

Department of Computer Science and Engineering
The Chinese University of Hong Kong

Final Year Project Report
2001-2002 First Term

LYU0103

Speech Recognition Techniques for
Digital Video Library

Supervisor: Professor Michael R. Lyu

Prepared by: Gao Zheng Hong (98795603)
Lei Mo (98796021)

Date of Submission: 1st, December 2001

Printed in Hong Kong


THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

This document could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the document. We may make improvements and/or changes in the product(s) and/or the program(s) described in this document at any time.

Table of Contents

Abstract.....	9
Chapter 1 : Introduction.....	10
1.1 Project Overview.....	10
1.1.1 VIEW Background.....	10
1.1.2 Video Information Processing.....	11
1.1.3 Speech Recognition Engine	11
1.2 Project Objectives	12
Chapter 2 : Introduction to Speech Recognition.....	14
2.1 Terminologies in SR System	14
2.1.1 Utterance	14
2.1.2 Speaker Dependence Vs. Speaker Independence.....	14
2.1.3 Vocabularies	15
2.1.4 Grammar.....	15
2.1.5 Isolated Word Recognition Vs. Continuous Speech Recognition	15
2.1.6 Difficulty Level	16
2.1.7 Accuracy.....	16
2.1.8 Training	17

2.2	Speech Recognition Process.....	17
2.2.1	Process Diagram.....	17
2.2.2	Process Explanation	18
2.3	Uses and Applications.....	20
2.3.1	Dictation.....	20
2.3.2	Command And Control	20
2.3.3	Telephony.....	21
2.3.4	Medical/Disabilities	21
2.3.5	Embedded System Applications.....	21
2.4	Challenges And Difficulties.....	21
2.4.1	Linguistic Variability.....	21
2.4.2	Speaker Variability.....	22
2.4.3	Channel Variability.....	22
2.4.4	Coarticulation.....	22
Chapter 3 : Speech Recognition Engines.....		24

3.1	CMU Sphinx 	24
3.1.1	Introduction to CMU Sphinx		24
3.1.2	Sphinx2 Vs. Sphinx3.....		24

3.1.3	SphinxTrain	25
3.2	Microsoft SAPI	25
3.2.1	Introduction to Microsoft SAPI	25
3.2.2	Responsibilities of SAPI and SR Engine	25
3.2.3	SAPI SR Objects and Interfaces.....	26
3.3	IBM ViaVoice.....	27
3.3.1	What is IBM ViaVoice?.....	27
3.3.2	System Characteristics	28
3.3.3	ViaVoice Native Architecture Overview.....	28
3.3.4	ViaVoice Speech Engine Architecture	28
3.3.5	Application Programming Interfaces	29
3.4	Comparisons of Different SR Systems.....	30
3.5	Why Choose ViaVoice?.....	31
3.6	Our Objective With ViaVoice.....	31
Chapter 4 : Audio Extraction		32
4.1	Microsoft DirectX	32
4.1.1	DirectShow Overview	32
4.1.2	DirectShow Application Programming.....	33
4.2	Implementation of Audio Extractor	35
4.2.1	Special Filters Employed in Our Program	35

4.2.2	Filter Graph of Audio Extractor	36
4.3	Dictation by Untrained ViaVoice Engine	38
Chapter 5 : Speech Segmentation.....		41
5.1	Why To Do Speech Segmentation	41
5.2	Overview of Different Approaches	42
5.3	Frame Energy Analysis	43
5.4	Segmentation Result.....	45
Chapter 6 : Visual Training Tool.....		46
6.1	WinTrain Features	47
6.2	Implementation of The Visual Training Tool.....	51
6.2.1	Document and View	51
6.2.2	Object Diagram	52
6.2.3	CWinTrainApp.....	52
6.2.4	CWinTrainDoc	53
6.2.5	CPlayerForm	54
6.2.6	CDictationView	55
6.2.7	CWinTrainView	56
6.2.8	CVideo.....	56
6.2.9	Customized Audio Library.....	58

Chapter 7 : String Alignment	61
7.1 String Alignment By Dynamic Programming.....	61
7.1.1 Longest Common Sequence (LCS).....	61
7.1.2 Definition of Edit Distance	61
7.1.3 Recurrence Relation	61
7.1.4 Pseudo code for the algorithm.....	62
7.2 Examples	62
Chapter 8 : Summary	64
8.1 Problems Facing.....	64
8.2 Future Plans.....	65
8.2.1 Visual Training Tool Enhancement	65
8.2.2 Gender Classification	65
8.2.3 Noise Removal	65
Acknowledgement	67
Appendix A: References	68
A-I: Books	68
A-II: Papers	69
A-III: Useful URL	70

A-IV: Useful Newsgroups And Forums..... 71

Appendix B: Progress Report..... 72

Abstract

Automatic Speech Recognition (ASR) is the process by which computer (or other type of machine) identifies spoken words and recognize what you are saying. It has been an interest of research by nearly four decades. It is amazing because it has a wide area of applications. If eventually computer can get 100 percent of recognizing words, it will bloom a new era of computer technology and even change people's daily life. However, there are many difficulties to overcome and speech recognition still has a long way to go.

This project is to apply multiple speech recognition techniques to improve accuracy of recognizing speeches in digital media contents.

Chapter 1 : Introduction

1.1 Project Overview

Our project about speech recognition techniques is a part of a mother project called VIEW, which is short for “Video over InternEt and Wireless” and supervised by Professor Michael R. Lyu. This project is to develop a multilingual digital video content hub for culture exchange and commercial deployment. This project includes a digital video library, which uses the outcome tool of our project to produce captions for the videos in it. In order to specify the role of our project, we will give a brief introduction to the VIEW project. For more detailed information, please visit <http://www.viewtech.org/>.

1.1.1 VIEW Background

The “Multilingual Digital Video Content Hub” and “VIEW”(Video over InternEt and Wireless) project is funded by Innovation and Technology Commission (ITF) Fund and Area of Excellent. The planning time of funding is from September 2000 to September 2002. The Principal Investigator is our supervisor Prof. Michael R. Lyu in department of Computer Science and Engineering, Chinese University of Hong Kong.

The main objective of the VIEW project is to develop a multilingual digital video content hub with emphasis on Chinese and English languages for cultural exchange and commercial deployment.

And also, it aims at to deliver software-enabling schemes for content creation, information summarization, multi-model searching and presentation.

The logical framework of the VIEW project consists of three major components:

- Back-end
 - Video Information Processing Engine
 - Indexing and Searching Engine
- Front-end
 - Visualization and Presentation Engine

1.1.2 Video Information Processing

There is a video information extraction engine that extracts multi-modal information from video including: transcript, key frames, characters, face, etc.

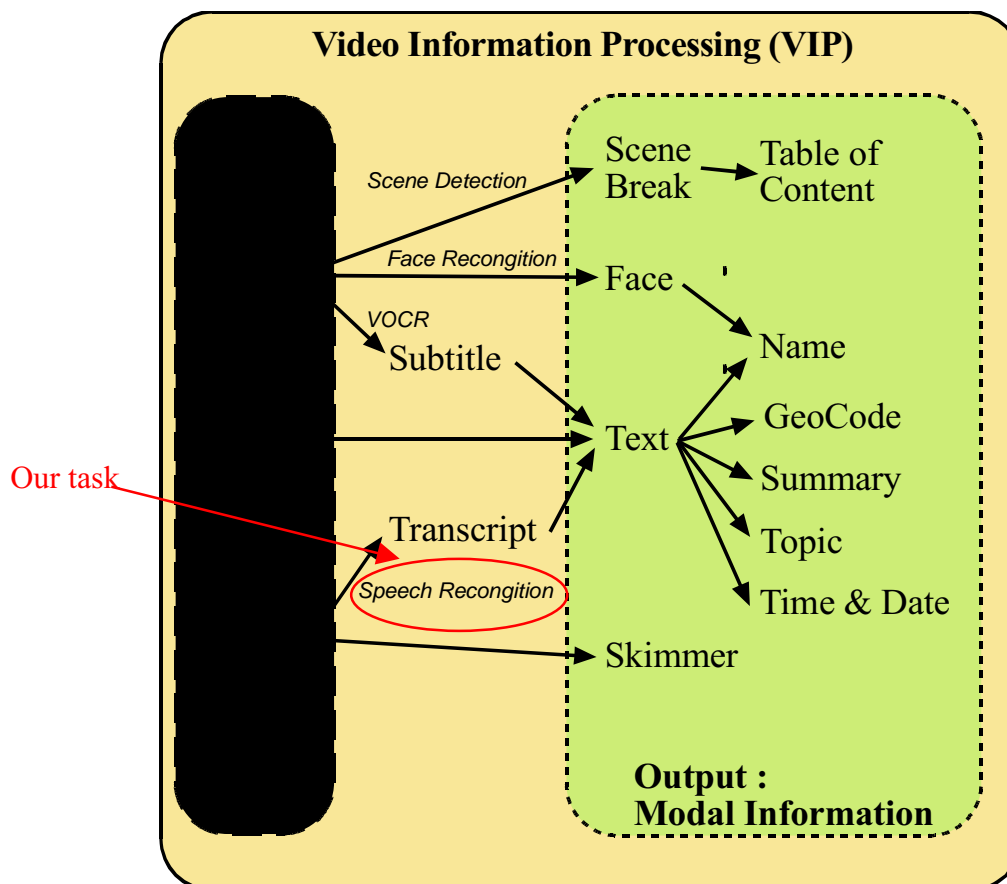


Figure 1.1 Video Information Processing

The information processing has major components including: speech recognition, video OCR, scene detection, face recognition, etc.

1.1.3 Speech Recognition Engine

A speech recognition engine is a software component, which is able to handle low-level speech recognition tasks. The nature of our project requires a state-of-the-art high quality speech recognition engine, which can dictate the speech in a large volume of video segments and produce text with an acceptable accuracy. Usually, a speech engine is a big component, which requires large amount of investment and human recourses to develop. Therefore, we are not going to develop

an engine ourselves. Instead, we use other commercial engines to provide a programmable low-level speech API for us to work on. There are several commercial speech engines available, such as Dragon Systems, Microsoft SAPI, and IBM ViaVoice. After a comparison study, we chose IBM ViaVoice.

1.2 Project Objectives

After introduce the Multilingual Digital Video Content Hub, our task in the project is not hard to see: apply speech recognition techniques for the video data obtained from digital video library to retrieve certain useful information, including the text of the speech and the timing of every word spoken.

The speech information will enable the video information processor to produce captions for videos and highlighting the word, which is currently being spoken, at correct time while playing videos. In order to achieve our goal, we need to perform multi-lingual speech recognition, timing information retrieval, and real-time dictation. More specifically, we need to embed the ViaVoice in the whole system and try to increase the accuracy of the speech recognition engine as much as possible. The ultimate goal is to display the video as well as showing the text of speaker's voice, as shown in figure 1.2.



Figure 1.2 Video with Speech Recognition Processing

Since speech recognition is heavily used in our project, we will give a brief introduction to it in the next chapter.

Chapter 2 : Introduction to Speech Recognition

Automatic Speech Recognition (ASR) is a process by which a computer (or an equivalent type of machine) identifies spoken words from an input voice device. Basically, SR means talking to your computer, AND having it correctly recognize what you are saying.

2.1 Terminologies in SR System

In order to understand SR technology, we have to learn some terms first:

2.1.1 Utterance

An utterance is the vocalization (speaking) of a word or words, which represents a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.

2.1.2 Speaker Dependence Vs. Speaker Independence

A speaker-dependent system is a system that must be trained on a specific speaker in order to recognize accurately what has been said (We will explain what means “training” later in the section). To train a system, the speaker is asked to record predefined words or sentences that will be analyzed and whose analysis results will be stored. This mode is mainly used in dictation systems where a single speaker is using the speech recognition system.

