

And in addition, we could restore F(u, v) by amplitude spectrum and phase spectrum:

$$F(u,v) = |F(u,v)|e^{j\varphi(u,v)} \tag{9}$$

As we mentioned before, our focus is primary on pre-processing and we try to figure out a simple, an efficient or an elegant way to do this part well. And in this paper our answer is Fast Fourier Transform. By combining amplitude spectrum of one image with phase spectrum of another image (both images belong to the same class), we get a mixed image that had main information of one image (the one with amplitude spectrum) and minor information of another image (the one with phase spectrum). Because our goal is to improve the recognition rate of facial expression recognition, extracting eigenvectors from a mixed image would be better than extracting eigenvectors from an original image (as there were more features in the that image), therefore enhances the accuracy of results.

C. Extracting Feature

As we discussed before, our facial expression recognition system is based on PCA. PCA is one of the oldest in analyzing multiple variables, which is derived from Karhumen-Loeve (KL) transformation. It is Pearson who first proposed PCA in 1901, and in 1963 Karhumen-Loeve made a huge improvement about the original version of PCA.

If the image elements are considered to be random variables, then the image may be seen as a sample of a stochastic process. The PCA basis vectors are defined as the eigenvectors of covariance matrix S:

$$S = \mathrm{E}[XX^T] \tag{10}$$

□Since the eigenvectors associated with the largest eigenvalues have face-like images, they also are referred to as eigenfaces. If we suppose the eigenvectors of S are V_1, V_2, ..., $V_{N□}$ and are associated respectively with the eigenvalues $m_1 \geq m_2 \geq \ldots \geq m_n$. Then:

$$X = \sum_{i=1}^{n} \hat{x}_i v_i \tag{11}$$

However, if the number of dimension is very big, calculating matrix S will be a tough task that may even make computer gets stuck. Fortunately, there is a good idea to fix it.

Assume that n is the number of samples, d is the number of dimension, and $Z_{□□□n \times d}$ is the matrix that every sample in sample matrix X minus mean Eigenvalue□

m, thus scatter matrix S should be $(ZZ^{T})_{d \times d}$. Now consider the matrix R = $(ZZ^{T})_{n \times n}$, for most occasion d is much greater than n, therefore the size of matrix R is much smaller than that of matrix S. So we have:

$$(ZZ^{T})\vec{v} = \lambda\vec{v} \tag{12}$$

And from Eq.12 we can get Eq.13 by premultiping Z^{T}:

$$(Z^{T}Z)(Z^{T}\vec{v}) = \lambda(Z^{T}\vec{v}) \tag{13}$$

III. CLASSIFY METHOD

Since we have extracted face features, the next step is to put the images into classes. Instead of choosing a classifier, we choose to calculate Euclidian distance of test images from all the training images and figure out the minimum value in order to find out the train image which is most similar to the test images. And in this paper, we assume neutral expression is the mean value. So according to this assumption, the more the distance from the neutral expression, the stronger the expression is. And Fig 1 shows the principle of this method.

