Department of Computer Science and Engineering

The Chinese University of Hong Kong

2003/2004 Final Year Project

Final Report

LYU0302

PVCAIS – Personal VideoConference Archives Indexing System

Supervisor

Professor Michael R. Lyu

Chan, Hiu Fai Philip
Ng, Chok Ming Lewis

15 April, 2004

Table of Content

Abstract

Videoconferencing becomes more and more popular in business activities, personal communications and educations, because of the rapid growth of Internet bandwidth and technical advances in multimedia coding. The participant of a videoconference usually wishes to keep the video archive for later reference. However, existing videoconferencing clients either do not record or only record the personal conference as ordinary audio and video files, in which the file name is the only place, but not the ideal place, to indicate the subject. Therefore, we can imagine that, in the near future, when a person accumulates many videoconference archives in his/her work, study and daily life, he/she may not be able to recall the details of a conference without watching it again. Since the visual and aural information is not directly searchable like text, it is also difficult to search out those archives with content of interest by normal searching methods. Therefore, the research of effective indexing personal videoconference archives is of timely importance. However, current efforts on multimedia indexing still focus on digital video libraries or distance learning. Reports on indexing videoconference archives have not been found so far as we investigated.

Our final year project will develop a well-designed Personal VideoConference Archives Indexing System - PVCAIS, which integrates many multimedia-indexing techniques to manage personal videoconference archives. Firstly, the contents of video, audio, text chat and whiteboard communications are received from the videoconferencing client and stored after removing the redundancies. Next, more information, e.g., participants, title, keywords, is extracted by face detection and recognition, speech recognition, Time-based Text Merging, etc. Afterwards, an index file containing the summary of the videoconference is generated. It stores the events of different channels and also the timestamps indicating when they happened. The whole indexing process should be automatic and transparent to users. Finally, a graphical user interface which allows the user to manage, search, browse the indexed videoconference archives conveniently will be developed. The browsing system should support synchronized playback in which all data channels are integrated with their timestamps.

This project applies many multimedia indexing functions to personal videoconference archives to generate a searchable content summary automatically. These techniques should be well designed, specific to the characteristics of videoconference, which facilities searching and browsing.

1. Introduction

1.1 Motivation

Due to the advantage of time and cost saving of using videoconference over traditional meeting, together with the rapid growth of Internet bandwidth and computing devices in the past few years, uptake of video conferencing to provide business conferences, remote seminars and courses, distance learning and personal informal communication is becoming more and more popular.

While the popularity of videoconferencing has grown exponentially, the ability to efficiently manage past videoconferencing archives has not. For most of existing videoconferencing systems, such as Microsoft NetMeeting and Microsoft LiveMeeting, past meeting record either cannot be kept or can be just kept as non-indexable file. So only the filename or meeting time can be used to search for the record. However, when we have more and more past meeting archives, it is difficult to retrieve those archives with content of interest in a reasonable time as we need to play back the whole content of each videoconferencing archive.

Therefore, the need to investigate on an effective videoconferencing archive indexing technique is of timely importance. Hence, we have designed this project which is to develop an experiment system PVCAIS – Personal VideoConference Archives Indexing System.

1.2 Project Objectives

Our project aims at illustrating how different media raw data(such as Video, Audio, Text Chat, Member Information, Whiteboard and File Transfer) can be obtained from a videoconferencing client, what multimedia-indexing algorithms(such as Key-frame Generation, Face Detection and Recognition, Speech Recognition) are suitable and practical to extract secondary information from the different raw data and how to integrate these algorithms

into the system to generate an index file for each videoconference. Consequently, we will design and implement a GUI of PVCAIS which allows users to search indexed videoconference archives and perform synchronized playback of a selected conference.

PVCAIS is designed to achieve the following targets:

- To provide the function of a basic videoconferencing client, such as audio, video, text, file and whiteboard communication
- To increase the reusability of the information extracted from the videoconference. For example, the video data extracted is used for Key-frame Generation, Face Recognition. By this way, more valuable information can be extracted from the same source of a videoconference
- To enable users to locate a certain videoconferencing archive effectively by using the searching system on the indexed file of each videoconference
- To reduce user work as much as possible by using automatic media data retrieval and secondary data generation done by the system. The user is basically transparent to these actions performed

2. Videoconference

2.1 Definition

Videoconference is a group or a person-to-person discussion, which employs real-time multimedia communication technology to enable participants at different geographical locations to see and hear each other as if they were together in one place.

In fact, videoconferences today may involve other media communication in addition to video or audio. They may include text chat, whiteboard, file transfer, and shared applications. Therefore, the term "Multimedia Conference" maybe more precious to describe it. However, by conversion, the term "VideoConference" is still used in this project.

A videoconference can be further classified as personal videoconference and group videoconference. A personal videoconference is a videoconference in which there is only one participant at each geographical location, where a group videoconference may consist of more than one participant at one geographical location. A personal videoconference is usually held among several participants and each participant sits in front of a personal computer with a view cam, a speaker and a microphone connected. A group videoconference is usually equipped with special hardware and communication lines and is held in a multimedia conference room, with a set of view cam and microphone installed for each participant in the conference room. And what our project focuses on is the personal videoconference only.

2.2 H.323 Standard

Most existing videoconferencing systems are based on the ITU-T H.323 standard. It includes videoconferencing techniques that can create incoming and outgoing channels for video and audio, share information via whiteboard, and exchange text messages in text chat.

H.323 specifies how terminals (such as personal computers), equipment, and services for multimedia communication over networks that do not provide a guaranteed quality of service (such as the Internet). H.323 terminals and equipment can carry real-time video, voice, and data, or any combination of these elements. Products that use H.323 for audio and video enable users to connect and communicate with each other over the Internet, just as people using different models of telephones can communicate using the same telephone standard. That means by both applying H.323 standard, videoconferencing programs developed for different OS or by different vendors can still communicate with each other.

H.323 is actually a series of standards that define:

- How calls are established.
- The framework for capability negotiation (the process of getting two machines to communicate with each other).
- How data is transmitted on the wire.
- Default audio and video compressor/decompressor (codec) components. Aaudio and video codecs encode and decode input/output from audio and video sources for communicating between nodes.

For low bandwidth use, the H.323 standard specifies standard codecs for audio (G.723) and for video (H.263) that enable H.323 products to send and receive voice and video images. H.323 also specifies use of the T.120 standard for data conferencing. More than 120 leading companies in the industry publicly announced their intent to support and implement H.323 in their products and services. This broad support establishes H.323 as the leading solution for audio and video conferencing over the Internet. Figure 2.1 shows the architecture of H.323 standard.

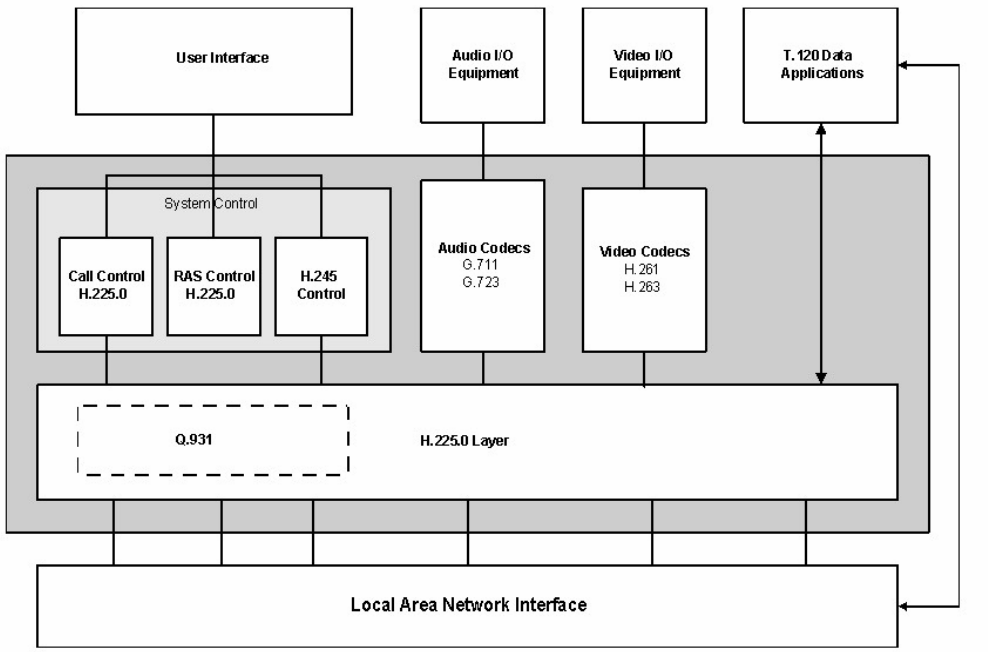

Figure 2.1 Architecture of H.323 standard

225.0	This standard defines a layer that formats the transmitted video, audio, data, and control streams for output to the network and retrieves the received video, audio, data, and control streams from the network. H.225.0 uses the packet format specified by IETF RTP and RTCP specifications for logical framing, sequence numbering, and error detection as part of audio and video transmissions. Support for RTP and RTCP enables the receiving node to synchronize the received packets in the proper order so the user hears or sees the information correctly.
H.245	The H.245 standard provides the call control mechanism that enables H.323-compatible terminals to connect to each other.
H.261	The H.261 standard specifies the format and algorithm for an alternative, higher bandwidth, video codec used to send or receive video images over network connections.
H.263	This standard specifies the format and algorithm for the default video codec used to send or receive video over low-bandwidth network connections.
G.711	The G.711 standard specifies the format and algorithm for an

	alternative, higher bandwidth, audio codec used to send or receive voice over network connections.
G.723	Audio codec, for 5.3 and 6.3 kilobits per second (Kbps) modes.
G.723.1	This standard specifies the format and algorithm for the default audio codec used to send or receive voice over low-bandwidth network connections.

Table 2.1 Components of H.323 standard

3. Design and Architecture of PVCAIS

3.1 Introduction

PVCAIS is a system that provides the convenient searching and browsing support for videoconferencing users on past videoconference archives. It also provides a videoconferencing client that provides real-time collaboration through standards-based data/audio/video Internet conferencing client support based on the standard of ITU-T H.323.

Data conferencing includes chat, whiteboard, and file transfer features. During each conference, the system first performs acquisition of media data from the videoconferencing client to generate raw videoconference archive files. Then it performs archive indexing by selecting a group of multimedia-indexing algorithms to generate indexed videoconference archives. By integrating these indexed videoconference archives, an index file for each videoconference is generated for later videoconference accessing, which includes searching and browsing on past archives. The idea of PVCAIS is illustrated in Figure 3.1.

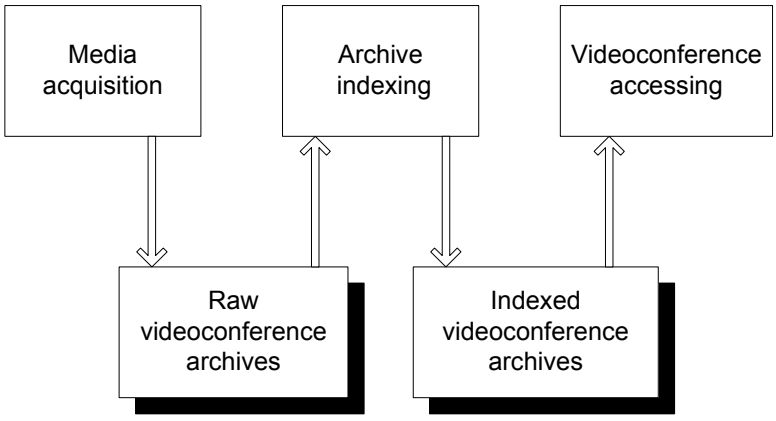

Figure 3.1 Top-level view of PVCAIS

3.2 PVCAIS Overview

PVCAIS can be divided into three main sections:

- Media Acquisition Module
- Archive Indexing Module
- Videoconferencing Accessing Module

The architecture of PVCAIS is shown in Figure 3.2, which consists of three separate processing modules with a linear processing order.

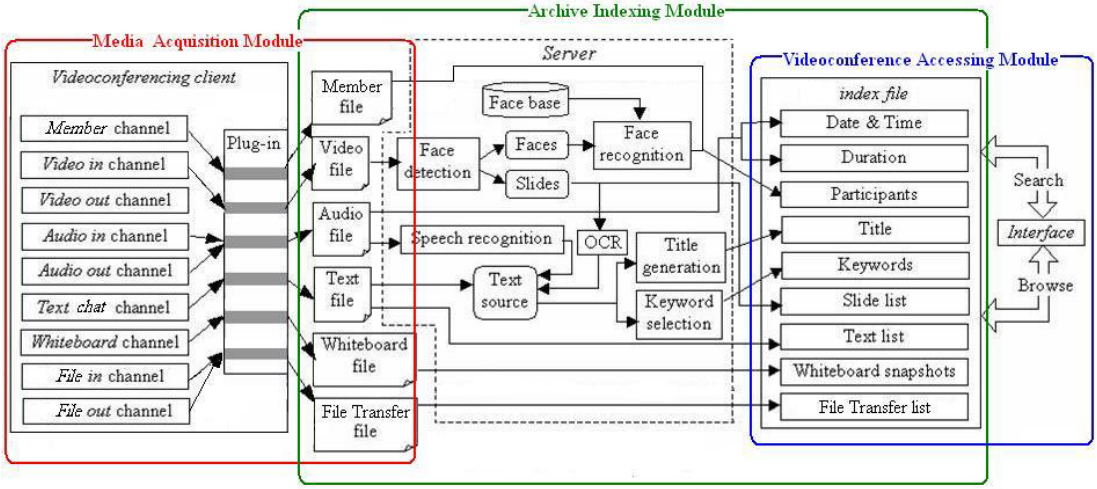

Figure 3.2 Overview of PVCAIS

The videoconferencing client is based on ITU-T H.323 standard which provides video, audio, text and whiteboard exchange between conference participants.