On the contrary, any speaker without any training procedure can use speaker-independent systems. Those systems are thus used in applications where it is not possible to have a training stage (telephony applications, for example).

It is also clear that the accuracy for the speaker-dependent mode is better compared to that of the speaker-independent mode. Adaptive system usually starts as speaker independent systems and utilize training techniques to adapt to the speaker to increase their recognition accuracy.

2.1.3 Vocabularies

Vocabularies (or dictionaries) are a list of words or utterances that can be recognized by the SR system. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult. It is because that the SR system has more opportunities to make some errors in larger size. Unlike normal dictionaries, each entry doesn't have to be a single word. They can be as long as a sentence or two. Smaller vocabularies can have as few as 1 or 2 recognized utterances (e.g. "Wake Up"), while very large vocabularies can have a hundred thousand or more!

A good SR system will therefore make it possible to adapt its vocabulary to the task it is currently assigned to. For example, enable a dynamic adaptation of the vocabulary.

2.1.4 Grammar

Grammars in SR system refer to the rules of the language model. Without grammar, every word will have an equal probability of occurrence at any point in utterance. As a consequence, it makes continuous speech recognition impossible. In fact, grammar places constraints on the sequence of words allowed, and the possible next word choice. That gives the system fewer choices at each point, improving the performance of recognition.

2.1.5 Isolated Word Recognition Vs. Continuous Speech Recognition

Isolated word recognition is the simplest speech recognition mode and it requires less CPU resources compared with continuous speech recognition mode. Each word is surrounded by a silence so that word boundaries are clearly justified. The system does not need to find the beginning and the end of each word in a sentence. The word is compared to a list of words models, and the model with the highest score is retained by the system. This kind of recognition is mainly used in telephony application to replace traditional DTMF methods.

Continuous speech recognition is much more natural and user-friendly. It assumes the computer is able to recognize a sequence of words in a sentence, or a sequence of sentences. But this mode requires much more CPU and memory, and the recognition accuracy is really lower compared with the preceding mode. Here lists the major

problems faced with continuous speech recognition:

- a. Speakers' pronunciation is less careful
- b. Word boundaries are not necessarily clear
- c. Speaking rate is less constant
- d. There is more variation in stress and intonation.
- e. Additional variability is introduced by the unconstrained sentence structure
- f. Co articulation is increased both within and between words
- g. Speech is mixed with hesitations, partial repetition, etc.

2.1.6 Difficulty Level

Based on speech recognition mode, we classify SR difficulty in 10 levels (Figure 2.1). The first level is most simple one, which means that it is speaker-dependent, able to recognize isolated words, and using a small vocabulary. 10 correspond to the most difficult level, which is speaker-independent continuous speech in a large size of vocabulary. State-of-the-art speech recognition systems with acceptable error rates are somewhere in between these two extremes.

	Isolated Words	Continuous Speech
Speaker Dependent	Small Vocabulary 1	Small Vocabulary 5
	Large Vocabulary 2	Large Vocabulary 7
Multi-Speaker	Small Vocabulary 2	Small Vocabulary 6
	Large Vocabulary 4	Large Vocabulary 7
Speaker Independent	Small Vocabulary 3	Small Vocabulary 8
	Large Vocabulary 5	Large Vocabulary 10

Figure 2.1 Difficulty level of Speech Recognition

2.1.7 Accuracy

The ability of a SR system can be measured by examining its accuracy. It means that

how well it will recognize the utterances, which includes not only correctly identifying an utterance, but also identifying if the spoken utterance is not in its vocabulary. Good ASR system has an accuracy of more than 90%. The acceptable accuracy of a system mainly depends on the application.

2.1.8 Training

As mentioned before, some of the SR system has the ability to adapt to a specific speaker. When the system has this ability, it can apply the training process. The ASR system is trained by letting that specific speaker to record predefined phrases and sentences and adjusting its comparison algorithm to match that particular speaker's voice. So consequently, training increases the accuracy of ASR system.

Training can also be used by speakers who have problems in speaking or pronouncing certain words. As long as the speaker can consistently repeat an utterance, ASR systems with training should be able to adapt it.

2.2 Speech Recognition Process

2.2.1 Process Diagram

The speech recognition process can be divided into different components, which is illustrated in figure 2.2:

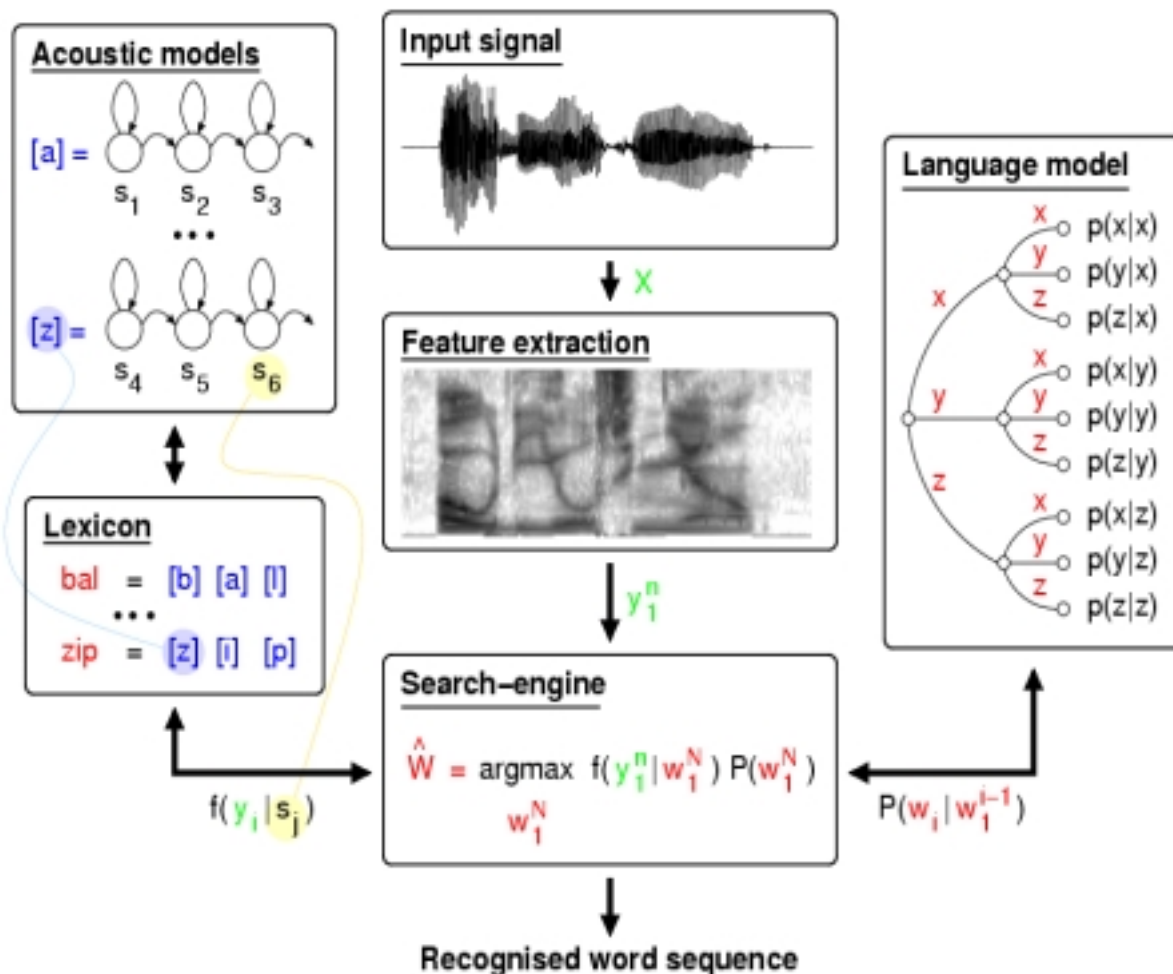


Figure 2.2 Speech recognition process

2.2.2 Process Explanation

2.2.2.1 Input Signal

First, we get the speaker's voice from an input device and save it as speech signals. The commonly used input device is a microphone. Note that the quality of the input device can influence the accuracy of SR system very much. The same applies to acoustic environment. For instance, additive noise, room reverberation, microphone position and type of microphone can all relate to this part of process.

2.2.2.2 Feature Extraction

The next block, which shows the feature extraction subsystem, is tried to deal with the problems created in first part, as well as deriving acoustic representations. The two

aims are separate classes of speech sounds, such as music and speech, and effectively suppress irrelevant sources of variation

2.2.2.3 Search Engine

The search engine block is the core part of speech recognition process. In a typical ASR system, a representation of speech, such as spectral or cepstral representation, is computed over successive intervals, e.g., 100 times per second. These representations or speech frames are then compared with the spectra frames, which were used for training. It is done by using some measurement of distance or similarity.

Each of these comparisons can be regarded as a local match. The global match is a search for the best sequence of words, in the sense that it is the best match to the data. And it is determined by integrating many local matches. The local match does not usually produce a single hard choice of the closest speech class, but rather a group of distances or probabilities corresponding to possible sounds. These are then used as part of a global search or decoding to find an approximation to the closest sequence of speech classes, or ideally to the most likely sequence of words.

2.2.2.4 Acoustic Model

The acoustic model is the recognition system's model for the pronunciation of words, crucial to translating the sounds of speech to text. In reality, the type of speaker model used by the recognition engine greatly affects the type of acoustic model used by the recognition system to convert the vocalized words to data for the language model (Context / Grammar) to be applied.

There are a wide variety of methods to build the pattern models, the three major types are:

- a. Vector Quantization
- b. Hidden Markov Models (HMM)
- c. Neural Networks

Due to the limitation of the size of this document, we are not going to present an in-depth discussion about the above methods here.

2.2.2.5 Language Model & Lexicon

Language model is another major component of the speech recognition process. It provides the knowledge source for the engine and helps to predict the next word.

The first component of the language model is the lexicon, which consists of the vocabulary. The vocabulary, as explained before, contains all of the possible words in the voice system that may be encountered.

The second component of the language model is the grammar. It defines the structure and format of the text allowed at any point in the utterance. Without grammar, every word in the lexicon would have an equal likelihood of occurrence at any point within the utterance, requiring algorithm that are too complex, making continuous speech impossible. These systems use the grammar to constrain the word choice at any point, which is more efficient.

2.3 Uses and Applications

Speech recognition system has a wide area of application. Here we just list some to get some points in mind:

2.3.1 Dictation

Dictation is the most commonly usage for ASR system. As a form of input, it is especially useful when someone's hands or eyes are busy. It allows people working in active environment such as hospitals to use computers. With ASR system, typing is not a necessary skill for using a computer. If we can successful get 100% accuracy of speech recognition, it would make computers accessible to people who do not want to learn the technical details to use them or just lazy to type words.

2.3.2 Command And Control

Imagine that in the near future, if we want to open a Netscape browser or click the start menu on Windows operating system, what we need to do is just say a word to the computer. People's hands are got free from mouse clicking forever.

2.3.3 Telephony

With the usage of ASR system, users can say the command in the telephone instead of pressing the buttons. It is more natural and convenient to do so.

2.3.4 Medical/Disabilities

Many people have difficulties in typing the words because of physical disabilities, such as blindness, muscular dystrophy and so on. Other people may have difficulties to hear the voice. With the ASR system, they can overcome these problems by talking to the engine or convert the caller's speech to text.

2.3.5 Embedded System Applications

It can also be used in embedded systems, for example a cell phone in the future can listen the command like "Call Jenny". Or a car can automatically drive itself by just hearing the voice form its host.

2.4 Challenges And Difficulties

Although SR is an amazing task, it has some challenges: mapping from a continuous time signal, the speech signal, to a sequence of discrete entities, such as words and sentences is the primary challenge of speech recognition processing.

