Project Code
04CS047

Project Title
Interacting With Small Personal Digital Devices by Voice:
Automatic Speech Recognition and Synthesis

(Volume 1 of 1)
Abstract

The Smart Recognizer (SR) is deployed as a speaker-dependent small-vocabulary speech recognition on mobile devices. The system can be easily reconfigured to work with arbitrary vocabularies. Applications of inputting Short Messages and operating by voice are built based on the algorithm. SR gets the MFCC features of voice signals, and utilizes asymmetric DTW to do feature matching. By carefully design, the memory requirements of the SR should fit to the average amount of today’s mobile phones and PDA, and the recognition precision is more than 85%.
Acknowledge

I would like to express my deepest gratitude to my final year project supervisor, Professor Horace H.S. IP, for his overall guidance, not only on this project but also on the my study in the last two years.

I would also like to express my sincere thanks to Assistant Professor Raymond Wong, for his valuable comments in doing the whole project. I am so fortunate that I was supervised by them.
# Table of Content

1. INTRODUCTION ........................................................................................................... 6  
   1.1 MOTIVATION ........................................................................................................... 6  
   1.2 SOLUTIONS ............................................................................................................. 7  

2. WORKING FLOW OF SR ................................................................................................. 9  

3. SPEECH FEATURE EXTRACTION ................................................................................. 11  
   3.1 FRAME BLOCKING .................................................................................................. 12  
   3.2 PRE-EMPHASIS: ..................................................................................................... 13  
   3.3 HAMMING WINDOW ............................................................................................. 14  
   3.4 FFT ........................................................................................................................ 14  
   3.5 MEL FILTER BANK ............................................................................................... 17  
   3.6 LOG COMPRESSION ............................................................................................ 18  
   3.7 DCT ...................................................................................................................... 19  
   3.8 NUMERICAL DIFFERENTIATION ......................................................................... 19  

4. FEATURE MATCHING .................................................................................................... 20  
   4.1 CEPSTRAL MEAN SUBTRACTION (CMS) ............................................................... 21  
   4.2 SPECTRAL SUBTRACTION (SS) ............................................................................. 21  
   4.3 CEPSTRAL MEAN NORMALIZATION (CMN) .......................................................... 22  
   4.4 “TO BE OR NOT TO BE” ...................................................................................... 22  
   4.5 DYNAMIC TIME WARPING (DTW) ...................................................................... 23  
      4.5.1 Symmetrical DTW ......................................................................................... 25  
      4.5.2 Asymmetrical DTW ...................................................................................... 27  
   4.6 AUTO CONFIGURATION ......................................................................................... 30  

5. SR SYSTEM DESIGN .................................................................................................... 31  
   5.1 USE CASE AND SCENARIO .................................................................................. 31  
   5.2 CLASS DIAGRAM .................................................................................................. 38  
   5.3 TECHNOLOGY AND TOOLS USED ...................................................................... 40  
      5.3.1 Microsoft EVC++ vs. Microsoft.NET compact framework .............................. 40  
      5.3.2 How to retrieve wave signals ........................................................................ 42  

6. EXPERIMENT .................................................................................................................. 44  
   6.1 VARIABLES CHOSEN ............................................................................................ 44  
   6.2 EXPERIMENTAL SETUP ....................................................................................... 45  
      6.2.1 Hardware ......................................................................................................... 45  
      6.2.2 Data Set ............................................................................................................ 46  
   6.3 BACKGROUND NOISE DEDUCTION ................................................................. 46  
   6.4 TWO DTW ALGORITHMS .................................................................................... 54  
   6.5 SINGLE CHARACTER ACCURACY TESTING .................................................... 55  

7. CONCLUSION .................................................................................................................. 59
8. REFERENCES ........................................................................................................61
9. APPENDIX ........................................................................................................62
   A. TEST CASES ................................................................................................62
1. Introduction

1.1 Motivation

Human-computer interactions often use a mouse and keyboard as machine input, and a computer screen or printer as output. However, when operating on small personal digital devices like mobile phones and PDA, the keyboard is too small and inconvenient. Speech, however, should have a highest priority in communication with such devices. When multimedia is available [1], combination of modalities can enhance accuracy, e.g., a visual talking face (animation) is more pleasant and intelligible than just a synthetic voice [2].

Currently mobile phones and PDA represent an innovative area and an attractive platform for speech recognition based features. Several manufacturers offer today mobile phones with voice interface. The growth in this segment is determined by several factors: functionality offered by speech recognition based features, robustness under typical application conditions, implementation cost, and end users’ acceptance. The expectation is that the competition on the market will migrate to added values for uncomplicated handling. Especially, voice control together with hands free operation seems to be very attractive. Terminals without any keypad may stand at the end of an evolution path where voice control is virtually the only method of user control. Speech recognition is expected to become one of the keys for mobile Internet access, like inputting the URL or messages (SMS and MMS) via voice. Significant progress was demonstrated recently in the standardization for such applications (W2C Voice Browser, VoiceXML).

The most powerful usage of my system is helping to input SMS messages, as well as controlling the mobile phone and PDA simply by voice. Mainland China has a huge market on SMS. During the 7 days’ break of Spring Festival this year, 10 billion SMS messages were sent. However, the market can be even bigger, for most of the middle-aged are afraid of sending SMS because of the complicated input method. They prefer to input the characters and control the devices by voices. Thus, a voice
recognition system implemented on mobile phones and PDA with very high accuracy in recognizing the speech, small vocabulary for auto configuration, fast response, and small memory allocation should be quite valuable and applicable. That is the reason to design and implement ‘SR’, and as the system is not perfect now, lots of further development is required.

1.2 Solutions

There are a few number of methods utilized, or will be utilized within 2 years on mobile phone and PDA to do speech recognition.

Most current mobile phones and smart-phones utilize some very simple comparisons on the file size and format to do speech recognition. Like in name-dialing, the user must speak the name or command one or twice to train the speech recognition engine before it can be used. The name is then saved as templates in the name-dialing phone book of the mobile phone. For those names not in the book, the recognition engine cannot recognize them.

A more advanced technique, speaker-dependent (SD) speech recognition with dynamic time warping (DTW) is quite suitable for mobile devices, and as it is of low complexity and language-independent, it will be frequently used by service providers in the coming few years [3, 4]. In my system, this technique is carefully evaluated and implemented.

The third one is speaker independent (SI) HMM-based (Hidden Markov Model) technology [5]. HMM based recognizers imply a much higher implementation complexity and need appropriate speech databases in many languages [6]. It is used for command-and-control and digit dialing, i.e., recognition of commands and phone numbers without requiring a training phase. Natural number (e.g., twenty-five, ninety-eight) recognition applies the same technology while improving the usability of voice dialing [7]. Currently a recognizer with an 18-word vocabulary was presented which is able to run on a 1.28 MHz system achieving a WER=0.5% in a quiet office.
environment [8]. It is expected that today’s memory and computational resources in 3G phones should facilitate deployment of such technology.

