UNIVERSITY
OF GÄVLE

FACULTY OF ENGINEERING AND SUSTAINABLE DEVELOPMENT

The Algorithms of Speech Recognition, Programming and Simulating in MATLAB

Tingxiao Yang

January 2012

Bachelor's Thesis in Electronics

Bachelor's Program in Electronics
Examiner: Niklas Rothpfeffer

Abstract

The aim of this thesis work is to investigate the algorithms of speech recognition. The author programmed and simulated the designed systems for algorithms of speech recognition in MATLAB. There are two systems designed in this thesis. One is based on the shape information of the cross-correlation plotting. The other one is to use the Wiener Filter to realize the speech recognition. The simulations of the programmed systems in MATLAB are accomplished by using the microphone to record the speaking words. After running the program in MATLAB, MATLAB will ask people to record the words three times. The first and second recorded words are different words which will be used as the reference signals in the designed systems. The third recorded word is the same word as the one of the first two recorded words. After recording words, the words will become the signals' information which will be sampled and stored in MATLAB. Then MATLAB should be able to give the judgment that which word is recorded at the third time compared with the first two reference words according to the algorithms programmed in MATLAB. The author invited different people from different countries to test the designed systems. The results of simulations for both designed systems show that the designed systems both work well when the first two reference recordings and the third time recording are recorded from the same person. But the designed systems all have the defects when the first two reference recordings and the third time recording are recorded from the different people. However, if the testing environment is quiet enough and the speaker is the same person for three time recordings, the successful probability of the speech recognition is approach to 100%. Thus, the designed systems actually work well for the basical speech recognition.

Key words: Algorithm, Speech recognition, MATLAB, Recording, Cross-correlation, Wiener Filter, Program, Simulation.

Acknowledgements

The author must thank Niklas for providing effective suggestions to accomplish this thesis.

Abbreviations

DC	Direct Current
AD	Analog to Digital
WSS	Wide Sense Stationary
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
STFT	Shot-Time Fourier Transform

Table of contents

Chapter 1

Introduction

1.1 Background

Speech recognition is a popular topic in today's life. The applications of Speech recognition can be found everywhere, which make our life more effective. For example the applications in the mobile phone, instead of typing the name of the person who people want to call, people can just directly speak the name of the person to the mobile phone, and the mobile phone will automatically call that person. If people want send some text messages to someone, people can also speak messages to the mobile phone instead of typing. Speech recognition is a technology that people can control the system with their speech. Instead of typing the keyboard or operating the buttons for the system, using speech to control system is more convenient. It can also reduce the cost of the industry production at the same time. Using the speech recognition system not only improves the efficiency of the daily life, but also makes people's life more diversified.

1.2 Objectives of Thesis

In general, the objective of this thesis is to investigate the algorithms of speech recognition by programming and simulating the designed system in MATLAB. At the same time, the other purpose of this thesis is to utilize the learnt knowledge to the real application.

In this thesis, the author will program two systems. The main algorithms for these two designed systems are about cross-correlation and FIR Wiener Filter. To see if these two algorithms can work for the speech recognition, the author will invite different people from different countries to test the designed systems. In order to get reliable results, the tests will be completed in different situations. Firstly, the test environments will be noisy and noiseless respectively for investigating the immunity of the noise for designed systems. And the test words will be chosen as different pairs that are the easily recognized words and the difficultly

recognized words. Since the two designed systems needs three input speech words that are two reference speech words and one target speech word, so it is significant to check if the two designed systems work well when the reference speech words and the target speech words are recorded from the different person.

Chapter 2

Theory

This theory part introduces some definitions and information which will be involved in this thesis. The author needs this compulsory information to support his research. By concerning and utilizing the theoretic knowledge, the author achieved his aim of this thesis. Including DC level and sampling theory, DFT, FFT, spectrum normalization, the cross-correlation algorithm, the autocorrelation algorithm, the FIR Wiener Filter, use spectrogram function to get the desired signals.

2.1 The DC Level and Sampling Theory

When doing the signal processing analysis, the information of the DC level for the target signal is not that useful except the signal is applied to the real analog circuit, such as AD convertor, which has the requirement of the supplied voltage. When analyzing the signals in frequency domain, the DC level is not that useful. Sometimes the magnitude of the DC level in frequency domain will interfere the analysis when the target signal is most concentrated in the low frequency band. In WSS condition for the stochastic process, the variance and mean value of the signal will not change as the time changing. So the author tries to reduce this effect by deducting of the mean value of the recorded signals. This will remove the zero frequency components for the DC level in the frequency spectrum.

In this thesis, since using the microphone records the person's analog speech signal through the computer, so the data quality of the speech signal will directly decide the quality of the speech recognition. And the sampling frequency is one of the decisive factors for the data quality. Generally, the analog signal can be represented as

$$x(t) = \sum_{i=1}^{N} A_i \cos(2\pi f_i t + \phi_i) \tag{1}$$

This analog signal actually consists of a lot of different frequencies' components. Assuming there is only one frequency component in this analog signal, and it has no phase shift. So this analog signal becomes:

$$x(t) = A\cos(2\pi ft) \quad (2)$$

The analog signal cannot be directly applied in the computer. It is necessary to sample the analog signal x (t) into the discrete-time signal x (n), which the computer can use to process. Generally, the discrete signal x (n) is always regarded as one signal sequence or a vector. So MATLAB can do the computation for the discrete-time signal. The following Figure 1 is about sampling the analog signal into the discrete-time signal:

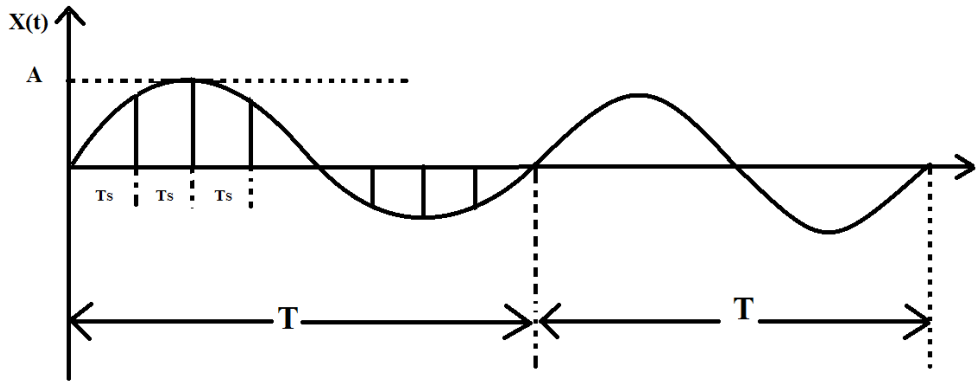

Figure 1: The simple figure about sampling the analog signal

As Fig.1 shown above, the time period of the analog signal x (t) is T. The sampling period of the discrete-time signal is T_S. Assuming the analog signal is sampled from the initial time 0, so the sampled signal can be written as a vector $x(n)=[x(0),x(1),x(2),x(3),x(4)\ldots x(N-1)]$. As known, the relation between the analog signal frequency and time period is reciprocal. So the sampling frequency of the sampled signal is $f_s=1/T_s$. Suppose the length of x (n) is N for K original time periods. Then the relation between T and T_s is $N\times T_s=K\times T$. So $N/K= T/T_s=f_s/f$, where both N and K are integers. And if this analog signal is exactly sampled with the same sampling space and the sampled signal is periodic, then N/K is integer also. Otherwise, the sampled signal will be aperiodic.

According to the sampling theorem (Nyquist theorem)[2], when the sampling frequency is larger or equal than 2 times of the maximum of the analog signal frequencies, the discrete-time signal is able to be used to reconstruct the original analog signal. And the higher sampling frequency will result the better sampled signals for analysis. Relatively, it will need faster processor to process the signal and respect with more data spaces. *In nontelecommunications applications, in which the speech recognition subsystem has access to high quality speech, sample frequencies of 10 kHz, 14 kHz and 16 kHz have been used. These*

sample frequencies give better time and frequency resolution [1]. In this thesis, for MATLAB program, the sampling frequency is set as 16 kHz. So the length of the recorded signal in 2 second will be 32000 time units in MATLAB. In next part, the theory of DFT and FFT will be introduced, which are important when trying to analyze spectrums in frequency domain. And it is the key to get the way to do the speech recognition in this thesis.

2.2 Time Domain to Frequency Domain: DFT and FFT

2.2.1 DFT

The DFT is an abbreviation of the *Discrete Fourier Transform*. So the DFT is just a type of Fourier Transform for the discrete-time x (n) instead of the continuous analog signal x (t). The Fourier Transform equation is as follow:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{j\omega n} \tag{3}$$

From the equation, the main function of the Fourier Transform is to transform the variable from the variable n into the variable ω, which means transforming the signals from the time domain into the frequency domain.

Assuming the recorded voice signal x(n) is a sequence or vector which consists of complex values, such as x(n)=R+I, where R stands for the real part of the value, and I stands for the imaginary part of the value. Since the exponent factor is:

$$e^{j\omega n} = \cos(\omega n) + j \cdot \sin(\omega n) \tag{4}$$

So:

$$x(n) \cdot e^{j\omega n} = (R+I) \cdot [\cos(\omega n) + j \cdot \sin(\omega n)] = R \cdot \cos(\omega n) + R \cdot j \cdot \sin(\omega n) + I \cdot \cos(\omega n) + I \cdot j \cdot \sin(\omega n) \tag{5}$$

Rearrange the real part and image part of the equation. We get:

$$x(n) \cdot e^{j\omega n} = [R \cdot \cos(\omega n) + I \cdot \cos(\omega n)] + [R \cdot j \cdot \sin(\omega n) + I \cdot j \cdot \sin(\omega n)] \tag{6}$$

So the equation (3) becomes:

$$x(\omega) = \sum [R\cos(\omega n) + I\cos(\omega n)] + \sum j[R\sin(\omega n) + I\sin(\omega n] \tag{7}$$

The equation (7) is also made of the real part and the imaginary part. Since in general situation, the real value of the signal x (n) is used. So if the imaginary part I=0. Then the Fourier Transform is

$$X(\omega)=\sum_{n=-\infty}^{\infty}[R\cos(\omega n)]+\sum_{n=-\infty}^{\infty}[jRsin(\omega n)] \tag{8}$$

The analyses above are the general steps to program the Fourier Transform by programing the computation frequency factor which consists of the real part and the imaginary part with the signal magnitude. But in MATLAB, there is a direct command "fft", which can be used directly to get the transform function. And the variable ω in equation (3) can be treated as a continuous variable.

Assuming the frequency ω is set in [0,2π], X (ω) can be regarded as an integral or the summation signal of all the frequency components. Then the frequency component X (k) of X (ω) is got by sampling the entire frequency interval ω= [0,2π] by N samples. So it means the frequency component $\omega_k = k\times\frac{2\pi}{N}$. And the DFT equation for the frequency component ω_k is as below:

$$X(k)=X(\omega_k)=\sum_{n=-\infty}^{\infty}x(n)e^{j\omega_k n}=\sum_{n=0}^{N-1}x(n)e^{j\frac{2\pi k}{N}n}, \quad 0\le k\le N-1 \tag{9}$$

This equation is used to calculate the magnitude of the frequency component. The key of understanding DFT is about sampling the frequency domain. The sampling process can be shown more clearly as the following figures.

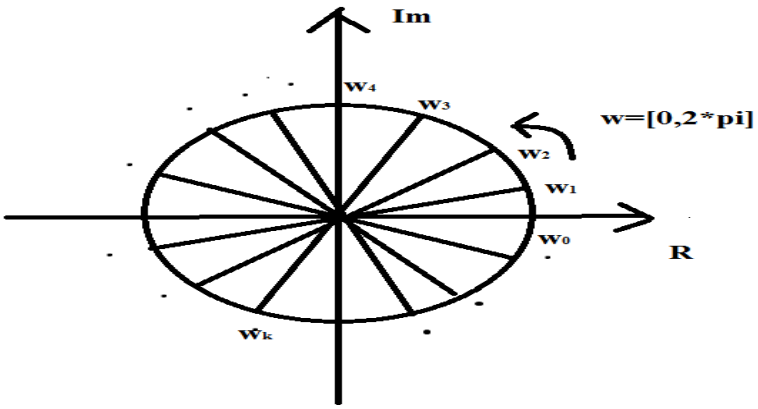

Figure 2: Sampling in frequency circle

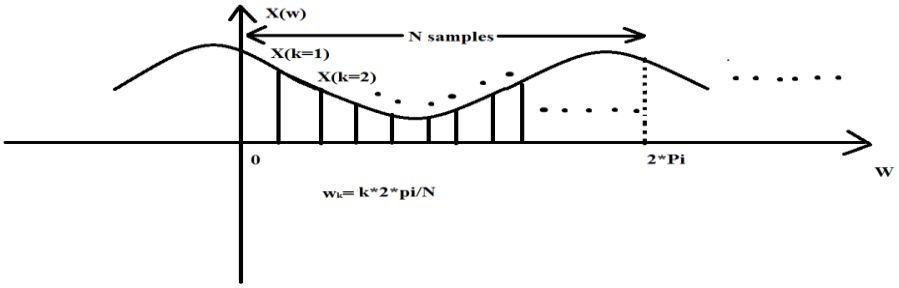

Figure 3: Sampling in frequency axis

In addition, MATLAB are dealing with the data for vectors and matrixes. Definitely, understanding the linear algebra or matrix process of the DFT is necessary. By observing the equation (3), except the summation operator, the equation consists of 3 parts: output X (ω), input x (n) and the phase factor $e^{j\omega_k n}$. Since all the information of the frequency components is from the phase factor $e^{j\omega_k n}$. So the phase factor can be denoted as:

$W_N^{kn} = e^{j\omega_k n}$, n and k are integers from 0 to N-1. (10)

Writing the phase factor in vector form:

$$W_N^{kn} = e^{j\omega_k n} = [W_N^{0k}, W_N^{1k}, W_N^{2k}, W_N^{3k}, W_N^{4k}, ..., W_N^{(N-1)k}] \quad (11)$$

And

$$x(n) = [x(0), x(1), x(2)..., x(N-1)] \quad (12)$$

So the equation (9) for the frequency component X (k) is just the inner product of the $(W_N^{kn})^H$ and x(n) :

$$X(k) = (W_N^{kn})^H \cdot x(n) \quad (13)$$

This is the vector form about calculating frequency component with using DFT method. But if the signal is a really long sequence, and the memory space is finite, then the using DFT to get the transformed signal will be limited. The faster and more efficient computation of DFT is FFT. The author will introduce briefly about FFT in next section.

2.2.2 FFT

The FFT is an abbreviation of the *Fast Fourier Transform*. Essentially, the FFT is still the DFT for transforming the discrete-time signal from time domain into its frequency domain. The difference is that the FFT is faster and more efficient on computation. And there are

many ways to increase the computation efficiency of the DFT, but the most widely used FFT algorithm is the *Radix-2 FFT Algorithm* [2].

Since FFT is still the computation of DFT, so it is convenient to investigate FFT by firstly considering the N-point DFT equation:

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn},\ \mathrm{k}=0,\ 1,\ 2\ldots\ \mathrm{N}-1 \tag{14}$$

Firstly separate x(n) into two parts: x(odd)=x(2m+1) and x(even)=x(2m), where m=0, 1, 2,…,N/2-1. Then the N-point DFT equation also becomes two parts for each N/2 points:

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn}=\sum_{m=0}^{N/2-1}x(2m)W_N^{2mk}+\sum_{m=0}^{N/2-1}x(2m+1)W_N^{(2m+1)k}=\sum_{m=0}^{N/2-1}x(2m)W_N^{2mk}+W_N^k\sum_{m=0}^{N/2-1}x(2m+1)W_N^{2mk}, \tag{15}$$

where $\mathrm{m}=0,\ 1,\ 2,\ ....,\ \mathrm{N}/2-1$

Since:

$$e^{j\omega_k n}=\cos(\omega_k n)+j\sin(\omega_k n). \tag{16}$$

$$e^{j(\omega_k+\pi)n}=\cos\left[(\omega_k+\pi)n\right]+j\cdot\sin\left[(\omega_k+\pi)n\right]$$

$$=-\cos(\omega_k n)-j\cdot\sin(\omega_k n)=-[\cos(\omega_k n)+j\cdot\sin(\omega_k n)]=-e^{j\omega_k n} \tag{17}$$

That is:

$$e^{j(\omega_k+\pi)n}=-e^{j\omega_k n} \tag{18}$$

So when the phase factor is shifted with half period, the value of the phase factor will not change, but the sign of the phase factor will be opposite. This is called symmetry property [2] of the phase factor. Since the phase factor can be also expressed as $W_N^{kn}=e^{j\omega_k n}$, so:

$$W_N^{(k+\frac{N}{2})n}=-W_N^{kn} \tag{19}$$

And

$$(W_N^{kn})^2=-W_{N/2}^{kn}=e^{j\frac{4\pi k}{N}n} \tag{20}$$

The N-point DFT equation finally becomes:

$$X(k)=\sum_{m=0}^{N/2-1}x_1(m)W_{N/2}^{mk}+W_N^k\sum_{m=0}^{N/2-1}x_2(m)W_{N/2}^{mk}=X_1(k)+W_N^kX_2(k),\ \mathrm{k}=0,\ 1\ldots\mathrm{N}/2 \tag{21}$$

$$X(k+N/2)=X_1(k)-W_N^kX_2(k),\ \mathrm{k}=0,\ 1,\ 2\ldots\mathrm{N}/2 \tag{22}$$

So N-point DFT is separated into two N/2-point DFT. From equation (21), $X_1(k)$ has $(N/2)\cdot(N/2)=(N/2)^2$ complex multiplications. $W_N^kX_2(k)$ has $N/2+(N/2)^2$ complex multiplications.