Voice recognition is complex processing, for both discrete and continuous speech. The difficulties affect the process in the construction of the engine, acoustic and language models, the training of the system by users, and in practical application of the system, the major challenges are:

2.4.1 Linguistic Variability

Several factors will affect the input audio signal. Although these factors may only change the voice signal slightly, it can also complicates the speech recognition ability to accurately interpret the signal. These factors are:

- a. Phonetics

- b. Phonology
- c. Syntax
- d. Semantics
- e. Discourse

2.4.2 Speaker Variability

The speaker variability not only refers to the variability between two different speakers, but also refers to the variability in speech, which can occur by an individual. The factors that affect the output include:

- a. Tempo
- b. Pronunciation
- c. Inflection
- d. Fatigue
- e. Stress
- f. Upper Respiratory Infection

2.4.3 Channel Variability

Even the quality and position of microphone and background environment will affect the output of ASR system dramatically. So we have to deal with mechanical disruption and outside interference of voice signal itself, which includes:

- a. Background Noise
- b. Transmission Channel
- c. Quality and position of microphone

2.4.4 Coarticulation

Speech sounds of words are known as *Phonemes*. It is the characteristics of these

phonemes, which allows listeners to identify the words. However, phonemes that make up words are not said in isolation. Every sound used for speech is affected by the sounds both preceding and following it. Thus, no phoneme is an “island”. Every phoneme will be influenced by the phoneme before and after it.

Coarticulation then is referred to the contextual effects of the phonemes on other phonemes:

- a. The articulation of each sound blends with the articulation of the neighboring sounds (before and after)
- b. It is known that the phonetic contextual effects may span multiple phonemes, but most often involves the closest neighboring phonemes.

Chapter 3 : Speech Recognition Engines

During the summer holidays in 2001, we investigated different speech recognition software such as *CMU Sphinx*, *Microsoft SAPI* and *IBM ViaVoice*. And we also visited ***IBM Research Lab China*** and ***Microsoft Research Institute of China*** in Beijing together with our supervisor Professor Michael R. Lyu this summer.

With the background knowledge of speech recognition in hand, and also the investigation and visiting experience, it is ready for us to do a summary on what we have seen and what we have learned.

We would introduce different speech recognition software in this chapter, which have their own advantages and drawbacks, and give a brief comparison at the end.

3.1 CMU Sphinx



3.1.1 Introduction to CMU Sphinx

Sphinx is originated at Carnegie Mellon University (CMU) and developed by a group of faculty members and students in department of computer science in that university. It is funded by DARPA (Defense Advanced Research Projects Agency).

Since the born the Sphinx Group is dedicated to release a set of reasonably mature, world-class speech components that provide a basic level of technology to anyone interested in creating speech-using applications without the once-prohibitive initial investment cost in research and development; the same components are open to peer review by all researchers in the field, and are used for linguistic research as well.

3.1.2 Sphinx2 Vs. Sphinx3

In early 2000, the Sphinx Group release Sphinx2, a real-time, large vocabulary, speaker-independent speech recognition system as free software. It is the engine used in the Sphinx Group's dialog systems that require real-time speech interaction, such as a many-turn dialog for travel planning. The pre-made acoustic models include American English and French in full bandwidth. Sphinx 2 is a decent candidate for

handheld, portable, and embedded devices, and telephone and desktop systems that require short response times.

At the same year, Sphinx 3, a slower, yet more accurate recognizer is released. It is used, for example, broadcast news transcription. An Acoustic Trainer, allowing anyone to create acoustic models for conditions, channels, and domains, is released as well.

3.1.3 SphinxTrain

SphinxTrain is the acoustic training environment for CMU Sphinx (both sphinx2 and sphinx3), which was first publicly released on June 7th, 2001. It is a suite of programs, script and documentation for building acoustic models from data for the Sphinx suite of recognition engines. With this contribution, people should be able to build models for any language and condition for which there is enough acoustic data.

3.2 Microsoft SAPI



3.2.1 Introduction to Microsoft SAPI

The Microsoft Speech API (SAPI) is a software layer allowing speech-enabled applications to communicate with speech engines (Speech Recognition (SR) engines and Text to Speech (TTS) engines). SAPI includes an Application Programming Interface (API) and a Device Driver Interface (DDI). Applications communicate with SAPI via the API layer and speech engines communicate with SAPI via the DDI layer.

3.2.2 Responsibilities of SAPI and SR Engine

A speech-enabled application and an SR engine do not directly communicate with each other -- all communication is done via SAPI.

SAPI takes responsibility for a number of aspects of a speech system, such as:

- Controlling audio input, whether from a microphone, files, or a custom audio source; and converting audio data to a valid engine format.
- Loading grammar files, whether dynamically created or created from memory, URL or file; and resolving grammar imports and grammar editing.
- Compiling standard SAPI XML grammar format, and conversion of custom grammar formats, and parsing semantic tags in results.
- Sharing of recognition across multiple applications using the shared engine. All marshalling between engine and applications is done by SAPI.
- Returning results and other information back to the application and interacting with its message loop or other notification method. This allows an engine to have a much simpler threading model than in SAPI 4, as SAPI 5 does much of the thread handling.
- Storing audio and serializing results for later analysis.
- Ensuring that applications do not cause errors -- preventing applications from calling the engine with invalid parameters, and dealing with applications hanging or crashing

The SR engine is responsible for:

- Using SAPI grammar interfaces and loading dictation.
- Performing recognition.
- Polling SAPI in order to learn about grammar and state changes.
- Generating recognitions and other events to provide information to the application.

3.2.3 SAPI SR Objects and Interfaces

In order for an SR engine to be a SAPI 5 engine, it must implement at least one COM object. Each instance of this object represents one SR engine instance. The main interface this object must implement is the `ISpSREngine` interface. SAPI calls the engine using the methods of this interface to pass details of recognition grammars and tell the engine to start and stop recognition etc. SAPI itself implements the interface `ISpSREngineSite`. A pointer to this is passed to the engine and the engine calls SAPI using this interface to read audio, return recognition results etc.

ISpSREngine is the main interface that needs implementing, but there are other interfaces that an engine may implement. The SR engine can implement the ISpObjectWithToken interface. This provides a mechanism for the engine to query and edit information about the object token in the registry used to create the engine.

There are two other interfaces than the engine can also implement. These each need to be implemented in separate COM objects, because SAPI needs to create and delete them independently of the main engine. These interfaces are:

- ISpSRAlternates, which can be used by the SR engine to generate alternates for dictation results. It is possible to generate alternates without this interface, but this interface allows alternates to be generated off-line, after the result has been serialized.
- ISpTokenUI allows engines to implement UI components that can be initialized from an application. These can be used to perform user training, add and remove user words, calibrate the microphone etc.

The engine can also implement another COM object allowing engine-specific calls between the application and the engine. This object can implement any interfaces, which the application is able to QueryInterface for.

3.3 IBM ViaVoice



3.3.1 What is IBM ViaVoice?

In 1994, IBM was the first company to commercialize a dictation system based on speech recognition. Since then IBM devoted to develop the most advanced technology in this field.

ViaVoice is the speech recognition system that developed by IBM and it supports 13 different languages. In September 1997, ViaVoice Chinese version came into being

and it draw much attention to the people who concerned. It successfully solves the problems like there are many characters with the same pronunciation and different meaning in Chinese, Chinese language has different tones, etc.

3.3.2 System Characteristics

Since the first Chinese version of ViaVoice speech recognition system was born, IBM devoted to increase the accuracy of system. In 1999, they enhanced the system to be a user independent, no vocabulary constraint, and continuous speech recognition system with high recognition accuracy. And now they have three different versions including Mandarin, Taiwanese and Cantonese.

3.3.3 ViaVoice Native Architecture Overview

The heart of a speech recognition system is known as the speech recognition engine. The speech recognition engine recognizes speech input and translates it into text that an application can understand. Application can access the engine through a speech recognition API. For ViaVoice, this API is known as Speech Manager API, or SMAPI, for short. SMAPI is a conventional API, meaning that the API is defined as a part of resource; in ViaVoice, SMAPI is defined as part of the speech engine. With an API, speech becomes a resource to all applications.

3.3.4 ViaVoice Speech Engine Architecture

The speech engine has a rather complex task to handle, that is, taking the raw audio input and translating it to recognized text that an application understands.

The audio input source module encapsulates the methods used by the engine to retrieve the audio input stream. By default, the engine retrieves its audio input from the standard microphone input device in the system. A developer can write a custom audio library so that the input of the engine would be a custom piece of hardware.

The acoustic processor takes raw audio data and converts it to the appropriate format for use. The acoustic processor consists of two components: the signal processor and the labeler.

In the ViaVoice engine, audio input picked up by the microphone is analyzed by the

signal processor. This raw audio data is captured at 22 kHz by default, but 11 kHz and 8 kHz sampling is also supported. It contains both speech data and background noise.

3.3.5 Application Programming Interfaces

ViaVoice SDK provides several programming interfaces that developers can use to incorporate speech into their application.

- *Speech Manager APIs (SMAPI)*: IBM speech recognition engine APIs
- *SMAPI Grammar Compiler API*: APIs used to compile grammars used by the speech recognition engine.

3.3.5.1 SMAPI

There is significantly more function in the ViaVoice engine beyond raw recognition of spoken words, including dynamic vocabulary handling, database functions to query and select installed users, languages, and domains, and the ability to add new words to the user's vocabulary. SMAPI supports:

- Verifying the API version
- Establishing a database session to query system parameters

SMAPI is provided as a library, which is linked into an application. The engine is a separate executable. The ViaVoice architecture supports many speech applications through a single engine, connected to one microphone.

3.3.5.2 SMAPI Grammar Compiler API

The ViaVoice SKD provides the ability for developers to compile grammars programmatically. This function is known as the SMAPI Grammar Compiler API and supports:

- Specifying parameters to the grammar compiler
- Compiling a grammar
- Obtaining error messages from the grammar compiler

The SMAPI Grammar Compiler APIs are provided as a library, which is linked into an application.

3.4 Comparisons of Different SR Systems

After introducing three different SR systems, we want to do a comparison to show their characteristics and drawbacks.

	CMU Sphinx	Microsoft SAPI	IBM ViaVoice
Characteristics	<ul style="list-style-type: none"> a. Open source b. Free software, no need for initial investment c. Good for researchers and developers d. With relatively high quality 	<ul style="list-style-type: none"> a. Applications and SR engine communicate with SAPI b. DDI and API remove implementation details making speech synthesis and SR engine and application convenient 	<ul style="list-style-type: none"> a. SAPI is define as a part of SR, which becomes the source of application b. Support 13 languages, including Cantonese and Chinese c. Developers can write audio library to handle input d. Support for Grammar Compiler APIs e. Support Dynamic vocabulary handling, database querying, add new words to the user's vocabulary
Drawbacks	<ul style="list-style-type: none"> a. Limited and hard to read documentation b. No Chinese version, although SphinxTrain can be used to build any language model, it needs large volume of acoustic data and investigation of the system c. SphinxTrain is only available in Unix system d. Acoustic build process can take many days (or longer) 	<ul style="list-style-type: none"> a. Has to implement COM objects and interfaces for SR engine to be a SAPI 5 engine b. Limited language version c. Do not support grammar compiler 	<ul style="list-style-type: none"> a. Constrained input audio data format b. Relatively low accuracy without training

Figure 3.1 Comparison of different SR systems

3.5 Why Choose ViaVoice?

As you can see from above table of comparison, IBM ViaVoice is the most distinguishable SR system. It has a high accuracy of dictation if fully trained. It uses 150,000-word base dictionary and user can add up to 64,000 words of their own. What's more important, it provide both Cantonese and Chinese version, which enable us to integrate it as a part of VIEW project.

When we visited IBM Research Lab China in Beijing this summer, we were really impressed by the outstanding performance of their product such as real-time dictation and telephony system, as well as the intelligence of the people there.