The Media Acquisition Module works together with the videoconferencing client to extract and save the raw content archives from all communication channels in real time. This module can be either embedded in the terminal as a Plug-in or separated from it.

After each videoconference finishes, the Archive Indexing Module performs a group of multimedia-indexing functions including Face Detection and Recognition, Speech Recognition, OCR, Title Generation and Keyword Selection to retrieve secondary information from the media files obtained by the Media Acquisition Module. The results are combined into an index file. Then all the related files of a videoconference are put into a single directory.

The Videoconference Accessing Module provides a user interface for users to search for past indexed videoconference archives. A list of searching result will be displayed. When a certain archive is selected, users can playback the content of that videoconference in a synchronized manner.

3.3 Media Acquisition Module

The Media Acquisition Module extracts and stores the raw content archives from videoconferencing client from all communication channels in real time. Afterwards, it will generate the media files.

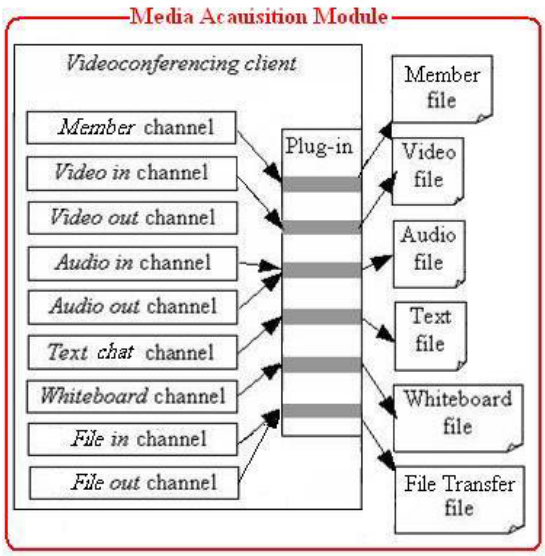

Figure 3.3 Structure of Media Acquisition Module

Before we can extract media information from videoconferencing client, we need to investigate into the media channels of the videoconferencing client.

There are 4 types of physical communication channels:

- audio channel : transmits incoming/outgoing video and audio information
- video channel : transmits incoming/outgoing video and audio information
- data channel : carries information streams for user applications, e.g. text chat, whiteboard and file transfer
- control channel : transmit system control information, e.g. H.2.45 control, call control and RAS(Registration, Admission and Status) control

In order to do indexing on the content of a videoconference, information in all of the above four channels should be extracted and stored in media files. In videoconference application, these physical channels can be divided into nine logical channels in the user's point of view. The nine logical channels are namely video-in, video-out, audio-in, audio-out, text-chat, whiteboard, file-in, file-out and control. Table 3.1 shows the nine logical channels, their corresponding physical channels and events that can be caught from each logical channel. These channels are synchronized, as shown in Figure 3.4.

Physical Channel	Logical Channel	Events
video	video-in	Scene change, face appeared, slide appeared
	video-out	Scene change, face appeared, slide appeared
audio	audio-in	Speech started, silence started, speaker changed
	audio-out	Speech started, silence started
data	text-chat	Text sent, text received, chat type changed
	whiteboard	Whiteboard updated
	file-in	File sent
	file-out	File received
control	control	Member joined, member left, channel created, channel closed

Table 3.1 Relation between physical channels and logical channels, and events of each logical channel

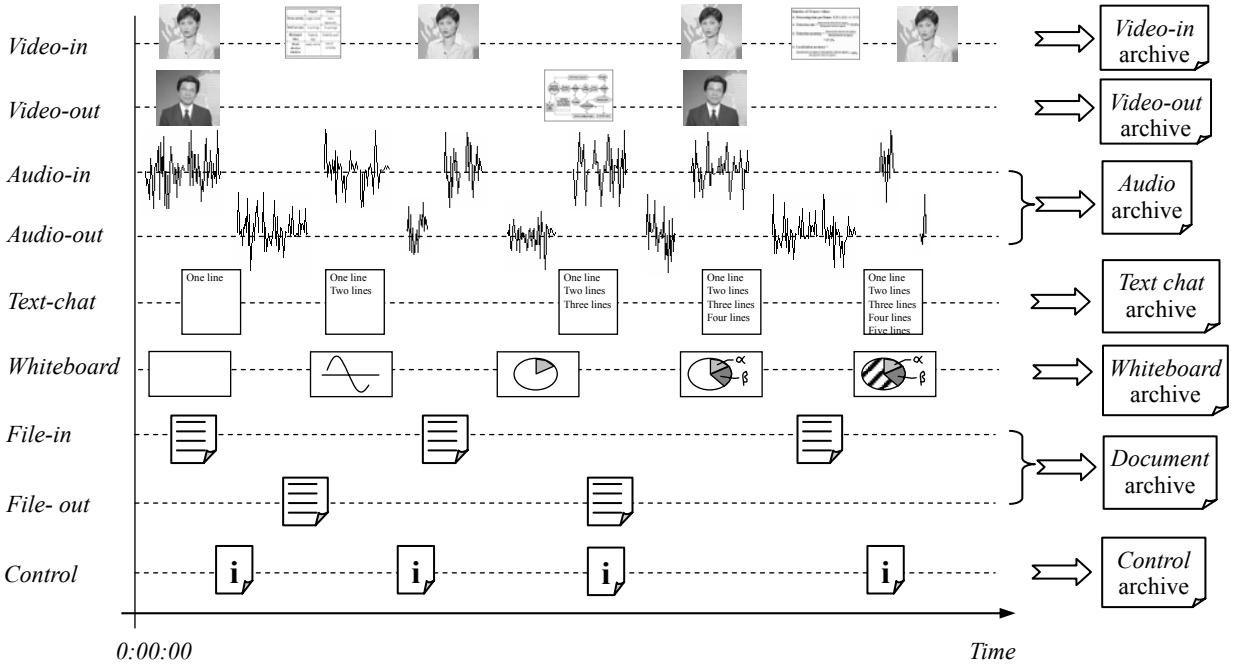

Figure 3.4 Illustration of the logical channels and media storage

When extracting the information from the channels, the system tries to reduce redundancy as much as possible in order to save storage space. Also, all events should be stored with a timestamp so that it can enable synchronized playback of the conference in the videoconference accessing module.

3.3.1 Video-in and Video-out Channels

The video-in channel receives the image shown to the user from the server, while the video-out channel sends out the local images.

Since video channel contains a lot of redundancies. The video channel shows either a human image or a slide with a relatively long duration. Thus, it is unnecessary to keep the entire video content in the media file. In fact, a set of key-frames with timestamp are enough to present the video content. The frame selection is based on the change of video content.

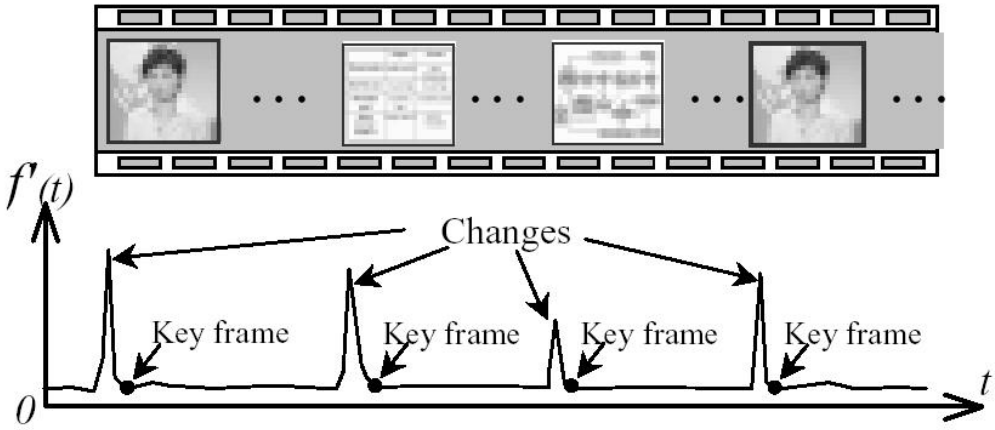

Figure 3.5 Key-frame selection algorithm on video channel

Let f(t) denotes the function of video content which varies with time t. After doing first derivate on the function, the changes of video content will be detected when f'(t) are peaks. The lowest point right after each peak is selected as key-frames. Note that f(t) should consider both the statistic and spatial distribution of colors in order to discriminate the changes between slides. After a key frame is marked, each key frame is stored as still image. The current timestamp and the image file name are written into the video media file.

3.3.2 Audio-in and Audio-out Channels

The audio-in channel and the audio-out channels contain incoming and outing voices, respectively.

The audio data in audio-in and audio-out channels will be mixed into one stream according to the timestamp right after the videoconference ends, as shown in Figure 3.6. The combined sound stream will be used as input for the speech recognition in the archive indexing module.

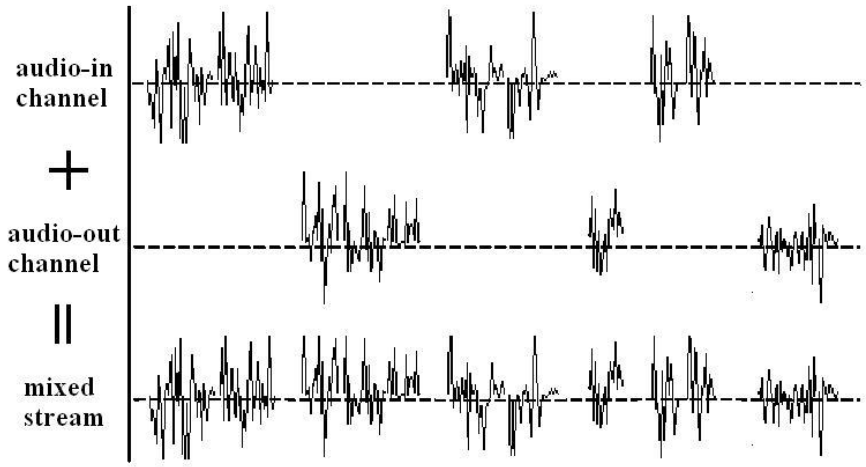

Figure 3.6 Mixing of audio-in and audio-out data

3.3.3 Text-chat Channel

The text-chat channel is shared by all participants for typed chat messages. The text-char archive is a file containing timestamp and corresponding information about a message sent or received. For example, Ann is doing videoconference with Bob and Cathy, for every Ann's outgoing message, the recipient's username and the message will be saved, while for every Ann's incoming message, the sender's username and the message will be saved. Ann can have an option to send message to a specific participant or "everyone" in which a multicast message will be sent to both Bob and Cathy in this sample.

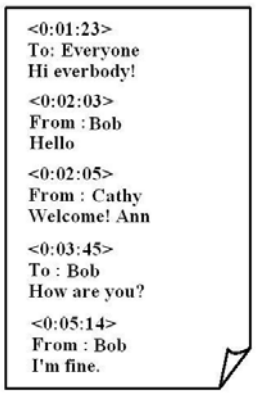

Figure 3.7 Sample text-chat archive file saved by Ann

3.3.4 Whiteboard Channel

The whiteboard channel is shared by all participants for handwritten contents. The whiteboard archive consists of an index file and a number of whiteboard snapshot image files. As update may take a period of time, the index file needs to store both the start-timestamp and end-timestamp for each whiteboard updated event and the path of the corresponding snapshot. The white board update events can be caught by monitoring the data transfer in the whiteboard channel. Another method is to compare the current sample with the last sample, as shown in Figure 3.8. By the function XOR(current sample, last sample), if it equals 0(they are different), then variable Update should be true, otherwise(they are the same), check whether current value of Update is True, if yes, set it into False as update ends. And that sample should be saved as a new whiteboard file.

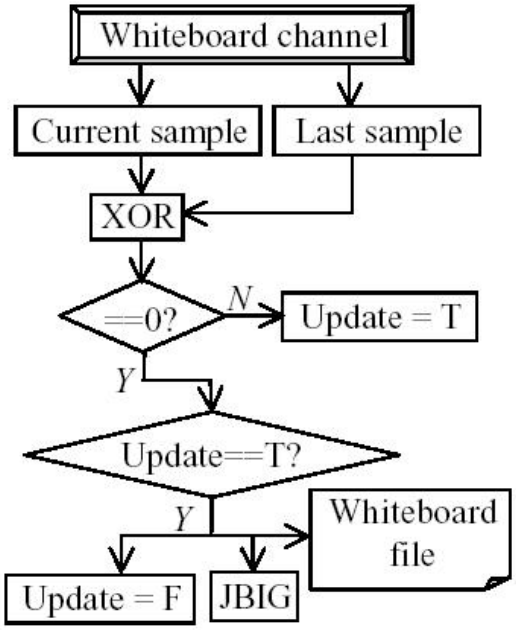

Figure 3.8 Whiteboard file generation algorithm by comparing samples

3.3.5 File Transfer Channel

The file-in and file-out channels are responsible for receiving and sending files, respectively. A copy of the file sent/received will be made into the archive directory. The document archive will store the timestamp and the corresponding file sending/receiving event. For file sending, the recipient's username and the path of the file in the archive directory will be stored; while for file receiving, the sender's username and the path of the file in the archive directory will be stored.