Also, the support vector machine (SVM) is also used in this area [9, 10]. It provides a significant improvement in performance on a static pattern classification task. With a large vocabulary, the error rate can be reduced to 4.0%.

However, the limited system resources of the mobile phones and PDA place strict restrictions on the memory and processing requirements of speech recognizers. DTW requires very few resources, which can be supported by mobile phones, but at the same time, the error rate is high. For most HMM-based recognizers, the memory requirements are too high for most mobile phones and PDA.

Thus, to avoid these problems and getting a better method for speech recognition on mobile phones and PDA, the Smart Recognizer (SR) is used. It has very low demands on system resources and allows flexible modification of the vocabulary.

In the following sections, section 2 illustrates the basic working flow of SR. The underlying technology is shown in section 3 and 4. Section 3 focuses on the feature extraction part, while section 4 deals with the feature matching. Section 5 introduces the system design of SR, as well as the user interfaces. Section 6 shows the experiment results to testify the efficiency and accuracy of SR.
2. Working flow of SR

SR is a low-resource high-performance recognizer adapted to the system and operating constraints of mobile phones and PDA. Modern mobile phones are powered by DSP and a microcontroller unit (MCU). The GSM base-band algorithms will be run by DSP, which will then perform acoustic processing functions, like echo cancellation and noise reduction. The MCU runs the user interface software and other applications such as address book management, games, or speech recognition. Permanent storage of data is typically provided by a flash memory which his shared by all applications. Therefore, the SR acoustic models cannot claim too much for themselves.

SR gives good solutions to the following three issues:

- Robust speech recognition in adverse conditions;
- Methods of memory reduction for increased implementation efficiency;
- Fixed-point code base for DSP-implementation.

And SR should also contain the following blocks:

- Noise-insensitive front-end reception allows effective speech recognition virtually anywhere where communication is possible.
- Scalable, high-quality phoneme databases make high quality speech recognition of English.

Language and Speaker dependent SR-engines allow rapid development of SR for any language without quality loss. Enough accuracy (at least 85%) should be achieved in real time on handheld units.

The basic work flow of SR is listed below:
At the highest level, the system contains two main modules: feature extraction and feature matching. Feature extraction is the process that extracts a small amount of data from the voice signal that can later be used to represent each character. Feature matching involves the actual procedure to identify the unknown character by comparing extracted features from previous voice input with the ones from a set of known characters. Details are discussed in later sections.

The same as other speech recognition systems, SR has to serve two distinguish phases. The first one is referred to the enrolment sessions or training phase while the second one is referred to as the operation sessions or testing phase. In the training phase, the certain speaker has to provide samples of his speech so that the system can build or train a reference model for that speaker. In case of speaker verification systems, in addition, a speaker-specific threshold is also computed from the training samples. During the testing (operational) phase (see Figure 1), the input speech is matched with stored reference model(s) and recognition decision is made.
3. Speech Feature Extraction

The purpose of this module is to convert the speech waveform to some type of parametric representation (at a considerably lower information rate) for further analysis and processing. This is often referred as the *signal-processing front end*.

The speech signal is a slowly timed varying signal (it is called *quasi-stationary*). When examined over a sufficiently short period of time (between 5 and 100 msec), its characteristics are fairly stationary. However, over long periods of time (on the order of 1/5 seconds or more) the signal characteristic change to reflect the different speech sounds being spoken. Therefore, *short-time spectral analysis* is the most common way to characterize the speech signal.

A wide range of possibilities exist for parametrically representing the speech signal for the speaker recognition task, such as Linear Prediction Coding (LPC), Mel-Frequency Cepstrum Coefficients (MFCC), and others. In this project, MFCC is chosen, because it is the best known and most popular. However, we have to spend more calculation time.
3.1 Frame Blocking

The digitized speech signal is blocked into overlapping frames. It is common to have 256 samples within one frame, so with the 8KHz frequency, a frame should contain 256/8000*1000 = 32 ms. A new frame contains the last half of the previous frame’s data, which is 16ms, and the first 16 ms of the next frame’s data. Thus, each frame is 32 ms in duration. The overlap decreases problems that might otherwise occur due to signal data discontinuity.
3.2 Pre-emphasis:

This stage spectrally flattens the frame using a first order filter. The transformation may be described as:

\[ Y[n] = x[n] - \alpha x[n-1], \quad 0.9 \leq \alpha \leq 1 \quad \text{and} \quad 0 < n < \text{SamplesPerFrame} \]

Here, \( x[n] \) refers to the \( n^{th} \) speech sample in the frame. Sphinx uses \( \alpha = 0.97 \) and the sampling rate is typically 8K 16-bit samples per second.

![Figure 3](image3.png) **Figure 3** Signal of speech “AB” before pre-emphasis (with Matlab)

![Figure 4](image4.png) **Figure 4** Signal of speech “AB” after pre-emphasis (with Matlab)
From the previous two figures (the signal of speech “AB”), it is clear that after this step, the higher frequencies are boosted, in order to compensate for subsequent attenuation of these frequencies, thus safeguarding the signal to noise ratio.

### 3.3 Hamming Window

In this stage a Hamming window is applied to the frame to minimize the effect of discontinuities at the edges of the frame during FFT. The transformation is:

\[
Y_1[n] = x[n] \times H[n], \quad 0 < n < \text{FrameSize}
\]

The vector \( H[n] \) is computed using the following equation.

\[
H[n] = 0.54 - 0.46 \times \cos \left( \frac{2 \times \Pi \times n}{\text{Framesize} - 1} \right)
\]

The constants, 0.54 and 0.46 used in the \( H[n] \) transform were obtained from the Sphinx source code.

There are several other windows, and the famous one is hybrid Hamming-Cosine window (see paper “1”). It has wider main lobe than Hamming window, but larger attenuation in the side lobes. However, to make the algorithm clear and easy to understand, the typical Hamming window is used.

### 3.4 FFT

The frame is padded with enough zeroes to make the frame size a power of two (call this \( N \)) and a Fourier transform is used to convert the frame from the time domain to the frequency domain.

\[
Y_2 = DFT(Y_1)
\]

The square of the magnitude is then computed for each frequency component. Thus the results are real numbers rather than the complex output produced by a discrete Fourier transform.
\[ Y_1[n] = \text{real}(Y_2[n])^2 + \text{imag}(Y_2[n])^2, \quad 0 < n \leq \frac{N}{2} \]

The result after this step is often referred to as spectrum or periodogram.

We can also see the reason for applying Hamming window from this step. It minimizes the bin "bleed" and "leakage" that occurs in frequency components that are not related to the sampling rate and fall outside the nearest FFT frequency bin. After Hamming window, the signals within one frame contains multiples of the periodicals, and are continuous in both side of the window.

The following figures (figure 4 and figure 5) show a frequency domain comparison of original signal and signal with Hamming window. Figure 4 shows two sets of signals before FFT transformation. Figure 5 shows the transformed frequencies.