3.6 Our Objective With ViaVoice

In late September, VIEW lab got the ViaVoice SR system from IBM China. Our first task is to get the SR engine to work. And then, based on the knowledge we learned and SMAPI developer's guide, we are trying to increase the accuracy of the speech recognition engine as much as possible and obtain the timing information of speech. We apply some techniques such as audio extraction, segmentation and training to the raw audio data.

Although we have get some work done, SR is not an easy task. We meet some difficulties when doing the project. It will be describe at the end of our report. In the next a few chapters, we will describe the implementation detail of our project.

Chapter 4 : Audio Extraction

Our project and also the speech recognition engine mainly deal with audio data. But in the digital video library, most data are stored as multimedia files, a mixture of both video and audio data. Therefore, the first step we need to take is to extract audio data from these multimedia files. Further more, we also need to apply some speech recognition techniques on these audio data. Thus, it is more convenient for us to store these data on some persistent storage rather than real-time extracting them out when needed. The IBM ViaVoice engine supports only monotonic, 22/16/8 kHz ACM data. Hence, we decided to store these audio data in monotonic, 22 kHz wave format.

Under Win32 environment, Microsoft DirectX provides a convenient multimedia library. We have developed our program for audio extraction right on the top of DirectX API. In the next section, we are going to introduce DirectX in more detail.

4.1 Microsoft DirectX

Microsoft® DirectX® is a set of low-level application programming interfaces (APIs) for creating games and other high-performance multimedia applications. Microsoft® DirectX® 8.1 is made up of the following components: DirectX Graphics, DirectX Audio, DirectInput, DirectPlay, DirectShow, and DirectSetup. Among these components, Microsoft DirectShow® provides for high-quality capture and playback of multimedia streams. It is the right component for us to achieve our goal. In order to write DirectShow program, we have to understand Microsoft DirectShow first.

4.1.1 DirectShow Overview

The Microsoft® DirectShow® application programming interface is a media-streaming architecture for the Microsoft Windows® platform. It supports a wide variety of formats, including Advanced Streaming Format (ASF), Motion Picture Experts Group (MPEG), Audio-Video Interleaved (AVI), MPEG Audio Layer-3 (MP3), and WAV files.

DirectShow is based on the Component Object Model (COM). Most provided components, including filters and graphs, are in the form of COM-compliant object.

DirectX programmers need to know at least how to use existing COM object.

4.1.2 DirectShow Application Programming

At the heart of DirectShow services is the modular system of pluggable components called filters, arranged in a configuration called a filter graph. A component called the filter graph manager oversees the connection of these filters and controls the stream's data flow.

Filter

The basic building block of DirectShow is a software component called a filter. A filter generally performs a single operation on a multimedia stream. For example, there are DirectShow filters that

- Read files
- Get video from a video capture device
- Decode a particular stream format
- Pass data to the graphics or sound card

Filters receive input and produce output. For example, if a filter decodes MPEG-1 video, the input is the MPEG-encoded stream and the output is an uncompressed RGB video stream.

Filter Graph

To perform a given task, an application connects several filters so that the output from one filter becomes the input for another. A set of connected filters is called a *filter graph*. As an illustration of this concept, the following diagram (figure 4.1) shows a filter graph for playing an AVI file:

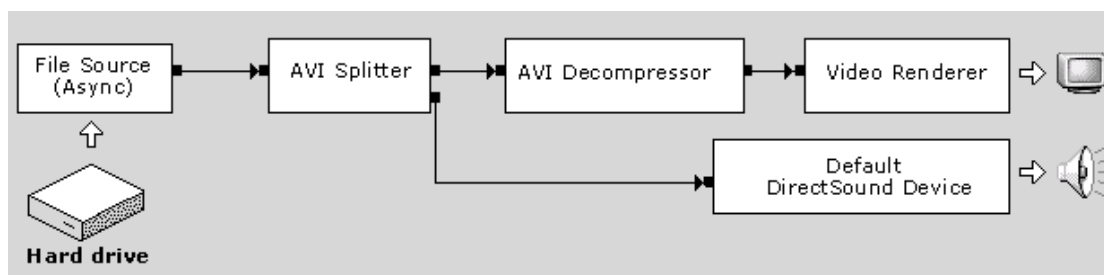


Figure 4.1 DirectShow Filter Graph

Filter Graph Manager

DirectShow provides a high-level component called the *Filter Graph Manager*. The Filter Graph Manager controls the flow of data through the graph. Our application makes high-level API calls such as "Run" (to move data through the graph) or "Stop" (to stop the flow of data). If we require more direct control of stream operations, we can access the filters directly through COM interfaces. The Filter Graph Manager also passes event notifications to the application, so that our application can respond to events, such as the end of a stream.

Writing a DirectShow application

A typical DirectShow application performs three basic steps, as illustrated in the following diagram.

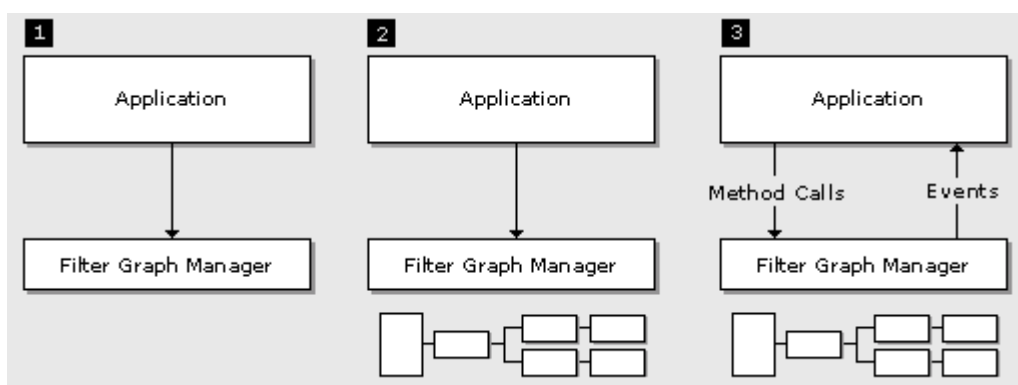


Figure 4.2 Process of Writing DirectShow Applications

1. Creates an instance of the Filter Graph Manager, using the **CoCreateInstance** function.

2. Uses the Filter Graph Manager to build a filter graph. (Other DirectShow helper components can be used here as well.)
3. Controls the filter graph and responds to events.

4.2 Implementation of Audio Extractor

The implementation of audio extractor employs some special filters to construct a filter graph, which is capable of converting other wave formats to the format desired by ViaVoice engine. This graph is also capable of storing wave streams into files.

4.2.1 Special Filters Employed in Our Program

Since our purpose is a little bit special, we need some special filters other than standard playback filters. There are two kinds of special filters needed. First, a transform filter is needed to produce the desired wave format. Second, a wave file writer is needed to write the extracted stream onto persistent storage. In our program, an ACM wrapper filter handles the transformation; a WavDest filter and a File Writer filter works together to handle the stream saving task.

ACM Wrapper Filter

The ACM Wrapper filter enables codes that use the Audio Compression Manager (ACM) to join a filter graph. It can act either as a decompression filter or as a compression filter. Decompression is only to PCM audio. It is used only for playback or wave format conversion.

In our program, we use it to convert the input PCM stream to monotonic, 22.05 kHz format. Since decompression is only to PCM audio, we connect the upstream to an audio decoder, which can produce PCM data. In DirectX, wave format is defined as a structure **WAVEFORMATEX**. The definition is

```
typedef struct {  
    WORD wFormatTag;           // Should be WAVE_FORMAT_PCM  
    WORD nChannels;           // Number of channels in the waveform-audio data  
    DWORD nSamplesPerSec;     // Sample rate, in samples per second (hertz)  
    DWORD nAvgBytesPerSec;    // required average data-transfer rate, in bytes per second
```

```
WORD nBlockAlign;      // the minimum atomic unit of data, in bytes
WORD wBitsPerSample;   // Bits per sample, usually equals 8 or 16
WORD cbSize;           // Size, in bytes, of extra format information appended
} WAVEFORMATEX;       // defines the format of waveform-audio data
```

In our program, the format information is store in a WAVEFORMATEX structure pointed by pFormat. The core part of conversion is the following two lines of code:

```
pFormat->nSamplesPerSec = 22050;
pFormat->nChannels = 1;
```

To reserve the validity of the format, we need to update other fields accordingly:

```
pFormat->nBlockAlign = pFormat->nChannels * pFormat->wBitsPerSample / 8;
pFormat->nAvgBytesPerSec = pFormat->nBlockAlign * Format->nSamplesPerSec;
```

Now the upstream PCM audio data should be converted to monotonic, 22.05 kHz format.

WavDest Filter & File Writer

The WavDest filter multiplexes an audio stream to produce a stream, which is capable of being written to a file. It takes a single audio stream as input, and its output pin must be connected to the File Writer filter, which is capable of writing a multiplexed stream to a file. These two filters are used to save the stream produced by the ACM wrapper into a file.

4.2.2 Filter Graph of Audio Extractor

Now we use an example to illustrate the flow of the audio extractor visually. Since nowadays the most popular video format is MPEG, we choose an MPEG-I file, called “*tvbnews.mpg*”, as the target to convert. The normal playback filter graph of the target file is shown in figure 4.3. On the top of the graph, a source filter is responsible for grabbing media stream from input file. Then the MPEG-I Stream Splitter divides the obtained media stream into two separate streams: video and audio. After that, the MPEG Audio Decoder and MPEG Video Decoder decode the corresponding stream data and pass them to output devices, namely Video Renderer and DirectSound Device, to render out.

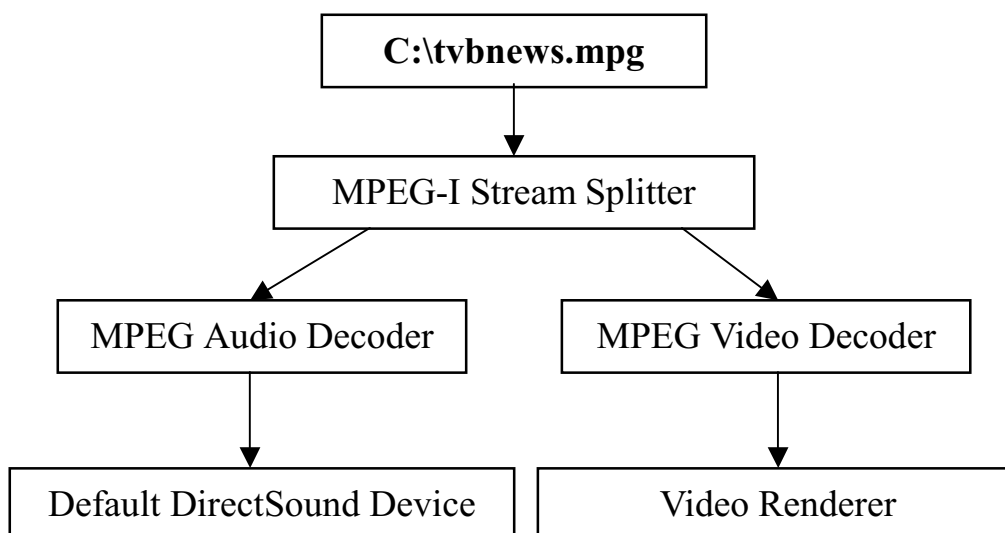


Figure 4.3 Filter Graph for Displaying Video Files

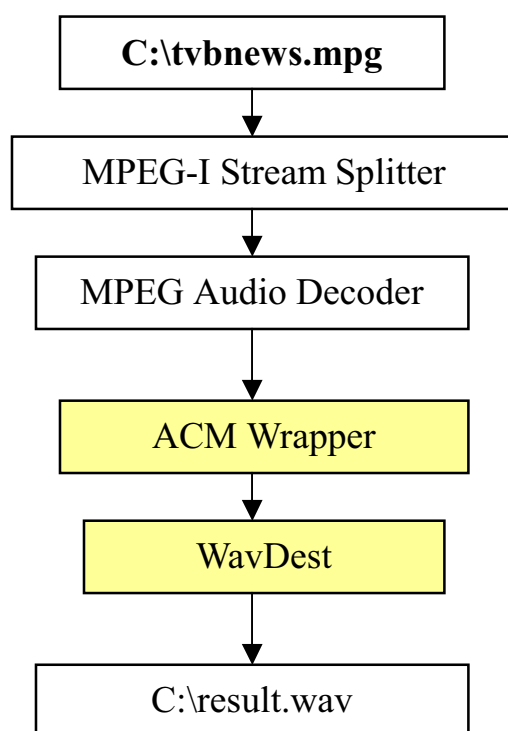


Figure 4.4 Filter Graph of Audio Extractor

The filter graph of audio extractor is shown in figure 4.4. The video data is no longer

needed. So the video decoder is removed. And there is no need to render the streams to users; the sound and video render devices are also removed. Instead, the output stream of the MPEG Audio Decoder is directed to an ACM Wrapper, which convert the wave format and passes the stream to a WavDest Filter, connected with a File Writer, to save the output to a file.