So the total number of complex multiplications for X(k) is $2\cdot(N/2)^2 + N/2 = N^2/2 + N/2$. For original N-point DFT equation (14), it has N^2 complex multiplications. So in the first step, separating x(n) into two parts makes the number of complex multiplications from N^2 to $N^2/2+N/2$. The number of calculations has been reduced by approximately half.

This is the process for reducing the calculations from N points to N/2 points. So continuously separating the $x_1(m)$ and $x_2(m)$ independently into the odd part and the even part in the same way, the calculations for N/2 points will be reduced for N/4 points. Then the calculations of DFT will be continuously reduced. So if the signal for N-point DFT is continuously separated until the final signal sequence is reduced to the one point sequence. Assuming there are $N=2^s$ points DFT needed to be calculated. So the number of such separations can be done is $s=\log_2(N)$. So the total number of complex multiplications will be approximately reduced to $(N/2)\log_2(N)$. For the addition calculations, the number will be reduced to $N\log_2(N)$ [2]. Because the multiplications and additions are reduced, so the speed of the DFT computation is improved. The main idea for Radix-2 FFT is to separate the old data sequence into odd part and even part continuously to reduce approximately half of the original calculations.

2.3 Frequency Analysis in MATLAB of Speech Recognition

2.3.1 Spectrum Normalization

After doing DFT and FFT calculations, the investigated problems will be changed from the discrete-time signals x(n) to the frequency domain signal X(ω). The spectrum of the X(ω) is the whole integral or the summation of the all frequency components. When talking about the speech signal frequency for different words, each word has its frequency band, not just a single frequency. And in the frequency band of each word, the spectrum ($|X(\omega)|$) or spectrum power ($|X(\omega)|^2$) has its maximum value and minimum value. When comparing the differences between two different speech signals, it is hard or unconvincing to compare two spectrums in different measurement standards. So using the normalization can make the measurement standard the same.

In some sense, the normalization can reduce the error when comparing the spectrums, which is good for the speech recognition [3]. So before analyzing the spectrum differences for different words, the first step is to normalize the spectrum $|X(\omega)|$ by the linear normalization. The equation of the linear normalization is as below:

y=(x-MinValue)/(MaxValue-MinValue) (23)

After normalization, the values of the spectrum $|X(\omega)|$ are set into interval [0, 1]. The normalization just changes the values' range of the spectrum, but not changes the shape or the information of the spectrum itself. So the normalization is good for spectrum comparison. Using MATLAB gives an example to see how the spectrum is changed by the linear normalization. Firstly, record a speech signal and do the FFT of the speech signal. Then take the absolute values of the FFT spectrum. The FFT spectrum without normalization is as below:

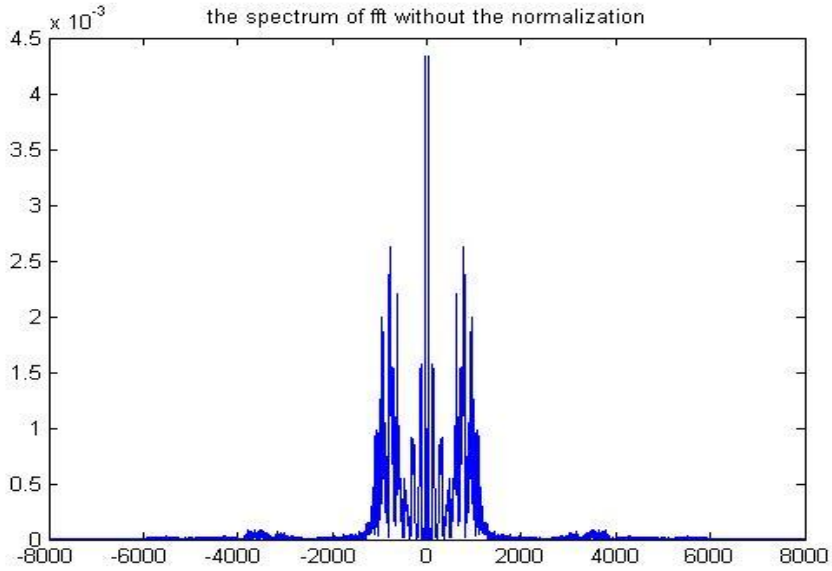

Figure 4: Absolute values of the FFT spectrum without normalization

Secondly, normalize the above spectrum by the linear normalization. The normalized spectrum is as below:

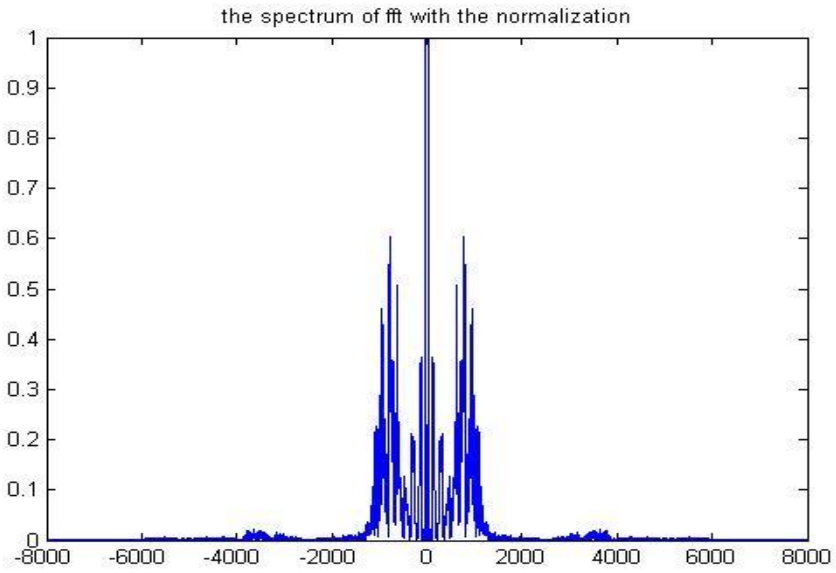

Figure 5: Absolute values of the FFT spectrum with normalization

From the Fig.4 and the Fig.5, the difference between two spectrums is only the interval of the spectrum $|X(\omega)|$ values, which is changed from [0, 4.5×10^{-3}] to [0, 1]. Other information of the spectrum is not changed. After the normalization of the absolute values of FFT, the next step of programming the speech recognition is to observe spectrums of the three recorded speech signals and find the algorithms for comparing differences between the third recorded target signal and the first two recorded reference signals.

2.3.2 The Cross-correlation Algorithm

There is a substantial amount of data on the frequency of the voice fundamental (F_0) in the speech of speakers who differ in age and sex. [4] For the same speaker, the different words also have the different frequency bands which are due to the different vibrations of the vocal cord. And the shapes of spectrums are also different. These are the bases of this thesis for the speech recognition. In this thesis, to realize the speech recognition, there is a need to compare spectrums between the third recorded signal and the first two recorded reference signals. By checking which of two recorded reference signals better matches the third recorded signal, the system will give the judgment that which reference word is again recorded at the third time. When thinking about the correlation of two signals, the first algorithm that will be considered

is the cross-correlation of two signals. The cross-correlation function method is really useful to estimate shift parameter [5]. Here the shift parameter will be referred as frequency shift.

The definition equation of the cross-correlation for two signals is as below:

$$r_{xy} = r(m) = \sum_{n=-\infty}^{\infty} x(n)y(n+m), m = 0, \pm 1, \pm 2, \pm 3, ... \quad (24)$$

From the equation, the main idea of the algorithm for the cross-correlation is approximately 3 steps：

Firstly, fix one of the two signals x(n) and shift the other signal y(n) left or right with some time units.

Secondly, multiply the value of x (n) with the shifted signal y (n+m) position by position.

At last, take the summation of all the multiplication results for x (n) · y (n+m).

For example, two sequence signals x(n) = [0 0 0 1 0], y(n)= [0 1 0 0 0], the lengths for both signals are N=5. So the cross-correlation for x(n) and y(n) is as the following figures shown:

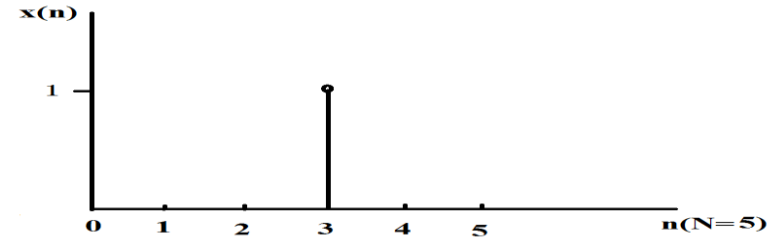

Figure 6: The signal sequence x(n)

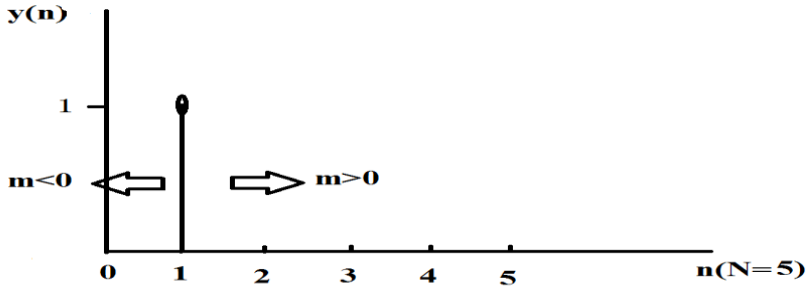

Figure 7: The signal sequence y (n) will shift left or right with m units

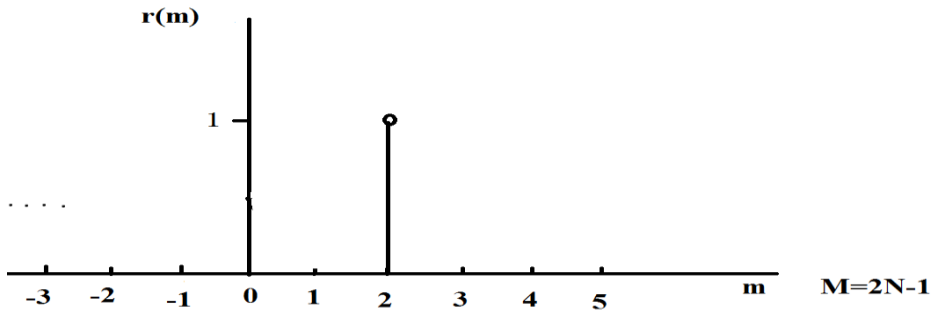

Figure 8: The results of the cross-correlation, summation of multiplications

As the example given, there is a discrete time shift about 2 time units between the signals x (n) and y (n). From Fig.8, the cross-correlation r(m) has a non-zero result value, which is equal 1 at the position m=2. So the m-axis of Fig.8 is no longer the time axis for the signal. It is the time-shift axis. Since the lengths of two signals x (n) and y (n) are both N=5, so the length of the time-shift axis is 2N. When using MATLAB to do the cross-correlation, the length of the cross-correlation is still 2N. But in MATLAB, the plotting of the cross-correlation is from 0 to 2N-1, not from –N to +N anymore. Then the 0 time-shift point position will be shifted from 0 to N. So when two signals have no time shift, the maximum value of their cross-correlation will be at the position m=N in MATLAB, which is the middle point position for the total length of the cross-correlation.

In MATLAB, the plotting of Fig.8 will be as below:

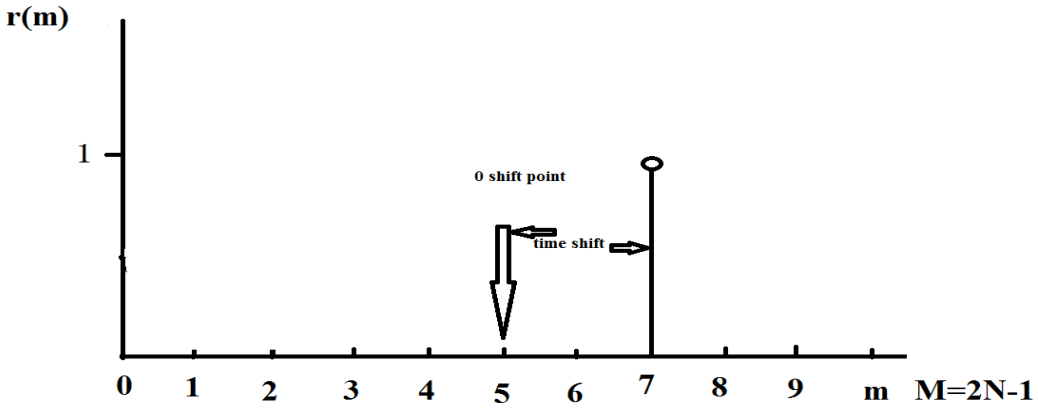

Figure 9: The cross-correlation which is plotted in MATLAB way (not real MATLAB Figure)

From Fig.9, the maximum value of two signals' cross-correlation is not at the middle point position for the total length of the cross-correlation. As the example given, the lengths of both signals are N=5, so the total length of the cross-correlation is 2N=10. Then when two signals have no time shift, the maximum value of their cross-correlation should be at m=5. But in Fig.9, the maximum value of their cross-correlation is at the position m=7, which means two original signals have 2 units time shift compared with 0 time shift position.

From the example, two important information of the cross-correlation can be given. One is when two original signals have no time shift, their cross-correlation should be the maximum; the other information is that the position difference between the maximum value position and the middle point position of the cross-correlation is the length of time shift for two original signals.

Now assuming the two recorded speech signals for the same word are totally the same, so the spectrums of two recorded speech signals are also totally the same. Then when doing the cross-correlation of the two same spectrums and plotting the cross-correlation, the graph of the cross-correlation should be totally symmetric according to the algorithm of the cross-correlation. However, for the actual speech recording, the spectrums of twice recorded signals which are recorded for the same word cannot be the totally same. But their spectrums should be similar, which means their cross-correlation graph should be approximately symmetric. This is the most important concept in this thesis for the speech recognition when designing the system 1.

By comparing the level of symmetric property for the cross-correlation, the system can make the decision that which two recorded signals have more similar spectrums. In other words, these two recorded signals are more possibly recorded for the same word. Take one simulation result figure in MATLAB about the cross-correlations to see the point:

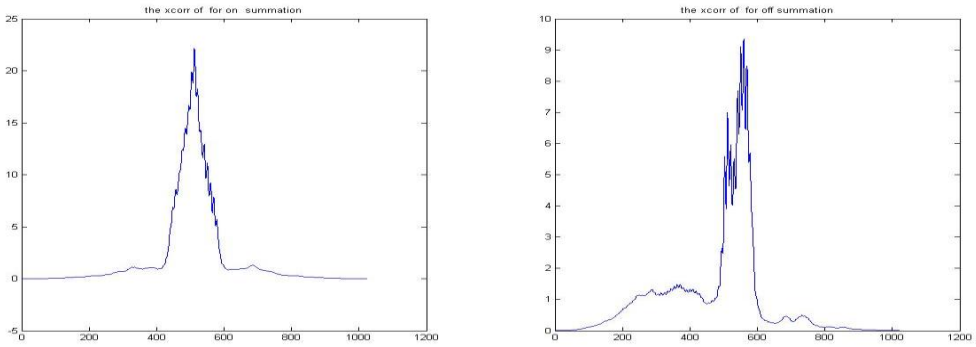

Figure 10: The graphs of the cross-correlations

The first two recorded reference speech words are "hahaha" and "meat", and the third time recorded speech word is "hahaha" again. From Fig.10, the first plotting is about the cross-correlation between the third recorded speech signal and the reference signal "hahaha". The second plotting is about the cross-correlation between the third recorded speech signal and the reference signal "meat". Since the third recorded speech word is "hahaha", so the first plotting is really more symmetric and smoother than the second plotting.

In mathematics, if we set the frequency spectrum's function as a function f(x), according to the axial symmetry property definition: for the function f(x), if x1 and x3 are axis-symmetric about x=x2, then f(x1) =f(x3). For the speech recognition comparison, after calculating the cross-correlation of two recorded frequency spectrums, there is a need to find the position of the maximum value of the cross-correlation and use the values right to the maximum value position to minus the values left to the maximum value position. Take the absolute value of this difference and find the mean square-error of this absolute value. If two signals better match, then the cross-correlation is more symmetric. And if the cross-correlation is more symmetric, then the mean square-error should be smaller. By comparing of this error, the system decides which reference word is recorded at the third time. The codes for this part can be found in Appendix.

2.3.3 The Auto-correlation Algorithm

In the previous part, it is about the cross-correlation algorithm. See the equation (24), the autocorrelation can be treated as computing the cross-correlation for the signal and itself

instead of two different signals. This is the definition of auto-correlation in MATLAB. The auto-correlation is the algorithm to measure how the signal is self-correlated with itself.