![Original signal](image1)

![Windowed signal](image2)

**Figure 5** Signals with and without Hamming window before FFT
Figure 6 Signals with and without Hamming window after the FFT transform
3.5 Mel Filter Bank

A set of triangular filter banks is used to approximate the frequency resolution of the human ear. They are also called the Triangular Bandpass Filters. The output is an array of filtered values, typically called mel-spectrum, each corresponding to the result of filtering the input spectrum through an individual filter.

The curve in figure 7 shows the ratio of frequencies before and after Mel filter bank.

![Mel-frequency to frequency curve](image)

**Figure 7** Characteristic curve of Mel frequency

The triangular mel-filters in the filter bank are placed in the frequency axis so that each filter's center frequency follows the mel scale, in such a way that the filter bank mimics the critical band, which represents different perceptual effect at different frequency bands. Additionally, the edges are placed so that they coincide with the center frequencies in adjacent filters. Pictorially, the filter bank looks like:
A common model for the relation between frequencies in Mel and linear scales is as follows:

\[ \text{melFrequency} = 2595 \times \log(1 + \text{linearFrequency}/700) \]

The constants that define the filterbank are the number of filters, the minimum frequency, and the maximum frequency [11]. The minimum and maximum frequencies determine the frequency range spanned by the filterbank. These frequencies depend on the channel and the sampling frequency that you are using. According to the CMU’s Sphinx-4, the Mel frequency scale is settled linearly from 200Hz up to 1000 Hz.

The transformation by using Mel filter banks is:

\[ Y_4[n] = \sum_{i=0}^{N/2} Y_3[i] \times \text{MelWeight}[n][i], \quad 0 < n < \text{NumberOfFilters} \]

For 8 KHz sampling rate, Sphinx uses a set of 31 Mel filters.

### 3.6 Log Compression

The range of the values generated by the Mel filter bank is reduced by replacing each value by its natural logarithm. This is done to make the statistical distribution of the spectrum approximately Gaussian - a requirement for the subsequent acoustic model. The transformation is:

\[ Y_5[n] = \ln(Y_4[n]), \quad 0 < n < \text{NumberOfFilters} \]
3.7 DCT

The discrete cosine transform is used to compress the spectral information into a set of low order coefficients. This representation is called the Mel-cepstrum. Currently Sphinx compresses the 40 element vector $Y_5$ into a 13 element cepstral vector. The transformation is:

$$Y_6 = DCT(Y_5)$$

3.8 Numerical differentiation

Acoustic modeling assumes that each acoustic vector is uncorrelated with its predecessors and successors. Since speech signals are continuous, this assumption is problematic. The traditional solution is to augment the cepstral vector with its first and second differentials.

$$\triangle C_m(t) = \frac{\sum_{\tau=-M}^{M} C_m(t+\tau)\tau}{[\sum_{\tau=-M}^{M}\tau^2]}$$

$M$ is set to 2 here.

Since the Mel cepstral vector is 13 elements long in Sphinx, after appending the differentials the final acoustic vector that is 39 elements in length.
4. Feature Matching

The state-of-the-art feature matching techniques used in speech recognition include Dynamic Time Warping (DTW), Hidden Markov Modeling (HMM), and Vector Quantization (VQ). In this project, the DTW approach will be used, due to ease of implementation and high accuracy in small vocabulary speech recognition.

![Figure 9 Figure matching](image)

The main steps are listed in figure 9. Both training and testing vectors, which represent the sound signal, will first pass the Cepstral Mean Subtraction module, which will remove some of the effects of noise. Although in this figure, training and testing vectors will pass the module at the same time, the training phase can be done offline. The distance between these two vectors are calculated by Dynamic Time Warping (DTW).
4.1 Cepstral Mean Subtraction (CMS)

The cepstral mean subtraction technique has been shown to provide an efficient way to remove the influence of the transmission channel (microphone in this project). It consists in estimating the mean feature vector for each speaker and subtracting it from each feature vector.

\[ \tilde{x}_c(t) = y_c(t) - \bar{b} \]

\[ \bar{b} = \frac{1}{T} \sum_{t=1}^{T} y_c(t) \]

There are two different cases:

The training phase: On the training set, the mean feature vector can be estimated off-line. The training set is first analyzed to generate the mean feature vector for each speaker. In a second step, these mean vectors are subtracted from the feature vectors.

The recognition phase: Since we are not supposed to have access to future data in the recognition phase, we can no more proceed in two steps. The mean feature vector is computed on-line and the estimated mean is directly subtracted from the current feature vector.

In addition, another two methods are also tried to decrease the effect of noise. Although they are not used for the long calculation time, details and the experiment results are also listed in the following.

4.2 Spectral Subtraction (SS)

A speech signal \( s \) is corrupted by additive noise \( n \). The combined signal, \( s+n \), is observed and an estimate of the noise amplitude spectrum, \( \hat{N}(e^{i\theta}) \) is also available.

An estimate of the original signal amplitude spectrum \( \hat{S}(e^{i\theta}) \) is given by:

\[ \hat{S}(e^{i\theta}) = |S(e^{i\theta}) + N(e^{i\theta})| - |\hat{N}(e^{i\theta})| \]

That is, the estimated amplitude of the signal at a given frequency is the real amplitude of the signal plus noise at that frequency minus the estimated noise
amplitude. The phase of the signal is assumed to be the same as that of the signal plus noise. This modification of the frequency domain information can be placed within the overlap and add framework to give the technique known as spectral subtraction.

4.3 Cepstral Mean Normalization (CMN)

CMN is a technique used to reduce distortions that are introduced by the transfer function of the transmission channel (e.g., the microphone). Using a transmission channel to transmit the input speech translates to multiplying the spectrum of the input speech with the transfer function of the channel (the distortion). Since the cepstrum is the Fourier Transform of the log spectrum, the logarithm turns the multiplication into a summation. Averaging over time, the mean is an estimate of the channel, which remains roughly constant. The channel is thus removed from the cepstrum by subtracting the mean cepstral vector. Intuitively, the mean cepstral vector approximately describes the spectral characteristics of the transmission channel (e.g., microphone).

4.4 “To Be Or Not To Be”

Because the background noise is not considered in this project, only utilizing SS is not that useful.