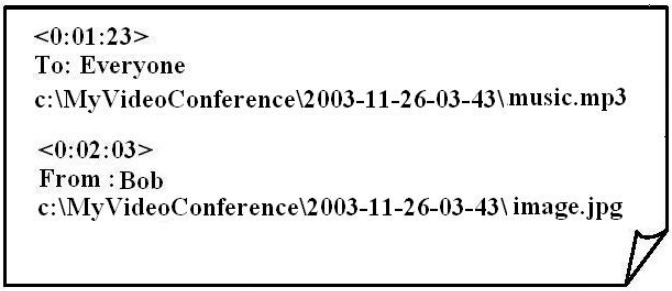

Figure 3.9 Sample document archive file

3.3.6 Control Channel

The control channel is used for the transfer of system control messages, such as member-joined-message, member-left-message, channel-created and channel-closed message. The Control archive is a file that stores the timestamp and the corresponding control message.

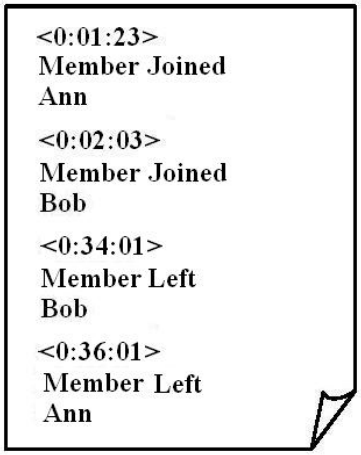

Figure 3.10 Sample control archive file

3.4 Archive Indexing Module

The Archive Indexing Module reads the seven raw content archive files generated in the Media Acquisition Module and automatically starts a series of multimedia-indexing functions to retrieve more information on the videoconference right after each videoconference ends. This module is not done in real-time as those indexing functions require a relative long running time and need quite a lot of system resources. This module should be transparent to users.

After all functions are done, all the related files for a videoconference are stored into a directory with meeting time as the directory name. The Archive Indexing Module will integrate all the archive files and generate a index file. The index file includes the date and time, the duration, the participant's names, the title, the keywords, the slide list, text list and whiteboard snapshots. Therefore, the index file is fully searchable.

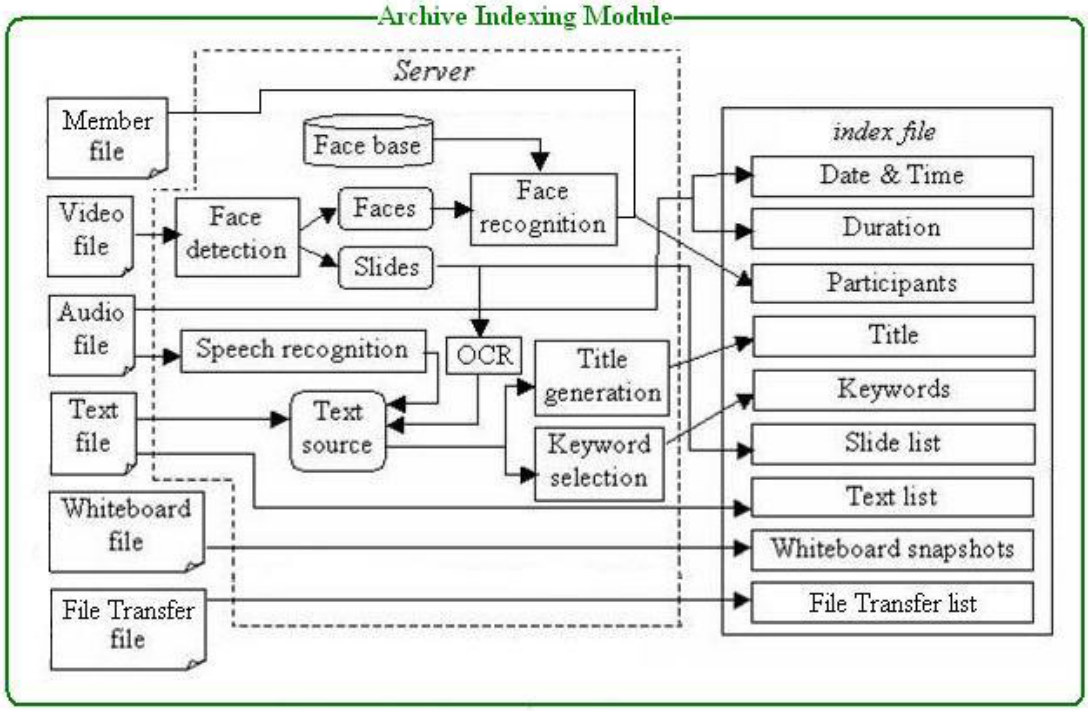

Figure 3.11 Structure of Archive Indexing Module

The multimedia-indexing functions include:

- Face Detection
- Face Recognition
- Speech Recognition
- OCR
- Time-based Text Merging
- Title Generation
- Keyword Selection

Multimedia-indexing function	**Input**	**Output**
Face Detection	video key-frame	A face region or a slide
Face Recognition	Face region	Username for the person recognized
Speech Recognition	Mixed sound stream of audio-in and audio-out channel	Speech script
OCR	Slide	Slide text
Time-based Text Merging	Speech script, chat script, whiteboard script and slide text	Text source
Title Generation	Text source	Title of the videoconference
Keyword Selection	Text source	Keywords of the videoconference

Table 3.2 Input and output of each multi-indexing function

3.4.1 Face Detection

Face detection takes key-frames generated in the Media Acquisition Module as input and distinguishes between face and slide. If a face is detected, the face region will be found out for face recognition. If a slide is found, the slide will be saved for OCR to recognize the slide text.

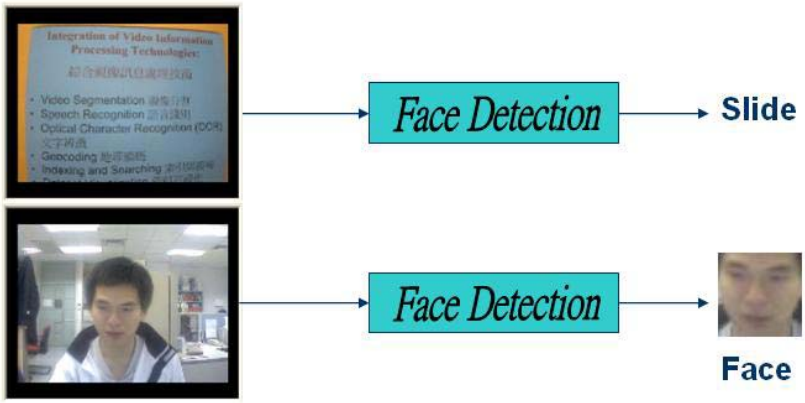

Figure 3.12 Face Detection

3.4.2 Face Recognition

Since videoconference video mainly involves human activities, face recognition plays an important role in multimedia indexing. Although the artificial intelligence technology to do face recognition still has difficulties when handling unlimited number of targets, it is suitable to be applied for videoconference indexing since the number of contacts of a user is limited.

The indexing system maintains a face base with usernames. If a face is recognized as an existing contact, contact's name will be added to the participant's list. If the face cannot be recognized as an existing contact, the system will interactively ask the users to input the identity for once and that face will be added into the face base. After that, the system will recognize him/her automatically.

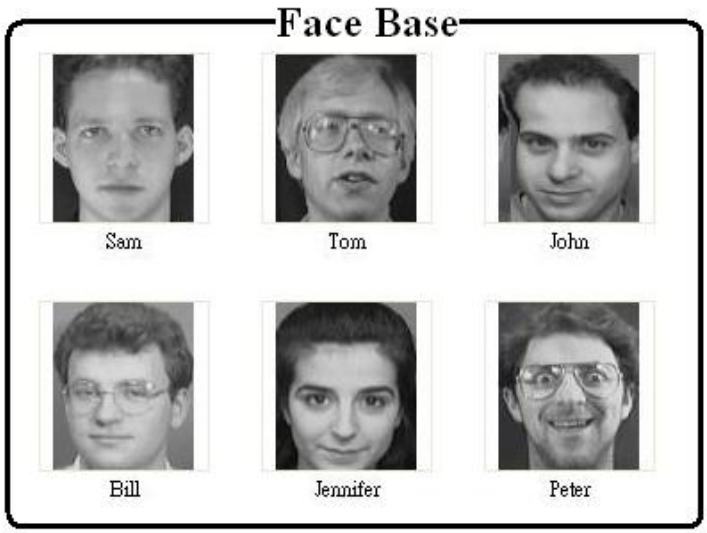

Figure 3.13 Sample face base for face recognition

3.4.3 Speech Recognition

The speech recognition process takes mixed stream of audio data as input and produces the speech script for the videoconference, which composes the majority of the text source. Accuracy of the speech recognition will greatly affect whether the script is useful or useless for later phase searching. Commercial products such as IBM ViaVoice and Microsoft SAPI can be used. However, the background noise and the varying microphone quality of participants make challenge to speech recognition.

3.4.4 OCR

The OCR processes the slide archives and recognizes text from them. The recognized text is stored as Slide text archive. The most difficult task is to identify text regions on the complex background in slide images. Also, the text in the slides may also be distorted.

3.4.5 Time-based Text Merging

After the speech script, chat script, whiteboard script and slide text are all extracted, the time-based text merging process will merge all these archives into the text source according the timestamp. The text source generated will be used for keyword selection and title generation.

3.4.6 Keyword Selection and Title Generation

The keyword selection takes the text source as input and produces keywords. The process can be done in two levels: local and global. Global keywords are selected from the whole videoconference text archives while local keywords can be generated by specifying a time period of interest, then only those texts within that period is used to generate keywords. Global keyword generation can enable quick response when searching while local keyword generation can be used to seek a point of interest.

The title generation also takes the text source as input and performs generation of the title for the whole videoconference by employing language processing technique.

3.5 Videoconference Accessing Module

The Videoconference Accessing Module serves as a user interface that provides users functions of managing indexed videoconferences, searching content of interest, playing back a synchronized version of a past videoconference. Figure 3.15 shows the structure of the Videoconference Accessing Module.

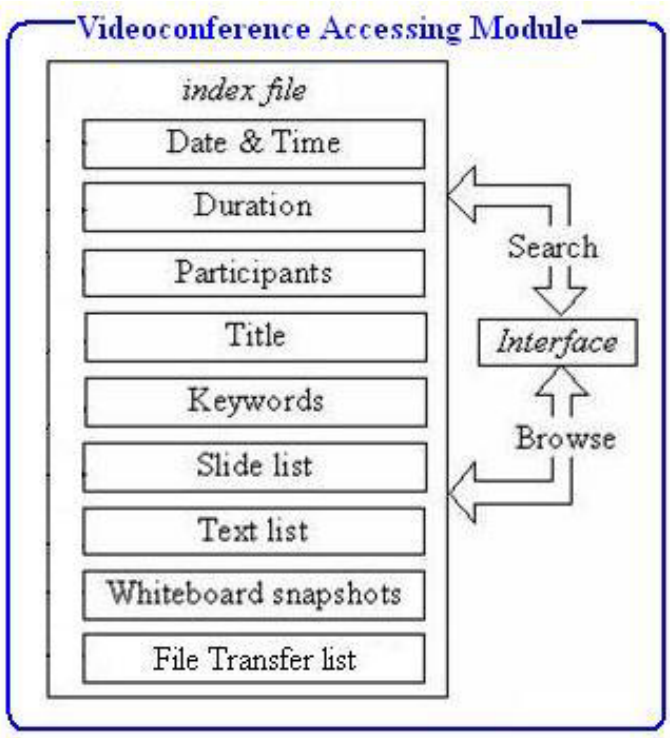

Figure 3.15 Structure of Videoconference Accessing Module

3.5.1 Implementation Schemes

The Videoconferencing Accessing Module can be implemented either as a stand-alone program or a web-based program. Stand-alone program allows users to manage those videoconference archives in the same computer of the module while a web-based program can enable users to manage their own videoconference archives through the Web browser with any computers which are connected to the Internet. Therefore, the web-based version is more advanced. However, the implementation is also more challenging as a web server is needed to be built to provide storage and all operations are called remotely. Also, security issues need to be handled to prevent malicious users.

3.5.2 Searching Interface

The search function allows users to search the videoconference of interest by inputting different searching criteria such as the name of participants,

period of the meeting time, title, keywords. Once the searching criteria is set, the Videoconference Accessing Module will access the index files of all the past videoconference archives. After searching, a list of videoconferences that match the criteria will be displayed. Users can then click on any one of the videoconference and the searching will open the playback interface to show the content of that conference in a synchronized manner.

3.5.3 Playback Interface

The playback interface provides a synchronized presentation of the conference. Users can play the videoconference and the Videoconference Accessing Module will access the index file and the related files in the disk. Different type of archives, such as local video key-frames, remote video key-frames, sound, whiteboard and text source will be shown according to their timestamp stored in the index file. By this way, users can view the whole past videoconference playback which is similar to the real-time videoconference.