4.3 Dictation by Untrained ViaVoice Engine

After audio extraction, the extractor generated a wave file named “*tvbnews.wav*”, which is in the right format (monotonic, 22 kHz), from the original MPEG file “*tvbnews.mpg*” (A frame of this file is shown in figure 4.4). Now the ViaVoice speech engine could process the speech data in “*tvbnews.wav*” using a customized audio library named AudWav. (For more information on AudWav audio library, please refer to Chapter 6, Section 6.2.9.) Figure 4.5 is a screen shot of the dictation interface.



Figure 4.4 Media File “*tvbnews.mpg*”



Figure 4.5 Dictation Result of “*tvbnews.wav*”

The results are shown in figure 4.6. There are 165 characters out of 249 characters that are recognized correctly. Thus the accuracy of untrained ViaVoice engine is approximately 66.3% relative to the speech in “*tvbnews.wav*”. We can see that the accuracy is not very high. Therefore, we need to do something to improve the performance of ViaVoice engine.

The next a few chapters will introduce the techniques we used to improve the accuracy of the ViaVoice speech engine.

<p>本港通縮的情況係持續惡化，四月份的消費物價指數下跌了百分之三點八。比較係三月份的百分之二點六，係再下跌多一點二個百分點。係八一年以來最大的跌幅。羅佩瓊報道。三月份開始，長途電話公司出現減價戰，加上有新報紙發行，引發多份報章減價促銷。政府發言人表示，隨著本地物價同埋成本持續下降，零售商普遍提供價格折扣，令到四月份的消費物價連續第六個月出現收縮。四月份消費物價指數下跌百分之三點八，比三月份的百分之二點六仲要低，其中跌幅最大的係衣服鞋襪，跌百分之二十幾。而其它耐用物品、燃料、電費、租金都有好大跌幅。而上昇的只有煙酒同埋交通費。亞洲電視記者羅佩瓊。</p>	<p>東江 供水、快節奏 我發聲 份一些能 物價指數 下跌近百分之三點八五的大 三月份約百分之 一 古典樂派 作客的多一點 糊個百分點 堤壩一年以來最大的 跌幅達的 田發伏 三月份開始 進德伊萬公司 雖然減價戰 加上有新報紙發行 一百多份報章 減價促銷 政府發言人表示 追豬分泌物 格曼成本 持續下降 零售商方面 提供價格折扣 令四月份消費物價 連續第六個月 除九十非凡 消費物價指數 下跌百分之三點八 比三月份約百分之二點六 重要大 其中跌幅最大的 去衣服鞋襪 跌百分之二十四 以其他耐用物品 燃料 電費 租金 由大跌幅已上升約 只有煙斗為簡化 亞洲電視記者 劉佩瓊</p>
--	--

(a) Result of Human Recognition

(b) Result of ViaVoice Engine

Figure 4.6 Recognition Results by Human and ViaVoice Engine

Chapter 5 : Speech Segmentation

As shown above, the dictation result produced by an untrained ViaVoice engine is full of errors. So we need to apply some speech recognition techniques to improve the accuracy of the engine. One of these techniques is to train certain number of models for the engine. It is being expected that such training will offer a perceptible increase in dictation accuracy.

Before actually train the engine, we need to some preprocessing on the audio data. The first one is to use speech segmentation and classification techniques to achieve silence removal.

5.1 Why To Do Speech Segmentation

There are mainly three reasons to do speech segmentation.

First, storage consumption is considerable for multimedia files. The silence in speech does consume disk space. Silence removal could save some storage.

Second, the training of ViaVoice engine is not done randomly. We need to provide some guidance to the engine. We plan to compare the real texts and the dictation result of the engine and pick up only those words, which are recognized correctly, to feed to the engine during the training process. Most of these real texts will be human input. Due to the usually long duration of media files, it will be more convenient to the speech interpreter, which is a human but not a machine, if the whole speech is segmented sentence by sentence and played back a single sentence once.

Third, during the training, we need to compare the real texts and the result produced by the speech engine to retrieve the words, which are recognized correctly. Such a comparison is done by a string alignment algorithm. The outcome of such a string alignment algorithm will be better if the length of the compared strings is shorter. And also the computation time increased with the length of input strings dramatically. Therefore, it will be better to do this string alignment sentence by sentence rather than on the whole lengthy speech.

5.2 Overview of Different Approaches

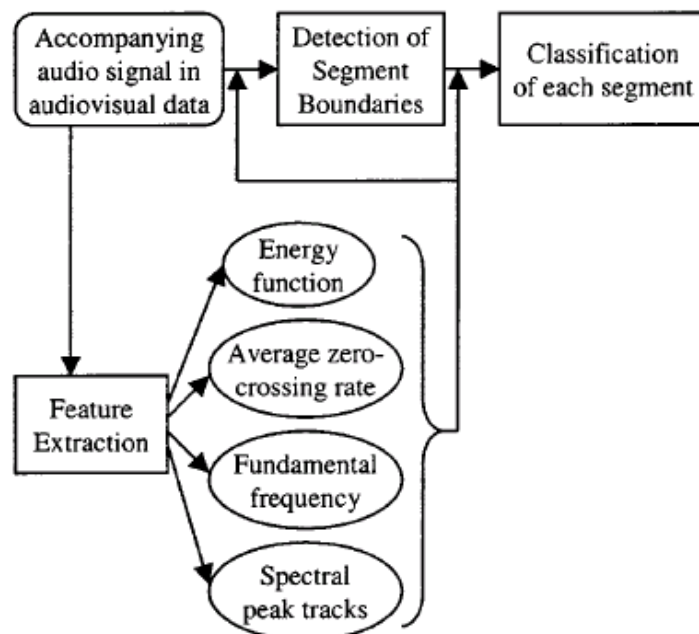


Figure 5.1 Speech Segmentation Using Feature Extraction

The basic idea of speech segmentation is to use certain features of the boundary data to determine the boundary position of each segment. This is called feature extraction. Figure 5.1 shows the flow of speech segmentation procedure using feature extraction. As shown in the above figure, there are different approaches to detect the boundaries of speech segments.

- *Energy function*: define a short-time energy function of an input audio signal. It provides a convenient representation of the amplitude variation over time.
- *Average zero-crossing rate*: for discrete-time signals, a zero-crossing is said to occur if successive samples have different signs. The rate at which zero-crossing occur is a simple measure of the frequency content of a signal. It can be used as another measure to distinguish between voiced and unvoiced speech because unvoiced speech usually has much higher ZCR values than voiced ones.
- *Fundamental frequency*: a harmonic sound consists of a series of major

frequency components including the fundamental frequency and those which are integer multiples of the fundamental one. We may divide sounds into two categories, i.e., harmonic and nonharmonic sounds. Whether an audio segment is harmonic or not depends on its source. The speech signal is a harmonic and nonharmonic mixed sound, since voiced components are harmonic while unvoiced components are nonharmonic. We can also use this characteristic to separate them.

5.3 Frame Energy Analysis

Upon the current phase of our project, the feature extraction target is mainly silence. The frame energy model is better at detecting silence than other approaches. Hence, we choose it for the implementation of this phase.

This approach works because the energy for unvoiced speech components is in general significantly smaller than those of the voiced components, as shown in Figure 5.3. The low energy parts in the graph show the position of silences in the speech. This feature provides a basis for distinguishing voiced speech components from unvoiced ones.

How to measure the energy of speech? In fact, there is some relationship between energy and wave amplitude of speech. Figure 5.2 shows the waveform of the same speech as in figure 5.3. It is not hard to see that the lower the amplitude, the higher the energy. And the amplitude data can be retrieved from the wave file. Hence, we established a threshold on wave amplitude to determine silence in the speech. If the amplitude of certain sample is below the threshold, which means its energy is considerably low, then it is classified as silence.

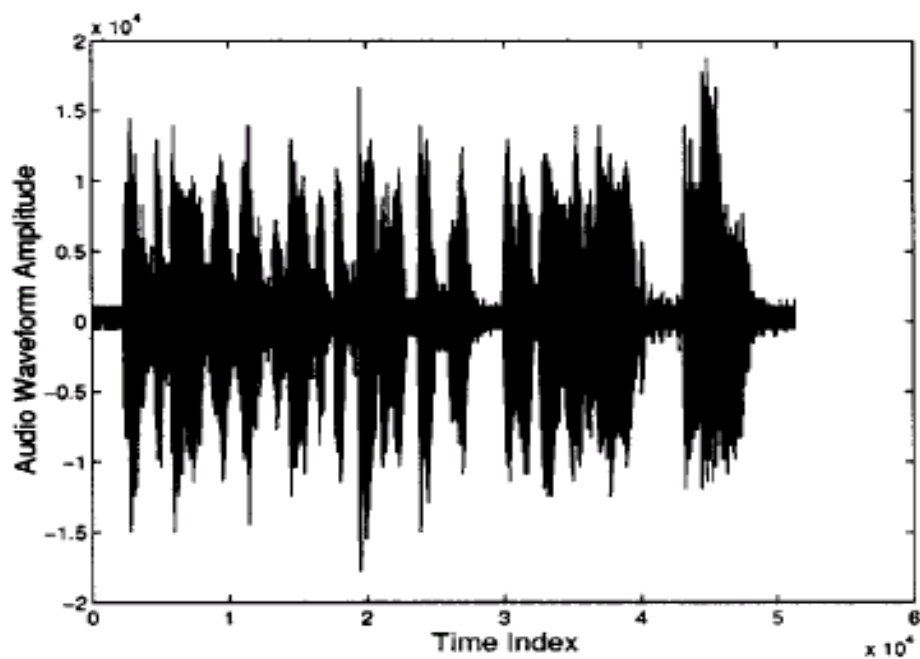


Figure 5.2 Waveform of a Speech Segment

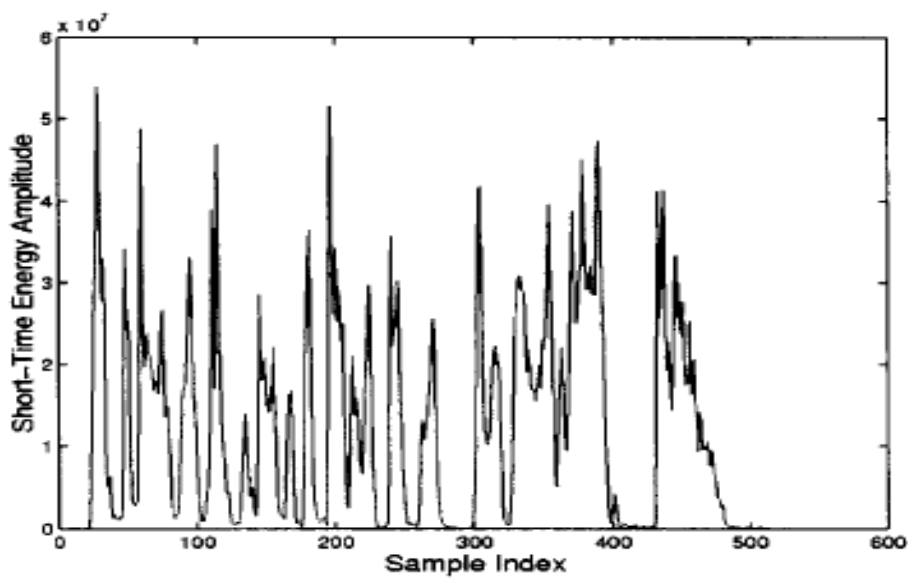


Figure 5.3 Frame Energy Graph of a Speech Segment

5.4 Segmentation Result

We implement the segmentation program using Matlab. A C/C++ version of this program is currently under development. Once again we come back to the old sample file “*tvbnews.mpg*” (figure 4.4 in Chapter 4), which is used as an example in Section 4.3. After extracting the audio data and saving in a file named “*tvbnews.wav*”, we segment the speech in this file into different parts based on the frame energy of the wave. The result is shown visually in figure 5.4. The small green dots in the wave diagram indicate silence, which, in turn, segments the speech into different parts. Every part of the green line on the top of the graph represents a segment of the original speech. There are 24 such segments.