<table>
<thead>
<tr>
<th>Baseline</th>
<th>SS</th>
<th>CMS</th>
<th>CMN</th>
<th>SS+CMS</th>
<th>SS+CMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.25%</td>
<td>NA</td>
<td>69.55%</td>
<td>67.05%</td>
<td>71.12%</td>
<td>70.9%</td>
</tr>
</tbody>
</table>

Table 1 The comparison between different subtraction methods

From Table 1 we can see that, CMS method is a little better than CMN, and when SS is also considered, there is a bit increase of the accuracy. But the payoff is a tremendous increase on the calculation. Thus in this project, only the CMS is utilized.
4.5 Dynamic Time Warping (DTW)

DTW is the temporal domain equivalent of instance-based learning. It begins with a set of template streams, each labelled with a class. Given an unlabelled input stream, the minimum distance between the input stream and each template is computed, and the class is attributed to the nearest template. The cleverness of dynamic time warping lies in the computation of the distance between input streams and templates. Rather than comparing the value of the input stream at time $t$ to the template stream at time $t$, an algorithm is used that searches the space of mappings from the time sequence of the input stream to that of the template stream, so that the total distance is minimised. This is not always a linear mapping; for example, we may find that time $t_1$ in the input stream corresponds to the time $t_1 + 5$ in the template stream, whereas $t_2$ in the input stream corresponds to $t_2 - 3$ in the template stream. The search space is constrained to reasonable bounds, such as the mapping function from input time to template time must be monotonically non-decreasing, in other words, the sequence of events between input and template is preserved.
In figure 10, the horizontal axis represents the time of the input stream, and the vertical axis represents the time sequence of the template stream. The path shown results in the minimum distance between the input and template streams. The shaded area represents the search space for the input time to template time mapping function. Any monotonically nondecreasing path within the space is an alternative to be considered. Using dynamic programming techniques, the search for the minimum distance path can be done in polynomial time:

$$O(N^2V)$$

where $N$ is the length of the sequence, and $V$ is the number of templates to be considered. Thus, for basic speech recognition DP has a small memory requirement, the only storage required by the search (as distinct from the templates) is an array that holds a single column of the time-time matrix.

Basically there are two DTW algorithms, symmetrical DTW and asymmetrical DTW. Both of them are implemented in this project, and the experiment shows that the latter one is much better.
4.5.1 Symmetrical DTW

If $D(i,j)$ is the global distance up to $(i,j)$ and the local distance at $(i,j)$ is given by $d(i,j)$

\[ D(i, j) = \min[D(i-1, j-1), D(i-1, j), D(i, j-1)] + d(i, j) \]

\[ D(1,1) = d(1,1) \]

The final global distance $D(n,N)$ shows the overall matching score of the template with the input. The input character is then recognized as the character corresponding to the template with the lowest matching score. (Note that $N$ will be different for each template.)

Figure 11 shows the three possible directions in which best match path may move.

![Figure 11 Best match path directions](image)

Computationally, it is already in a form that could be recursively programmed. However, unless the language is optimised for recursion, this method can be slow even for relatively small pattern sizes. So I define my own method which is both quicker and requires less memory storage uses two nested for loops. This method only needs two arrays that hold adjacent columns of the time-time matrix.

In figure 12, the cells at $(i, j)$ and $(i, 0)$ have different possible originator cells. The path to $(i, 0)$ can only originate from $(i-1, 0)$. However, the path to $(i, j)$ can originate from the three standard locations as shown.
The pseudocode for this process is:

Calculate first column (predCol)

\[
\text{for } i = 1 \text{ to number of input feature vectors} \\
\quad \text{curCol}[0] = \text{local cost at } (i, 0) + \text{global cost at } (i-1, 0) \\
\quad \text{for } j = 1 \text{ to number of template feature vectors} \\
\quad \quad \text{curCol}[j] = \text{local cost at } (i, j) + \text{minimum of global costs at } (i-1, j), (i-1, j-1) \text{ or } (i, j-1) \\
\quad \text{end} \\
\text{preCol} = \text{curCol} \\
\text{end}
\]

Minimum global cost is in \text{curCol[number of template feature vectors]}
4.5.2 Asymmetrical DTW

Although the basic DP algorithm has the benefit of symmetry (i.e. all frames in both input and reference must be used) this has the side effect of penalising horizontal and vertical transitions relative to diagonal ones. One way to avoid this effect is to double the contribution of \( d(i,j) \) when a diagonal step is taken. This has the effect of charging no penalty for moving horizontally or vertically rather than diagonally. This is also not desirable, so independent penalties \( d_h \) and \( d_v \) can be applied to horizontal or vertical moves. In this case there gives the so called asymmetrical DTW:

\[
D(i, j) = \min[D(i-1, j-1) + 2d(i, j), D(i-1, j) + d(i, j) + d_h, D(i, j-1) + d(i, j) + d_v]
\]

\( d_h = 0.5 \) and \( d_v = 0.3 \)

Here, values for \( d_h \) and \( d_v \) may be found experimentally.

This approach will favour shorter templates over longer templates, so a further refinement is to normalize the final distance score by template length to redress the balance. In addition, each frame in this algorithm of the input pattern is used once and only once. By comparison to the symmetric DTW, the three possible directions where the best match plan may move from cell\((i, j)\) are listed in figure 13.

![Figure 13 Three possible directions in asymmetric DTW](image)

However, it should be noted that special cases do occur. Consider the path as it originates from \((0, 0)\). As the path must move to column 1 then the global distances to
rows 1, 2 … n of the first column are meaningless, as illustrated in Figure 13. The shaded area shows the region into which the path can never move. The rectangular boxes show the ‘special cases’ for the path. As in symmetrical DTW, row 0 is a special case as normal. Additionally, in asymmetric DTW, row 1 is also treated specially.

![Figure 14 a special case for asymmetric DTW](image)

Point (i, j) in Figure 14 can be considered a ‘special case’. Special cases arise by virtue of the fact that the path must to move along the pattern file one frame at a time. This means that the match path can only arrive in such a 'special case' cell from a limited number of locations. As can be seen from Figure 5, such special cases appear in pairs which are located progressively higher in each subsequent column until a pair (or half of a pair if the column height is even) is located at the top of a column. All the remaining columns can be dealt with as normal. The special cases, when they occur, are at j = 2i-1 and 2i. The global cost for each is:

- 2i-1: local cost + minimum global distance at predCol[j-1] and predCol[j-2]
- 2i: local cost + global distance at predCol[j-2]

For both normal cells and special case cells, the rows 0 and 1 are dealt with in a different manner than normal.
The algorithm to find the least global cost is now more complicated than the symmetrical version. Therefore, it is easier to explain in pseudocode immediately rather than explaining the process in words. The pseudocode for the process is:

```pseudocode
predCol[0] = local cost at (0, 0)
for i=1 to number of input feature vectors
    curCol[0] = local cost at (i, 1) + global cost at (i-1, 0)
    for j=1 to (minimum of number of template feature vectors and 2i+1)
        store j in highestJ
        if cell(i, j) is a special case then
            if row 1 then
                curCol[j] = local cost at (i, j) + global cost at (i-1, j-1)
            else
                if cell(i, j) is the lower of the special case pair
                    curCol[j] = local cost at (i, j) + minimum of global costs at (i-1, j-1) or (i-1, j-2)
                else
                    curCol[j] = local cost at (i, j) + global cost at (i-1, j-2)
            end
        else
            if row 1 then
                curCol[j] = local cost at (i, j) + minimum of global costs at (i-1, j) or (i-1, j-1)
            else
                curCol[j] = local cost at (i, j) + minimum of global costs at (i-1, j), (i-1, j-1) or (i-1, j-2)
        end
    end
preCol = curCol
end
minimum global cost is value in curCol[highestJ]
```
4.6 Auto Configuration

When user is inputting a word character by character, SR will predicate it, and then show it out. A small vocabulary with about 800 common words is utilized to implement this function. The index of the vocabulary is listed in figure 15.