4. Implementation

4.1 Overview

In this chapter, we will talk about our implementation work of PVCAIS based on the design in the last chapter.

Nowadays, there is a lot of videoconference client software in the market. However, as they are commercial products, the producers will not disclose the source codes of their products. In order to obtain the raw data of a conference, we need to build our own videoconference client first.

Most parts of our videoconference client are built on top of the NetMeeting 3 SDK. Before looking into the implementation, let's have a brief description of NetMeeting 3 and NetMeeting 3 SDK.

After talking on building our own videoconference client, we will talk about the implementation work done on the three modules of PVCAIS: Media Acquisition Module, Archive Indexing Module and Videoconference Accessing Module.

4.2 NetMeeting

NetMeeting is part of the Microsoft Internet Explorer family and is included in Windows 95 or above versions of Windows. NetMeeting is a software tool that enables user to meet and talk with others on a network. That network can be a closed corporate network or the public Internet. Beside video and audio, NetMeeting offers many other conferencing features that can make the virtual conference more interactive. These features include application sharing, chat, file transfer and whiteboard. NetMeeting even provides remote desktop sharing to allow user to remotely control a Windows PC. Another important feature of NetMeeting is its user interface. Microsoft has managed to integrate all of these communication methods into one interface that follows the Internet

Explorer style. This can provide great consistency. All these features make NetMeeting very popular and even become the standard of videoconference program. We will have a brief description for these six features.

4.2.1 Video

The most complex feature of NetMeeting is its video conferencing capability. Video conferencing is the capacity to exchange live video between two sites. In this case, user uses his or her own computer to capture, play back, send and receive live digital video. It is the most complex feature because of the volume of information involved with digital video. At the same time, it is one of the most funny and exciting components of the program. It enables user to see the person he or she is collaborating with.

4.2.2 Audio

The audio feature exchanges digital audio with the remote party. This feature can be called as "computer telephone", because it turns user's computer into a telephone. In many cases, the quality of a phone call on the Internet can be as good as an ordinary phone call, but there are many conditions that can interfere with it. Its greatest advantage is that a user is able to communicate with another user anywhere in the world using NetMeeting, by simply connecting to his or her local Internet service.

4.2.3 Text Chat

This feature is like an ordinary chat room on the Web, which enables user to send text to other members of a conference. Each line of message typed will be preceded by the user's name so that the text can be identified. This was the first type of interactive online communication and NetMeeting has integrated it to be one of the communication features.

4.2.4 Whiteboard

This feature is a digital representation of an ordinary whiteboard or dry-erase board used at office or home. User can draw with a pen, create lines and shapes, send images, enter text, and use many other capabilities that make it a good collaboration tool for a number of applications. For instances, engineers can collaborate on technical drawings, doctors can look at X rays, and anyone can simply have fun with it.

4.2.5 File Transfer

One of the simpler features of NetMeeting is the capability to send and receive files. With this feature, user can send any file that is on his or her local computer or network to the participants of the conference. User can also receive a file sent by a member of the conference and save it to his or her local disk or network.

4.2.6 Application Sharing

This feature enables user to select an application on his or her desktop to share with other people in the conference. This allows the participants of the conference to see and interact with the application. For example, user can share Microsoft Word and every participant will get a visual copy on their desktops. Beyond that, participants can also press the buttons and change the text. This is very helpful if members of the conference are writing a report together.

4.3 NetMeeting SDK

The NetMeeting 3.0 SDK is an extension of NetMeeting. It provides an interface for programmers and Web developers to integrate conferencing capabilities into their applications. The NetMeeting SDK provides many individual components that can perform basic communication functions, for example, file transfer and whiteboard.

Programmers can simply integrate these components into their applications to have these functions supported. The applications built by The NetMeeting SDK are NetMeeting clients, they can communicate with the original NetMeeting program.

4.3.1 NetMeeting Components

Following is the Windows NetMeeting diagram.

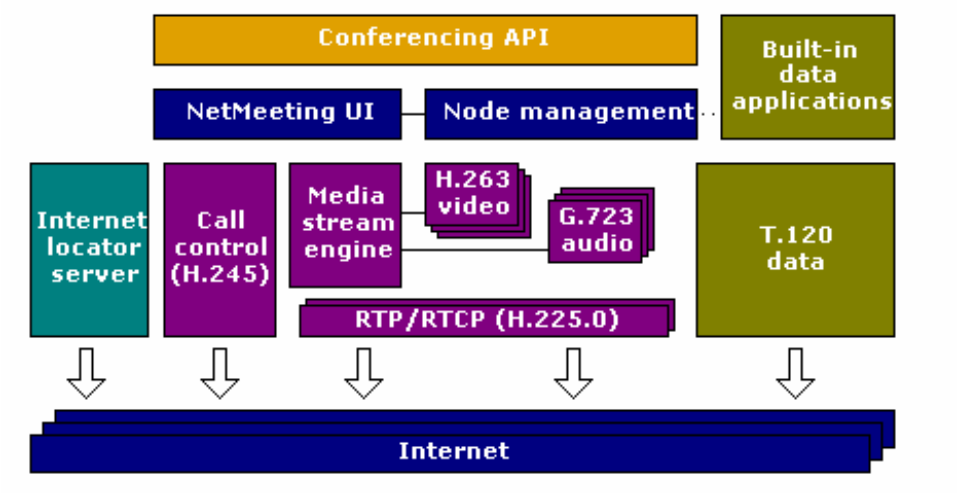

Figure 4.1 NetMeeting architecture

4.3.1.1 Conferencing API

The NetMeeting conferencing API, shown in orange in the diagram, takes advantage of all the functionality provided at lower layers including the NetMeeting user interface (UI).

4.3.1.2 Control Layer

The following components, shown in blue in the diagram, tie together the NetMeeting functionality provided by lower layers.

Node Management: Keeps track of conferences and conference participants.

NetMeeting UI: Displays NetMeeting functionality to users.

4.3.1.3 Audio/Video Components

The following components, shown in violet in the diagram above, manage the audio and video functionality in NetMeeting.

Call Control: Sets up the audio and video portion of the NetMeeting call using the H.245 standard for call control.

RTP/RTCP: Handles real-time streaming of audio and video over the Internet using the H.255.0 standard.

H.263 Video: Compresses and decompresses video data using a H.263-compliant video codec.

G.723 Audio: Compresses and decompresses audio data using a G.723-compliant audio codec.

Media Stream Engine: Coordinates capture, compression, and transmission of audio and video data. On the receiving side, this component receives, decompresses, and replays audio and video.

4.3.1.4 Data Conferencing Components

The following data conferencing components, shown in gold in the diagram above, handle transmission of data among different conference participants. These include file transfer, application sharing, whiteboard, and chat data, as well as applications written using the NetMeeting SDK's Data Channel.

T.120 data: Sends and receives data between different conference participants using the T.120 protocol for data

conferencing.

Built-in data applications: Use the T.120 data component to send and receive information. (These applications include chat, file transfer, and whiteboard.)

4.3.1.5 Internet Locator Service Component

The Internet Locator Service (ILS) server component, shown in the color teal, communicates with ILS on the network (LAN or Internet) to get directory listings and find users.

4.3.2 SDK Access to NetMeeting Components

The following diagram shows the NetMeeting components that are accessible through the various APIs supplied by the NetMeeting SDK. The COM Conference Interfaces are mainly used in our system.

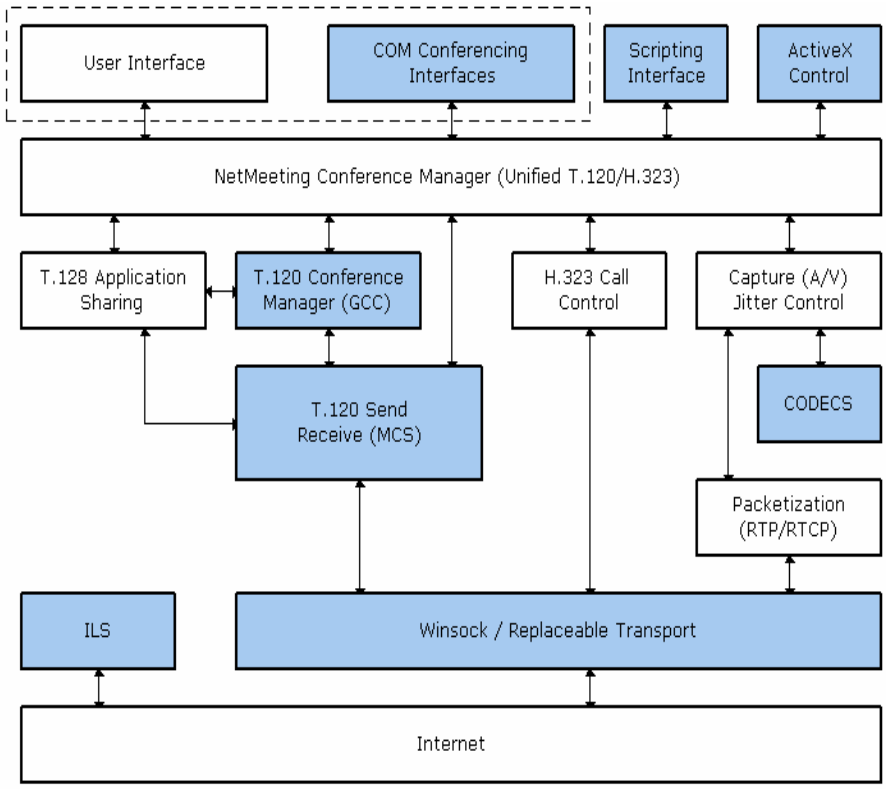

Figure 4.2 NetMeeting SDK Architecture

4.3.2.1 COM Conferencing Interfaces

The NetMeeting COM API provides many interfaces and functions to use with NetMeeting client applications developed in C/C++ or other programming languages that support COM. These interfaces and functions allow developer to use the components provided by the API easily to build their own video conference program. This can save a lot of time. We will now have a detailed description on the interfaces that we have used to build our system.

INmConference Interface

The INmConference interface represents a Windows NetMeeting 3 conference. Conferences can be hosted by the local computer or set up through a conference server. The way to set up a conference will affect the way in which data is distributed and how members can join and leave. A conference can be active or idle. Only one conference can be active at a time. An idle conference has no members and uses no channels, while an active conference always has one or more members.

INmConferenceNotify Interface

The INmCOnferenceNotify is the event sink for INmConference. When new member is joined, the MemberChanged function will be called.

INmCall Interface

The INmCall interface methods are used to manage and retrieve information about incoming and outgoing calls during NetMeeting conferences. These methods allow user to accept, reject, and cancel new calls.

INmCallNotify Interface

The INmCallNotify Interface provides the event notification sink methods to handle events from INmCall methods.

INmFt Interface

The INmFt interface is used to access information of a file being transferred, including its transfer state, size, and owner.

INmChannelFt Interface

The INmChannelFt interface methods are used to manage the file exchange between members of a conference.

INmChannelFtNotify Interface

The INmChannelFtNotify is the event sink for the INmChnnaleFt. When a file is being sent or received, the FtUpdate method will be called.

INmManager Interface

The INmManager interface represents the overall NetMeeting conference manager. The conference manager initializes the NetMeeting run time for its environment and controls how the underlying NetMeeting functionality interacts with NetMeeting client software.

INmManagerNotify Interface

The INmManagerNotify interface is the event sink for INmManager. The NetMeeting run time calls INmManagerNotify methods when certain events are triggered that specifically concern the management of the overall conference.

INmMember Interface

The INmMember interface is used to access information about a participant, usually on a remote system, in a conference. All types of communication channels also provide a way to retrieve the Member objects (instance of this interface) that are connected to the channels.

4.4 Overview of COM

Nearly the whole API of the NetMeeting 3 SDK is provided in terms of COM interfaces and objects, so let's have a look on what COM is.

COM stands for The Component Object Modal. It is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. COM is the foundation technology for Microsoft's OLE(compound documents), ActiveX(internet enabled components), etc..

Although COM is object-oriented, it is not a language, it is in fact a standard. It does not specify how an application should be structured. Language, structure, and implementation details are left to the application developers. COM specifies an object model and programming requirements that enable COM objects (also called COM components, or simply objects) to interact with each other. These objects can be within a single process, in other process, or even on remote machines. They can be written in different languages, and may be structurally different. That is why COM is referred to as a binary standard – it is a standard that applies after a program has been translated to binary machine code.

The only language requirement for COM is that code is generated in a language that can create structures of pointers and, either explicitly or implicitly, calls functions through pointers. Object-oriented languages such as C++ and SmallTalk provide programming mechanisms that simplify the implementation of COM objects, but languages such as C Pascal, Ada, Java, and even BASIC programming environments can create and use COM objects.

COM defines the essential nature of a COM object. In general, a software object is made up of a set of data and the functions that manipulate the data. A COM object is one in which access to an object's data is achieved exclusively through one or more sets of related functions. These function sets are called interfaces, and the functions of an interface are called methods. Further, COM requires that the only way to gain access to the methods of an interface is

through a pointer to the interface.

Besides specifying the basic binary object standard, COM defines certain basic interfaces that provide functions common to all COM-based technologies. It also provides a small number of API functions that all components require. COM has now expanded its scope to define how objects work together over a distributed environment, and added security features to ensure system and component integrity.