The equation for the auto-correlation is:

$$r_x(k) = r_{xx}(k) = \sum_{k=-\infty}^{\infty} x(n)x(n+k) \tag{25}$$

The figure below is the graph of plotting the autocorrelation of the frequency spectrum $|X(\omega)|$

.

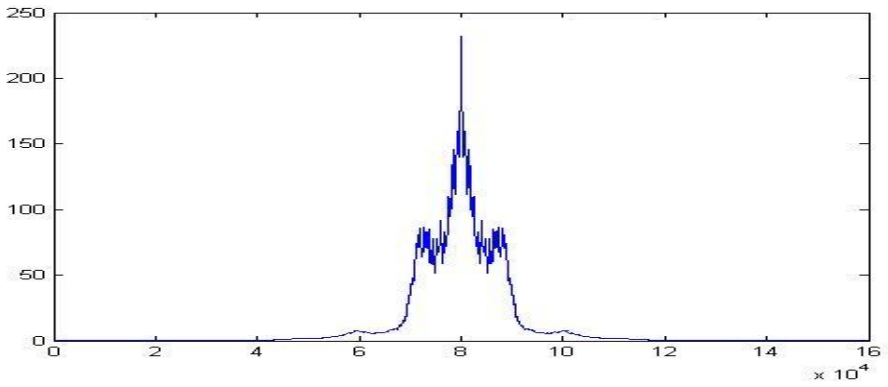

Figure 11: The autocorrelation for $|X(\omega)|$

2.3.4 The FIR Wiener Filter

The FIR Wiener filter is used to estimate the desired signal d (n) from the observation process x (n) to get the estimated signal d (n)'. It is assumed that d (n) and x (n) are correlated and jointly wide-sense stationary. And the error of estimation is e (n) =d (n)-d (n)'.

The FIR Wiener filter works as the figure shown below:

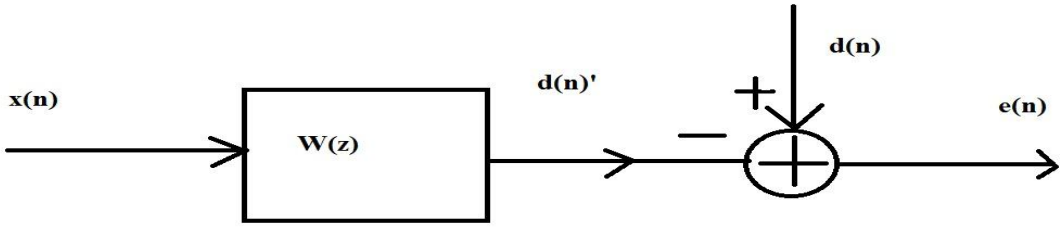

Figure 12: Wiener filter

From Fig.12, the input signal of Wiener filter is x (n). Assume the filter coefficients are w (n). So the output d (n)' is the convolution of x (n) and w (n):

$$d(n)' = w(n) * x(n) = \sum_{l=0}^{p-1} w(l)x(n-l) \tag{26}$$

Then the error of estimation is:

$$e(n) = d(n) - d(n)' = d(n) - \sum_{l=0}^{p-1} w(l)x(n-l) \tag{27}$$

The purpose of Wiener filter is to choose the suitable filter order and find the filter coefficients with which the system can get the best estimation. In other words, with the proper coefficients the system can minimize the mean-square error:

$$\xi = E\left\{\left|e(n)\right|^2\right\} = E\left\{\left|d(n) - d(n)'\right|^2\right\} \tag{28}$$

Minimize the mean-square error in order to get the suitable filter coefficients, there is a sufficient method for doing this is to get the derivative of ξ to be zero with respect to w*(k). As the following equation:

$$\frac{\partial \xi}{\partial w^*(k)} = \frac{\partial}{\partial w^*(k)} E\left\{e(n)e^*(n)\right\} = E\left\{e(n)\frac{\partial e^*(n)}{\partial w^*(k)}\right\} = 0 \tag{29}$$

From equation (27) and equation (29), we know:

$$\frac{\partial e^*(n)}{\partial w^*(k)} = -x^*(n-k) \tag{30}$$

So the equation (29) becomes:

$$\frac{\partial \xi}{\partial w^*(k)} = E\left\{e(n)\frac{\partial e^*(n)}{\partial w^*(k)}\right\} = -E\left\{e(n)x^*(n-k)\right\} = 0 \tag{31}$$

Then we get:

$$E\left\{e(n)x^*(n-k)\right\} = 0 \quad \text{, k=0, 1…., p-1} \tag{32}$$

The equation (32) is known as *orthogonality principle* or *the projection theorem* [6].

By the equation (27), we have

$$E\left\{e(n)x^*(n-k)\right\} = E\left\{\left[d(n) - \sum_{l=0}^{p-1} w(l)x(n-l)\right]x^*(n-k)\right\} = 0 \tag{33}$$

The rearrangement of the equation (33):

$$E\left\{d(n)x^*(n-k)\right\} - E\left\{\sum_{l=0}^{p-1} w(l)x(n-l)x^*(n-k)\right\} = r_{dx} - \sum_{l=0}^{p-1} w(l)r_x(k-l) = 0 \tag{34}$$

Finally, the equation is as below:

$$\sum_{l=0}^{p-1} w(l) r_x(k-l) = r_{dx} \quad ; \quad \text{k=0,} \quad 1\ldots \quad \text{p-1}$$

(35)

With $r_x(k) = r_x^*(-k)$, the equation may be written in matrix form:

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & \cdots & r_x^*(p-3) \\ \vdots & \vdots & & \vdots \\ r_x(p-1) & r_x(p-2) & \cdots & r_x^*(0) \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ \vdots \\ w(p-1) \end{bmatrix} = \begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \\ r_{dx}(2) \\ \vdots \\ r_{dx}(p-1) \end{bmatrix} \tag{36}$$

The matrix equation (36) is actually Wiener-Hopf equation [6] of:

$$R_x w = r_{dx} \tag{37}$$

In this thesis, the Wiener-Hopf equation can work for the voice recognition. From equation (37), the input signal x (n) and the desired signal d (n) are the only things that need to know. Then using x (n) and d(n) finds the cross-correlation r_{dx}. At the same time, using x (n) finds the auto-correlation $r_x(n)$ and using $r_x(n)$ forms the matrix R_x in MATLAB. When having the R_x and r_{dx}, it can be directly found out the filter coefficients. With the filter coefficients it can continuously get the minimum mean square-error ξ. From equations (27), (28), and (32), the minimum mean square-error ξ is:

$$\xi_{\min} = E\left\{e(n)d^*(n)\right\} = E\left\{\left[d(n) - \sum_{l=0}^{p-1} w(l)x(n-l)\right]d^*(n)\right\} = r_d(0) - \sum_{l=0}^{p-1} w(l) r_{dx}^*(l) \tag{38}$$

Apply the theory of Wiener filter to the speech recognition. If we want to use the Wiener-Hopf equation, it is necessary to know two given conditions: one is the desired signal d (n); the other one is the input signal x(n).

In this thesis, it is assumed that the recoded signals are wide-sense stationary processes. Then the first two recorded reference signals can be used as the input signals x1(n) and x2(n). The third recorded speech signal can be used as the desired signal d(n). It is a wish to find the best estimation of the desired signal in the Wiener filter. So the procedure of applying Wiener filter to the speech recognition can be thought as using the first two recorded reference signals

to estimate the third recorded desired signal. Since one of two reference signals x1(n), x2(n) is recorded for the same word as the word that is recoded at the third time. So using the one of two reference signals which is recorded for the same word as the third time recording to be the input signal of Wiener filter will have the smaller estimation minimum mean square-error $\xi_{\min}$ according to equation (38).

After defining the roles of three recorded signals in the designed system 2, the next step is just to find the auto-correlations of reference signals, which are r_{x1} (n), r_{x2} (n) and find the cross-correlations for the third recorded voice signal with the first two recorded reference signals, which are r_{dx1} (n), r_{dx2} (n). And use r_{x1} (n), r_{x2} (n) to build the matrix R_{x1}, R_{x2}. At last, according to the Wiener-Hopf equation (37), calculate the filter coefficients for both two reference signals and find the mean values of the minimum mean square-errors with respect to the two filter coefficients. Comparing the minimum mean square-errors, the system will give the judgment that which one of two recorded reference signals will be the word that is recorded at the third time. The better estimation, the smaller mean value of $\xi_{\min}$

2.3.5 Use spectrogram Function in MATLAB to Get Desired Signals

The spectrogram is a time-frequency plotting which contains power density distribution at the same time with respect to both frequency axis and time axis. In MATLAB, it is easy to get the spectrogram of the voice signal by defining some variables: the sampling frequency, the length of Shot-Time Fourier Transform (STFT) [7] and the length of window. In previous parts of this paper, DFT and FFT have been introduced. The STFT is firstly to use the window function to truncate the signal in the time domain, which makes the time-axis into several parts. If the window is a vector, then the number of parts is equal to the length of the window. Then compute the Fourier Transform of the truncated sequence with defined FFT length (nfft).

The Fig.13 below is the spectrogram for the recorded speech signal in MATLAB, with defined fs=16000, nfft=1024, the length of hanning window is 512, and the length of overlap is 380. It is necessary to mention that the length of window has to be smaller or equal than 1/2 the length of the STFT (nfft) when programming in MATLAB.

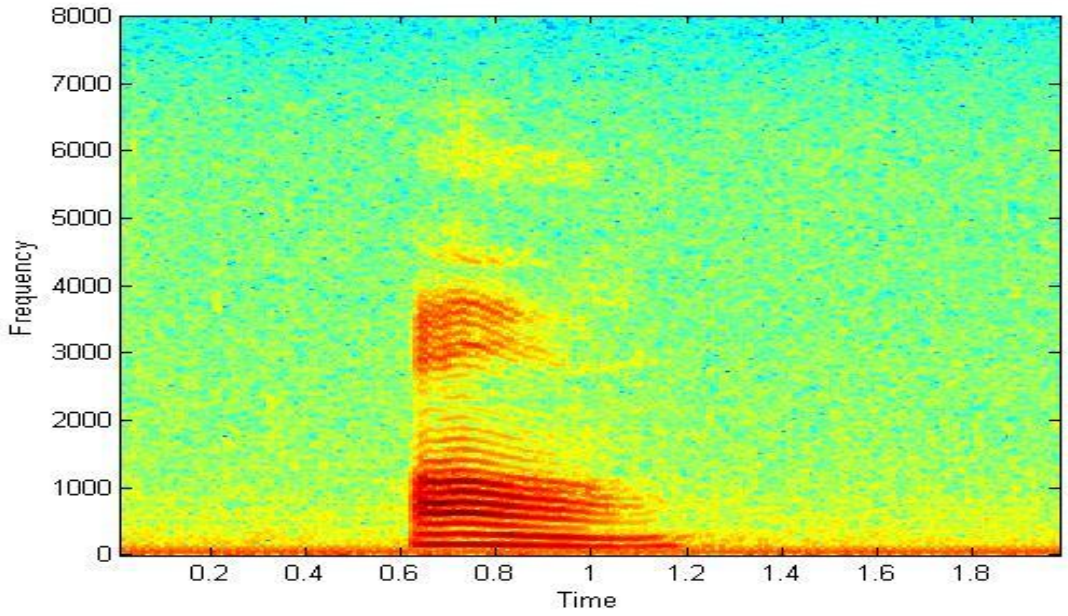

Figure 13: The spectrogram of recorded speech word "on"

From Fig.13, the X-axis is the time-axis and the Y-axis is the frequency-axis. The resolution of the color represents the gradient of the power distribution. The deeper color means the higher power distribution in that zone. From Fig.13, the most power is located at the low frequency band. The following figure is plotted in MATLAB for a 3-Dimension spectrogram of the same recorded speech word "on"

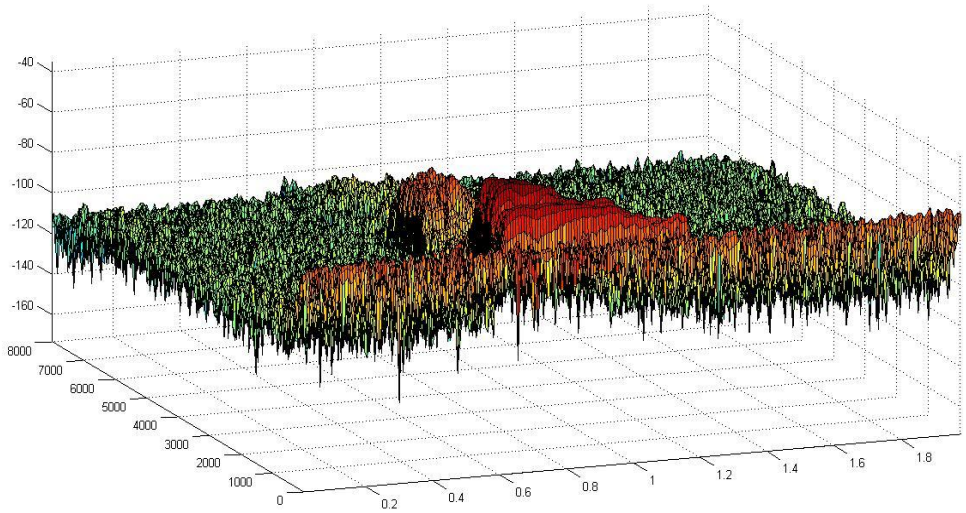

Figure 14: The 3-Dimension spectrogram of recorded speech word "on"

Basically the Fig.14 is exactly the same as the Fig.13 except that the power distribution can be viewed from the heights of the power "mountains". Now considering the speech recognition,

the point here is not the graph how it looks like, but the function of getting spectrogram. The procedure of making spectrogram in MATLAB has an important conception: use the window to truncate the time into short time parts and calculate the STFT. So it is convenient to use the spectrogram function in MATLAB to get the frequency spectrum purer and more reliable. Firstly see the figure as below:

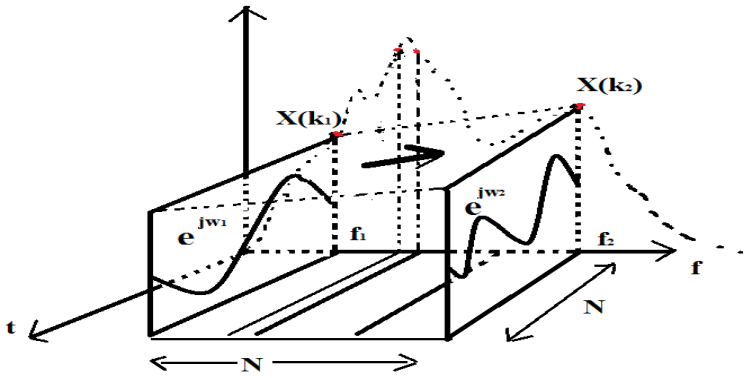

Figure 15: 3-Dimension relation graph of the DFT

From Fig.15, the spectrum in frequency domain can be treated as the integral or the summation of all the frequency components' planes. For each frequency component's plane, the height of the frequency component's plane is just the whole time domain signal multiply the correlated frequency phase factor ejw. From Fig.15, if the time domain signal is a pure periodic signal, then the frequency component will be the perfect one single component plane without touching other frequency plane, such as $e^{j\omega_1}$ and $e^{j\omega_2}$ planes shown in Fig.15. They are stable and will not affect of each other. But if the signal is aperiodic signal, see the figure as below:

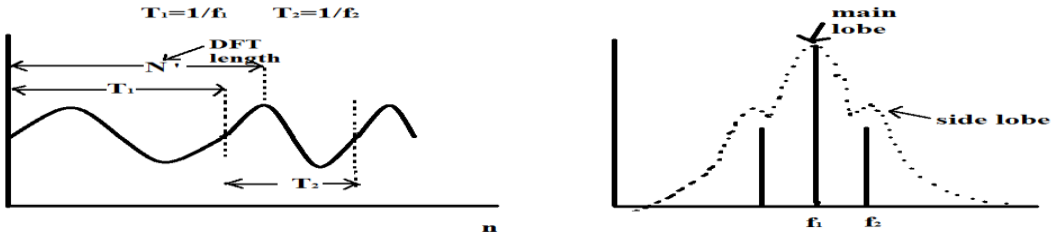

Figure 16: Aperiodic signal produces the leakage by DFT for the large length sequence

From Fig.16, ths sginal is an aperidic signal, the frequency changed after one period T1. If we still treat this aperidic signal as one single plane, and directly compute the DFT of it, the result of DFT for data sequence with the length of N'−T1 would like moving its frequency

component power into the frequency component which has the same frequency as this data sequence. The result of DFT is a power spectrum. The behavior of this power flowing is called leakage. Since the signal is discrete in the real signal processing, one time position has one value state. And when recording the speech signal, the speech signal is a complex signal which contains a lot of frequences. So the recorded speech signal will be aperiodic signal due to the change of the pronoucation, it will have the leakage in the frequency spectrum including the power of the interfering noise. From Fig.16, after time T1 the frequency of the signal is changed in the time period T2. As the frequency changing of the aperiodic signal, the spectrum will not be smooth, which is not good for analysis.