![Figure 15 Auto figuration flow](image)

26 arrays are used, one representing a set of words with the same initial letter. At first, the indexing structure of one array is based on the times a word is cited; however, when considering the time consuming of adjusting the array, a fixed indexing structure is implemented. The arrays are ordered as the dictionary: words are sorted on a character-by-character comparison, and those with smaller ASCII values have higher ranks.
5. SR System Design

5.1 Use Case and Scenario

The use case diagram (figure 16) illustrates the scenarios when users utilize SR to input speech signal.

![Use Case Diagram](image)

**Figure 16** Use Case Diagrams

(a) Training

*Main Success Scenario:*

1. User clicks one button and speaks to the microphone. (See figure 17)
2. SR gets the voice signal and passes voice signals to Voice Retrieval part for feature extraction. Voice signal should be input within 1.2 seconds after one button is clicked.
3. User does not have chance to listen to the recorded voices. It is to simplify the procedure of SR.
4. SR shows a message box to inform user the recording is finished, and the voices have been stored. (See figure 18)
5. User clicks “OK” on the message box, and then clicks another button to make a record, or click “OK” button on the rightmost of the window to finish recording.
**Figure 17** User Interface of SR speech input

**Figure 18** User Interface of SR speech input with message box
(b) Testing

Main Success Scenario:

1. User inputs a character by keyboard, sketching or voice.
2. User clicks ‘Tool’-> ‘Start Rec’ to start the record (figure 19). For each character user wish to input, he should speak to the microphone within 1.2 seconds after the occurrence of last input character. User does not need to click ‘Stop Rec’ to stop the record of one character.
3. User simply touches the screen or keyboard to stop the record.
4. SR does auto configuration to the words user has input, showing the predicated word on the screen (figure 20).
6. User clicks ‘Edit’ -> ‘Find…’ to search for a word (figure 22), clicks ‘Edit’ -> ‘Replace’ to replace an existing word with another one (figure 23), and clicks ‘Go To’ to go to a special line (figure 24).
7. SR highlights the word user hope to find and replace (figure 25 and 26).
Figure 19 Start the voice input

Figure 20 Auto configuration
Figure 21 Edit

Figure 22 Find a word
Figure 23 Replace an existing word

Figure 24 Go to a specified line
Figure 25 Result of executing a Find command

Figure 26 Result of executing a Replace command
5.2 Class Diagram

Figure 27 Simplified Class Diagram
Figure 27 is a simplified class diagram. Classes automatically generated by MFC are not listed here.

**Class Spectrum**

It is the class calculating spectrum of voice signals. The sampled signals will be passed to this class, and features are extracted by methods discussed in the previous sections.

**Class File I/O**

It is the class dealing with I/O with Pocket PC’s filing system. There are two formats of files used in this system, one is wave file (.wav) and the other is data feature file (.dat). Functions ReadWav(), WriteWav() handles the former format, read quantized data from buffer, and output the wave file. ReadFeature() and WriteFeature() handle the data feature file. All the input and output are in binary format. Function Assertion() generating warnings in case files are missed.

**Class CSpeechInputDlg**

It is the class handling users’ input in the training session. Quantized data will be retrieved from the hardware driver, and stored in array Sample[]. Functions StartRecord(), which make a connection with the wave reader runs when user requires the input of voices, while function EndRecord() is run automatically 1.2 seconds later. It close the connection and put the sampled data into Sample[].

**Class CNodepadView**

It is the class retrieving users’ voice and outputting the recognized character. Same functions StartRecord() and EndRecord() appear again in this class. Function DTW() will do feature matching by asymmetric DTW.

**Class CGotoDlg**

The object of this class is called when the ‘Go To’ command is set in class CNodepadView.

**Class CFindDlg**

The object of this class is called when the ‘Find’ or ‘Replace’ command is set in class CNodepadView.
5.3 Technology and tools used

5.3.1 Microsoft EVC++ vs. Microsoft.NET compact framework

In this project, Microsoft Embedded Visual C++ is utilized as the core development language. A brief overview of this language and a simple comparison with Microsoft .Net will be discussed in later sections.

Microsoft Embedded Visual C++ provides an Integrated Development Environment (IDE) designed for developing applications for Windows CE devices. It consists of an integrated set of windows, tools, menus, toolbars, directories, and other elements to help create, test, and debug a Windows CE application. It is based on the Win32 application programming interface (API). The fundamentals of programming for Windows CE closely parallel programming for Windows 98, Windows NT, and Windows 2000. As with the other Windows operating systems, Windows CE is an event-driven programming model. A Windows CE-based program receives messages, interprets the messages, and acts on the messages.

A Windows CE program has one or more Windows that receive and process messages in a message loop. The Windows can be visual or, for an application that does not require a user interface, non visible. Each window has a window handle (hwnd) associated with a message processor that handles the messages for the window. You can also use the window handle to call any related function.

Like any other Windows-based program, a Windows CE program has two primary functions, a message processor (usually called WndProc) and WndMain, which provides an entry point to the program. The WndProc function processes messages for the Window. In general, an application processes only those messages that are relevant to it, and passes other messages back to the operating system. In addition to being the primary message process for an application, WinMain also handles initialization and shutdown.

There is another tool called Microsoft .Net platform can perform the similar functions. But why Embedded C++ is chosen?
From figure 28 we can see that both Embedded VC++ and VS.NET are on the same software platform, using the same set of drivers. However, the gap between them is the code and compiling method. Embedded VC++ utilizes native code, while VS.NET is mainly comprised of managed code. The following table shows the differences between these two different coding systems:

<table>
<thead>
<tr>
<th>Native Code</th>
<th>Managed Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>C/C++ &amp; Win32 API</td>
<td>C#/VB.NET &amp; .NET CF</td>
</tr>
<tr>
<td>EXEs &amp; DLLs: <em>Native CPU instructions</em></td>
<td>EXEs &amp; DLLs: <em>IL instructions; JIT to native</em></td>
</tr>
<tr>
<td>CPU instructions</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---</td>
</tr>
<tr>
<td>Portable source code</td>
<td>Portable binary code</td>
</tr>
<tr>
<td>Manual cleanup</td>
<td>Garbage Collection</td>
</tr>
<tr>
<td>ActiveX / COM</td>
<td>COM not supported</td>
</tr>
<tr>
<td>No run-time required (OS is the runtime)</td>
<td>Run-time required: \textit{In ROM for all Windows Mobile devices}</td>
</tr>
</tbody>
</table>

\textbf{Table 2} Comparison between Native Code and Managed Code

Managed Code requires more run-time memory, and extra execution time to transfer the JIT to native CPU instructions, just like the operation of virtual machine in Java. As SR is only run on devices with Pocket PC 2003 operating system, operation via JIT is a waste of time. Thus for the purpose of high efficiency and low memory allocation, Native Code is used in this system.