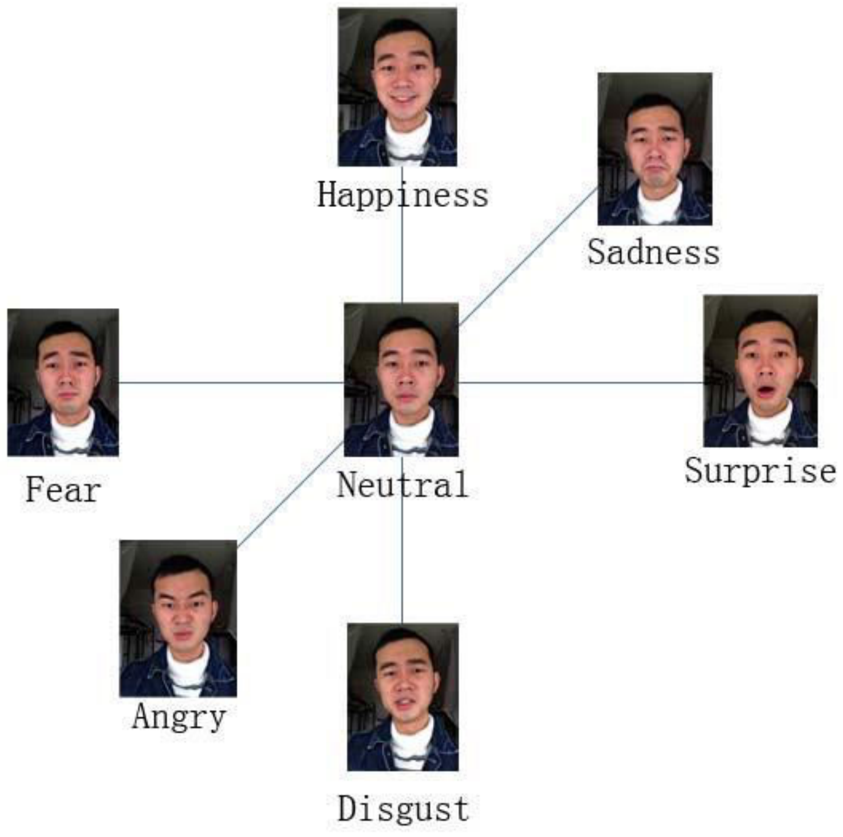


Fig. 1. Euclidian distance of test images

To harness PCA in our face recognition system, firstly our program read a sample of training database (m pixel * n pixel) and saved it in a sample matrix Face Container. Then by applying PCA, we converted m*n dimensions matrix into k dimensions matrix, and those k dimensions would represent all the dimensions in a training matrix, which would reduce a lot of operations in the following part.

IV. TEST RESULTS

In this section, all the works were done in Matlab2011b in a PC with 2.5GHz Intel Core i5 and 4GB 1600MHz DDR3 RAM.

Our database in this paper is self-made image database, which contains 252 images of 7 facial expression classes (anger, disgust, fear, happiness, sadness, surprise and neutral) of 3 individuals (Da Ding, Ruxin Du, Hao Zhang). Each individual includes 84 images, and facial expression class has 12 images. In the experiment we pick half of the database (126 images) randomly as train images, and the rest half is used as test images. Fig.2. shows our image database.

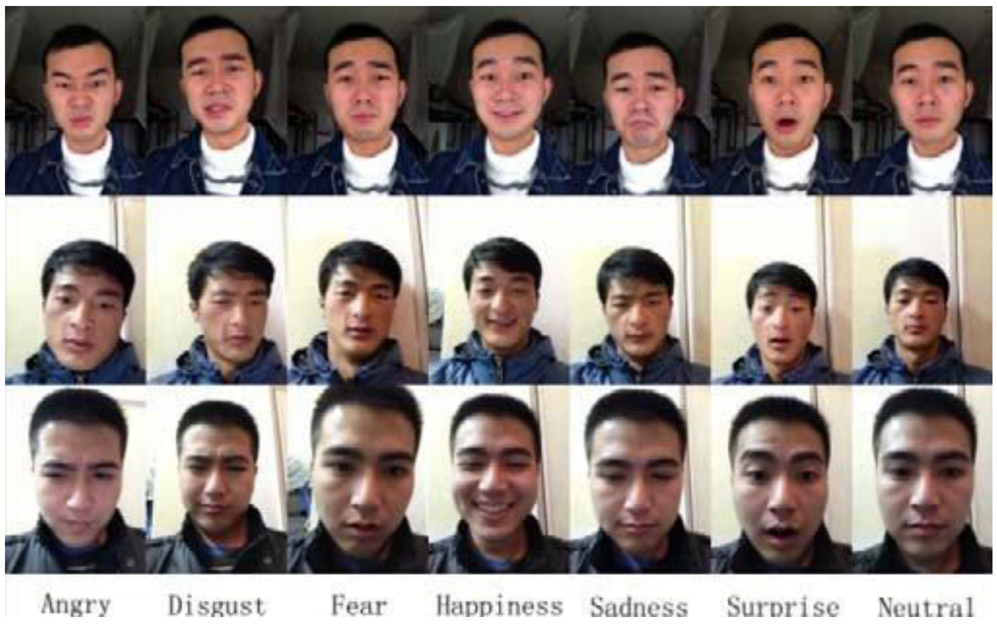
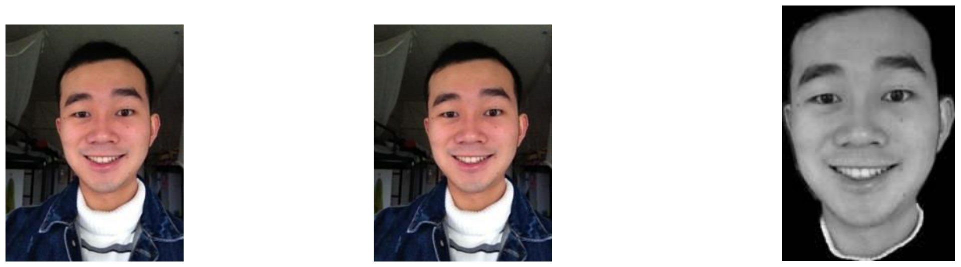


Fig. 2. Our database including 7 facial expression of 3 individuals

Firstly, our purpose is to detect and extract individual's face from the background. After lighting compensation, extracting skin, and find skin color blocks, we find the identified face which has been converted to gray images. One example is displayed below.