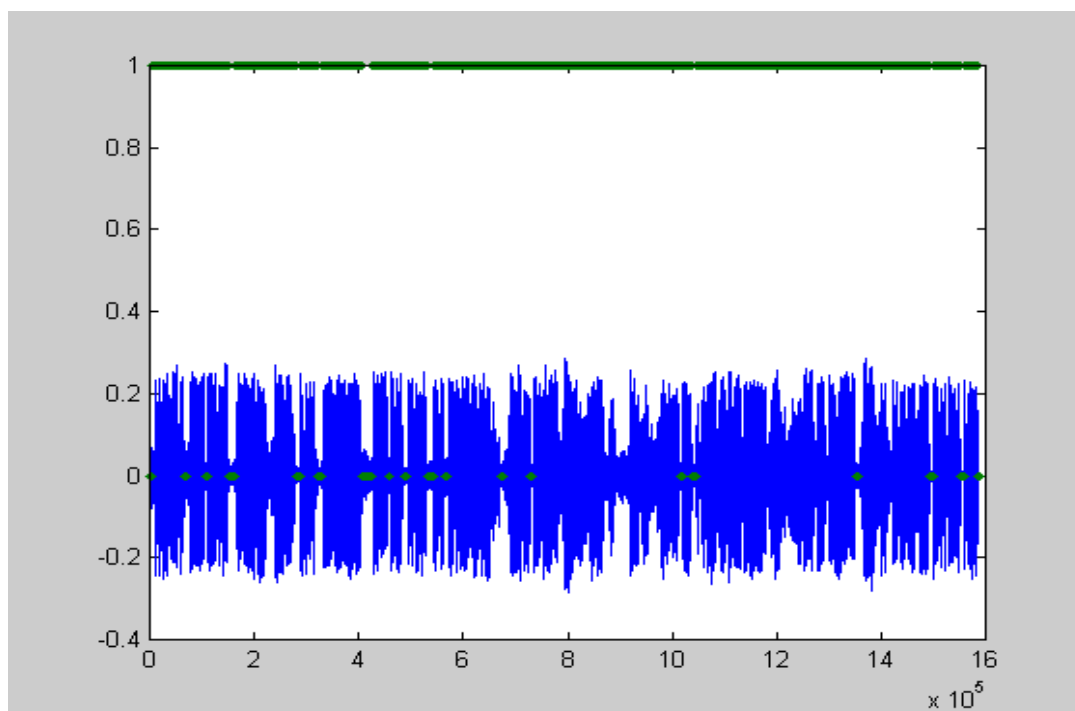


Figure 5.4 Result of Segmentation on file “*tvbnews.wav*”

The result is stored in a file named “*tvbnews.txt*”, which records the start and end sample index of each segment. For example, the first two numbers stored in “*tvbnews.txt*” are 9901 and 133100. This means the first segment begins at the 9901st wave sample and ends at the 133100th wave sample. And also the 24 segments are saved separately as 24 small wave files for further manipulation.

Chapter 6 : Visual Training Tool

As mentioned before, training of a speech recognition system means letting the specific speaker to read a predefined paragraph of text and adjusting the language model to match that particular speaker's voice. However, in our system, we cannot employ the speaker to record his/her voice beforehand. For example, when we get the audio file from TVB news, it is impossible to find the speaker and ask him/her to do the training.

So in order to increase the accuracy of the ViaVoice system, our solution is training the system "manually". After preprocessing, the speech data are not ready for training yet. We need to compare the output of the speech engine with the correct texts and find those correctly recognized words. Before we can do this comparison, we need to collect large amount of text data. Such input work will require considerable effort under the condition that lack of the help of certain appropriate tool. Thus, we decided to implement a visual training tool to help us train the ViaVoice engine.

First, the tool displays the video, which contains that speaker's voice we want to train. When the user hears what the speaker is saying, he/she can type in the texts of what they have heard. In addition, the dictation output of IBM ViaVoice engine is also shown in the interface.

In order to let user catch up what the speaker is saying and record it correctly, we display the video sentence by sentence.

Second, we should use some algorithm to compare the two outputs (one by ViaVoice engine and one by user input). For those characters that are recognized as correct by speech recognition engine, we record them and use the data to do the training.

This tool is named WinTrain, which was developed using Microsoft Visual C++. The next section is going to introduce the features of the WinTrain tool.

6.1 WinTrain Features



Figure 6.1 Main Interface of WinTrain

This tool is running on Microsoft Windows with DirectX and IBM ViaVoice Runtime installed. The main window is split into three parts, as shown in figure 6.1. The left upper window is the video window, where the opened media file is played. The window just below the video window is the dictation window, where the result text of speech recognition is displayed. The right window is a text editor. It is for the user to input captions of the media file. After the document is saved, user input captions are also saved.

This tool can also process the speech segmentation information. Prior to open a media file, the tool will search for a text file with the same name of the media file. For example, if the media file is "*tvbnews.mpg*", the expected segmentation information file is "*tvbnews.txt*". If this file is present and is in a valid format, the tool will split the whole media content into segments, whose positions and durations are specified in

the segment file. If there is more than one segment in a media file, the previous and next button in the video window will be available and the video will be played one segment each time. At the end of each segment, the player will stop automatically until the user presses the next button. If the user presses the previous button, the currently playing segment will be stopped and the last segment will be load and played. The texts in the text editor are refreshed automatically according to the segment that is currently played. Only the texts corresponding to the currently played segment are shown in the editor.

Figure 6.2 (a through c) shows a multi-segment media being played by the tool. The segment being played in (a) is the first segment of the media. After the user presses the next button, the currently played segment will be switched to the second segment, which is shown in figure 6.2(b). If the user presses the next button again, the third segment will be played, as shown in figure 6.2(c). The user could go back to (b) from (c) or go back to (a) from (b) by pressing the previous button. Note that the editor text is also refreshed corresponding to the currently played segment.



(a) First Segment



(b) Second Segment



(c) Third Segment

Figure 6.2 (a ~ c) Demonstration on Video Segment Switching

There is a dictation button in the toolbar of the main window, as shown in figure 6.3. After this button is pressed, the speech of the currently played segment will be passed to the ViaVoice speech recognition engine, and the result will be shown in the dictation window. The result will be used further with the string alignment algorithm.



Figure 6.3 Dictation Button

The visual train tool is also capable of processing multiple media files simultaneously, as shown in figure 6.4.

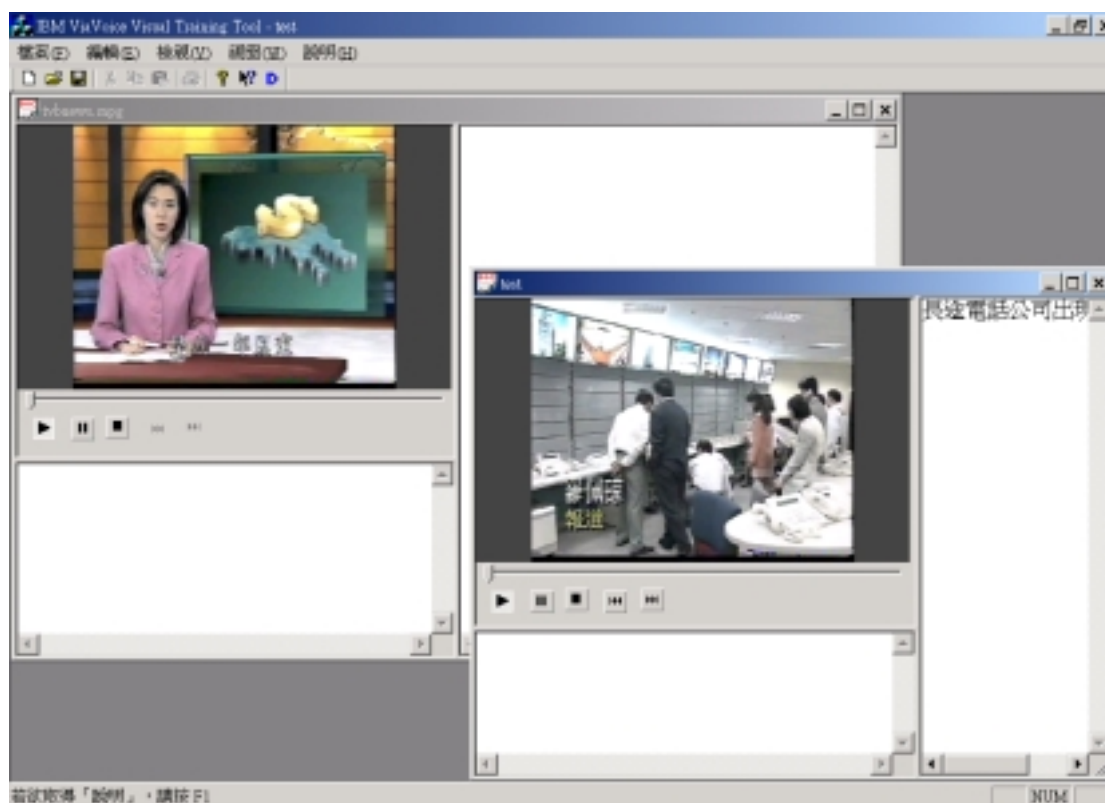


Figure 6.4 Multiple Documents Demonstration

Now we are going to discuss the implementation of the visual training tool.

6.2 Implementation of The Visual Training Tool

The visual training tool is developed using Microsoft Foundation Classes. The document/view architecture is adopted.

6.2.1 Document and View

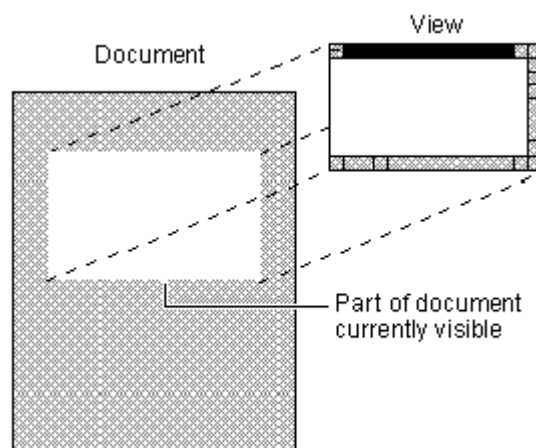


Figure 6.5 Document and View Architecture

The document/view implementation separates the data itself from its display and from user operations on the data. All changes to the data are managed through the document class. The view calls this interface to access and update the data.