Using windows can improve this situation. Windows are weighting functions applied to data to reduce the spectrum leakage associated with finite observation intervals [8]. It's better to use the window to truncate the long signal sequence into the short time sequence. For short time sequence, the signal can be treated as "periodic" signal, and the signal out of the window is thought as all zeros. Then calculate the DFT or FFT for the truncated signal data. This is called Short Time Fourier Transform (STFT). Keep moving the window along the time axis, until the window has truncated through the whole spectrum. By this way, the window will not only reduce the leakage of the frequency component, but also make spectrum smoother.

Since moving step of the window is always less than the length of the window. So the resulted spectrum will have the overlaps. Overlaps are not bad for the analysis. The more overlaps, the better resolution of the STFT, which means the resulting spectrum is more realiable.

Using the spectrogram function in MATLAB can complete this procedure, which always gives a returned matrix by using "specgram" function in MATLAB. So the "specgram" can be directly used as the "window" function to get the filtered speech signal. After using the "specgram", the useful and reliable information of the recorded signals for both time domain and frequency domain can be got at the same. The next step to be considered is just to compare the spectrums of the third recorded signal with the first two recorded reference signals by computing the cross-correlation or using the Wiener Filter system as previously introduced. This's how the spectrogram works for the speech recognition in this thesis.

When using the "s=specgram(r, nfft, fs, hanning(512),380);" command in MATLAB, it will get a returned matrix, in which the elements are all complex numbers. Use MATLAB to plot the spectrogram for better understanding. The figure plotted in MATLAB is as below:

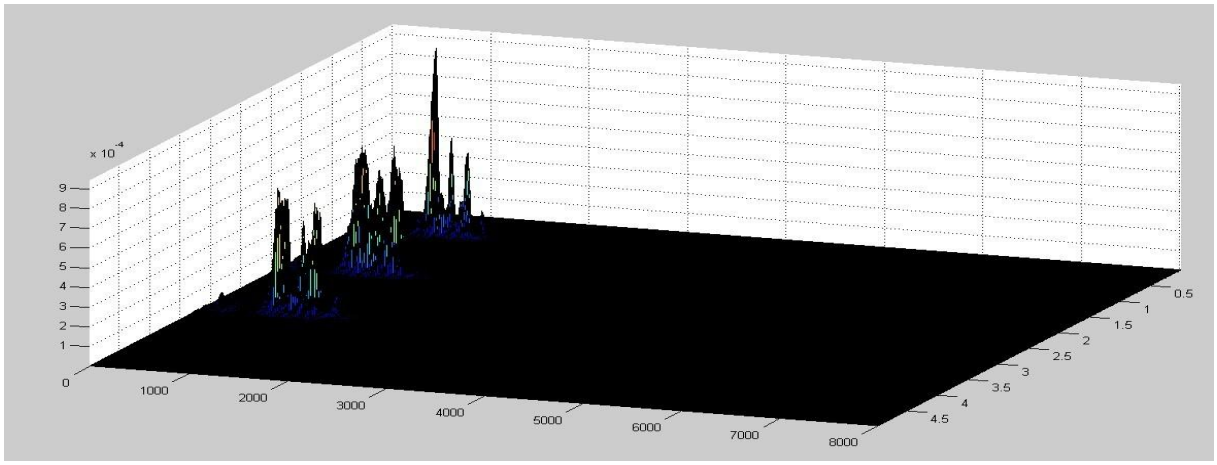

Figure 17: The spectrogram of speech "ha…ha…ha"

To be better understanding of the figure for the matrix, modify the figure as below:

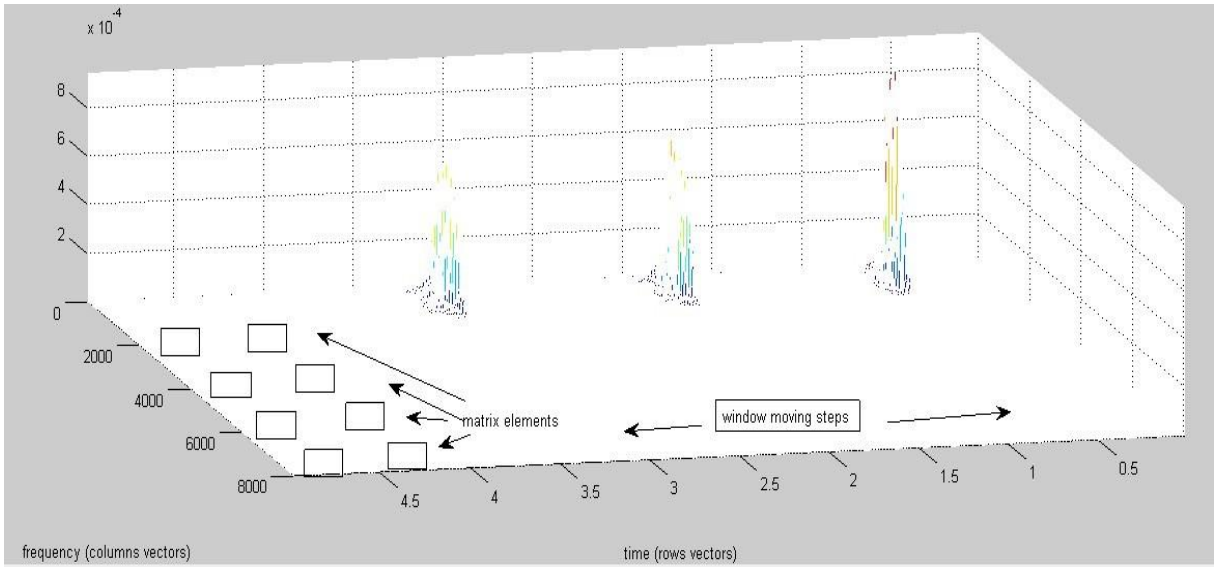

Figure 18: The modified figure for Figure 17

From Fig.18, The vector length of each row is related to the moving steps of the window. By checking the variable information in MATLAB of this exmaple, the returned matrix is a 513×603 matrix. Since the sampling frequency for the recording system is set as fs=16000, so the length of the voice signal is 16000×5=80000(recorded in 5 seconds). And the length of hanning window is set as 512. The overlap length setting is 380. And for the DFT/FFT periodic extension, the window function actually computes the length of 512+1=513.So the

moving step length is 513−380=133. So the number of time window steps is calculated as 80000/133=≈602, which is almost the same as the number of coulums for the matrix in MATALB.

It is shown that the moving window divided the time length of the original signal from 80000 into the short time length 603. So to count the number of conlums for the matrix is actually to see the time position. And to count from the number of rows of the matrix is actually to view the frequency position.

So for the element Sij=A in the matrix, the "i" is the frequency position (the number of rows.), and the "j" is the time position (the number of column). "A" is the FFT result for that time window step. From the previous discussion, the FFT/DFT will result complex numbers. So "A" is a complex number. In order to find the spectrum magnitude (height of the spectrum) of FFT/DFT, it needs to take the absolute value, $|A|$. Assuming the returned matrix is an M×N matrix, when comparing the spectrums between the third recorded speech sginal and the first two recorded reference signals, it is viewed from the frequency axis (the number of rows M). For one single frequency (single row), this row's vector not only contains one element. It means the row's elements will all have their own spectrum contributions for different times section at this single frequency (at this number of row). So viewing from the frequency axis, it will show that N values' ploting at this frequency or N peaks overlapped at this frequency. So when plotting for the whole speech frequency band, the spectrum is actually N overlapped spectrums. Run the program code of this thesis in MATLAB, it will show the speech spectrums for three recordings as bellow:

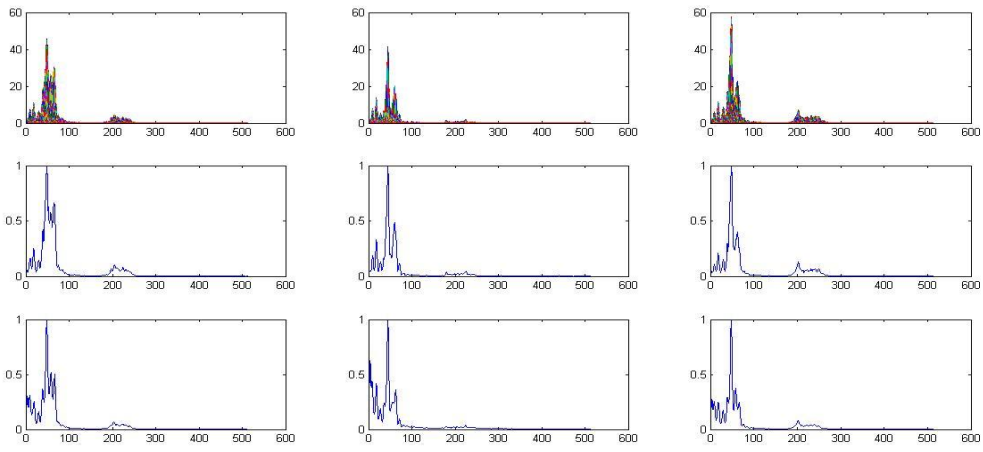

Figure 19: The spectrum viewed from "frequency axis"

From Fig.19, the graphs of the first row are directly plotted by the absolute values of the matrix. The graphs of the second row are plotted by taking the maximum value for each row vector of the matrix to catch the contour profile of the first row's spectrums plotted in the figure. The graphs of the third row are plotted by taking the summation of each row's elements. The first row graphs and the second row graphs as shown are not exactly the representations for the real frequency spectrum. Since they are just the maximum value of each frequency, so the information of spectrums is just for the moment when the magnitude of spectrum is maximum. By taking the summation calculation of each row, the information of spectrums is for the whole time sections and the noise effect will be reduced. So the third row graphs are the real spectrums' representations. From Fig.19, the differences between the third row graphs and the other two rows' graphs are not obvious when plotted in spectrums. But the obvious differences can be viewed when plotting the signals in time domain. See the figure as below:

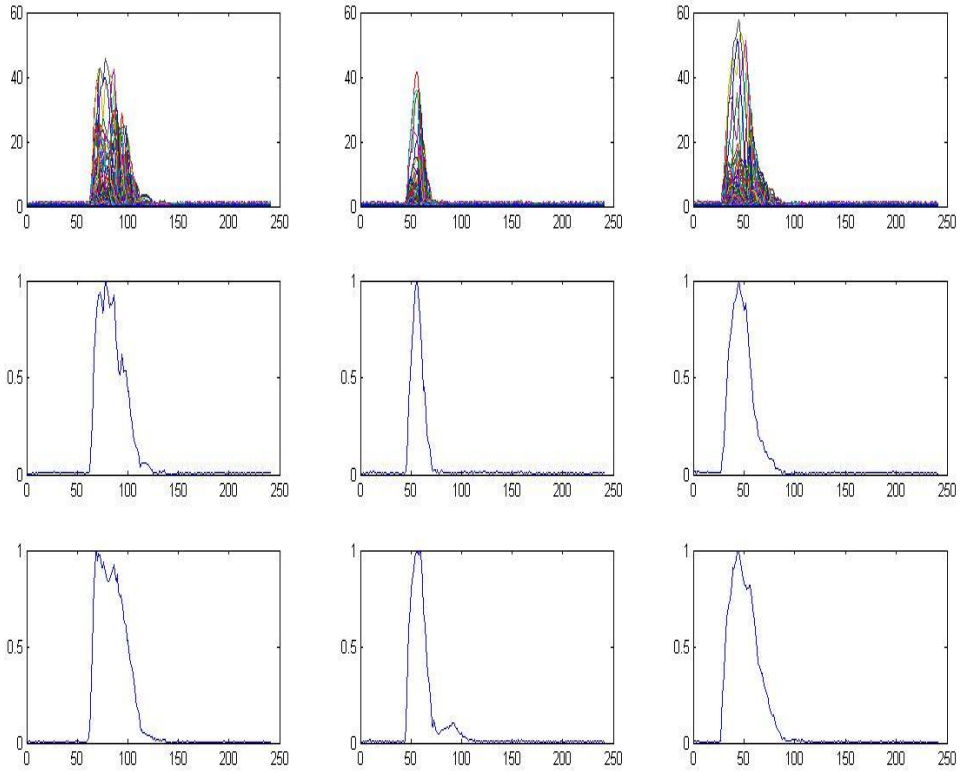

Figure 20: The speech signals viwed from "time axis"

From Fig.20, compare the graphs in the second column. There is a ripple in the third row at about time section 100. But we can't see this ripple from graphs plotted in the first and second row. This is due to the noise level higher than this ripple of the voice signal. By taking the summation operation, the ripples of signal will come out from the noise floor. After the linear normalization, this result will be clearer. So when comparing the signals' spectrums, it needs to compare the summation spectrums, which are more accuracy and reliable.

Chapter 3

Programming steps and Simulation Results

In this thesis there are two designed systems (two m files of MATLAB) for speech recognition. Both of these two systems utilized the knowledge according to the Theory part of this thesis which has been introduced previously. The author invited his friends to help to test two designed systems. For running the system codes at each time in MATLAB, MATLAB will ask the operator to record the speech signals for three times. The first two recordings are used as reference signals. The third time recording is used as the target signal. The corresponding codes for both systems can be found in Appendix.

3.1 Programming Steps

3.1.1 Programming Steps for Designed System 1

(1) Initialize the variables and set the sampling frequency fs=16000.
Use “wavrecord” command to record 3 voice signals. Make the first two recordings as the reference signals. Make the third voice recording as the target voice signal.

(2) Use “spectrogram” function to process recorded signals and get returned matrix signals.

(3) Transpose the matrix signals for rows and columns, take “sum” operation of the matrix and get a returned row vector for each column summation result. This row vector is the frequency spectrum signal.

(4) Normalize the frequency spectrums by the linear normalization.

(5) Do the cross-correlations for the third recorded signal with the first two recorded reference signals separately.

(6) This step is important since the comparison algorithm is programed here. Firstly, check the frequency shift of the cross-correlations. Here it has to be announced that the frequency shift is not the real frequency shift. It is processed frequency in MATLAB. By the definition of the spectrum for the "nfft", which is the length of the STFT programmed in MATLAB, the function will return a frequency range which is respect to the "nfft". If "nfft" is odd, so the returned matrix has $\frac{nfft+1}{2}$ rows; if "nfft" is even, then the returned matrix has $\frac{nfft}{2}+1$ rows. These are defined in MATLAB. Rows of the returned "spectrogram" matrix are still the frequency ranges. If the difference between the absolute values of frequency shifts for the two cross-correlations is larger or equal than 2, then the system will give the judgment only by the frequency shift. The smaller frequency shift means the better match. If the difference between the absolute values of frequency shifts is smaller than 2, then the frequency shift difference is useless according to the experience by large amounts tests. The system needs continuously do the comparison by the symmetric property for the cross-correlations of the matched signals. The algorithm of symmetric property has been introduced in the part of 2.3.2. According to the symmetric property, MATLAB will give the judgment.

3.1.2 Programming Steps for Designed System 2

(1) Initialize the variables and set the sampling frequency fs=16000.

(2) Use "wavrecord" to record 3 voice signals. Make the first two recordings as the reference signals. Make the third voice recording as the target voice signal.

(3) Use "spectrogram" function to process recorded signals and get the returned matrix signals.

(4) Transpose the matrix signals for rows and columns, and take "sum" operation of the matrix and get a returned row vector for each column summation result. This row vector represents the frequency spectrum.

(5) Normalize the frequency spectrums by the linear normalization.

(6) From this step, the system will be different to system 1. The system is programed for the winner filter mode. Firstly calculate the auto-correlations of 3 signals: the first two recorded reference signals and the third recorded target signal. Secondly, set the total order number is 20. And use a "for" loop to detect each order result. For certain filter order, define the auto-correlation length from N to N+p. By the definition of the Wiener filter equation (36), the sizes or the lengths of the auto-correlation matrix and auto-correlation vector both should be p. Since the position N is the maximum value position of the auto-correlation, so "r(N)=r(0)". To be more clearly, this is explained the part 2.3.2, which introduced the relation between the maximum value and the position for the cross-correlation. After defining R_x and r_{dx}, the next step is directly to calculate the filter coefficients for each reference signal.

(7) After finding the filter coefficients for each reference signal, calculate the minimum mean square-error for each reference signal. Compare the mean value of the minimum mean square-errors for the order range from 0 to 20. The better estimation should have the smaller minimum mean square-errors. The theory of the Wiener filter has been introduced in the part of 2.3.4.

3.2 Simulation Results

In this section, the author simulated two designed systems within the help of friends who are from different countries. As the thesis introduced previously, the only task of operator is to run the program and record three speech signals. The first two recordings are used as reference signals. The third recording is used as the target signal for which MATALB should give the judgment. In the following results, the author uses "reference signals" to stand for the first two recordings and uses "target signal" to stand for the third recording. The words in the quotes stand for the contents of recordings. The author tried to test designed systems for both easily recognized words and difficultly recognized words. "From time 1 to time 10, 'on'" in the following of the thesis means the operator simulated 10 times and the third recording

word is "on" in the first 10 times simulations. Both the contents of the reference words and the target word are known, the author wants to test if the judgment that is given by MATLAB is correct as we known. The statistical simulation results will be put in tables and will also be plotted. In this Simulation Result part, only the plotted results will be shown in the following content. The related result tables will be given in Appendix B. Since the author programmed in MATLAB to plot figures for each system to help the analysis when simulating at each time, and the resulting figures for each system are got by the same principles, so the author will only put the simulation figure once time at the beginning of simulation results for each system.

