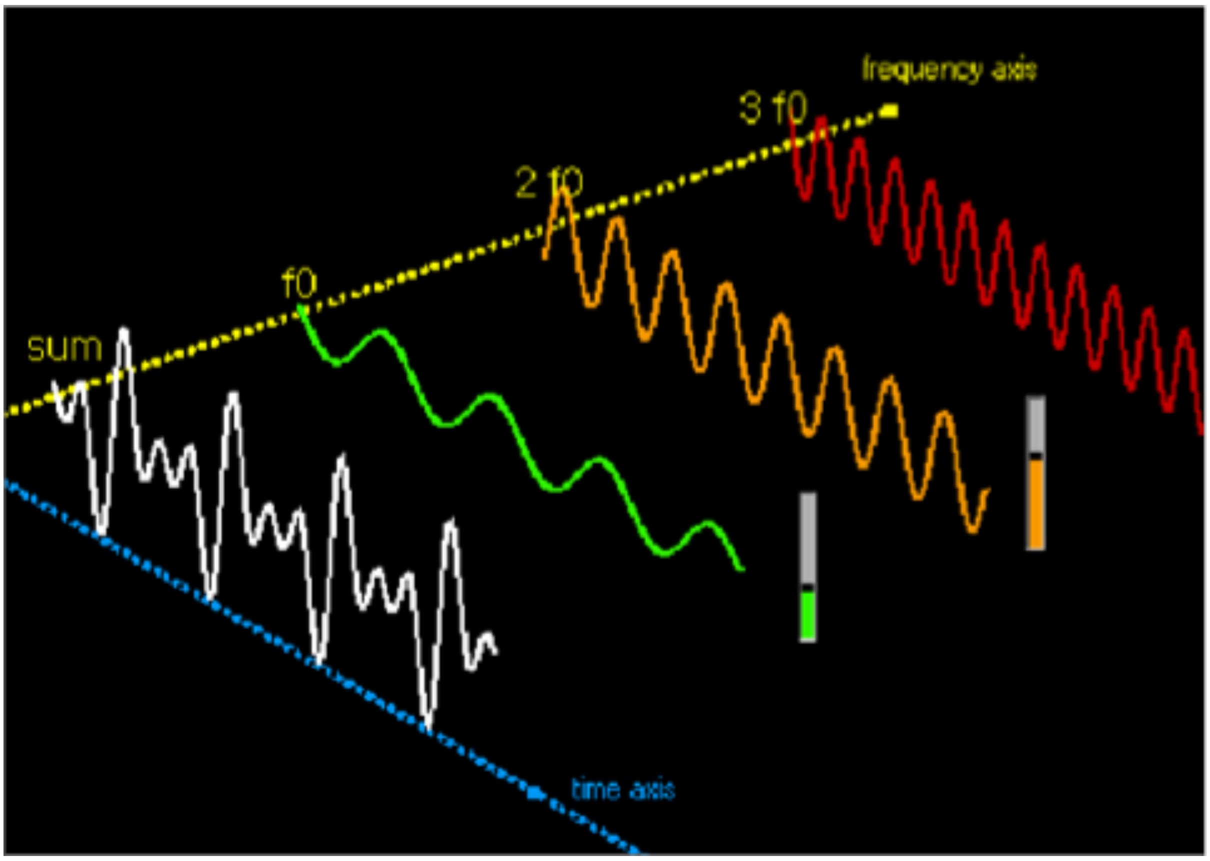
Modern Facial Recognition Systems

Using Fast Fourier Transforms



Mathematics Project

Aayush Nandkeolyar

Ankit Agrawal

Amogh G

Kumar Rohit

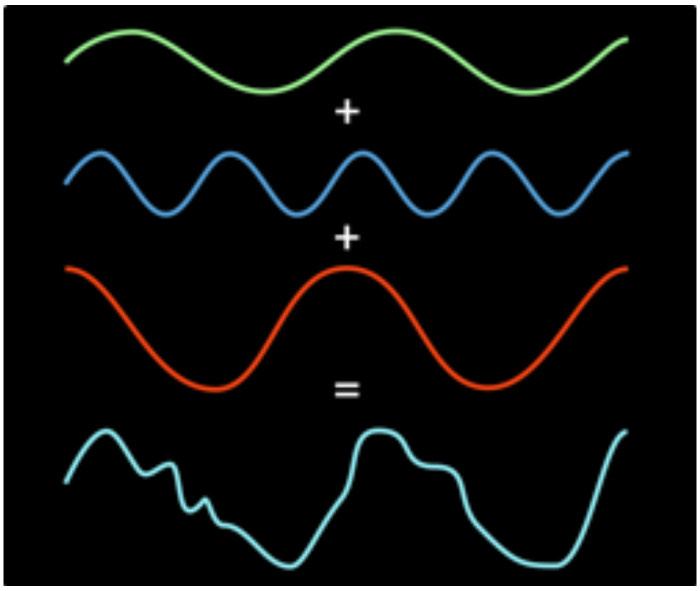
Introduction

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 a 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. 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.