Documents, their associated views, and the frame windows that frame the views are created by a document template. The document template is responsible for creating and managing all documents of one document type.

The key advantage to using the MFC document/view architecture is that the architecture supports multiple views of the same document particularly well. In the visual training tool, there are three separated views, including the video view, the dictation view, and the editor view. All these three views would be associated with a single document object. The document stores the data. All views access the document and display the data they retrieve from it. This scenario would be difficult to code without the separation of data from view, particularly if the views stored the data themselves.

6.2.2 Object Diagram

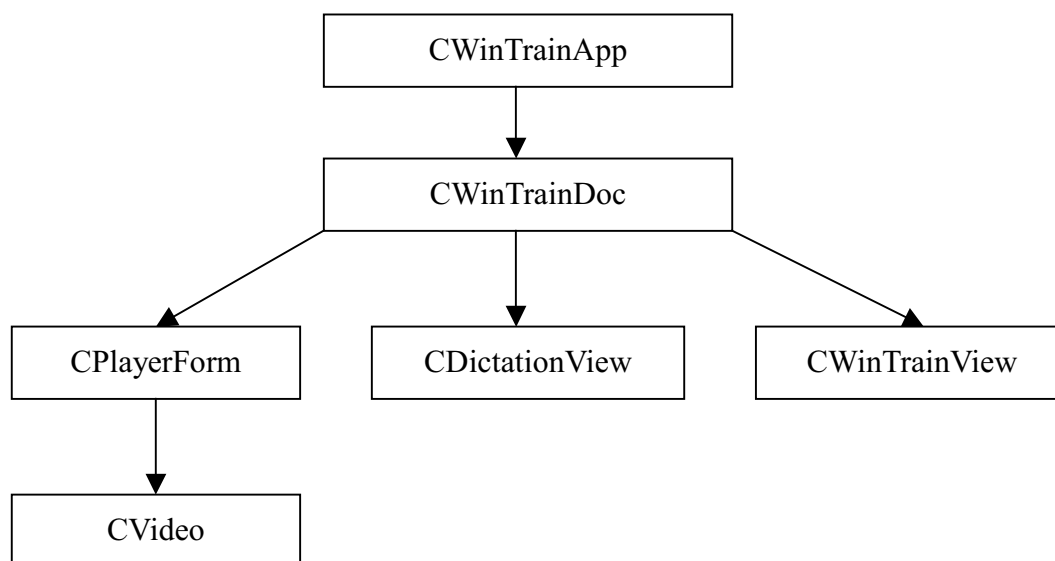


Figure 6.6 Object Diagram of The Visual Training Tool

6.2.3 CWinTrainApp

This is the class of the main application. There will be one and only one `CWinTrainApp` object. This object maintains a document template, which manages the interaction between the document and its associated views. The prototype of this class is

```
class CWinTrainApp : public CWinApp
{
public:
    CWinTrainApp();
    virtual BOOL InitInstance();
    virtual int ExitInstance();

// Implementation
    afx_msg void OnAppAbout();
    DECLARE_MESSAGE_MAP()
private:
    BOOL m_bATLInited;
    BOOL InitATL();
}
```

6.2.4 CWinTrainDoc

This is the one and only one type of document in the visual training tool. It manages the document data, including the media file, the segmentation information, and the user-entered texts. All views are updated according to the document object. The prototype of this class is

```
class CWinTrainDoc : public CDocument
{
protected: // create from serialization only
    CWinTrainDoc();
    DECLARE_DYNCREATE(CWinTrainDoc)
// Overrides
    public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
        virtual BOOL OnSaveDocument(LPCTSTR lpszPathName);
// Implementation
    public:
        void UpdateTextView(LONG index);
        void SetTextAt(LONG index);
        void SetTextAt(LONG index, CString text);
        CList<Entry, Entry&> & GetEntryList();
        CString GetMediaPath();
        virtual ~CWinTrainDoc();
protected:
        BOOL GetSegmentInfo(CString fileName);
        CList<Entry, Entry&> m_EntryList;
        CPlayerForm *GetPlayerForm();
        CEditView *GetEditView();
        CArray <CString, CString> m_TextArray;
        CFile m_textFile;
        CString m_textFileName;
        CString m_mediaFileName;
        afx_msg void OnAppDictate();
```

```
        DECLARE_MESSAGE_MAP()  
    }  
}
```

6.2.5 CPlayerForm

This form is the video view of the document. One of its responsibilities is to lay out the video window and control panel. There is a CVideo object embedded. The user inputs, such as play and stop and so on, are passing to it. This form is also responsible for update the CVideo object upon changed made to the document. The prototype of this class is

```
class CPlayerForm : public CFormView  
{  
protected:  
    CPlayerForm();          // protected constructor used by dynamic creation  
    DECLARE_DYNCREATE(CPlayerForm)  
// Form Data  
public:  
    enum { IDD = IDD_PLAYER_FORM };  
    CBitmapButton m_StopButton;  
    CBitmapButton m_PrevButton;  
    CBitmapButton m_NextButton;  
    CBitmapButton m_PauseButton;  
    CBitmapButton m_PlayButton;  
    CStatic m_Screen;  
// Operations  
public:  
    LONG GetCurrentEntry();  
    void SetMediaPath(CString path);  
// Overrides  
    public:  
        virtual void OnInitialUpdate();  
    protected:  
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support  
// Implementation  
protected:
```

```
CBitmap m_PlayBitmap;  
CVideo * m_Video;  
virtual ~CPlayerForm();  
  
//message map functions  
afx_msg void OnPlayButton();  
afx_msg void OnPauseButton();  
afx_msg void OnStopButton();  
afx_msg void OnNextButton();  
afx_msg void OnPreviousButton();  
DECLARE_MESSAGE_MAP()  
}
```

6.2.6 CDictationView

This view implements the dictation view of the visual training tool. If a user presses the dictation button, the currently played speech will be fed to ViaVoice engine to produce the text. And the result is displayed on this view. The prototype of this class is

```
class CDictationView : public CEditView  
{  
protected:  
    CDictationView();          // protected constructor used by dynamic  
creation  
    DECLARE_DYNCREATE(CDictationView)  
  
// Overrides  
protected:  
    virtual void OnDraw(CDC* pDC);    // overridden to draw this view  
  
// Implementation  
protected:  
    virtual ~CDictationView();  
    DECLARE_MESSAGE_MAP()  
}
```

6.2.7 CWinTrainView

This view implements a text editor, which is used for a user to type down what he hears from the media speech. The whole texts are also segmented, as the same way as the speech. Only texts related to the currently played speech segment are displayed.

The prototype of this class is

```
class CWinTrainView : public CEditView
{
protected: // create from serialization only
    CWinTrainView();
    DECLARE_DYNCREATE(CWinTrainView)

// Attributes
public:
    CWinTrainDoc* GetDocument();

// Overrides
public:
    virtual void OnDraw(CDC* pDC); // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);

// Implementation
public:
    virtual ~CWinTrainView();
}
}
```

6.2.8 CVideo

This class implements a customized media player, which supports segmentation of the media content. The segmentation information is provided during construction. If no such information supplied, the default segmentation is to regard the whole media

content as a single segment. All operations other than previous and next, including play, pause, and stop, are relative to the current active segment. A user can switch between segments only by pressing the previous or the next button. The concept of media segment is encrypted inside a class named CMediaSegment. A list of CMediaSegment and the index of current active segment are maintained in class CVideo in order to perform the switching between segments. This player requires Microsoft DirectShow to work. All formats supported by DirectShow are therefore supported by this player. The prototypes of class CMediaSegment and CVideo are

```
class CMediaSegment
{
public:
    CMediaSegment();
    CMediaSegment(REFTIME start, REFTIME stop);
    ~CMediaSegment();

public:
    REFTIME GetStartTime();
    REFTIME GetStopTime();
    void SetStartTime(REFTIME start);
    void SetStopTime(REFTIME stop);
};

typedef CArray<CMediaSegment, CMediaSegment&> MediaEntries;

class CVideo
{
public:
    CVideo(CString file, CWnd * pWnd, MediaEntries & entries, CRect rect =
        CRect(0,0,0,0));
    ~CVideo();

public://function
    LONG GetCurrentEntry();
    LONG GetPreviousEntry();
    LONG GetNextEntry();
```

```
void Previous();
void Next();
int Seek(LONGLONG i);
double GetAvgTimePerFrame(){return sPF;};
LONGLONG GetWidth(){return Width;};
LONGLONG GetHeight(){return Height;};
LONGLONG GetLengthInFrames ();
void Initial(CRect);
void ReleaseDirect();
HBITMAP GetCurrentBitmap(int);
char* GetImagePtr (long seek_to_frame);
void Play();
void Stop();
void ToEnd();
void ToStart();
void Pause();
void SpeedUp();
void SpeedDown();
void SetVolume(long);
long GetVolume();
void SetWindowPosition(CRect rect);
long GetDuration();
long GetPosition();
void GetWindowPosition(long* left,long*top,long*,long*);
long GetVideoStatus();
LONGLONG SetPosition(GUID,LONGLONG);
};
```

6.2.9 Customized Audio Library

The dictation button in the toolbar of the main window of the visual training tool enables real time dictation. This task requires passing audio data to the speech engine, getting the dictation result, and displaying the texts in the dictation window.

As introduced in Chapter 3, Section 3.3.4, the audio input source module encapsulates the methods used by the engine to retrieve the audio input stream. By default, the

engine retrieves its audio input from the standard microphone input device in the system. A developer can write a custom audio library so that the input of the engine would be a custom piece of hardware. The audio library functions must be packaged as a library, On Windows this is a Dynamically Linked Library (DLL), on Unix based systems and Macintosh it is a shared library. The implementations of the audio library functions are determined by the requirements of the application developer. The audio library will be loaded by the engine upon the first client connection. The engine will then resolve audio library exported function addresses. The audio library will be unloaded by the engine when the last client disconnects and a new client connects specifying a different audio source. The engine will also unload the audio library when the engine is terminating.

The acoustic processor takes raw audio data and converts it to the appropriate format for use. The acoustic processor consists of two components: the signal processor and the labeler.

The following 10 functions must be implemented in an audio library.

- AudioConnect Function
- AudioCreate Function
- AudioDestroy Function
- AudioDisconnect Function
- AudioGetPCM Function
- AudioPutPCM Function
- AudioStartPlayback Function
- AudioStartRecording Function
- AudioStopPlayback Function
- AudioStopRecording Function

In our case, audio sources are media files. Therefore, we need to a customized audio source library, which takes raw audio data from media files and converts these data into formats that will be accepted by the IBM ViaVoice engine.

Currently, The visual training tool uses an audio library named AudWav. This library is capable of retrieving audio data from wave files only. Thus we need to store every segment of a media file as a single wave file. These small wave files may not be

needed for other tasks. Another audio library, namely AudMedia, is under development. This library will support arbitrary media files, given that they are supported by DirectShow, and will be capable of retrieving audio data from media files directly.

Chapter 7 : String Alignment

Using our virtual training tools, we can get both the output text of ViaVoice SR engine and the typing from user. So the next task we need to do is comparing the two pieces of strings and find the matching. Once we get the characters that recognized as correct by ViaVoice SR engine, we can use the data to do the training.

We use string alignment algorithm to compare the two strings of text, it is a kind of dynamic programming. So we will give a brief introduction to the algorithm first, followed by an example.

7.1 String Alignment By Dynamic Programming

7.1.1 Longest Common Sequence (LCS)

Given two strings A1 and A2 over a finite alphabet, the problem is to find a longest sequence, which is a subsequence to each string.