3.2.1 The Simulation Results for System 1

(1) The information of the first statistical simulation results for system 1 is as following:

Reference signals: "on" and "off":
Target signal: From time 1 to time 10, "on".
From time 11 to time 20, "off".
Speaker: Tingxiao Yang (from China) for both reference signals and the target signal.
Environment around: almost no noise

The figure 21 is about frequency spectrums for three recorded signals, but the axis is not the real frequency axis since the figure is got by STFT. The information of STFT can be found in the part 2.3.5. The method of getting this figure can be also found in the part 2.3.5.

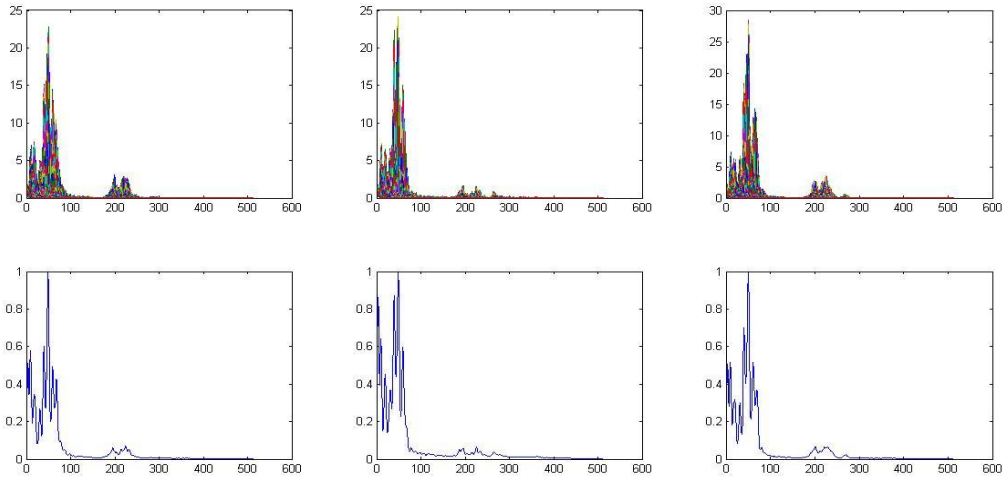

Figure 21: Frequency spectrums for three speech signals: "on", "off", "on"

The figure of cross-correlations between the target signal "on" and the reference signals is as below (the reference signal of the left plotting is "on"; the reference signal of the right plotting is "off"):

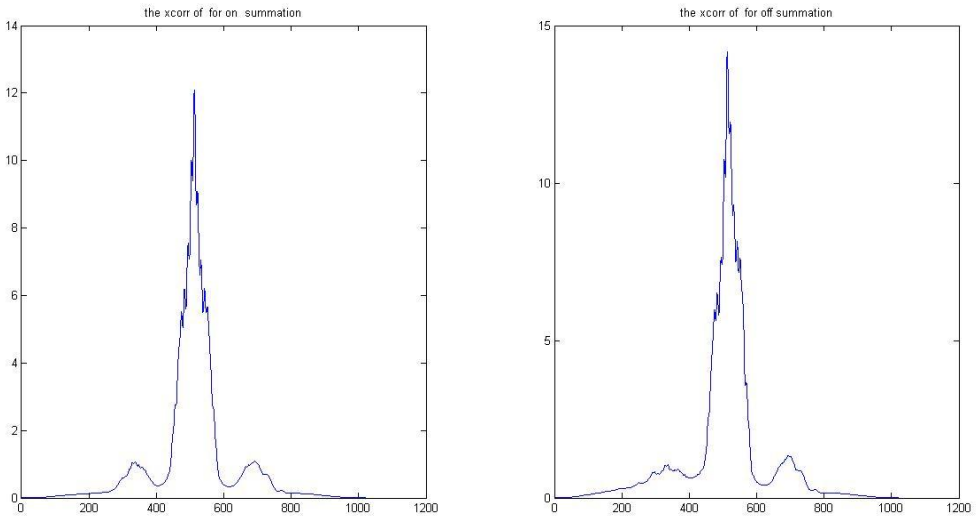

Figure 22: Cross-correlations between the target signal "on" and reference signals

As Fig.22 shown above, there is no big difference between two graphs, since the pronunciations of "on" and "off" are close.

According to the simulation results (Table 1, Appendix B), the plotted simulation result for frequency shift is as below:

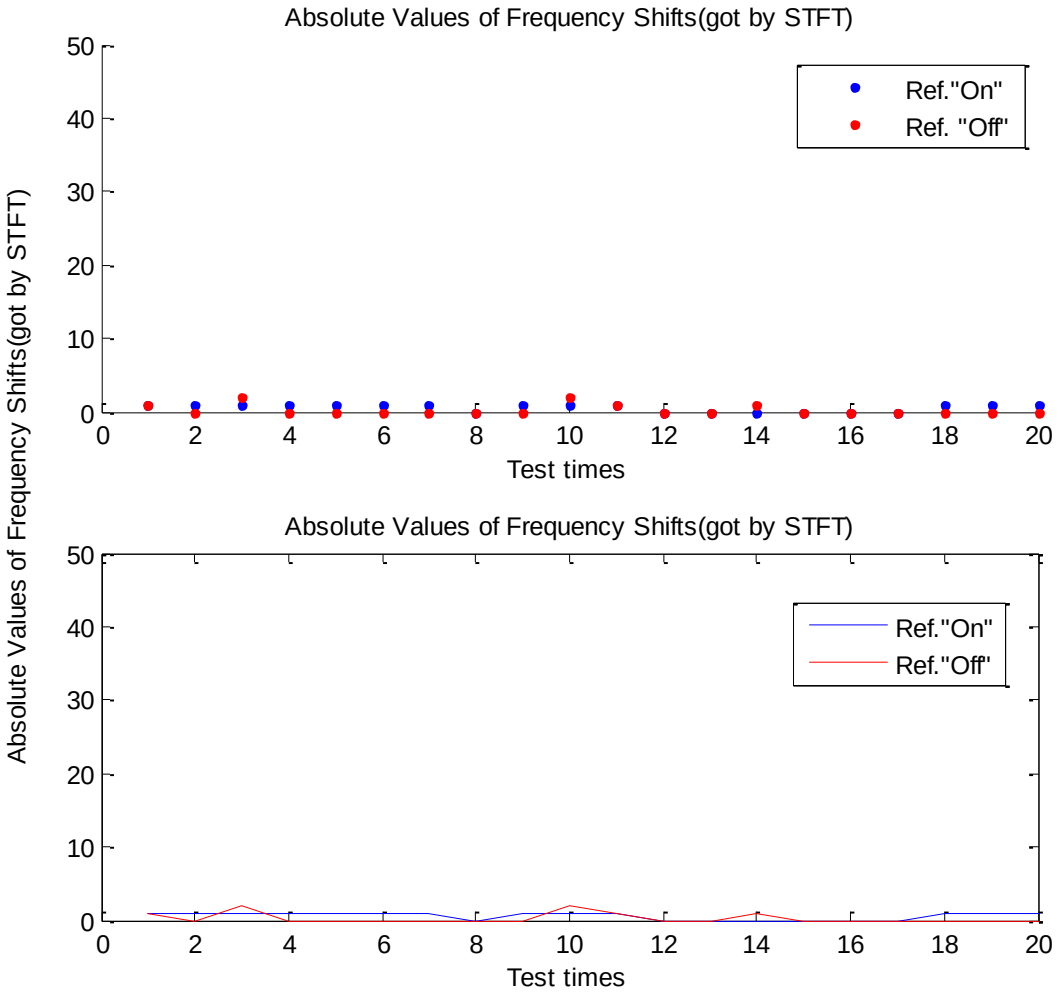

Figure 23: Frequency shifts in 20 times simulations for reference "on" and "off"

From Fig.23, it has shown that it is hard to give the judgments with frequency shifts. The frequency shifts are very close between the speech words "on" and "off". So the designed system will give the judgments according to the symmetric errors. The plotted simulation result for symmetric errors is as below:

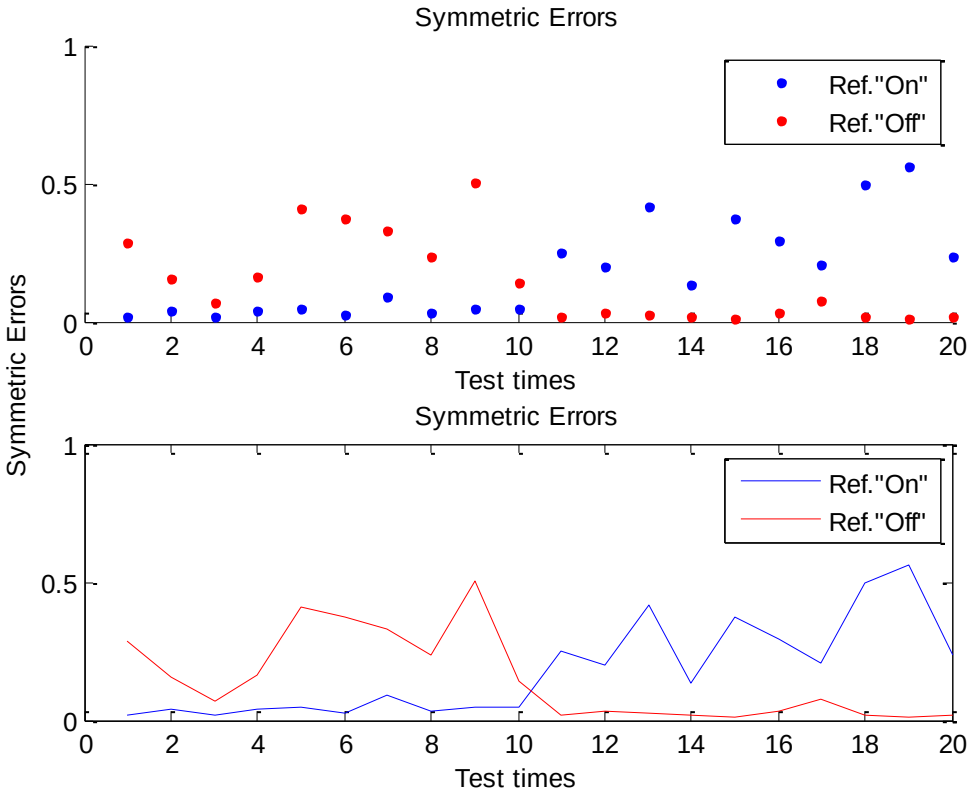

Figure 24: Symmetric errors in 20 times simulations for reference "on" and "off"

As shown in Fig. 24, the blue curve is simulated result when the reference speech word is "on". The red curve is the simulated result when the reference speech word is "off". As information given at the beginning, the target speech word is "on" in the first 10 times simulations and the target speech word is "off" in the second 10 times simulations. From Fig.24, it is shown that in the first 10 times simulations the reference "on" curve has lower value and in the second 10 times the reference "off" curve has lower value. The results have shown that when the reference speech signal and the target speech signal are matched, the symmetric errors are smaller. The judgments are totally correct.

(2) The information of the second statistical simulation results for system 1 is as following:

Reference signals: "Door" and "Key":
Target signal: From time 1 to time 10, "Door".
From time 11 to time 20, "Key".
Speaker: Tingxiao Yang (from China) for both reference signals and the target signal.
Environment around: almost no noise

The figure 25 about frequency spectrums for three recorded signals is got by the same way as the figure 21.

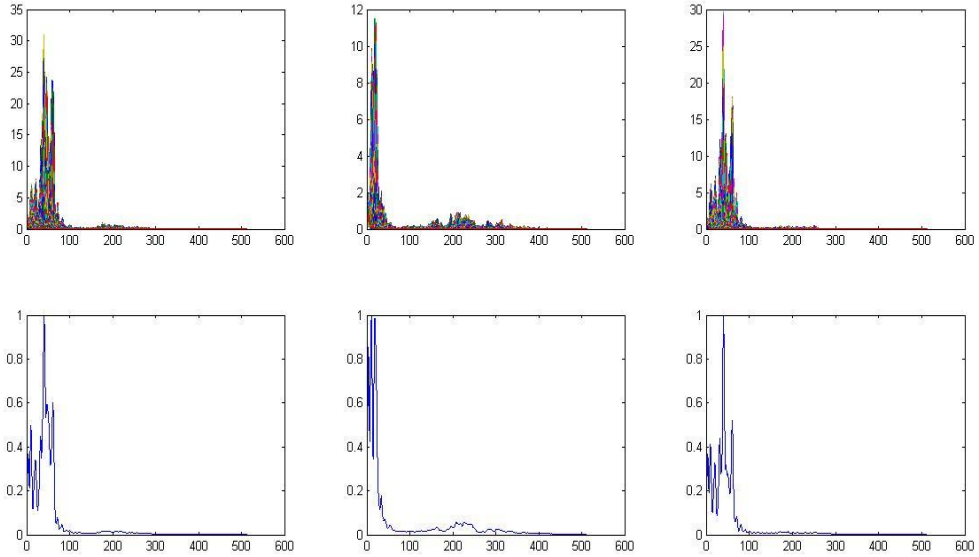

Figure 25: Frequency spectrums for three signals: “Door”, “Key”, and “Door”

The figure of cross-correlations for the target signal “Door” with reference signals is as below (the reference signal of the left plotting is “Door”; the reference signal of the right plotting is “Key”):

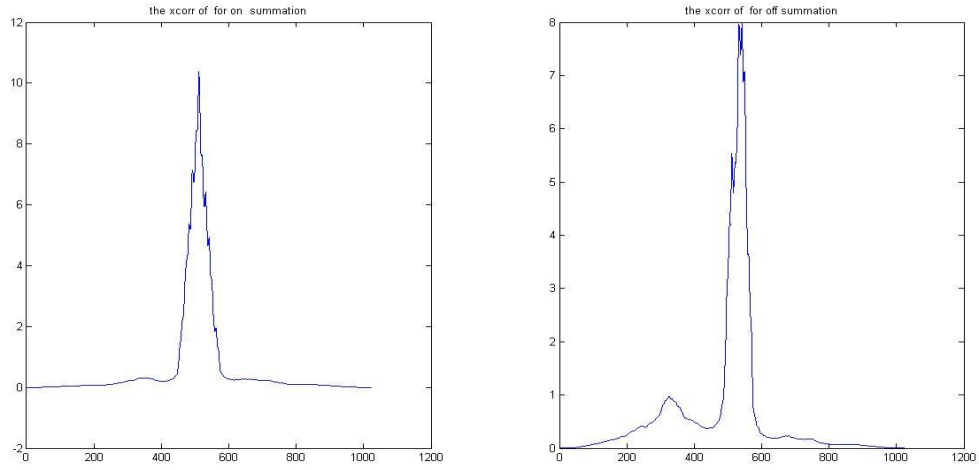

Figure 26: Cross-correlations for the target signal “Door” with reference signals

As Fig.26 shown above, there is large difference between two graphs. Since the pronunciations of "Door" and "Key" are totally different. As introduced in theory part 2.3.2, the better matched signals have better symmetric property of the cross-correlation. The Fig.26 approved this point.

According to the simulation results (Table 2, Appendix B), the plotted simulation result for frequency shift is as below:

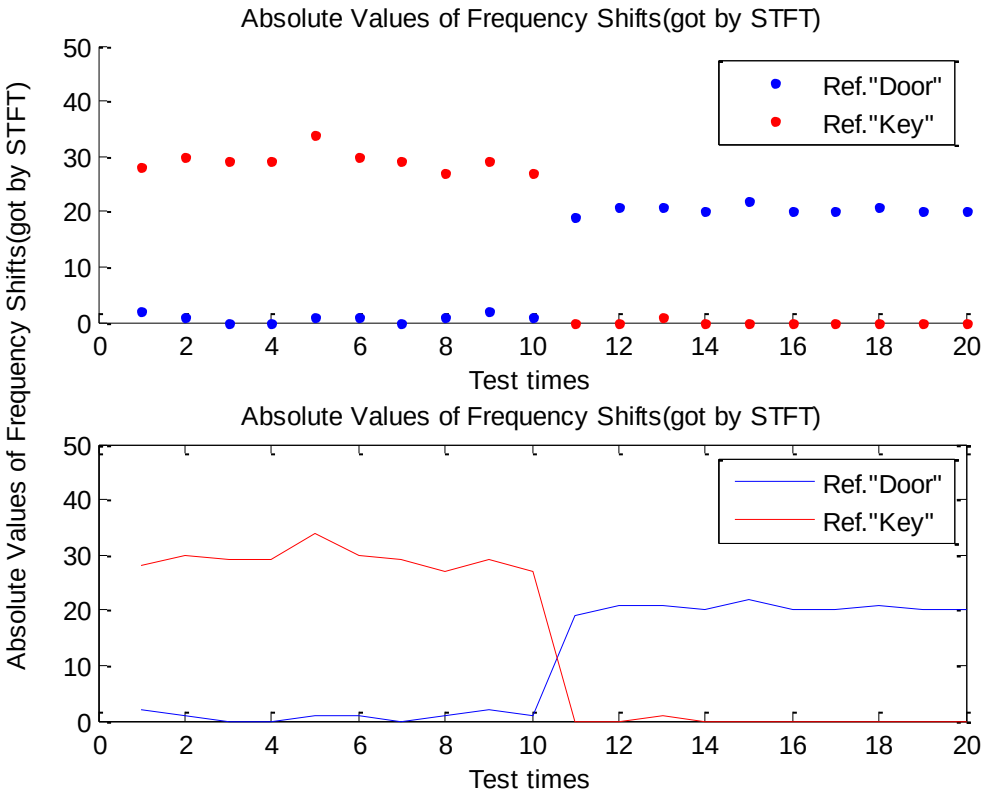

Figure 27: Frequency shits in 20 times simulations for reference "Door" and "Key"

From Fig.27, it can be seen that the frequency shifts have large differences. So the designed system will directly give the judgments according to the frequency shifts.