4.4.1 COM Objects and Interfaces

COM is a technology that allows objects to interact across process and machine boundaries as easily as objects within a single process interact. COM enables this by specifying that the only way to manipulate the data associated with an object is through what we called an interface on the object. When this term is used, it refers to an implementation in code of a COM binary-compliant interface that is associated with an object. An object that implements an interface means that the object uses code that implements each method of the interface and provides COM binary-compliant pointers to those functions to the COM library. COM then makes those functions available to any client who asks for a pointer to the interface, whether the client is inside or outside of the process that implements those functions.

4.4.2 Interfaces and Interface Implementations

COM makes a fundamental distinction between interface definitions and their implementations. An interface is actually a contract that consists of a group of related function prototypes whose usage is defined but whose implementation is not. These function prototypes are equivalent to pure virtual base classes in C++ programming. An interface definition specifies the interface's member functions, called methods, their return types, the number and types of their parameters, and what they must do. There is no implementation associated with an interface.

An interface implementation is the code a programmer supplies to carry out the actions specified in an interface definition. Implementations of many of the interfaces a programmer could use in an object-based application are included in the COM libraries. Programmers are, however, free to ignore these implementations and write their own. An interface implementation is to be associated with an object when an instance of that object is created, and provides the services that the object offers.

Interface, in fact, define a contract between an object and its client. The contract specifies the methods that must be associated with each interface, and what the behavior of each of the methods must be in terms of input and output. The contract generally does not define how to implement the methods in an interface. Another important aspect of the contract is that if an object supports an interface, it must support all of its methods in some way. Not all the methods in an implementation need to do something – if an object does not support the function implied by a method, its implementation may be a simple return, or perhaps the return of a meaningful error message – but the methods must exist.

COM uses the word "interface" in a sense different from that typically used in C++ programming. A C++ interface refers to all of the functions that a class supports and that clients of an object can call to interact with it. A COM interface refers to a predefined group of related functions that a COM class implements, but does not necessarily represent all the functions that the class supports.

4.5 SMIL

4.5.1 Introduction of SMIL

In the Videoconference Accessing Module, we will provide an interface for users to search for a conference by different criteria, such as the text in text chatting, the file transfer record or even the drawings in the whiteboard. We will provide a list of conferences to users that match the

searching criteria. Users can then choose any of the conferences in the list to review. However in a videoconference, beside the video and audio channel, there are other channels that contain important content of the videoconference, these include whiteboard drawings, file transfer record and text chatting content. All these channels may be opened at the same time during the videoconference, for example, when the users of the videoconference are discussing on the design of a product, they may at the same time talk and draw on the whiteboard. If we just let the users to review the video and audio of the videoconference, important information may be omitted. However, how can we review all this separate channels at the same time and in a synchronized way? The answer is to use SMIL.

SMIL stands for Synchronized Multimedia Integration Language. The W3C is the chief sponsor for the development of SMIL. SMIL enables simple authoring of interactive audiovisual presentations. Its basic idea is to name media components for text, images, audio and video with URLs and to schedule their presentation either in parallel or in sequence. A typical SMIL presentation has the following characteristics:

- The presentation is composed from several components that are accessible via URLs, for example, files stored on a Web server.

- The components of the presentation file can be many different types, including audio, video, image or text. The start and finish times of the components can be organized in a parallel way of relative to other components in the presentation. For example, two video can be shown on the same time or one after another in the presentation screen.

- There can be control buttons to control the flow of the presentation, for example, by pressing the “Play” button, the presentation starts, by pressing the “Pause”, the presentation can be paused. Users can also start at any point of the presentations. Playing SMIL presentation is just like playing an ordinary video or audio file.

- There can be hyperlinks embedded in the presentation that users can follow them.

SMIL is an easy-to-learn HTML-like language, and so many SMIL presentations are written just using a simple text-editor. SMIL allows user to put text, still images, video, audio and animation into the presentation. This can make the presentation much more interactive. Before the development of SMIL, it is very difficult to combine and use different media to present an idea at the same time. Now with SMIL, presentation can be much more interesting and powerful.

Figure 4.3 A SMIL document playing in Real Player

4.5.2 Advantages of Using SMIL

There are many advantages to use SMIL as a presentation authoring tools, these include:

1. SMIL is text-based

   Since SMIL is text-based, you do not need to invest any new software in order to produce SMIL presentation. You can write a SMIL presentation with any text editor.

2. Put together customized presentations

   Since a SMIL file is a simple text file, you can write a program to generate the SMIL file automatically for different visitor. Therefore, you can create different presentation parts, providing a customized SMIL file for different visitors based on their preferences.

3. SMIL is led by the W3C

   The W3C is an organization focused on defining standards that the general public can follow. Therefore W3C tries to design a specification that is beneficial to everyone and is easy to use.

4. Use clips from different locations

   Since a SMIL file lists a separate URL for each clip, the media clips of the presentation can be stored on different servers.

5. Layout the presentation

   When there are different types of media in the presentation, SMIL provides methods for you to organize the layout of your presentation. For example, you can display text under a video clip, the text can be a translation to the language of the video clip.

6. Time Control

   SMIL provides methods for you to control the timeline of your presentation. For example, you can play a video 3s after another audio

has started.

7. Create interactive multimedia presentation
   By using SMIL, you can embed many different types of multimedia format into a single presentation. For example, your presentation can be divided into different regions, you can display a video on one of the region, and display an animation on another regions.

4.5.3 Syntax of SMIL

4.5.3.1 Header and Layout

The syntax of a SMIL document is very similar to that of a HTML document. SMIL has three main elements. Namely <smil>, <head>, and <body>. The <smil> element is the outer container of the whole SMIL document. Within <smil> element, it is mainly divided into two sections, the head and the body. The head section is under the <head> tag and the body section is under the <body> tag. Although SMIL is very similar to HTML, there are two main differences between them, they are:

1. Unlike HTML, SMIL is case-sensitive, therefore all the tags should be written in lower case.

2. SMIL is XML-based, therefore all the tags should be ended properly.

In the header of the SMIL document, normally there are three types of elements:

1. The <meta> element
   This element is used to state the properties of the SMIL document and set the values of these properties. These properties include author of the document, key words, expiration date, etc.

2. The <switch> element

One of the greatest ideas proposed with SMIL is to adapt your presentation to the end-user system capabilities. For example, if the user's modem has a 14000 bps modem, it would be better to make it download light pictures. You could also want to translate parts of the presentation depending on the user language. The <switch> element makes it possible. This element allows the author of the document to specify a set of alternative elements from which only one acceptable element should be chosen.

3. The <layout> element

In the <layout> section, the author can determine the positions of the media in the presentation. Under this section, there are many tags to let you control the layout of your presentation.

3.1 The <root-layout> tag

This tag let you set the width and height for the window in which your presentation will be rendered. The following statements will create a window with a 300x200 pixels dimension and a white color background.

```
<smil>
 <head>
  <layout>
   <root-layout width="300" height="200"
    background-color="white" />
  <layout>
 </head>
 <body>
 </body>
</smil>
```

3.2 The <region> tag

The presentation window rendered by the previous tag can be further divided into several different regions, each region can be used to display different media. The <region> tag is used to create the regions. The following statement will create a region namely "region 1" with relative position (125,125) from the left top corner of the presentation window and size equal 64x64 pixels dimensions.

<region id="region 1" left="125" top="125" width="64" height="64" />

4.5.3.2 The Body, Timing and Synchronization

In the body section, author can design what and when to be displayed in the regions defined in the header of the SMIL document. There are two main types of media object. Media like video and audio are called "continuous media", while media like text and image are called "discrete media". The following tags are used to describe the media to be displayed in the presentation.

1. The <img> tag

This tag is used to display a still image in a region of the presentation window. The following statement will display am image in the region with ID "region 1".

**

The src="file" attribute describes the location of the source image.

2. The <text> tag

This tag is used to display plain text onto a region of the presentation window. The following statement will display the text from the text file called "text1.txt" into a region called "region 1".

<text src="text1.txt" region="region 1" />

3. The <textstream> tag

   This tag is used to display a stream of text for example, RealText, on to the presentation window. The following statement will display a RealText file onto "region 1" region of the window. The behavior of the text stream is then determined by the RealText file.

   <textstream src="test.rt" region="region 1"/>

4. The <audio> tag

   This tag is used to play an audio file in the presentation. The following statement will play an audio file called "test.wav".

   <audio src="test.wav" />

5. The <video> tag

   This tag is used to play a video file onto a region of the presentation window. The following statement will display a video onto "region 1" of the presentation.

   <video src="test.mpg" region="region 1"

Besides displaying the media, SMIL provide timing controls to let the author design when and how the media files are displayed.

1. begin="s" and end="s" attributes

   **

   In the above statement, the begin="6s" attribute describes when the image starts to display. The end="8s" attribute describes when the image finishing displaying. Besides specifying the time to start or

end, the media can be set to display after certain events have occurred or finished. For example:

```
<body>
 <img src="test1.jpg" region="region 1" begin="2s" end="6s"
  id="image1" />
 <img src="test2.jpg" region="region 2" id="image 2"
  begin="id(image 1)(3s)" />
 <img src="test3.jpg" region="region 1" id="image 3"
  begin="id(image 1)(end)" />
</body>
```

In the above source, the image, "test2.jpg", will starts to display 3 seconds after image "test1.jpg" finishes displaying. Image "test3.jpg" will start to display right after "test1.jpg" finishes displaying and will be displayed on the same region as "test1.jpg". The "begin" and "end" can also be used with other tags that are used to display media.

2. The dur="s" attribute

   Besides specifying when to end the display, author can specify the duration of the media by setting the dur="s" attribute. In the following statement, the image will begin to display when the presentation starts and will be displayed continuously for 6 seconds.

```
<img src="test1.jpg" region="region 1" dur="6s" id="image1"
 />
```

3. The <seq> tag

   Another way to display a group of media in a sequence is by using the <seq> tag. The media objects grouped by a pair of <seq> and </seq> tag will be displayed one after another. For example:

```
<body>
```

```
<seq>
  <img src="test1.jpg" region="region 1" dur="2s" />
  <img src="test2.jpg" region="region 1" dur="2s" />
  <img src="test3.jpg" region="region 1" dur="2s" />
  <img src="test4.jpg" region="region 1" dur="2s" />
 </seq>
</body>
```

In the above source, the four images will be displayed in the same region one after one, with duration of 2 seconds.

4. The <par> tag

   Sometimes author may want to display two media objects at the same time. Then the <par> tag can be used. By grouping media objects by a pair of <par> and </par>, all these media objects can be displayed at the same time.

   For example:

```
<body>
  <par>
    <video src="test1.mpg" region="region 1" />
    <video src="test2.mpg" region="region 2" />
    <audio src="test3.wav" />
  </par>
</body>
```

In the above source, the two video files and the audio file can be played at the same time in different regions of the presentation window.

4.5.3.3 Media Support

Although SMIL specification can support nearly all the multimedia formats currently used, the actual performance depends on the players

or browsers that are used to play the SMIL document. For example, some players can support the mp3 format audio file while some can only support the Microsoft wav format audio file. The following is a table of multimedia formats that are supported by SMIL, the corresponding tag used by the format and what can be seen on different players.

Media	Tag	RealPlayer	GRiNS	Soja
GIF	img	OK	OK	OK
JPEG	img	OK	OK	OK
Microsoft Wav	audio	OK	OK	-
Sun Audio	audio	OK	OK	OK
Sun Audio Zipped	audio	-	-	OK
MP3	audio	OK	-	-
Plain text	text	OK	OK	OK
Real text	textstream	OK	-	-
Real movie	video	OK	-	-
AVI	video	OK	OK	-
MPEG	video	OK	OK	-
MOV	video	OK	-	-

Table 4.1 Media formats supported by SMIL

4.5.4 Players and Browsers of SMIL

SMIL has become a very popular standard. Many famous players have already supported it. The following is a list of players that can support SMIL.

1. RealPlayer
   http://www.real.com

2. QuickTime

http://www.apple.com/quicktime/

3. Internet Explorer
   http://www.microsoft.com/window/ie/default.asp

4. SOJA (Helio)
   http://www.helio.org

5. GRINS (CWI)
   http://www.cwi.nl/SMIL/

4.6 RealText

4.6.1 Introduction to RealText

Although SMIL is very powerful and can support many different types of multimedia format, its support on text is very simple. Therefore, some vendors try to produce more powerful text format that can be embedded into SMIL. One of them is RealText.

RealText is a RealSystem product for streaming text from files of live sources. With RealText, you can create presentations consisting of text alone, or combine text with other media to create, for example, closed-captioned video. RealText is a mark-up language similar to HTML to describe how and when the text displays. RealPlayer can be used to play the RealText file. Since RealText file is basically a text file, therefore like SMIL, any simple Text Editor can be used to create a RealText file. Below is a RealText example:

```
<window duration="30" bgcolor="yellow">
Mary had a little lamb,
<br/><time begin="3"/>little lamb,
<br/><time begin="6"/>little lamb.
<br/><time begin="9"/>Mary had a little lamb,
```

```
<br/><time begin="12"/>whose fleece was white as snow.
<br/><time begin="15"/><clear/>Everywhere that Mary went,
<br/><time begin="18"/>Mary went,
<br/><time begin="21"/>Mary went,
<br/><time begin="24"/>Everywhere that Mary went,
<br/><time begin="27"/>That lamb was sure to go.
</window>
```

4.6.2 Basic Syntax of RealText

4.6.2.1 Window Tag Attributes

RealText provides a number of window styles that you can choose depending on how you want to display text. These include:

1. Generic

   A generic window has no preset parameters. You can use it to create any RealText display allowed by the RealText mark-up. You can display and erase lines of text, scroll text through the window, or have text crawl from side to side, for example.