7.1.2 Definition of Edit Distance

Let P be a pattern string and T a text string over the same alphabet. The edit distance between P and T is the smallest number of changes sufficient to transform a substring of T into P, where the changes may be:

Substitution -- two corresponding characters may differ: KAT -> CAT

Insertion -- add a character to T that is in P: CT -> CAT

Deletion -- delete from T a character that is not in P: CAAT -> CAT

7.1.3 Recurrence Relation

Let $D[i,j]$ be the minimum number of differences between $P_1, P_2, P_3, \dots, P_i$ and the segment of T ending at j. $D[i,j]$ is the minimum of three possible way to extend smaller strings:

If $(P_i = T_j)$, then $D[i-1, j-1]$, else $D[i-1, j-1] + 1$. We either match or substitute the i-th and j-th characters, depending upon whether they do or do not match.

$D[i-1,j] + 1$. There is an extra character in the pattern to account for, so we do not advance the text pointer and pay the cost of an insertion.

$D[i,j+1] + 1$. There is an extra character in the text to remove, so we do not advance the pattern pointer and pay the cost of a deletion.

The boundary condition is $D("", "") = 0$ and $D(0,I) = I$.

7.1.4 Pseudo code for the algorithm

The pseudo code of the algorithm is shown below. The matrix D is n by m where n = number of P and m = number of T .

```

editdistance(P,T)
  for i = 0 to n do D[i,0] = i
  for i = 0 to m do D[0,i] = i
  for i = 1 to n do
    for j = 1 to m do
      D[i,j] = min( D[i-1,j-1] + matchcost(p_i, t_j), D[i-1,j] + 1, D[i,j-1] + 1)

```

7.2 Examples

Here is an example using the dynamic programming for sequence alignment. The sample text is part of the speech of the video file “*tvbnews.mpg*”, the target file used through out this document. String 1 is the user input texts using our visual training tool, while string 2 is the output of IBM ViaVoice engine.

String 1 (User input):

三月份開始，長途電話公司出現減價戰，加上有新報紙發行，引發多份報紙減價促銷。

String 2 (Output of SR engine):

三月份開始，進德伊萬公司雖然減價戰，加上有新報紙發行，一百多份報章減價促銷。

So the longest common sequence (LCS) is:

三月份開始，公司減價戰，加上有新報紙發行，多份報減價促銷。

The edit distance between these two strings is 9.

Chapter 8 : Summary

8.1 Problems Facing

The main difficulty we are facing now comes with the hard nature of speech recognition. Although speech recognition has already been studied for decades, there are still lots of problems to overcome. The accuracy of state-of-the-art speech recognition systems is not very high. Although we have chosen IBM ViaVoice engine, which we consider as the best commercial speech engine that we could find, the outcome of this engine is still with considerable number of errors, as shown in Chapter 4, Section 4.3.

One of the goals of our project is to increase the accuracy of ViaVoice speech recognition system. This means that dealing with speech recognition becomes one of jobs naturally, although it is not an easy task.

First, the accuracy is influenced by many factors. Some of them are out of our control. For example, the quality of input voice signals may influence the outcome of the speech engine significantly. But the main source speeches that we are dealing with are media files, and there is no way for us to control the qualities of them.

Second, there is a distance between theory and practice in speech recognition. For example, although training should improve the outcome of speech recognition theoretically, we are still not able to tell how significantly training would improve recognition result or even whether there will be an improvement until we really finish the training process.

Third, in order to proceed in training of voice data, we need large pool of text data related to a set of media contents. These data need to be typed by human based on the video file and therefore requires a considerable amount of work. Fortunately, we were informed that the VIEW lab would hire some helpers to do the job.

Based on these difficulties, we make our future plans, which will be introduced in the next section.

8.2 Future Plans

8.2.1 Visual Training Tool Enhancement

Although the visual training tool is already ready for use, there could still be several enhancements of the visual training tool.

First, the AudMedia audio library will be finished on the Win32 platform as a Dynamic Linking Library. As discussed in Chapter 6, Section 6.2.9, this library will enable the tool to do real time dictation without saving separate media segments into separate wave files.

Second, the string alignment program will be integrated into the visual training tool. Currently, the string alignment program is a separate program. Thus, a user need to obtain the real texts and the recognition output from the training toll, and then pass them to the string program manually. After the string alignment program becomes part of the tool, this process will become as simple as click and wait.

8.2.2 Gender Classification

As mentioned before, the accuracy of speaker dependent recognition is much better than speaker independent one. Because the speeches we are dealing with are mainly come with media files, it is not possible to identify every speaker present. But we could still offer some limited information of speakers to the speech engine. This is done through gender classification.

We are planning to train separate language models, one for men and the other for women. And by some means, the program is capable of figuring out the gender of the current speaker and tells the speech engine to use the corresponding language model for recognition.

Once again, due to the distance between theory and practice, the effect of gender classification is still need to be proved in practice.

8.2.3 Noise Removal

As mentioned in Section 8.1, the quality of input signals influents the recognition

accuracy significantly. Therefore, we could expect a great improvement of the recognition accuracy if we could improve the quality of input signals successfully.

Noise is the part of input signals that is undesired by a speech recognition engine. Thus, it becomes our target to remove from input signals. But this is not an easy task. We are still investigating it. And the outcome still needs experiments to prove.

Acknowledgement

We would like to express our gratitude to Prof Michael R. Lyu, our project supervisor. He has given us many useful suggestions and directions throughout this project.

We would also like to express our thanks to Jackie Chun Keung CHAU, who has helped us to plan the tasks and give us useful suggestions, Edward Hon-hei YAU, who has offered us lot of helps on multimedia programming, and Min Cai, who offered many valuable suggestions for the implementation of this project.

We are fortunate to work in the VIEW lab directed by Professor Michael R. Lyu. Our team of colleagues also deserves our gratitude for helping us in numerous ways while we worked on this project.

Appendix A: References

A-I: Books

- *Fundamentals of Speech Recognition*; Lawrence Rabiner & Biing-Hwang Juang Englewood Cliffs NJ: PTR Prentice Hall (Signal Processing Series), c1993, ISBN 0-13-015157-2
- *Speech recognition by machine*; W.A. Ainsworth London: Peregrinus for the Institution of Electrical Engineers, c1988
- *Speech synthesis and recognition*; J.N. Holmes Wokingham: Van Nostrand Reinhold, c1988
- *Speech Communication: Human and Machine*, Douglas O'Shaughnessy; Addison Wesley series in Electrical Engineering: Digital Signal Processing, 1987.
- *Electronic speech recognition: techniques, technology and applications*, edited by Geoff Bristow, London: Collins, 1986
- *Readings in Speech Recognition*; edited by Alex Waibel & Kai-Fu Lee. San Mateo: Morgan Kaufmann, c1990
- *Hidden Markov models for speech recognition*; X.D. Huang, Y. Ariki, M.A. Jack. Edinburgh: Edinburgh University Press, c1990
- *Speech Recognition: The Complete Practical Reference Guide*; T. Schalk, P. J. Foster: Telecom Library Inc, New York; ISBN 0-9366648-39-2; 377 pages; paperback only. Covers speech recognition in a telephony environment and wish to use call processing hardware based in PCs. It is written using Dialogic hardware as the example for the hardware.
- *Automatic speech recognition: the development of the SPHINX system*; by Kai-Fu Lee; Boston; London: Kluwer Academic, c1989

- *An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition*, S. E. Levinson, L. R. Rabiner and M. M. Sondhi; in Bell Syst. Tech. Jnl. v62(4), pp1035--1074, April 1983
- *Automatic Speech and Speaker Recognition: Advanced Topics*, C.H. Lee, F.K. Soong and K.K. Paliwal (Eds.), Kluwer, Boston, 1996.
- *Review of Neural Networks for Speech Recognition*, R. P. Lippmann; in Neural Computation, v1(1), pp 1-38, 1989.

A-II: Papers

- *Audio Content Analysis for Online Audiovisual Data Segmentation and Classification*, by Tong Zhang Member IEEE, and C. C. Jay Kuo, Fellow, IEEE
- *A Robust and Fast Endpoint Detection Algorithm for Isolated Word Recognition*, Yiying Zhang, Xiaoyuan Zhu, Yu Hao State Key Laboratory of Intelligent Technology and Systems, Tsinghua University
- *The Carnegie Mellon University Distributed Speech Recognition System*. Speech Technology Adams, D. and Bisiani, R. 3(2), April, 1986.
- *BEAM: An Accelerator for Speech Recognition*. In International Conference on Acoustics, Speech and Signal Processing. Bisiani, R IEEE, May, 1989.
- *The use of recurrent neural networks in continuous speech recognition* Tony Robinson, Mike Hochberg and Steve Renals, Cambridge University Engineering Department, August, 1993
- *The application of dynamic programming to connected speech recognition* H.F. Silverman and D.P. Morgan, IEEE ASSP Magazine, July 1990
- *Connectionist probability estimators in HMM speech recognition* S.Renals, N.Morgan, IEEE Transactions on speech and audio processing, Jan, 1994

- *Automatic Speech Recognition: The Development of the SPHINX System.*
Boston: Kulwer Academic Publishers, 1989
- *Speaker-independent isolated word recognition using dynamic features of speech spectrum* Furui, IEEE Transactions on Acoustics, Speech, and Signal Processing, Feb, 1986

A-III: Useful URL

- CMU Sphinx <http://fife.speech.cs.cmu.edu/speech/>
- IBM ViaVoice (Chinese version)
<http://www-900.ibm.com/cn/ibm/crl/project/project1.html>
- IBM ViaVoice (Cantonese speech recognition)
<http://www-900.ibm.com/cn/ibm/crl/project/cvp.html>
- IBM ViaVoice (Putonghua speech recognition)
<http://www-900.ibm.com/cn/ibm/crl/project/vp.html>
- Speech Recognition Introduction <http://www.macalester.edu/~lkim/ai/sr.html>
- Online Bibliography of Phonetics and Speech Technology Publications
http://www.informatik.uni-frankfurt.de/~ifb/bib_engl.html
- MIT's Spoken Language Systems Homepage <http://www.sls.lcs.mit.edu/sls/>
- Speech Technology <http://www.speechtechnology.com>
- Speech Control (Speech Controlled Computer Systems: Microphones, headsets, and wireless products for ASR) <http://www.speechcontrol.com/>
- Microphone.com: Microphones and accessories for ASR
<http://www.microphones.com/>
- Say I Can.com: "The Speech Recognition Information Source."
<http://www.sayican.com/>

A-IV: Useful Newsgroups And Forums

Discussion forums dedicated to computer and speech

- US: <http://www.speech.cs.cmu.edu/comp.speech/>
- UK: <http://svr-www.eng.cam.ac.uk/comp.speech/>
- Aus: <http://www.speech.su.oz.au/comp.speech/>

Forums and newsgroups dedicated to users of speech software

- <http://www.speechtechnology.com/users/comp.speech.users.html>
- <news:comp.speech.research>

Newsgroup dedicated to speech software and hardware research

- <news:comp.dsp>

Newsgroup dedicated to digital signal processing

- <news:alt.sci.physics.acoustics>

Appendix B: Progress Report

Time	Task
June, 2001	Background study: CMU Sphinx III speech recognition engine
July, 2001	Background study: Microsoft SAPI
August, 2001	Visiting IBM Research Laboratory and Microsoft Research Institute, Beijing, China together with our supervisor Michael R. Lyu
Week1-3	Study background knowledge of Chinese Speech Recognition; Wait for IBM ViaVoice to come
Week4-5	Construct the homepage of FYP; Learn to use Microsoft DirectX(C++)
Week6-7	Study preprocessing of speech recognition; Extract wave file from media content
Week8	Get the IBM ViaVoice SR engine to work
Week9-10	Visual training tool pre-released: able to display video, output of ViaVoice engine, and user input panel to correct the errors AudMedia project launching
Week11-12	Speech segmentation finished; Enhance the visual training tool: process segmentation information of media content Develop AudMedia library