(3) The information of the third statistical simulation results for system 1 is as following:

Reference signals: "on" and "off":

Target signal: From time 1 to time 10, "on".

From time 11 to time 20, "off".

Speaker: Marcus.Eliasson (from Sweden) for both reference signals and the target signal.

Environment around: there is some noise sometimes

Since "on" and "off" have small frequency shifts' difference (Fig.23 and Table 1), so the designed system will only give the judgments with symmetric errors. The plotted simulation result (data is given in Table 3, Appendix B) is as below:

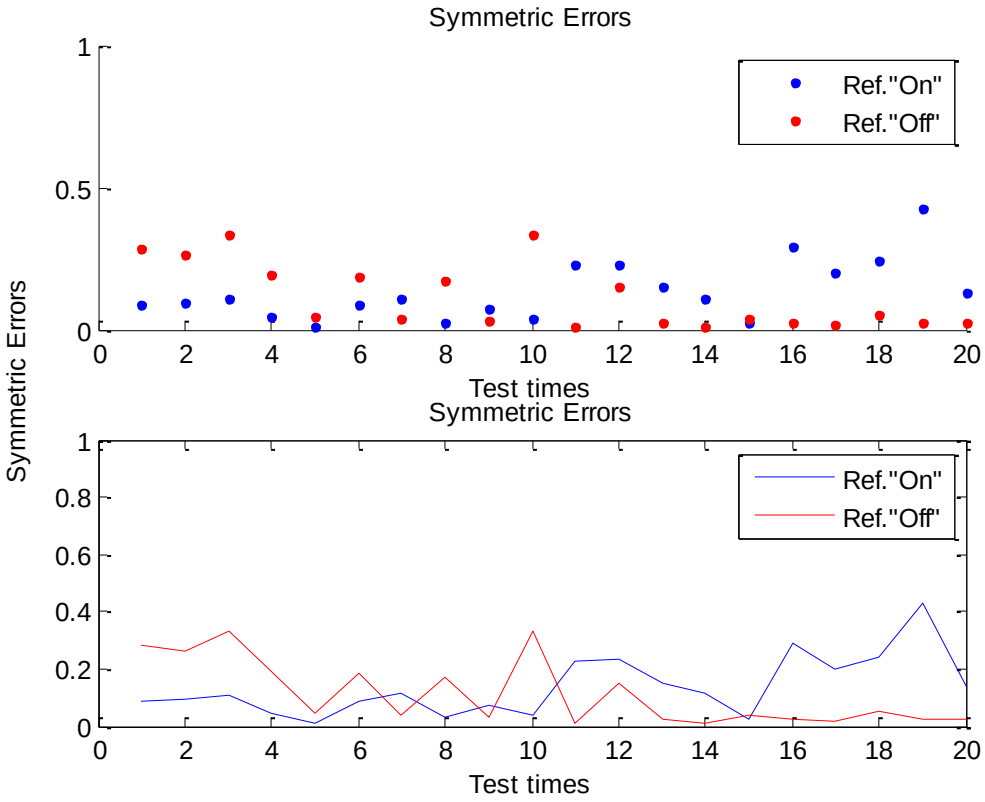

Figure 28: Symmetric errors in 20 times simulations for reference "on" and "off" (noisy)

(4) The information of the fourth statistical simulation results for system 1 is as following:

Reference signals: "Door" and "Key":
Target signal: From time 1 to time 10, "Door".
From time 11 to time 20, "Key".
Speaker: Marcus.Eliasson (from Sweden) for both reference signals and the target signal.
Environment around: there is still some noise sometimes

The plotted simulation result (data is given in Table 4, Appendix B) is as below:

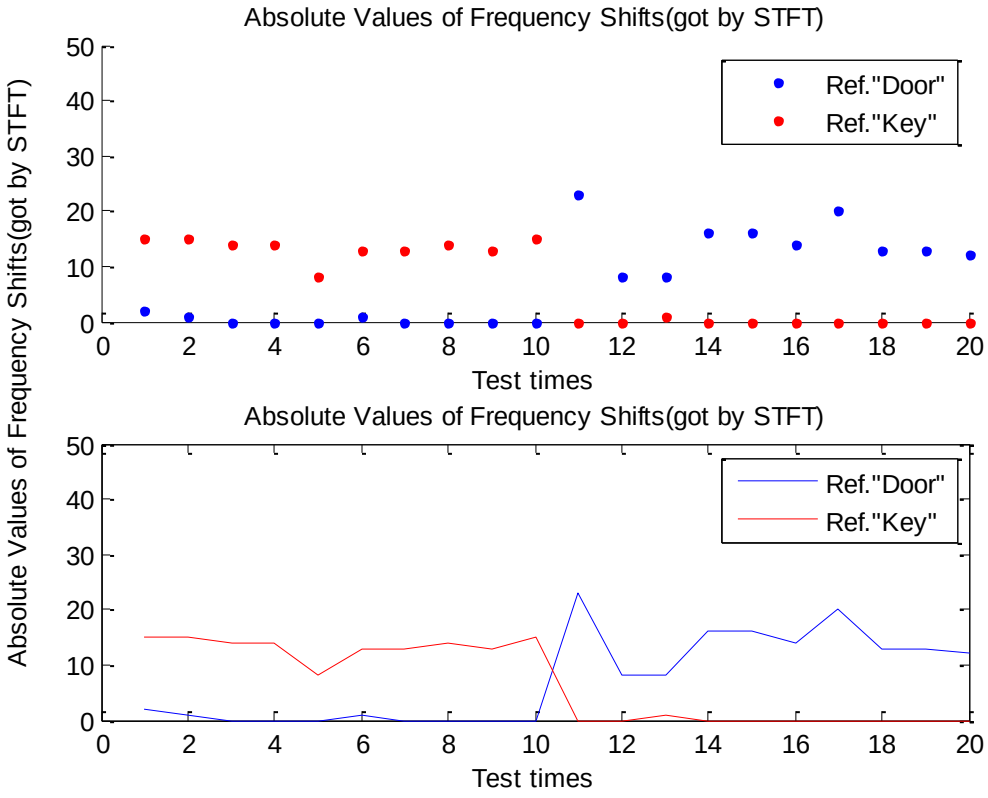

Figure 29: Frequency shits in 20 times simulations for reference "Door" and "Key" (noisy)

(5) The information of the fifth statistical simulation results for system 1is as following:

Reference signals: From time 1 to time 10:"on" and "off"

From time 11 to time 20:"Door" and "Key"

Target signal: From time 1 to time 5, "on"

From time 6 to time 10, "off"

From time 11 to time 15, "Door"

From time 16 to time 20, "Key"

Speakers: Marcus.Eliasson(from Sweden) for reference signals

Tingxiao Yang (from China) for the target signal

Environment around: almost no noise

Notice: The reference signals and the target signal are recorded by the different persons.

As mentioned in part 3.1.1, when the system 1 can give the judgment with frequency shift, then the system will not calculate the symmetric errors. When the system 1 cannot give the

judgment with frequency shifts, then the system will calculate the symmetric errors. According to this principle, the author got the simulation result as below:

Table 5 indicates the simulation results for reference signals "Door" and "Key" as the information given at the beginning of this section (5).

Test times	frequency_on_shift	frequency_off_shift	Error1	Error2	**Final judgments**
1	2	8	No need	No need	on
2	7	8	0.2055	0.4324	on
3	8	9	0.2578	0.2573	off
4	9	17	No need	No need	on
5	8	9	0.2304	0.3640	on
6	0	0	0.3268	0.6311	on
7	0	0	0.3193	0.3210	on
8	0	0	2.2153	0.9354	off
9	0	0	0.4603	0.1481	off
10	0	0	0.1189	0.0741	off
11	8	22	No need	No need	Door
12	8	0	No need	No need	Key
13	8	25	No need	No need	Door
14	8	24	No need	No need	Door
15	8	24	No need	No need	Door
16	-15	0	No need	No need	Key
17	-15	0	No need	No need	Key
18	-14	0	No need	No need	Key
19	-14	0	No need	No need	Key
20	-15	0	No need	No need	Key
Total successful probability(total in 20 times)			**80%**		

Table 5: Simulation results for speech words "On", "Off", "Door" and "Key"

Since the simulation results are not good as the expected values. So only the table results are shown here in case the plotted result is too confusing.

3.2.2 The Simulation Results for System 2

(1) The information of the first statistical simulation results for system 2 is as following:

Reference signals: "on" and "off":

Target signal: From time 1 to time 10, "on".

From time 11 to time 20, "off".

Speaker: Tingxiao Yang (from China) for both reference signals and the target signal.

Environment around: almost no noise

The figure of the minimum mean square-errors between the target signals and reference signals is as below (data is given in Table 6, Appendix B):

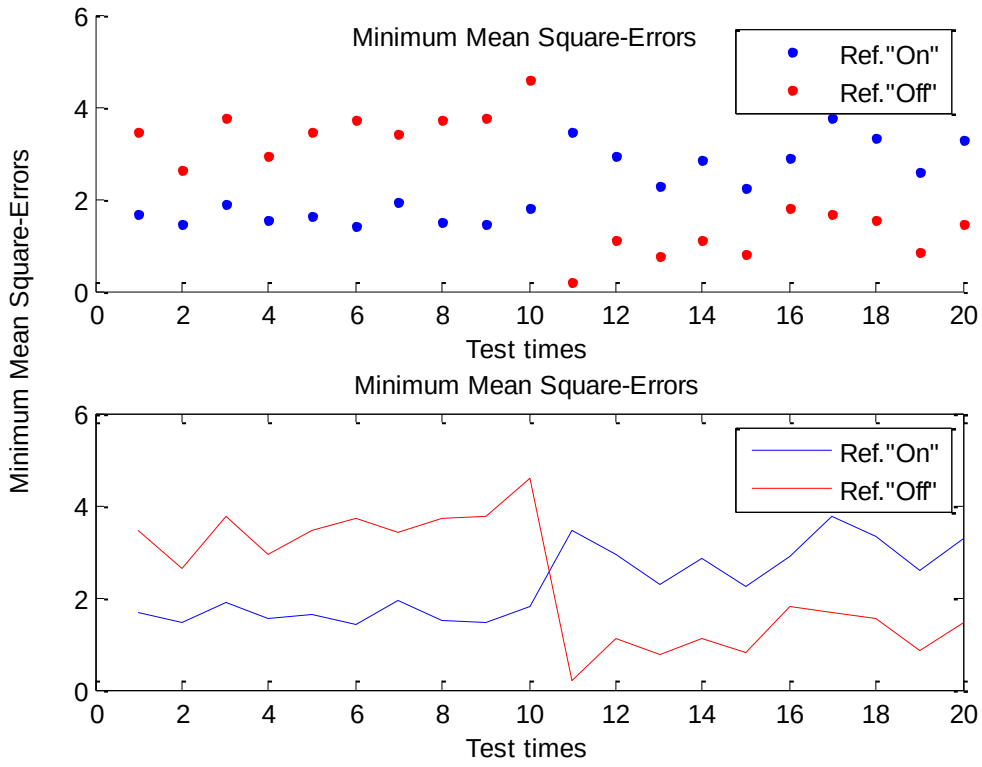

Figure 30: Minimum mean square-errors for the target signals with reference signals

.

(2) The information of the second statistical simulation results for system 2 is as following:

Reference signals: "Door" and "Key":

Target signal: From time 1 to time 10, "Door"

From time 11 to time 20, “Key”

Speaker: Tingxiao Yang (from China) for both reference signals and the target signal

Environment around: almost no noise

The figure of the minimum mean square-errors for target signals with reference signals is as below (data is given in Table 7, Appendix B):

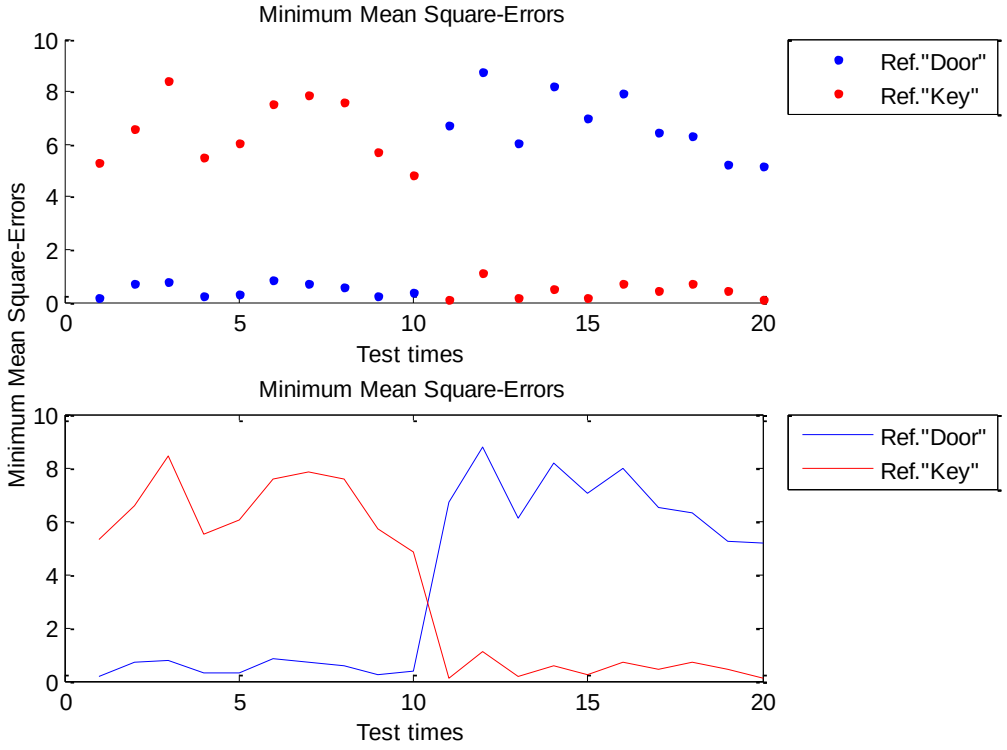

Figure 31: Minimum mean square-errors for the target signals with reference signals

(3) The information of the third statistical simulation results for system 2 is as following:

Reference signals: “on” and “off”

Target signal: From time 1 to time 10, the voice is “on”

From time 11 to time 20, the voice is “off”

Speaker: Babak.Kazemi (from Iran) for both reference signals and the target signal

Environment around: almost no noise

The figure of the minimum mean square-errors for target signals with reference signals is as below (data is given in Table 8, Appendix B):

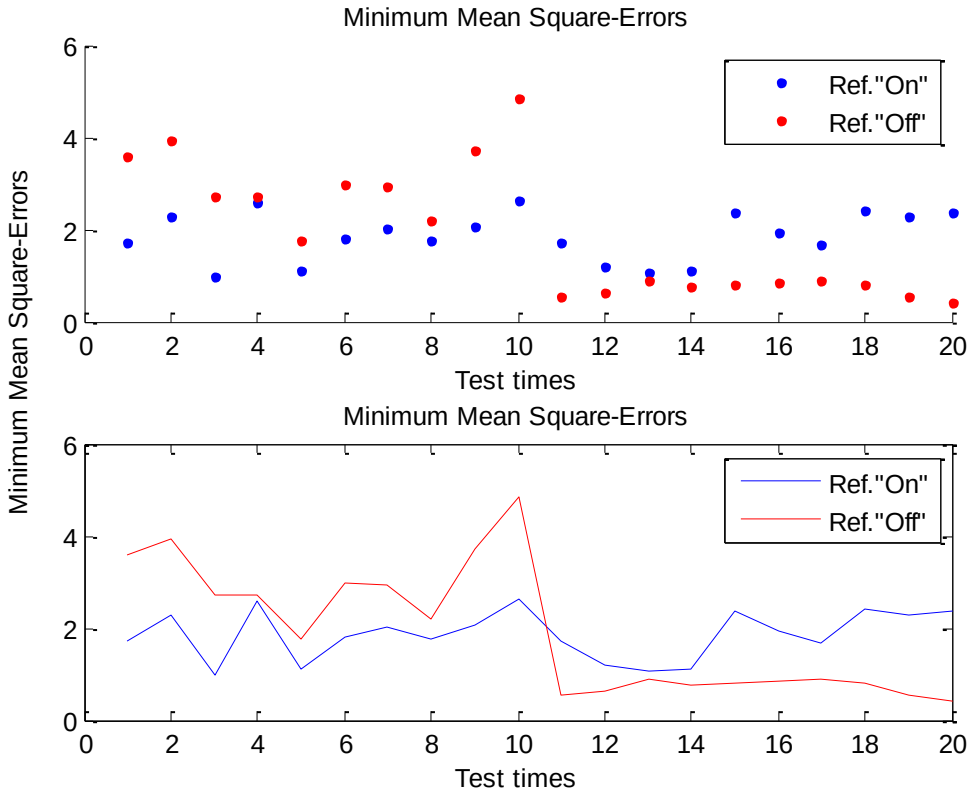

Figure 32: Minimum mean square-errors for the target signals with reference signals

(4) The information of the fourth statistical simulation results for system 2 is as following:

Reference signals: “Door” and “Key”:

Target signal: From time 1 to time 10, the voice is “Door”.