2. ScrollingNews

   A ScrollingNews window is preset to have text scroll from the bottom of the window to top at a set rate for the entire presentation. The text does not crawl from side to side, though.

3. TickerTape

   Text in a TickerTape window crawls from the right side of the window to the left. It can also loop back around to the right. It does not scroll up or down, however. Text displays next to the window's top or bottom edge.

4. Marquee

   The Marquee window is like the TickerTape in that text crawls from

right to left and can loop. It is different in that text is centered vertically within the window.

5. TelePrompter

   A TelePrompter window behaves like a generic window except that text arriving at the bottom edge of the window causes the text above it to move up just enough to display the new line.

The above window types can set by using the following statement:

<window type="tickertape" duration="2:05:00.0">

Beside the window types, other properties can also be set within the <window> tag. These include:

1. duration=”dd:hh:mm:ss:xyz”

   The duration attribute specifies the time, relative to the start of the presentation, that this RealText stream stops playing. The default is 60 seconds. The RealPlayer timing slider is keyed to this value, which is in 24-hour format, where `dd` is days, `hh` is hours, `mm` is minutes, `ss` is seconds, `x` is tenths of seconds, `y` is hundredths of seconds, and `z` is milliseconds.

2. width=”pixels”

   The `width` attribute determines the window width in pixels. The default is 500 for TickerTape and Marquee windows, 320 for other window types. SMIL layout tags can specify a playback region width that overrides the width set here.

3. height=”pixels”

   The `height` attribute sets the window height in pixels. The default is 30 for TickerTape and Marquee windows, 180 for other window types. SMIL layout tags can specify a window height that overrides the height set here.

4. bgcolor=”color”

   The bgcolor="*color*" attribute determines the window's background color. The default is black for TickerTape windows and white for all other window types.

5. scrollrate=”pixels per second”

   The scrollrate attribute sets the number of pixels per second that the text moves vertically. The default is 10 for ScrollingNews windows and 0 for all other window types.

6. crawlrate=”pixels per second”

   The crawlrate attribute specifies the number of pixels per second that the text moves horizontally. The default is 20 for TickerTape and Marquee windows, 0 for other window types.

7. loop=”true|false”

   The loop="true|false" attribute is available only in TickerTape and Marquee windows, where it defaults to true. When set to true, this attribute tells RealPlayer to buffer all text and redisplay ("loop") it if and when the stream runs dry, which occurs when the text has moved out of the window and no new text has arrived.

4.6.2.2 Text Tags

RealText provides many mark-up tags that define how the streaming text looks and operates. A tag's default value applies if you do not specify a tag value. You can place mark-up tags anywhere on a line.

1. <time begin/> and <time end/> Tags

   The <time/> tags control the RealText presentation timeline by determining when a text component appears and disappears, respectively, relative to the start of the presentation:

<time begin="dd:hh:mm:ss.xyz"/>
<time end="dd:hh:mm:ss.xyz"/>

2. <clear/> Tag

   This tag clears the existing text buffers to remove all text from the window:

   <clear/>

3. <pos x/> and <pos y/> Tags

   These tags position the text horizontally and vertically, respectively:

   <pos x="pixels"/>
   <pos y="pixels"/>

4. …

   The … tags add space between text. In TickerTape and Marquee windows, they move the "cursor" to the right edge of the window. In all other window types, the and each cause the next text to display two lines down.

5.

   The
 tag adds space between text. In TickerTape and Marquee windows, it moves the "cursor" to the right edge of the window. In all other window types, this tag causes the text that follows to display on the next line.

6. …

   The … tags display the enclosed text bloded.

7. …

   The … tags display the enclosed text italicized.

8. <u>…</u>

The <u>…</u> tags display the enclosed text underlined.

9. Tag

   The tag lets you specify text characteristics:

   …text…
   bgcolor=”color”

   This <font> tag attribute set the text background color.

10. Color=”color”

    This <font>tag attribute lets you control the font color. It supports all color values available in HTML.

11. face=”font name”

    This <font> tag attribute controls the text font.

    Size=”n” This <font> tag attribute lets you control the font size. You can use relative sizes or absolute sizes.

4.7 Videoconference Client

We have developed our own videoconference client which supports all the basic functions that a videoconference should have, including Video and Audio streaming, Text Chat, File Transfer and Whiteboard. All parts except Whiteboard are done by NetMeeting SDK. Also, we have developed a Face Verification Login feature for our videoconference client which will verify the face of a user against his/her login ID before he/she can use the videoconference client.

4.7.1 Face Verification Login

The Face Verification Login feature is designed to make our system more secure. Each user of our system has to register a unique user ID to our system before their first login. Also, they need to train their frontal faces

by the Face Training Interface in order to for our system to support the Face Verification Login feature.

4.7.1.1 Face Recognition Algorithm

We have chosen to implement Eigenface algorithm because it is the most popular one due to its effectiveness. Eigenface was first proposed by Sirovich and Kirby in 1987.

Eigenface's main idea is to get the features in mathematical sense instead of physical face feature by using mathematical transformation. By this way, the representation of image can be much more compact and efficient for comparison. The method it uses is Principal Component Analysis (PCA).

Pentland and Turk of MIT refined Eigenface by adding preprocessing and procedures of face detection in 1991. In their approach, faces are represented by a small number of eigenvectors which contain the major variations in the faces. They created approximate reconstruction of face images with a weighted sum of the eigenvectors obtained from PCA.

At elementary level, face image is expressed as a one dimensional column vector of concatenated rows of pixels:

$X = [x_1, x_2, ..., x_n]^T$, where n is the total number of pixels in the image

To compare two images, the simplest method is to compare the images pixel by pixel. However, n is usually large (even for a 100 × 100 pixels image, n would be 10,000). The comparison is therefore time-consuming. Moreover, not all pixel values are important for comparison as only a few of them represent the characteristic of a face. Therefore, PCA is used to keep the representation of image compact and efficient for comparison.

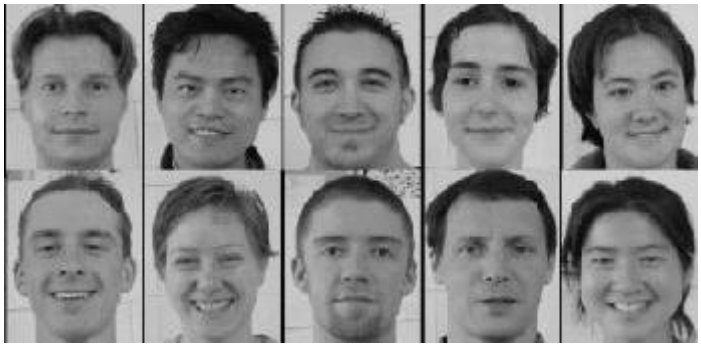

Figure 4.4 Sample training set

Face recognition using Eigenface can be divided into 2 phases. The first phase is the training phase. In this phase, a group of individual face images is used as the training set. Figure 4.4 shows a sample training set of 8 people. The size, orientation and light intensity of each image should be standardized. Each of the images in the training set is represented by a vector of size N by N, where N is the size of the image. With the training images, a set of M eigenvectors is found by using PCA.

PCA is done by first finding ψ by averaging the training set $\{T_1, T_2, \ldots\ldots T_M\}$ with T_i representing each of the vector in the set. Then a matrix $A = \{\varphi_1, \varphi_2, \ldots\ldots \varphi_M\}$ with column vector $\varphi_i = T_i - \psi$ can be formed, which represents the difference vector of the train images and the average face ψ. Then the covariance matrix $C = AA^T$ and the eigenvector and the associated eigenvalues of C can also be found.

After the eigenvectors have been calculated, eigenvalues of each eigenvector are sorted. They are called eigenfaces. The eigenfaces with the largest number of eigenvalues are chosen. These M' eigenfaces are considered the best eigenvector to represent a face. The span of the M' eigenfaces is called face space. Figure 4.5 shows a sample face space.

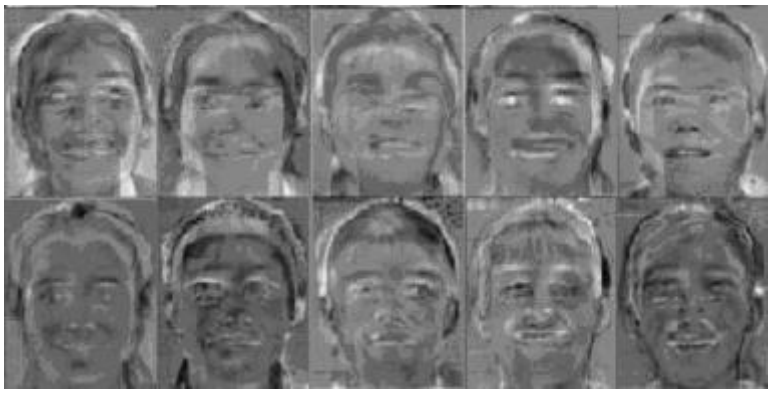

Figure 4.5 Sample face space

The second phase is the recognition phase. In this phase, a new face image for recognition is obtained. To recognize this image, we first subtracted the image by the average face ψ. Then we calculate the dot product of the input vectors with the eigenfaces, which makes a projection of the input image onto the face space. Similarly, we can make projections of the training image onto the face space. The Euclidean distances between point of the input image and points of training set are then computed. The training set image with the minimum distance from the input image should be the best match.

Experiment result shows that eigenface approach reach 96% correct classification averaged over lighting variation, 85% correct averaged over orientation variation and 64% correct averaged over size variation.

4.7.1.2 Face Training Interface

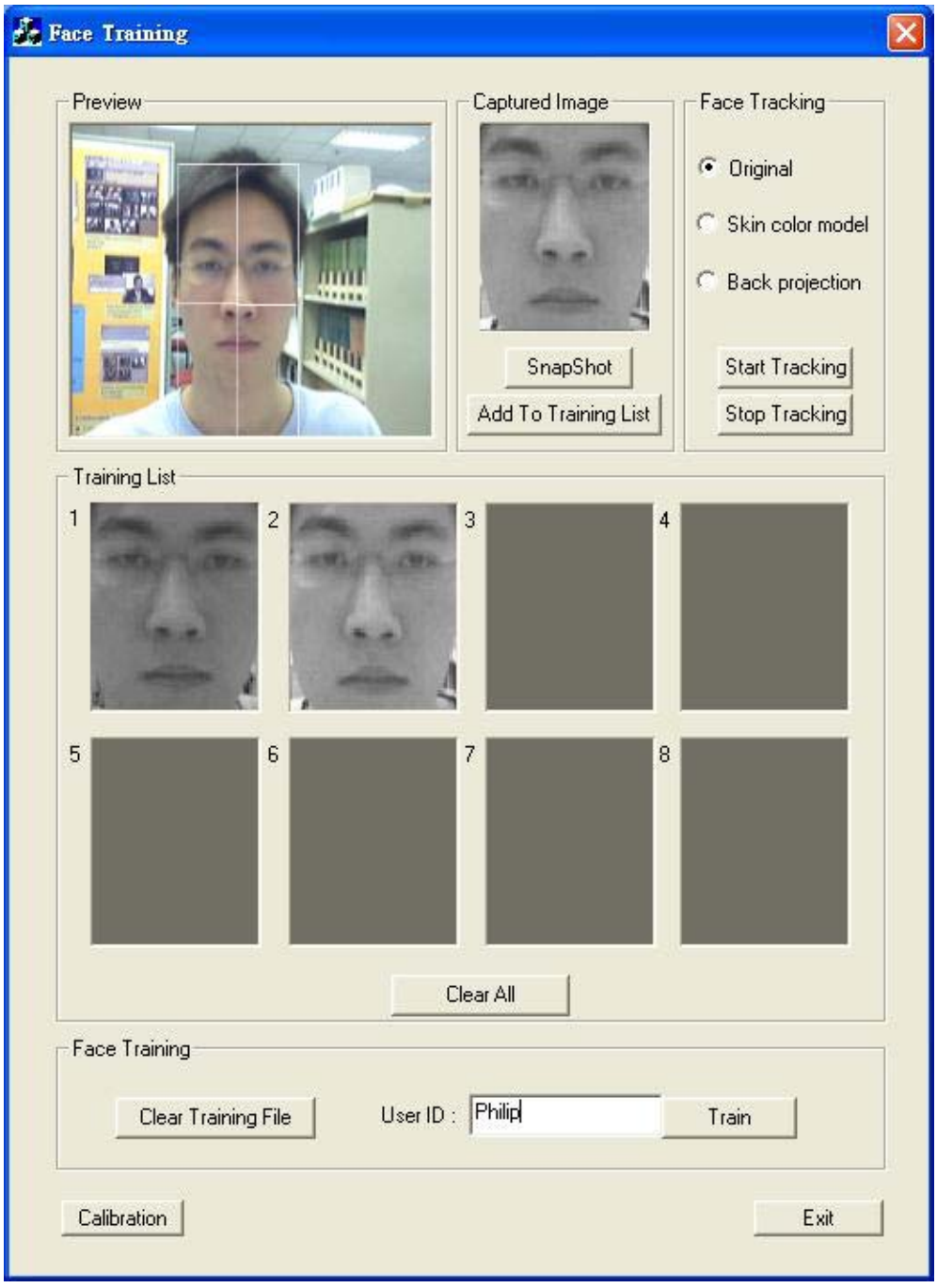

Figure 4.6 Face Training user interface

In the Face Training User Interface, users can preview the live video capture of the web cam in the Preview Window. The "calibration" button is used to fine tune the setting of the web cam. When users click

start tracking, the system will try to find if human face exists in the search space. The system will locate the possible area of face by a white square and users can press the button "Snapshot" to capture that image. The captured image will be shown in the Captured Image Window. If user confirms that the Face Detection is done properly, he/she can press "Add to Training List" to put the image into the Training List. The system requires users to put at least eight images into the Training List. After eight images are collected, users should input their User ID and press "Train" to start the training process.