\textit{5.3.2 How to retrieve wave signals}

By using Microsoft Windows Multimedia Package, the Pocket PC can retrieve voice signals from the microphone. An interface called “Waveform Audio API interface” are utilized to gain the greatest control possible over audio I/O devices. It has the following features:

- Querying and opening waveform audio I/O devices
- Allocating audio data blocks
- Playing waveform audio files
- Handling errors generated by audio functions
- Using windows messages to manage waveform audio playback
- De-allocating memory blocks associated with audio data
- Closing waveform audio output devices
Through the above features, SR controls the audio I/O device, retrieving the voice signals from an embedded microphone on the Pocket PC, and quantizing them. There are totally xx steps, and the details are listed below.

1. To record a sound correctly, first determine what drivers are available for audio I/O on your Windows CE–based device, and then open those drivers for recording or playback. In SR, the `WaveInOpen` and `WaveOutOpen` functions are used to choose the device that is best able to retrieve and play the specified data format. In addition, they fill the memory location with a device handle. Use this device handle to identify the open waveform audio I/O device when calling other audio functions.

2. After determining the capabilities of your Windows CE–based device, one can allocate memory for the audio data blocks. Use the WAVEHDR structure to allocate the memory that the `waveInAddBuffer` and `waveOutWrite` functions need to play sound. The following table shows the functions that prepare headers.

3. The data stored in the buffer has already been quantized. Pocket PC only supports 8K Hz sampling frequency. The next step is to read the data out of the buffer. SR provides a fast reader to achieve this goal. It works based on the classical wave file format, which is standardized by Microsoft.

4. After retrieving the data, the audio I/O connection will be closed. A set of functions are called, the first one is `waveInStop`, to close the microphone. Then `waveInReset` is called to reset the state of the microphone, so it can respond later. The last one is `waveInClose`, which close the buffer, implying the end of the whole process to retrieve voice signal.

5. In addition, a timer is utilized in SR to control the interaction to the microphone. When user starts the record, the I/O channel will be automatically closed after 1.2 seconds. Any voice signal sent by microphone within this period will be seemed to be the voice input from user.
6. Experiment

6.1 Variables Chosen

Based on the analysis in the previous sections, there are several variables to be quantized which are listed in the following table.

\textit{preEmphasisAlpha}: the weighting to flatten the frame in pre-emphasis.

\textit{frameSampleAmt}: number of samples within one frame in frame blocking.

\textit{frameDublicatedAmt}: the duplicated sampling amount of frames.

\textit{hammingWindowAppha}: the weighting value used in Hamming Window, obtained from Sphinx IV.

\textit{sampleRate}: the sampling rate of the voice signal.

\textit{minFreq}: the minimum frequency covered by the filter bank, obtained from Sphinx IV.

\textit{maxFreq}: the maximum frequency covered by the filter bank, obtained from Sphinx IV.

\textit{numOffilter}: number of filters in the filter bank, obtained from Sphinx IV.

\textit{cepstralVecAmt}: dimension of the feature vector.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pi</td>
<td>3.1415926</td>
</tr>
<tr>
<td>preEmphasisAlpha</td>
<td>0.95</td>
</tr>
<tr>
<td>frameSampleAmt</td>
<td>256</td>
</tr>
<tr>
<td>frameDublicatedAmt</td>
<td>128</td>
</tr>
<tr>
<td>hammingWindowAppha</td>
<td>0.46</td>
</tr>
<tr>
<td>numOfFilter</td>
<td>31</td>
</tr>
<tr>
<td>sampleRate</td>
<td>8000Hz</td>
</tr>
<tr>
<td>minFreq</td>
<td>10Hz</td>
</tr>
<tr>
<td>maxFreq</td>
<td>1000Hz</td>
</tr>
</tbody>
</table>

\textbf{Table 3 Variables}
NO experiment is settled to quantize the above values. They are set by the existing standards from Sphinx IV, an ASR system developed by CMU. The sampling rate can only be set to 8 KHz, which severely decrease the recognition accuracy.

6.2 Experimental Setup

6.2.1 Hardware

The system is installed on HP iPAQ Pocket PC hx4700. It boasts the latest OS - Microsoft Mobile 2003 SE, Intel’s 624MHz high speed processor, 62 MB flash memory, and a voice channel with 8K Hz frequency. By comparison with its colleagues, hx4700 is the one with best OS supporting and fastest processing speed. However, the whole processing capability is only equal to the PC in late 1990’s. The lack of hardware support decreases accuracy of recognition, and also increase the computation time, especially in the calculation of floating-point. The detailed analysis will be shown in the later sections. The detailed description is listed in table 4.

Table 5 lists the comparison of performance between current Pocket PC and IBM’s ThinkPad T40. Processor speed and memory speed are tested here, as they influence the accuracy and computation time of SR. Processor speed test measures the performance of a Pocket PC's processor and memory subsystems by performing both integer and floating-point calculations. Memory speed test measures the performance of a Pocket PC's internal file system. Neither the amount of onboard memory nor the presence of storage cards impact this test. The result can more or less shows the weak computation capability current Pocket PC has.
<table>
<thead>
<tr>
<th>Model</th>
<th>HP iPAQ Pocket PC hx4700</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installed RAM</td>
<td>62MB</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel 624MHz Xscale</td>
</tr>
<tr>
<td>Built-in devices</td>
<td>Speaker (8K Hz)</td>
</tr>
<tr>
<td>OS Provided</td>
<td>Microsoft Windows Mobile</td>
</tr>
<tr>
<td></td>
<td>2003 Second Edition</td>
</tr>
<tr>
<td>Data Service</td>
<td>SMS</td>
</tr>
</tbody>
</table>

**Table 4** Design of the Pocket PC

<table>
<thead>
<tr>
<th>Functions</th>
<th>Hx4700</th>
<th>ThinkPad T40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write 1MB file</td>
<td>2,005ms</td>
<td>860 ms</td>
</tr>
<tr>
<td>Read 1MB file</td>
<td>67ms</td>
<td>30ms</td>
</tr>
<tr>
<td>Copy 1Mb file</td>
<td>1,929ms</td>
<td>605ms</td>
</tr>
<tr>
<td>Write 10Kb * 100 files</td>
<td>2,700ms</td>
<td>1,200ms</td>
</tr>
<tr>
<td>Read 10Kb * 100 files</td>
<td>200ms</td>
<td>55ms</td>
</tr>
<tr>
<td>Copy 10Kb * 100 files</td>
<td>2,305ms</td>
<td>904ms</td>
</tr>
<tr>
<td>Calculating Pi to 1 million places</td>
<td>300,000ms</td>
<td>93000ms</td>
</tr>
<tr>
<td>Calculating Fibonacci number to 1 thousand places</td>
<td>20,000ms</td>
<td>3,000ms</td>
</tr>
</tbody>
</table>

**Table 5** Comparison of Performance

### 6.2.2 Data Set

There are 12 different subsets used for testing of the recognition accuracy. 9 of them are from BBC website, and the left 3 sets composites of equal number of the 26 distinct characters.