From time 11 to time 20, the voice is “Key”.

Speaker: Babak.Kazemi (from Iran) for both reference signals and the target signal.

Environment around: almost no noise

The figure of the minimum mean square-errors for target signals with reference signals is as below (data is given in Table 9, Appendix B):

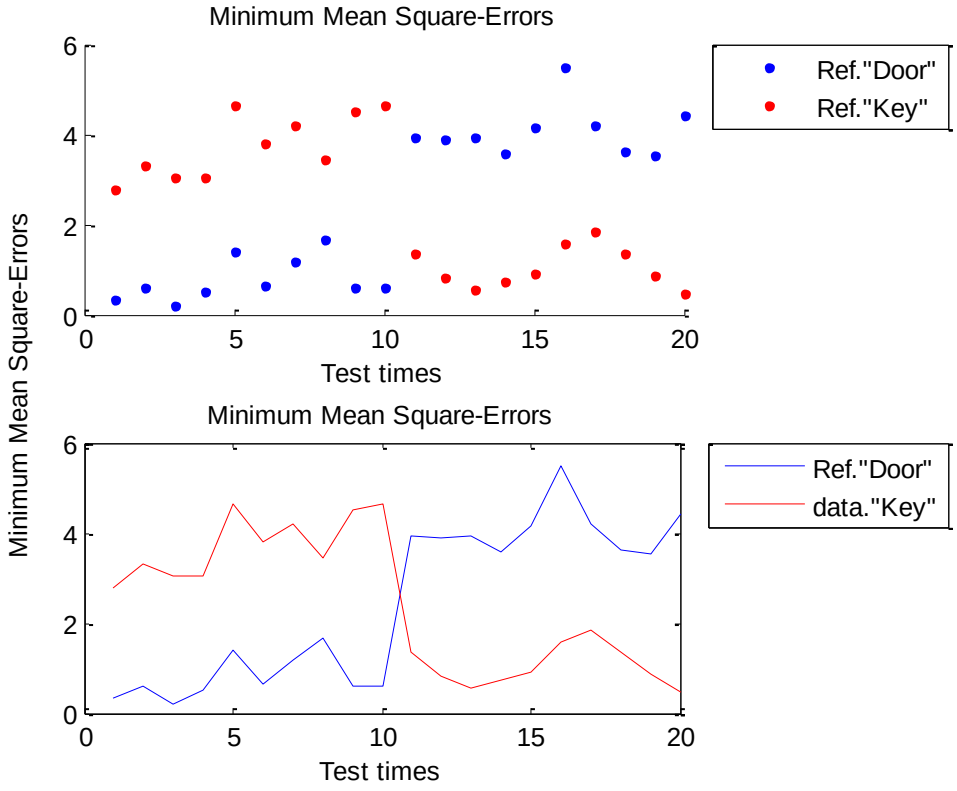

Figure 33: Minimum mean square-errors for the target signals with reference signals

(5) The information of the fifth statistical simulation results for system 2 is as following:

Reference signals: “on” and “off”

Target signal: From time 1 to time 10, the voice is “on”

From time 11 to time 20, the voice is “off”

Speakers: Babak.Kazemi (from Iran) for the target signal

Tingxiao Yang (me:from China) for reference signals

Notice: The reference signals and the target signal are recorded by the different persons.

The figure of the minimum mean square-errors for target signals with reference signals is as below (data is given in Table 10, Appendix B):

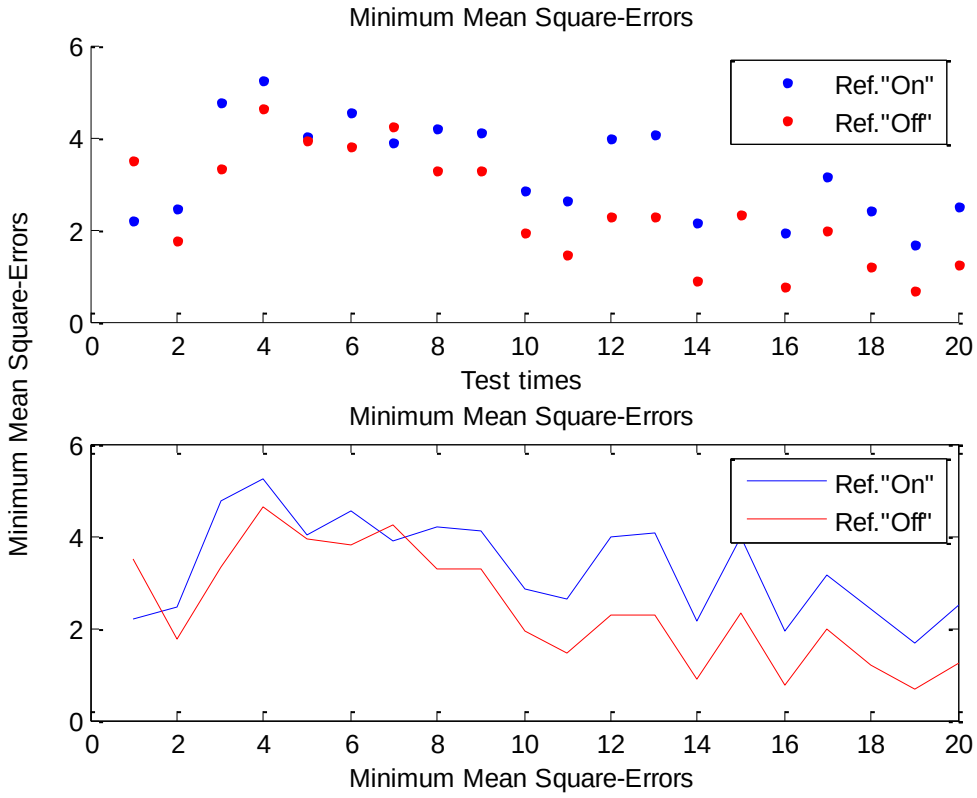

Figure 34: Minimum mean square-errors for the target signals with reference signals

Chapter 4

Discussion and Conclusions

In this section, the author will discuss what have done in the result section and analyze the simulation results that are shown in previous chapter. At the end, according to the discussions of the simulation results, the author will give the related conclusions.

4.1 Discussion

4.1.1 Discussion about The Simulation Results for The designed System 1

There are 5 tables for the result figures in the previous chapter for the designed system 1. Table 1 and Table 2 are simulated by the author in order to see how the designed system works for the different pairs of words, the easily recognized words and the difficultly recognized words. From Table 1(Fig.23), it can be observed that the frequency shift is not reliable, since the pronunciations of "on" and "off" are really closed. At this situation, the designed system will give the judgments by comparing the errors of the symmetric property of the cross-correlations (Fig.24). The principle of this recognition method is introduced in the part 2.3.2 of this thesis. If the pronunciations of two words are totally different, such as "Door" and "Key", the designed system 1 will make the judgments directly by the frequency shifts, which we can observe from Table 2(Fig.27). The designed system will not calculate the errors anymore. By observing large amounts of simulation results, the author programmed the system in MATLAB to rely frequency shifts when the difference between the absolute values of frequency shifts for the different reference signals is larger or equal to 2. Otherwise, the designed system 1 will continuously calculate the errors of the symmetry. Table 3(Fig.28) and Table 4(Fig.29) are simulated by the author's friends who are from different countries. The purpose of these results is to show that the designed system actually works for the different people who are from different countries. And from the Table 3(Fig.28) and Table 4(Fig.29), it can be seen that the designed system didn't work that well when there is noise around. This shows that the designed system is easy to be disturbed by the noise. The Table 5 results are

simulated by two people. One is responsible for the reference signals' recordings, the other one is for the target signal's recordings. It can be observed that the designed system almost doesn't work for this situation.

4.1.2 Discussion about the Simulation Results for the Designed System 2

There are also 5 tables of simulation results for the designed system 2. The purposes of these simulations are the same as the designed purposes for the system1. The system 1 is designed by observing large amounts of the plots of the cross-correlations. The system 2 is designed by using the reference signals to model the target signal. By comparing the errors between the real target signal and the modeling target signal got from the different reference signals, the system 2 gives the judgment that which reference signal is more similar to the target signal. The author designed Wiener filter to realize this signal modeling process. In this system, the reference signals are used as the auto-correlation sources, which are the inputs of the Wiener filter. And the target signal is used as the desired signal. By the Wiener filter equation $R_x w = r_{dx}$, it can be known that when applying this equation, it actually gives the assumption that the input signal x (n) is correlated to the desired signal d (n). In other words, the reference signals should be correlated to the target signal. But if one person gives reference signals and the other one person gives the target signal, then the reference signals and the target signal are not correlated to each other. So the designed system 2 doesn't work well when the reference signals and the target signal are recorded by different people.

4.2 Conclusions

For general conclusions, the designed systems for speech recognition are easily disturbed by the noise, which can be observed from Table 3(Fig.28). For the designed system 1, the better matched signals have the better symmetric property of their cross-correlation. This conclusion can be got from Table 1(Fig.23 and Fig.24) and Table 2(Fig.27).For the designed system 2, if the reference signal is the same word as the target signal, so using this reference signal to model the target will have less errors. This conclusion can be proved by the all the simulation results for the designed system 2. When both reference signals and the target signal are

recorded by the same person, two systems work well for distinguishing different words, no matter where this person is from. But if the reference signals and the target signal are recorded by the different people, both systems don't work that well. So in order to improve the designed systems to make it work better, the further tasks are to enhance the systems' noise immunity and to find the common characteristics of the speech for the different people. Otherwise, to design some analog and digital filters for processing the input signals can reduce the effects of the input noise and to establish the large data base of the speech signals for different words. And studying more advanced algorithms for signal modeling [9] can give a lot of helps to realize the better speech recognition.

References

[1] Joseph Picone, "Signal Modeling Techniques In Speech", Systems and Information Sciences Laboratory, Tsukuba Research and Development Center, Tsukuba,Japan.

[2] John G.Proakis, Dimitris G.Manolakis, Digital Signal Processing, Principles, Algorithms, and Applications, 4th edition,Pearson Education inc., Upper Saddle River.

[3] Luis Buera, Antonio Miguel, Eduardo Lleida, Oscar Saz, Alfonso Oretega, "Robust Speech Recognition with On-line Unsupervised Acoustic Feature Compensation", Communication Technologies Group (GTC),13A, University of Zaragoza, Spain.

[4] Hartmut Traunmüller , Anders Eriksson, "The frequency range of the voice fundamental in the speech of male and female adults", Institutionen för lingvistik, Stockholms universitet, S-106 91 Stockholm, Sweden.

[5] Jian Chen, Jiwan Gupta,"Estimation of shift parameter of headway distributions using crosscorrelation function method", Department of Civil Engineering, The University of Toledo.

[6] John Wiley, Sons,Inc. Statistical Signal Processing And Modeling, Monson H Hayes,Georigia I nstitute of Technology.

[7] Henrik V. Sorensen, C. Sidney Burrus, "Efficient Computation of the Short-Time Fast Fourier Transform", Electrical and Computer Engineering Department, Rice University, Houston.

[8] Fredric J. Harris,Member, IEEE, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier transform", Proceedings of the IEEE, VOL 66, No.1, JANUARY, 1978

[9] Joseph W. Picone, senior member, IEEE "Signal Modeling Techniques in Speech" Recognition

Appendix A

The whole program code of MATLAB is as below:

System 1 codes:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%initial vaules
%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
on=0;
off=0;
fs=16000;          %sampling frequency, in 1 second  take 16000 samples
test=0;
duration=2;        %recording time
%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Press any key to start %g seconds of recording.on.\n',duration);

pause;

fprintf('Recording...\n');

r=wavrecord(2*fs,fs);
%duration*fs: the length of the recorded data: take 2*fs samples need 2 seconds

r=r-mean(r);


fprintf('Press any key to start %g seconds of recording. off.\n',duration);

pause;

fprintf('Recording...\n');

y=wavrecord(2*fs,fs);

fprintf('Press any key to start %g seconds of recording. voice.\n',duration);



y=y-mean(y);
pause;

fprintf('Recording...\n');

voice=wavrecord(2*fs,fs);
voice=voice-mean(voice);
fprintf('Finished recording.\n');
```

```
%%%%%%%%%%%%% use specgram to get the signals spectrom information in
%%%%%%%%%%%%% frequency domain

nfft = min(1023,length(r));  %%%%%%%define our the length of STFT
% define window length and the overlap to decide the window step
s=specgram(r, nfft, fs, hanning(511),380);
s2=specgram(y, nfft, fs, hanning(511),380);
s3=specgram(voice, nfft, fs, hanning(511),380);


%%%% PART 1....frequency information
%%%% FREQUENCY  SPECTRUM %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% s is complex matrix
absolute=transpose(abs(s)); %%%%take abs make it real and ease to plot
absolute2=transpose(abs(s2));
absolute3=transpose(abs(s3));


%%%% FREQUENCY  SPECTRUM %%%%%%%%%%%%%%%%%% for the
frequency spectrum ,
%If A is a matrix, sum(A) treats the columns of A as vectors,
%returning a row vector of the sums of each column
% after transpose, the rows and columns has been swapt,
% take the summation of transposed matrix ,we get summation the time axis
% to return a frequency spectrum
a4=sum(absolute)  %%%% get time-freuqency related spectrum
a5=sum(absolute2)
a6=sum(absolute3)
%%% %FREQUENCY  SPECTRUM %%%%%%%%%%%%%
normalize%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% the spectrom and also decrease the noise effect%%

a4_norm=(a4-min(a4))/(max(a4)-min(a4));
a5_norm=(a5-min(a5))/(max(a5)-min(a5));
a6_norm=(a6-min(a6))/(max(a6)-min(a6));
%%%%%%%%%%%%%%%%%%%%%%%%transpose row to colume
vector%%%%%%%%%%%

F4=transpose(a4_norm); % just always used to deal with vector data
F5=transpose(a5_norm);
F6=transpose(a6_norm);
%%%%%%%%%FREQUENCY  SPECTRUM :compare
crosscorrelation%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[x3,lag3]=xcorr(F6,F4);
[mx3,indice3]=max(x3);
frequency_on_shift=lag3(indice3)

[x4,lag4]=xcorr(F6,F5);
```

```
[mx4,indice4]=max(x4);
frequency_off_shift=lag4(indice4)



%%%%%%%%%%%%%%%%%%%%%%%%%%%% frequency domain spectrum plot
%%%%%%%%
figure(1)% 2x3 matrix ploting ,first row ploting the original spectrum
        % second row ploting summated real spectrum
subplot(2,3,1)
plot(abs(s))
subplot (2,3,2)
plot(abs(s2))
subplot (2,3,3)
plot(abs(s3))
subplot(2,3,4)

plot(F4);
subplot(2,3,5)
plot(F5);
subplot(2,3,6)
plot(F6);

figure(2)
subplot(1,2,1)
plot(x3)
title('the xcorr of  for on  summation');
subplot(1,2,2 )
plot(x4)
title('the xcorr of  for off summation');



%%%%%%%%%%% test 6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% frequency spectrum compare
% firstly check about about the frequency shift
% I only trust the juedement when freqeuencyshift difference is larger or
% equal to 2.
% The smaller of the frequency shift, the better match signals
if abs(abs(frequency_on_shift)-abs(frequency_off_shift))>=2
   if abs(frequency_on_shift)>abs(frequency_off_shift)
      off=off+1;
       test=0;
   else
      on=on+1;
       test=1;
   end
   %%%%%%%%%%display result
   frequency_on_shift
```

```
frequency_off_shift
  test
   if on>off
   display('answer is on')
   else
   display('answer is off')
   end

 %%%%%%%%%%%%%%%%%%%%
 % when can't give the judement by the frequencyshift, we can directly do
 % the judgement from the symetric property of cross-correlation for the
 % matched signal shape.

else

if indice3<length(x3)/2
  q=1:(indice3-1);
 p=indice3+length(q):-1:indice3+1
 length(p);
 length(q);
 x3_left=x3(q);
 min(x3_left);
 x3_right=x3(p);
 min( x3_right);
error1= mean((abs(x3_right-x3_left)).^2);
else
    q=1+frequency_on_shift*2:indice3-1
     p=length(x3):-1:indice3+1
     length(q);
     length(p);
      x3_left=x3(q);
  x3_right=x3(p);
error1= mean((abs(x3_right-x3_left)).^2);
   end

if indice4<length(x4)/2
  q2=1:indice4-1
 p2=indice4+length(q2):-1:indice4+1
 x4_left=x4(q2)
 length(q2);
 length(p2);
 min(x4_left);

 x4_right=x4(p2)
 min( x4_right);
error2= mean((abs(x4_right-x4_left)).^2)
else
    q2=1+frequency_off_shift*2:indice4-1
     p2=length(x4):-1:indice4+1
     length(q2);
```