4.7.1.3 Face Verification Interface

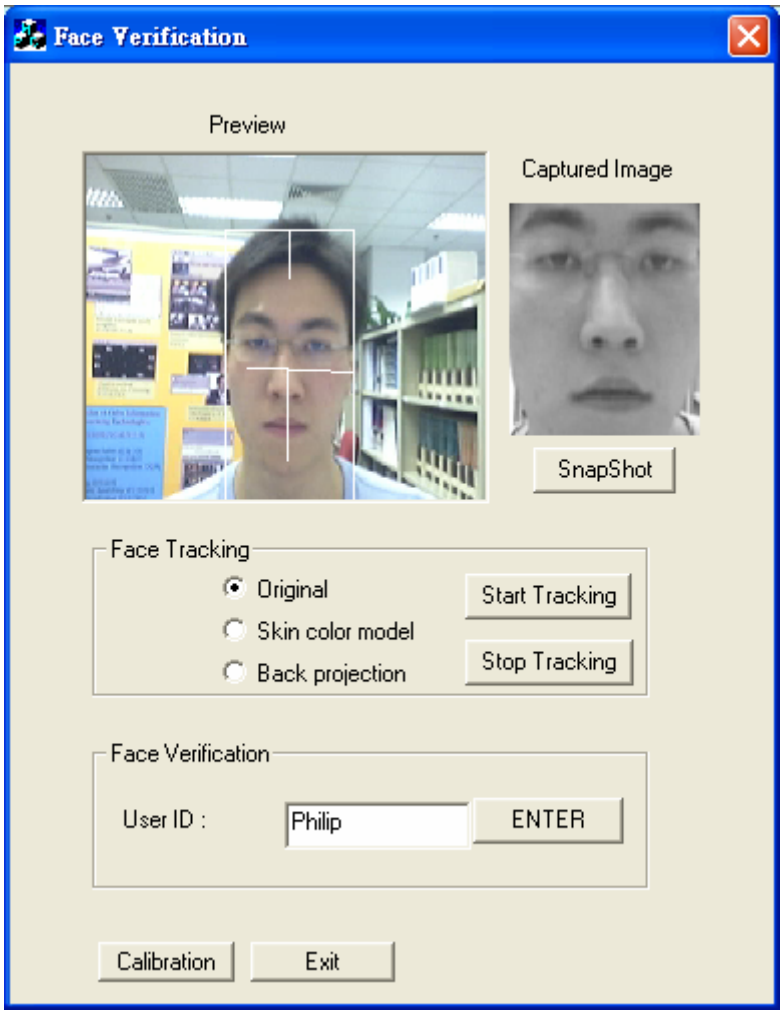

Figure 4.7 Face Verification user interface

Before using the videoconference client, users are required to do the Face Verification login, similar to Face Training, users can preview the live video captured by the web cam in the Preview Window and start Face Tracking by pressing the "Start Tracking" button. After user's face is located correctly, user can press "snapshot" button and see the captured image in the Captured Image Window. Then user can enter their user ID and click "login" to start the Face Verification. The system will then verify the input image against their identity. If the verification result is "true" with a high certainty, the user can login successfully and the system will then launch the videoconference client. If the verification result is "false", the system will show "Face Not Match". There is also a case that the system cannot tell whether the result is "true" or "false" due to the input image quality, then the system will show "Please Try Again" and users should try to do the image capturing again to obtain another face image for verification.

4.7.2 Main Interface

In the Main Interface of our videoconference client, users can use the "Connect" button to start a conference. Users need to enter the IP address of the computer that they want to connect into the box on the top left corner. After the conference is created, the button will be changed into "Disconnect" and it is used to end the conference. The "Initialize" button is used to initialize the data collection functions of the system after the conference starts. Users can choose to play or stop viewing of the local video and remote video by pressing the button below the video windows. And the audio streaming is automatically started after the conference begins. Users can also use the buttons "Text Chat", "File Transfer" and "Whiteboard" to launch these additional videoconference functions.

The "Start Audio Capture" button is used to record both the local and remote audio data for later reference. Similarly, when the "Start Capturing Video" is pressed, the key-frames of both local and remote video will be collected. The "Stop Capturing Video" button is used to stop the

key-frames capturing function. After the conference is finished, user can press the "Voice Recognition" button to perform voice recognition on the recorded audio data of the conference. The result will be stored in a text file. There is also a check box on the lower left corner. When this box is checked, the system will perform face detection and recognition on the key-frames recorded from the remote video right after the conference ends. When the user disconnects, the conference ends and the system will ask him/her to input a title for that conference for later searching purpose. The User Interface of our videoconference client is shown in Figure 4.8.

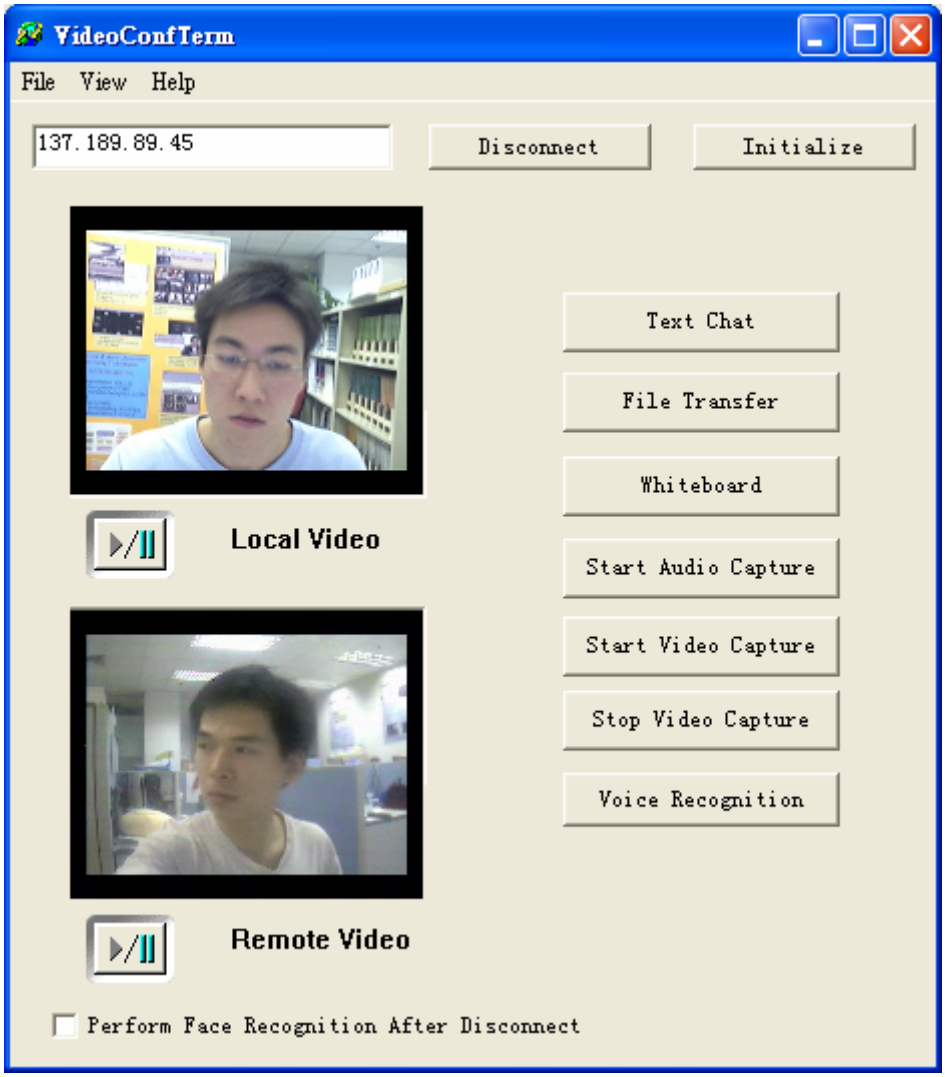

Figure 4.8 Main window of the videoconference client

4.7.3 Text Chat

The Text Chat function is built using NetMeeting SDK. It is used to send/receive text messages to/from other participants of the conference. For sending message, users can choose to send to a specific participant or select "everyone" to send a multicast message to everyone in that videoconference. Users can enter the text to be sent in the box at the bottom and press "enter" to send the message. For each sent/received message, time, sender/receiver name and message content will be shown in the user interface. Figure 4.9 shows the user interface of the chatting tool.

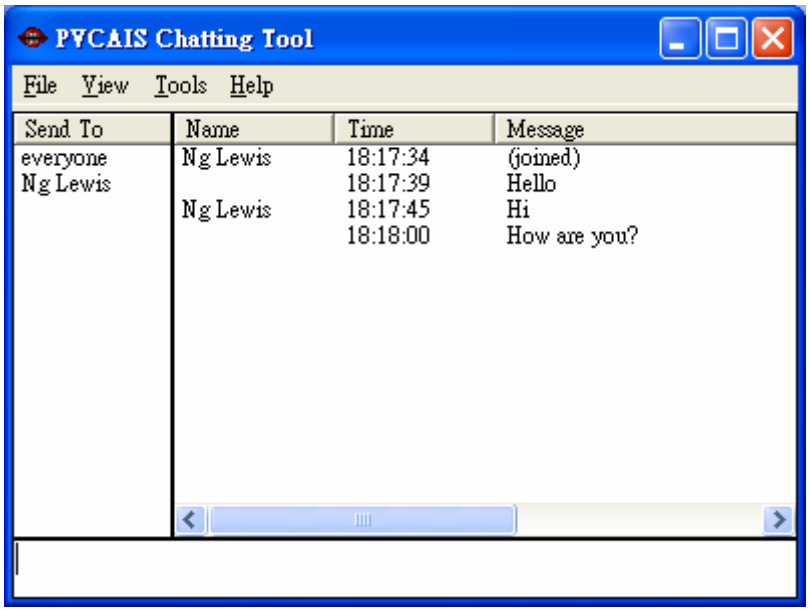

Figure 4.9 Chatting window

4.7.4 File Transfer

The File Transfer function is built using NetMeeting SDK. It is used to send/receive files to/from other participants of the conference. For sending files, users can choose to send to a specific participant by typing the username of the receiver. Also, users can type "everyone" to send file to everyone in that videoconference. Users can select a file to be sent in the file selection window. For the receiver, our system will automatically receive the incoming file. Then it will ask the user if he or she wants to store the file or not.