The detailed subsets are listed in Appendix A. The following figures show the frequency of each character within each data set. The first 8 data sets are chosen randomly, and the distribution of the frequencies fit to the classic relativity frequency
of letters in English test (figure 40), thus they are eligible testing data sets. However, as the distribution of frequencies vary too much, the other 3 data sets for the purpose of testing the accuracy of recognizing those less appeared characters like ‘j’ and ‘z’.

**Figure 29** Relative Frequency of letters in Data Set 1

**Figure 30** Relative Frequency of letters in Data Set 2
Figure 31 Relative Frequency of letters in Data Set 3

Figure 32 Relative Frequency of letters in Data Set 4
Figure 33 Relative Frequency of letters in Data Set 5

Figure 34 Relative Frequency of letters in Data Set 6
Figure 35 Relative Frequency of letters in Data Set 7

Figure 36 Relative Frequency of letters in Data Set 8
Figure 37 Relative Frequency of letters in Data Set 9

Figure 38 Relative Frequency of letters in Data Set 10
Figure 39 Relative Frequency of letters in Data Set 11

Figure 40 Relative Frequency of letters in English
6.3 Background Noise Deduction

Three different methods for background noise deduction are introduced from section 4.1 to 4.4. A simple conclusion is made in section 4.4 that, CMS method is a little better than CMN, and when SS is also considered, there is a bit increase of the accuracy. But the payoff is a tremendous increase on the calculation. Thus in this project, only the CMS is utilized. To testify the result, some experiments are conducted and the results are listed here.

The first 8 data sets are used, which are paragraphs from BBC. Five different methods are used, which are: SS, CMS, CMN, SS + CMS, and SS + CMN.

Figure 41 shows the precision curve of all the 26 characters from the above data sets. From the map we can see that, the above functions reducing the background noise can help to improve the recognition precision by about 5% to 10%. But what kind of methods should we use? When referring to the precision, SS+CMN is the best choice, however, when we further consider the calculation time (listed in table 6), SS + CMN needs much more calculation time than simply utilizing CMS. Every time it is used, it will take 0.2 seconds to complete the calculation. In a respond system like SR, such a computation time is not affordable. So the payoff is that, by using two times of the calculation time to get a 2% improvement in precision. Because the efficiency is also a very important issue in SR, SS+CMN is ignored and CMS is finally chosen.

![Precision Map](image)

*Figure 41 Recognition precision of the characters ‘a’ to ‘z’*
6.4 Two DTW algorithms

In section 4.5, two different DTW are introduced. One is symmetric DTW, and the other one is asymmetric DTW. Here, ‘symmetric’ and ‘asymmetric’ are the signs for two different weighting functions. Both functions are derived from the distance travelled (in grid units) in the last step of the path. The symmetric form combines the $i$ and $j$ directions while the asymmetric form uses just the $i$ direction. The sum of this function over the length of the path gives a measure of how long a path is. Besides the analysis of the strength and weakness of these two algorithms made in section 4.5, experiments are also conducted.

In figure 42, the red line represents the asymmetric DTW, while the blue one is symmetric DTW. It is obvious that the asymmetric DTW will retrieve a better result.

However, we should also see that, with an average precision of 82.34%, asymmetric DTW is also not that good. A backend HMM algorithm may help to improve the recall and precision. It will be the further development.

<table>
<thead>
<tr>
<th>CMS</th>
<th>CMN</th>
<th>SS+CMS</th>
<th>SS+CMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>52ms</td>
<td>55ms</td>
<td>150ms</td>
<td>200ms</td>
</tr>
</tbody>
</table>

*Table 6 Computation time for the 4 functions*
Finally, I apply SR to the 11 testing sets in Appendix A. The 11 testing sets are separated into two groups. Figure 43 shows the recall-precision curves for queries of each character in first 8 testing set, while figure 44 shows the corresponding curves for the remaining 3 testing sets. Figure 45 displays the average recall-precision curves for the above two groups. The reason to separate into two diagrams is that, the former group has different number of characters while the latter one contains equal amount. Thus, figure 43 can be seemed as how accurate SR works in real applications, while figure 44 is used to deeply testify retrieval of each character.

In the first set of testing data, several characters are not taken into account, because of their very low appearance. According to figure 40, they are ‘j’, ‘k’, ‘q’, ‘x’, and ‘z’. However, in the second group of testing cases, we can see that the recognition rate of these 5 characters are quite high, thus the average recall-precision curve of the second group of testing set, also called ‘Ideal Set’, is much flatter.
Figure 43 recall-precision curves for each character in first 8 testing sets

Figure 44 recall-precision curves for each character in last 3 testing sets
From the above figures, we can see that the ratio of recall/precision curve is not that flat. When the recall is 0.8, the precision value is only 0.76 for real set and 0.84 for ideal set. Such a precision is far from enough when comparing with the existing speech recognition systems operated on PC.

Why the curve runs in this way? There are two reasons.

First, the low sampling rate on mobile devices, which is only 8K Hz, greatly decreases the accuracy. To clearly illustrate this and discover if it is the problem with the algorithm, another experiment is done. The SR system is modified, and run on a notebook IBM T40. The training and testing procedures remain the same, while the only change is the sampling frequency of the voice. Surely, the running time is shortened as PC has much better computation capability on calculation of integer and floating point, but here we just care about the accuracy. The comparison of the four curves is shown in figure 46, where we can see that the system itself is robust and efficient, but when applied to the extreme environment, it will become less accurate. Later, when the PDA and mobile phone apply sampling rate of 16K Hz, SR will achieve excellent recognition accuracy and face a very bright future.
Another for the low precision result is that, during the training phase, user only needs to speak the character once. If there are a large number of sets of training data, the accuracy will increase obviously. I have not tried how accurate SR will become if several sets of training data can be applied, because SR is designed for the ease of use. No user hopes to train one character for many times, so it is a payoff. Some surveys need to be conducted, testify the acceptable times of input user will do to train SR. But at this stage, SR will only take one input for each character.
7. Conclusion

Critical Review

The techniques in speech recognition have been developing for more than 20 years, and seem to be quite mature. There are hundreds of different algorithms nowadays. However, it is a long way to go when applying to mobile devices with limited memory and low process capacity. Currently there is a boom in designing such applications, like Siemens VSR engine, Microsoft Voice Command 1.5, and Intel’s speech recognition engine for 3G phones. I try to develop a system with similar functions of the above engines in my own way, choosing the most suitable techniques according to the processing conditions of current Pocket PC. For example, according to the limited memory, only algorithms supporting small vocabulary can be used. In addition, as HMM includes a very time-consuming training session, it is replaced by DTW. (But HMM can also be applied in further development.) Referring to section 6, different algorithms are compared and chosen according to the experiment results on both the accuracy and calculation time. Furthermore, I also pay much attention to the interface design, which is the key issue in system development on mobile devices. All the buttons and text areas are carefully allocated, and whether the functions should run automatically or be driven by users’ interactions are also considered.