```
    length(p2);
     x4_left=x4(q2);
 x4_right=x4(p2);
error2= mean((abs(x4_right-x4_left)).^2)
end
if error1>error2
   off=off+1;
   test=0;
else
   on=on+1;
   test=1;
end



%%%%%%%%%%%%%   results %%%%%%%%%%%%%%%%%%%%%%
on
off
error1
 error2

 frequency_on_shift
 frequency_off_shift

test

if on>off
   display('answer is on')
   if on==off
      display('donno what u have said')
   end
else
   display('answer is off')
end
end
```

System 2 codes:

```
%%%%%%%%%%%%%% wiener filter  voice recogniton of frequency spectrum %%%%%
%%%%%%%%%%%%%% wiener filter method to realize voice recognition
%%%%%%%%%%
%%% linear prediction %%%%%%%%%%%%%%%

%%%% let voice signal to be d(n)
% use reference signal 1 and signal 2 to estimate d(n)_hat
%%%% find the error e=mean(abs(d(n)-d(hat))^2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%initial vaules
%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
on=0;
```

```
off=0;
fs=16000;          %sampling frequency, in 1 second  take 16000 samples

duration=2;        %recording time
%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Press any key to start %g seconds of recording.on.\n',duration);

pause;

fprintf('Recording...\n');

r=wavrecord(2*fs,fs);
%duration*fs total samples : take 2*fs samples need 2 seconds

r=r-mean(r);


fprintf('Press any key to start %g seconds of recording. off.\n',duration);

pause;

fprintf('Recording...\n');

y=wavrecord(2*fs,fs); %duration*fs :take 2*fs samples need 2 seconds

%%%%%%%%%%%%  control the number of how manay times u want to test
for T=1:20
fprintf('Press any key to start %g seconds of recording.voice.\n',duration);
y=y-mean(y);
pause;

fprintf('Recording...\n');

voice=wavrecord(2*fs,fs);
voice=voice-mean(voice);
fprintf('Finished recording.\n');

%%%%%%%%%%%%% use specgram to the signals spectrom information both in
%%%%%%%%%%%%% frequency domain and time domain.

nfft = min(1023,length(r));  %%%%%%%define our the length of STFT
s=specgram(r, nfft, fs, hanning(511),380);
s2=specgram(y, nfft, fs, hanning(511),380);
s3=specgram(voice, nfft, fs, hanning(511),380);

%%%% PART 1....frequency information
%%%% FREQUENCY  SPECTRUM %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% s is complex matrix
absolute=transpose(abs(s)); %%%%take abs make it real and ease to plot
absolute2=transpose(abs(s2));
```

```
absolute3=transpose(abs(s3));


%%%% FREQUENCY  SPECTRUM %%%%%%%%%%%%%%%%%% for the
frequency spectrum ,
%%%%%%%%%%%%%%%%%%%%%%

a4=sum(absolute)  %%%% get time-freuqency related spectrum
a5=sum(absolute2)
a6=sum(absolute3)
%%% %FREQUENCY  SPECTRUM %%%%%%%%%%%%%%%%%%%%
%%%%%%%% normalize the spectrom and also decrease the noise effect%%

a4_norm=(a4-min(a4))/(max(a4)-min(a4));
a5_norm=(a5-min(a5))/(max(a5)-min(a5));
a6_norm=(a6-min(a6))/(max(a6)-min(a6));
%%%%%%%%%%%%%%%%%%%%%%transpose row to colume
vector%%%%%%%%%%%

F4=transpose(a4_norm);
F5=transpose(a5_norm);
F6=transpose(a6_norm);
%%%%%%%%%%%%%%%%% F4 is x1
%%%%%%%%%%%%%%%%%% F5is x2
%%%%%%%%%%%%%%%%%% F6 is d(n)
N=length(F6);
rx1=xcorr(F4,F4);
rx2=xcorr(F5,F5);
d=F6;
rdx1=xcorr(F6,F4);
rdx2=xcorr(F6,F5);
rd=xcorr(d,d);
%%%%%%%%%%%% use x1 to estimate d(n) ===============
for p=1:20;

rx_1=rx1(N:N+p-1);
rdx_1=rdx1(N:N+p-1);
Rx1=toeplitz(rx_1, rx_1)

det(Rx1);%%%%%%%%%%%%%%%% check Rx is not singular
I=inv(Rx1);
w=I*rdx_1;

e1(p)=rd(N)-transpose(w)*rdx_1
end


%%%%%%%%%%%%% use x2 to estimate d(n) ===============
for p=1:20;
```

```
rx_2=rx2(N:N+p-1);
rdx_2=rdx2(N:N+p-1);
Rx2=toeplitz(rx_2, rx_2)

det(Rx2);%%%%%%%%%%%%%%% check Rx is not singular
I=inv(Rx2);
w=I*rdx_2;

e2(p)=rd(N)-transpose(w)*rdx_2
end
figure(1)
subplot(211)
title('reference on')
plot(e1);grid;
subplot(212)
title('reference off')
plot(e2);grid;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% frequency domain spectrum plot
%%%%%%%%
figure(2)
subplot(2,3,1)
plot(abs(s))

subplot (2,3,2)
plot(abs(s2))
subplot (2,3,3)
plot(abs(s3))
subplot(2,3,4)
plot(F4);
subplot(2,3,5)
plot(F5);
subplot(2,3,6)
plot(F6);

m1=mean(e1)
m2=mean(e2)

if m1<m2
   display('answer is on')
   if m1==m2
      display ('donno what to say')
   end
else
    display ('what u said is off')
end
end
```

Appendix B

The definitions of "error1" and "error2" in the following table can be found in the part 2.3.2 of this thesis. "frequency_on_shift" and "frequency_off_shift" in the following table are the frequency shifts which are got by comparing the spectrums as shown in the following Figure.21. "No need" means the designed system can give the judgment just by the frequency shift without calculating the defined errors.

Table 1 indicates the simulation results for reference signals "on" and "off" as the information given at the Simulation Results part 3.2.1(1).

Test times	frequency_on_shift	frequency_off_shift	Error1	Error2	**Final judgments**
1	1	1	0.0207	0.2858	on
2	-1	0	0.0434	0.1564	on
3	-1	2	0.0164	0.0664	on
4	-1	0	0.0400	0.1665	on
5	-1	0	0.0446	0.4071	on
6	-1	0	0.0280	0.3758	on
7	-1	0	0.0932	0.3273	on
8	0	0	0.0357	0.2381	on
9	-1	0	0.0483	0.5041	on
10	-1	-2	0.0465	0.1386	on
11	-1	-1	0.2493	0.0202	off
12	0	0	0.2002	0.0343	off
13	0	0	0.4141	0.0271	off
14	0	1	0.1310	0.0146	off
15	0	0	0.3697	0.0133	off
16	0	0	0.2942	0.0322	off
17	0	0	0.2096	0.0731	off
18	-1	0	0.4953	0.0164	off
19	-1	0	0.5644	0.0141	off
20	-1	0	0.2353	0.0194	off

Total successful probability(on)	100%
Total successful probability(off)	100%

Table 1: Simulation results for speech words "on" and "off"

Table 2 indicates the simulation results for reference signals "Door" and "Key" as the information given at the Simulation Results part 3.2.1(2).

Test times	frequency_door_shift	frequency_key_shift	Error1	Error2	**Final judgments**
1	-2	28	No need	No need	Door
2	-1	30	No need	No need	Door
3	0	29	No need	No need	Door
4	0	29	No need	No need	Door
5	-1	34	No need	No need	Door
6	-1	30	No need	No need	Door
7	0	29	No need	No need	Door
8	-1	27	No need	No need	Door
9	-2	29	No need	No need	Door
10	-1	27	No need	No need	Door
11	-19	0	No need	No need	Key
12	-21	0	No need	No need	Key
13	-21	-1	No need	No need	Key
14	-20	0	No need	No need	Key
15	-22	0	No need	No need	Key
16	-20	0	No need	No need	Key
17	-20	0	No need	No need	Key
18	-21	0	No need	No need	Key
19	-20	0	No need	No need	Key
20	-20	0	No need	No need	Key
Total successful probability(door)			**100%**		
Total successful probability(key)			**100%**		

Table 2: Simulation results for speech words "Door" and "Key"

Table 3 indicates the simulation results for reference signals “on” and “off” as the information given at the beginning of the Simulation Results part 3.2.1 (3).

Test times	frequency_on_shift	frequency_off_shift	Error1	Error2	**Final judgments**
1	0	0	0.0888	0.2858	on
2	0	0	0.0979	0.2645	on
3	0	0	0.1073	0.3327	on
4	0	0	0.0430	0.1958	on
5	0	0	0.0075	0.0476	on
6	0	0	0.0885	0.1834	on
7	0	0	0.1121	0.0390	off
8	0	0	0.0281	0.1699	on
9	0	0	0.0755	0.0286	off
10	0	0	0.0389	0.3312	on
11	0	0	0.2289	0.0075	off
12	0	0	0.2316	0.1493	off
13	0	0	0.1519	0.0228	off
14	0	0	0.1123	0.0072	off
15	0	0	0.0240	0.0360	on
16	-1	0	0.2900	0.0245	off
17	0	0	0.1984	0.0162	off
18	0	0	0.2414	0.0526	off
19	0	-1	0.4284	0.0246	off
20	-1	0	0.1334	0.0269	off
Total successful probability(on)			**80%**		
Total successful probability(off)			**90%**		

Table 3: Simulation results for speech words “On” and “off”

Table 4 indicates the simulation results for reference signals “Door” and “Key” as the information given at the beginning of the Simulation Results part 3.2.1 (4).

Test times	frequency_door_shift	frequency_key_shift	Error1	Error2	**Final judgments**
1	-2	15	No need	No need	Door

2	-1	15	No need	No need	Door
3	0	14	No need	No need	Door
4	0	14	No need	No need	Door
5	0	8	No need	No need	Door
6	-1	13	No need	No need	Door
7	0	13	No need	No need	Door
8	0	14	No need	No need	Door
9	0	13	No need	No need	Door
10	0	15	No need	No need	Door
11	-23	0	No need	No need	Key
12	-8	0	No need	No need	Key
13	-8	-1	No need	No need	Key
14	-16	0	No need	No need	Key
15	-16	0	No need	No need	Key
16	-14	0	No need	No need	Key
17	-20	0	No need	No need	Key
18	-13	0	No need	No need	Key
19	-13	0	No need	No need	Key
20	-12	0	No need	No need	Key
Total successful probability(Door)			**100%**		
Total successful probability(Key)			**100%**		

Table 4: Simulation results for speech words “Door” and “Key”

Table 5 indicates the simulation results for reference signals “Door” and “Key” as the information given at the beginning of the Simulation Results part 3.2.1 (5).

Test times	frequency_on_shift	frequency_off_shift	Error1	Error2	**Final judgments**
1	2	8	No need	No need	on
2	7	8	0.2055	0.4324	on
3	8	9	0.2578	0.2573	off
4	9	17	No need	No need	on
5	8	9	0.2304	0.3640	on
6	0	0	0.3268	0.6311	on

7	0	0	0.3193	0.3210	on
8	0	0	2.2153	0.9354	off
9	0	0	0.4603	0.1481	off
10	0	0	0.1189	0.0741	off
11	8	22	No need	No need	Door
12	8	0	No need	No need	Key
13	8	25	No need	No need	Door
14	8	24	No need	No need	Door
15	8	24	No need	No need	Door
16	-15	0	No need	No need	Key
17	-15	0	No need	No need	Key
18	-14	0	No need	No need	Key
19	-14	0	No need	No need	Key
20	-15	0	No need	No need	Key
Total successful probability(total in 20 times)			**80%**		

Table 5: Simulation results for speech words "Door" and "Key"

In the following tables, "m1" is the mean value of the minimum mean square-errors for reference signal 1. "m2" is the mean value of the minimum mean square-errors for reference signal 2. The definition of the mean value of the minimum mean square-errors can be found in the part 2.3.4 of this thesis.

Table 6 indicates the simulation results for reference signals "on" and "off" as the information given at the beginning of this section (1).

Test times	m1	m2	**Final judgments**
1	1.6740	3.4707	on
2	1.4442	2.6448	on
3	1.9087	3.7704	on
4	1.5448	2.9563	on
5	1.6103	3.4758	on
6	1.3971	3.7114	on
7	1.9205	3.3964	on
8	1.4944	3.7154	on

9	1.4716	3.7707	on
10	1.7879	4.5664	on
11	3.4775	0.1948	off
12	2.9213	1.0938	off
13	2.3013	0.7820	off
14	2.8370	1.1304	off
15	2.2277	0.7933	off
16	2.8922	1.8087	off
17	3.7633	1.6842	off
18	3.3211	1.5603	off
19	2.5852	0.8402	off
20	3.2708	1.4720	off
Total successful probability(on)		**100%**	
Total successful probability(off)		**100%**	

Table 6: Simulation results for speech words "on" and "off"

Table 7 indicates the simulation results for reference signals "Door" and "Key" as the information given at the beginning of this section (2).

Test times	m1	m2	**Final judgments**
1	0.1837	5.2746	Door
2	0.7070	6.5936	Door
3	0.7565	8.4193	Door
4	0.2680	5.5084	Door
5	0.2973	6.0271	Door
6	0.8471	7.5534	Door
7	0.7039	7.8490	Door
8	0.5523	7.5952	Door
9	0.2467	5.7104	Door
10	0.3792	4.8257	Door
11	6.7231	0.0990	Key
12	8.7500	1.0829	Key
13	6.0756	0.1670	Key
14	8.1771	0.5392	Key

15	7.0094	0.2012	Key
16	7.9720	0.7285	Key
17	6.4771	0.4326	Key
18	6.3291	0.6853	Key
19	5.2181	0.4563	Key
20	5.1473	0.1231	Key
Total successful probability(Door)		**100%**	
Total successful probability(Key)		**100%**	

Table 7: Simulation results for speech words "Door" and "Key"

Table 8 indicates the simulation results for reference signals "on" and "off" as the information given at the beginning of this section (3).

Test times	m1	m2	**Final judgments**
1	1.7351	3.5948	on
2	2.2663	3.9466	on
3	0.9809	2.7017	on
4	2.6011	2.7017	on
5	1.1137	1.7504	on
6	1.8093	2.9836	on
7	2.0433	2.9250	on
8	1.7423	2.1840	on
9	2.0782	3.7172	on
10	2.6252	4.8561	on
11	1.7214	0.5358	off
12	1.2078	0.6105	off
13	1.0685	0.8707	off
14	1.1108	0.7572	off
15	2.3774	0.7926	off
16	1.9559	0.8530	off
17	1.6756	0.8925	off
18	2.4116	0.8229	off
19	2.2664	0.5321	off
20	2.3713	0.4023	off

Total successful probability(on)	**100%**
Total successful probability(off)	**100%**

Table 8: Simulation results for speech words "on" and "off"

Table 9 indicates the simulation results for reference signals "Door" and "Key" as the information given at the beginning of this section (4).

Test times	m1	m2	**Final judgments**
1	0.3277	2.7586	Door
2	0.5835	3.3163	Door
3	0.2032	3.0460	Door
4	0.4941	3.0519	Door
5	1.4109	4.6609	Door
6	0.6575	3.7882	Door
7	1.1808	4.2020	Door
8	1.6448	3.4522	Door
9	0.5781	4.5291	Door
10	0.5794	4.6320	Door
11	3.9298	1.3690	Key
12	3.8828	0.8265	Key
13	3.9405	0.5655	Key
14	3.5953	0.7455	Key
15	4.1770	0.9070	Key
16	5.4789	1.5823	Key
17	4.1875	1.8230	Key
18	3.6388	1.3420	Key
19	3.5240	0.8551	Key
20	4.4077	0.4852	Key
Total successful probability(Door)			**100%**
Total successful probability(Key)			**100%**

Table 9: Simulation results for speech words "Door" and "Key"

Table 10 indicates the simulation results for reference signals "on" and "off" as the information given at the beginning of this section (5).

Test times	m1	m2	**Final judgments**
1	2.1943	3.5103	on
2	2.4698	1.7392	on
3	4.7760	3.3472	off
4	5.2285	4.6510	off
5	4.0235	3.9339	off
6	4.5611	3.8235	off
7	3.8988	4.2199	on
8	4.2026	3.2715	off
9	4.1196	3.2929	off
10	2.8686	1.9450	off
11	2.6504	1.4479	off
12	3.9870	2.2611	off
13	4.0748	2.2867	off
14	2.1320	0.8903	off
15	3.9950	2.3133	off
16	1.9331	0.7774	off
17	3.1565	1.9995	off
18	2.4163	1.1888	off
19	1.6946	0.6655	off
20	2.5083	1.2284	off
Total successful probability(on)			**30%**
Total successful probability(off)			**100%**

Table 10: Simulation results for speech words “on” and “off”