(a) Original image *(b) After lighting compensation* *(c) Identified face*

Fig. 3. Example of detecting face

Since we have detect faces (which assures that there is actually one face in the image), we used FFT to combine amplitude spectrum of one image with phase spectrum of the next image (the last one will combine with the first one). All the mixed images we got are shown in Fig.4 compared with original images.

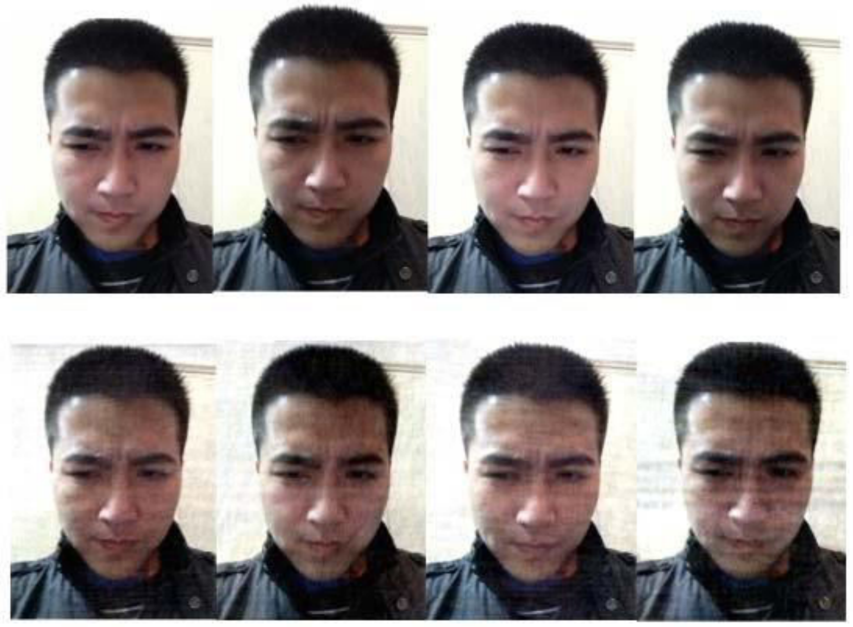


Fig. 4. Original version (above) and images after FFT processing (below)

After applying FFT, we then use PCA to extract facial expression feature and calculate Euclidian distance of images in order to put the train images into right class. Our final results are displayed below as Table I.

TABLE I

RESULTS OF RECOGNITION RATE USING FFT+PCA

Individual / Facial expression	Da Ding	Ruxin Du	Hao Zhang
Angry	100%	100%	100%
Disgust	100%	83.3%	100%
Fear	100%	100%	71.4%
Happiness	83.3%	100%	83.3%
Sadness	100%	100%	83.3%
Surprise	100%	100%	100%
Neutral	100%	100%	100%
Average	97.6%	97.6%	90.4%

And then we compare our method (FFT+PCA) to normal method (PCA only), and set k (dimensions of Eigenvector) to10, 20, 30, 40 respectively. The results are shown in Table II.

TABLE II

COMPARISON TO PCA ONLY METHOD

Value of k	FFT+PCA	PCA only	Improvement
10	96.0%	90.5%	6.14%
20	94.4%	89.7%	5.31%
30	95.2%	92.1%	3.45%
40	97.6%	93.7%	4.24%

From Table 1 and Table 2, we can clear see that our method (FFT+PCA) is more reasonable than PCA only, and the results indicate that our approach is satisfactory.

V . CONCLUSION

Based on the analysis, we provide a full system of face expression recognition (FFT+PCA). Noticing that few researches focus on pre-processing of images, which will also enhance the result of classification, we use Fourier Transform as a way to do pre-processing. Through FFT, we combine amplitude spectrum of one image with phase spectrum of another image as a mixed image. Then we harness PCA to extract Eigenvectors and calculate Euclidian distance to put training images into qualified class.