To sum up, this system gets the voice signals with 8K Hz sampling frequency from the microphone, and then extracts the features which are called Mel-scale Frequency Cepstral Coefficients, or in short, MFCC, by 8 major steps. By using these coefficients, the dimension of the data decreases from about 1000 to 39. In addition, Cepstral Mean Subtraction (CMS) is applied to deduce the influence of the microphone, as well as the background noise. Asymmetric DTW is then used to do feature matching by the coefficients. To make the system more user friendly, auto configuration is applied with the support of a database of 800 commonly used vocabularies. The whole system only requires 700KB memory, and reaches the precision of 85%.
The whole system can operate on any series of Pocket PC with Microsoft WinCE.NET Operating System. The deployment is quite simple. However, the average recognition precision needs further improvement. There are several techniques can be applied to SR in the future, and hopefully they will increase the precision, and at the same time decrease the calculation time.

**Further Development**

- **A more powerful backend**

  CHMM will take place of the DTW [12]. This new back-end will decode the features into best-matching phonemes, which are put together to form words. In addition, CHMM parameters and the pronunciation models are stored offline.

  The back-end can take two classic steps, likelihood evaluation and Viterbi decoding. The former step is a prerequisite for the latter one. Markov properties of HMMs provide an efficient dynamic programming algorithm, called alpha-recursion, to evaluate the complete data likelihoods.

  However, using HMM implies more calculation time in training. The poor calculation capacity of Pocket PC may not be able to support it. Anyway, it is a very good way to improve the accuracy, and if the whole system can become speaker-independent, the pay of time-consuming training is worthy.

- **Different feature extraction methods**

  Generally speaking, there are three methods to do feature extraction, which are MFCC analysis, LPC analysis and PCA. In this system, only MFCC is tried, and in the future, the left two analysis methods can be tried. Surely LPC will be much faster than MFCC, for it does not need the FFT transform, but some paper also reports that MFCC analysis will achieve a more accurate result. PCA is a classic method, utilizing in many different areas. Before using PCA, kernel functions can be called to project the original voice signals to a higher dimensional subspace, to help distinguishing the different groups of voice signals in view of their complex characteristics. But the problem with Kernel is that, it is quite slow. Whether it fits to my system under Pocket PC is a problem.
8. References


Appendix

A. Test Cases

Test Case 1
Despite the dip in price growth, the Nationwide's forecast of a "soft landing" for the market is in line with other recent surveys that have suggested the market is not heading for a crash. On Monday, property website Hometrack said price falls were luring more buyers onto the market, with the number of people on estate agents' books rising 6.2% in March.

Test Case 2
At a regional level, there continues to be a North-South divide with prices growing most quickly in the north of England, and in Wales, Scotland and Northern Ireland, Mr Bannister said. London recorded the slowest rise. However, he added that price growth had slowed "rapidly" in the north-west of England, once one of the UK's hotspots, and the region appears to be following the trend seen in London a year earlier. "Affordability seems to be key to this," he added.
"A mortgage payments as a proportion of take-home pay has been increasing across all of the regions over time, but hit higher levels in London... much earlier and this is when prices began to slow."

**Test Case 3**

The MPs called on the Post Office to review its policy after MPs heard 75% of cash machines in its branches charge customers.

Sub-postmasters have no control over what type of machine they install, and no account is taken of the needs of the local community, the committee was told.

"The Post Office has a responsibility to move towards greater provision of free machines, particularly when there are no banks nearby," the report ruled.

During sometimes heated exchanges MPs quizzed the chief executives of the UK's biggest High Street banks and fee-charging machine providers.

**Test Case 4**

"I am glad that MPs recognise that 97% of cash machine withdrawals are free and that UK consumers get a good deal, particularly when compared to other countries," a spokeswoman said.

Consumer groups welcomed the committee's findings.

"Easy access to cash, free of charge, must be protected," said Laurence Baxter, senior policy adviser at Which.

"The massive growth in charging threatens people's right to get hold of their money for free."
Test Case 5

Last April, frustration at the firm's problems ended in a boardroom coup. Eurotunnel's entire Anglo-French board was replaced with an all-French team after a revolt by rebel shareholders.

Shareholders, some of whom have lost 90% of the value of their investments, argued that the then management's restructuring efforts would not significantly cut the group's debt.

Eurotunnel is presently able to issue bonds for any interest it is unable to pay, but at the end of the year will be obliged to start paying all the interest on its debt in cash.

Test Case 6

Mr Hurd, currently head of smaller US computer services group NCR, will replace Carly Fiorina, who was ousted from HP in February.

HP's shares closed up 8.1% in Tuesday trading in New York, following initial speculation of Mr Hurd's appointment.

Mr Hurd, 48, is credited with turning around the fortunes of NCR since he took over the top job in March 2003.

Ms Fiorina, one of the most powerful businesswomen in the US, left HP after a dispute with the company's board over future strategy.

Mr Hurd will take her place on 1 April.

While this is a huge stretch for Mark (in terms of the scale and size of HP), he is a very capable executive who did a great job with Teradata, part of NCR," said Bruce Richardson, senior vice president of AMR Research.

Test Case 7

Born in 1957 in New York City

Educated on tennis scholarship at Baylor University, Texas

Joined NCR in 1980 after brief spell as pro tennis player
Became NCR's CEO in March 2003, after serving as president since 2001 and being named chief operating officer in 2002.

He has accepted a $2m joining bonus and a four-year contract offering him $1.4m a year and the chance to buy 700,000 shares, all of which will be sellable by the end of the contract.

**Test Case 8**

"While this is a huge stretch for Mark (in terms of the scale and size of HP), he is a very capable executive who did a great job with Teradata, part of NCR," said Bruce Richardson, senior vice president of AMR Research.

Mr Richardson said it was "a surprise pick, but one with high potential".

In contrast to Ms Fiorina's marketing background, Mr Hurd is firmly rooted in technology manufacturing, representing a return to HP's tech-centric roots.

He has eschewed her "superstar CEO" style in favour of a more down-to-earth approach.

Aside from the difference in scale - HP has annual sales of some eighty bn USD, more than thirteen times NCR's - there are similarities between the two firms - and one person quoted by the Wall Street Journal said Hurd "has run a mini HP".

Part of NCR's recent success has been in landing lucrative service contracts, an area in which HP has struggled against competitors such as IBM.

It has also lost ground to Dell in particular in the personal computer marketplace.

The company's $19bn purchase of rival Compaq - a move pushed through directly by Ms Fiorina - was not considered a success.

It was this, and her failure to hit HP's profits targets, which led to her dispute with the company's board and subsequent departure.

**Test Case 9**

*Each character has 20 samplings.*
Test Case 10

*Each character has 5 samplings.*

Test Case 11

*Each character has 10 samplings.*