Pre-processing methods include adjusting face position, converting into grey- level images, or make some image corrections, but most of these methods don't consider a way that enhancing features of a person. But modern methods try to utilise a suitable method in pre-processing. And the method is Fast Fourier transform (FFT). By using Fast Fourier Transform, we process original images and combine amplitude spectrum of one image with phase spectrum of another image to enhance features before extracting eigenvectors.

Fourier Transforms



Almost every imaginable signal can be broken down into a combination of simple waves. This fact is the central philosophy behind Fourier transforms (FT), they take a signal and express it in terms of the frequencies of the waves that make up that signal. When sound is recorded digitally the strength of the sound wave itself *can* be recorded (this is what a ".wav" file is), but more often these days the Fourier transform is recorded instead. At every moment a list of the strengths of the various frequencies is written down. This is more or less what an mp3 is (with lots of other tricks). It's not until a speaker has to physically play the sound that the FT is turned back into a regular sound signal.

However, acoustics are just the simplest application of FT's. An image is another kind of signal, but unlike sound an image is a "two dimensional" signal. A different kind of FT can be still found, and it is likewise two dimensional. When this was first done on computers it was found that, for pretty much any picture that isn't random static, most of the FT is concentrated around the lower frequencies. In a nutshell, this is because most pictures don't change quickly over small distances, so the higher frequencies aren't as important. This is the basic idea behind .jpeg" encoding and compression.

Pre-Processing

~ Fourier Transform

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:

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

Since we are dealing with images, and in fact digital images, so for digital images we will be working on **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:

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

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 fundamental 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 utilising periodicity of matrix W, some terms in DFT calculation can be combined; by utilising 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.

~ Amplitude Spectrum and Phase Spectrum

Here 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:

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

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:

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

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:

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

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 here the 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 and improve current systems, 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.

~ Extracting Feature

As we discussed before, traditional facial expression recognition systems are based on PCA. PCA is one of the oldest in analysing 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 Eigen faces. If we suppose the eigenvectors of S are V_1 , V_2 , ..., V_n and are associated respectively with the eigenvalues $m_1 \geq m_2 \geq ..... \geq m_N$

$$X = \sum_{i=1}^{n} \hat{x_i}\, v_i \tag{11}$$

However, if the number of dimensions 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:

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

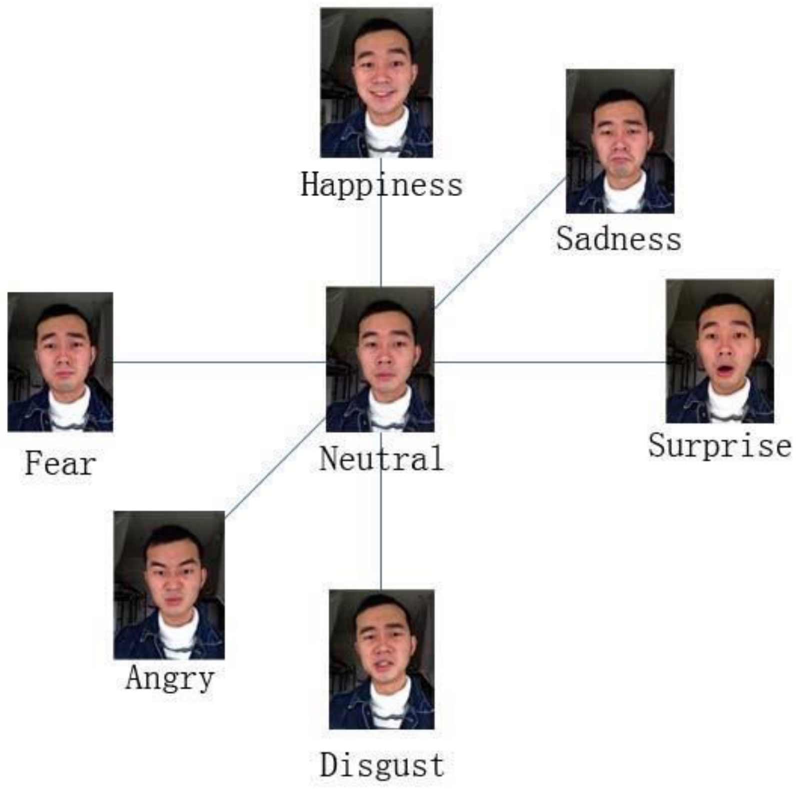
And from Eq.12 we can get Eq.13 by multiplying with Z^T:

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

Classifying Faces

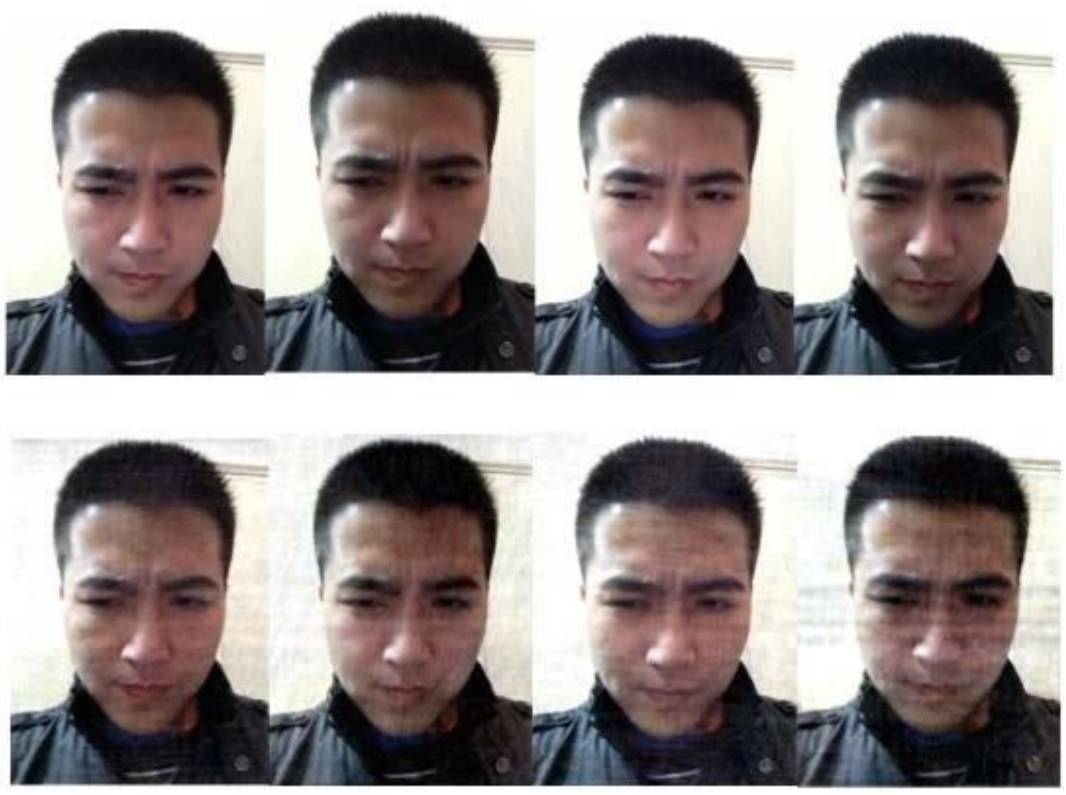
Since we have extracted face features, the next step is to put the images into classes. Instead of choosing a classifier, we choose to calculate Euclidian distance of test images from all the training images and figure out the minimum value in order to find out the train image which is most similar to the test images. Now, 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.

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.



Application

Firstly, our purpose is to detect and extract individual's face from the background.



After lighting compensation, extracting skin, and find skin colour blocks, we find the identified face which has been converted to grey images. One example is displayed below.

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.

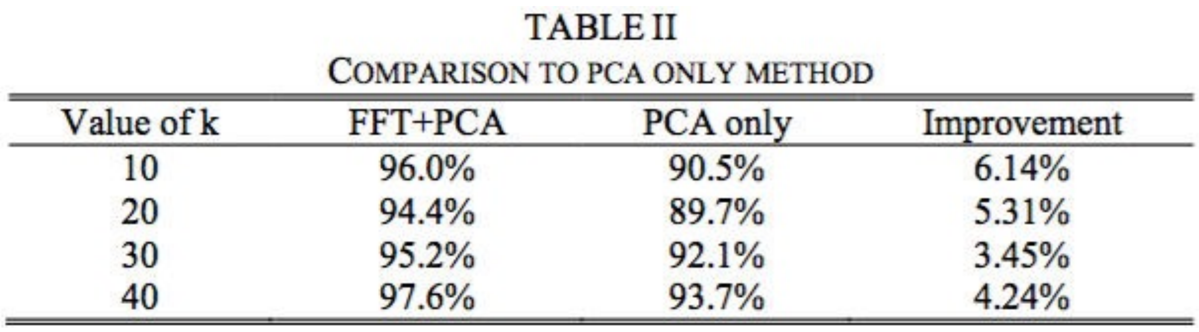


TABLE II

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, we can clear see that the new method (FFT+PCA) is more reasonable than PCA only, and is the reason that it's adopted in use currently.

Conclusion

Based on the analysis above, modern methods these days use a full system of face expression recognition (FFT+PCA). 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.

Thus we are able to recognise people based on images and predefined datasets which has improved progress in numerous fields and mainly in the fields of Safety and Security.