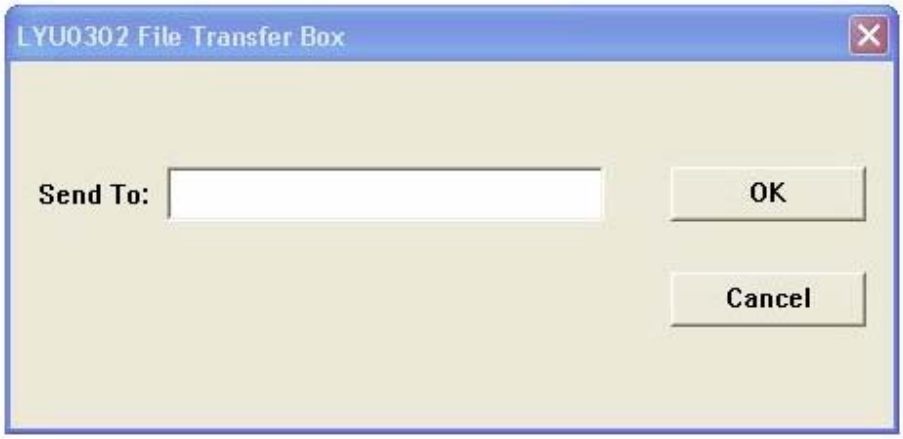

Figure 4.10 File transfer window

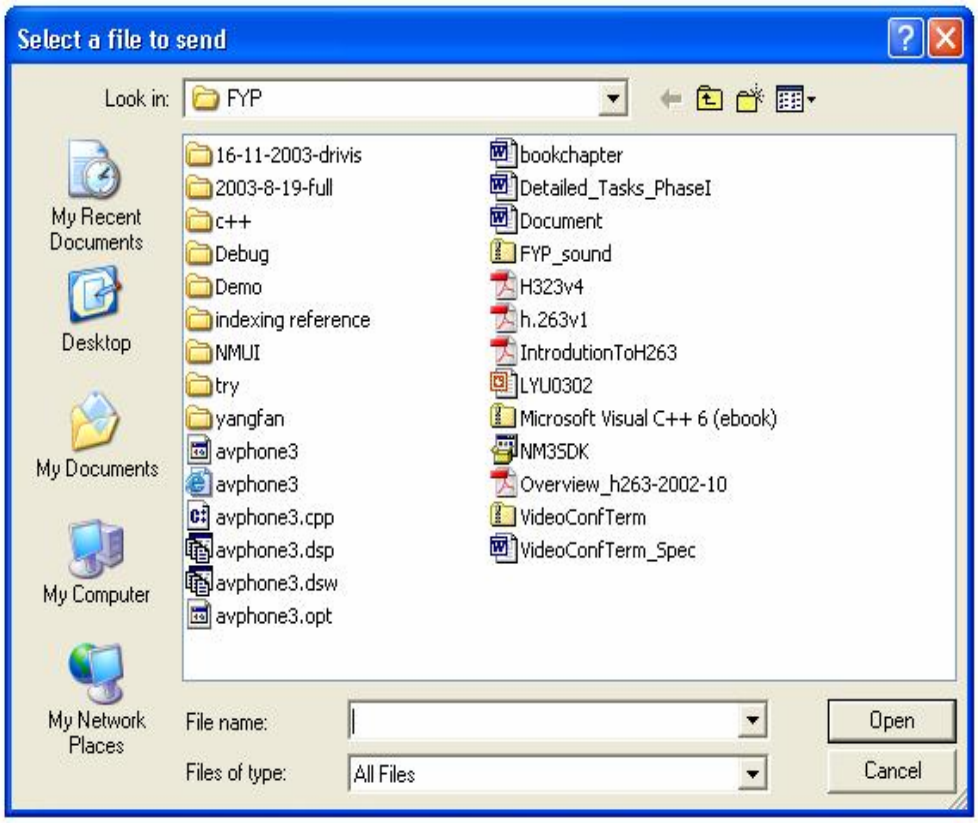

Figure 4.11 File selection window

Figure 4.12 File receiving window

4.7.5 Whiteboard

Since NetMeeting API does not provide any functions and reference on its whiteboard, therefore we have no way to collect the data from NetMeeting's whiteboard. However, whiteboard is a very important feature of videoconference software. Ideas that cannot be expressed verbally can be drawn on the whiteboard. So, in order to provide this feature in our system, we have designed and built our own whiteboard.

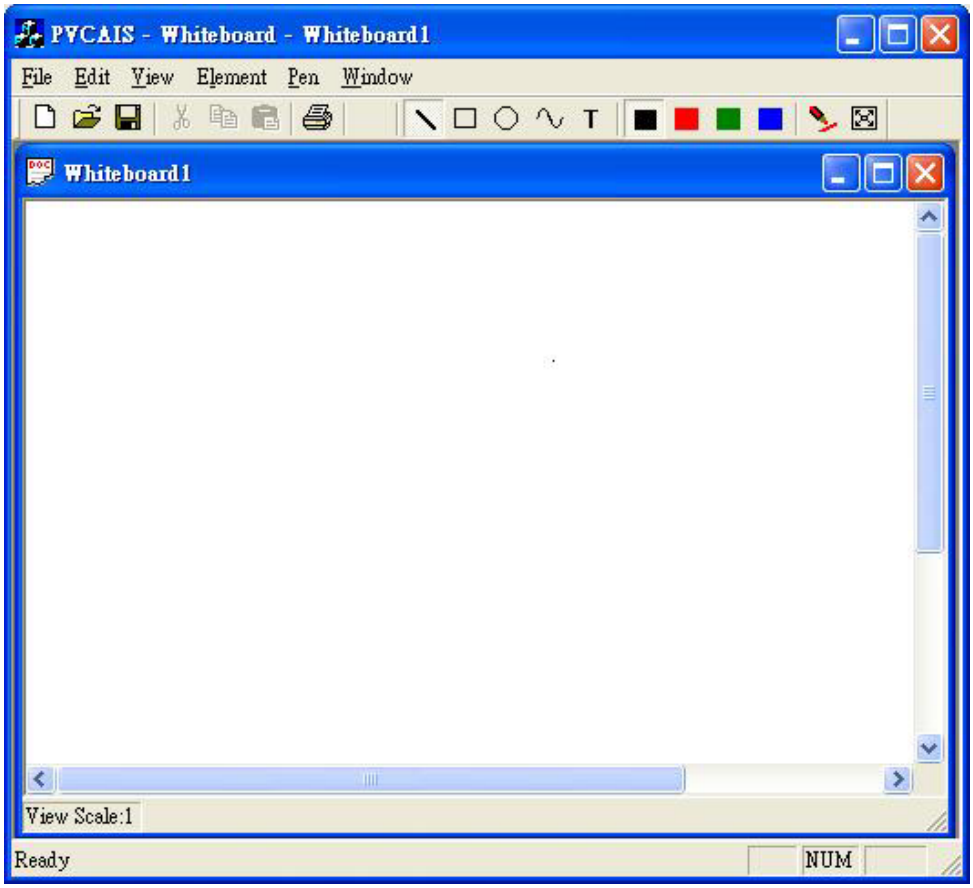

Figure 4.13 PVCAIS - whiteboard

Our whiteboard supports the all basic functions as a normal whiteboard in the market. Users can draw straight lines, circles, rectangles and curves in the whiteboard. Users can also change the pen size and color, currently available color are red, black, green and blue.

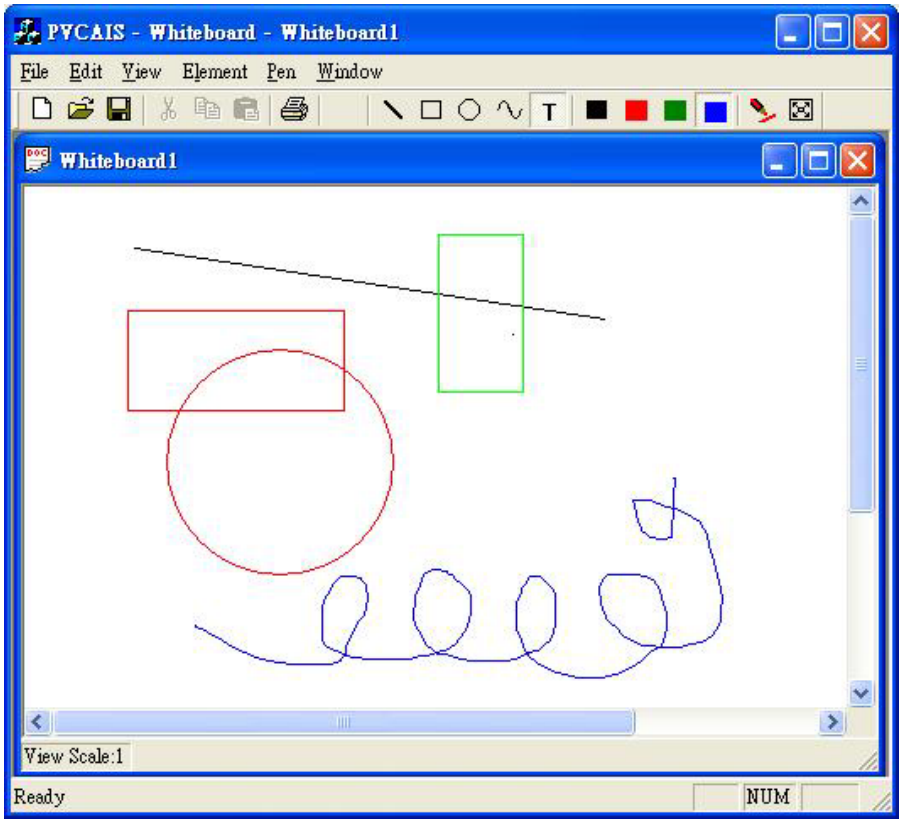

Figure 4.14 Whiteboard drawings

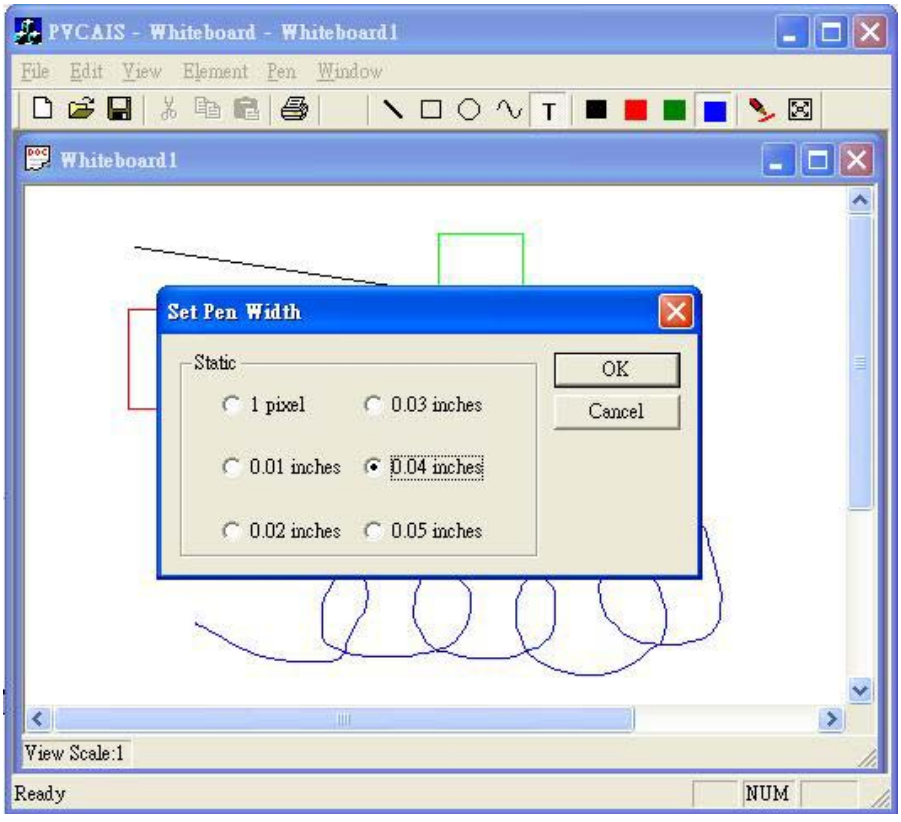

Figure 4.15 Whiteboard set pen width function

Besides drawing on the whiteboard, users can also input text into the drawing to explain what has be drawn.

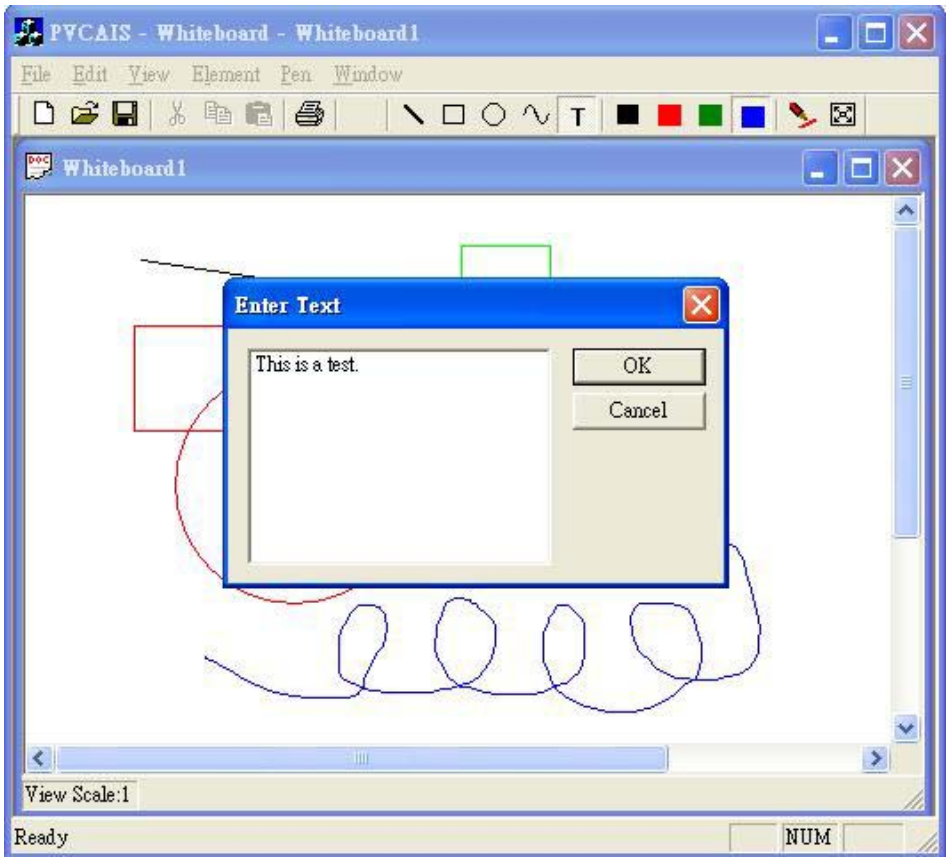

Figure 4.16 Text can be input to the whiteboard

Each circle, rectangle, curve, straight line and text on the whiteboard are independent image objects and can freely moved to any place in the whiteboard or deleted from the whiteboard.

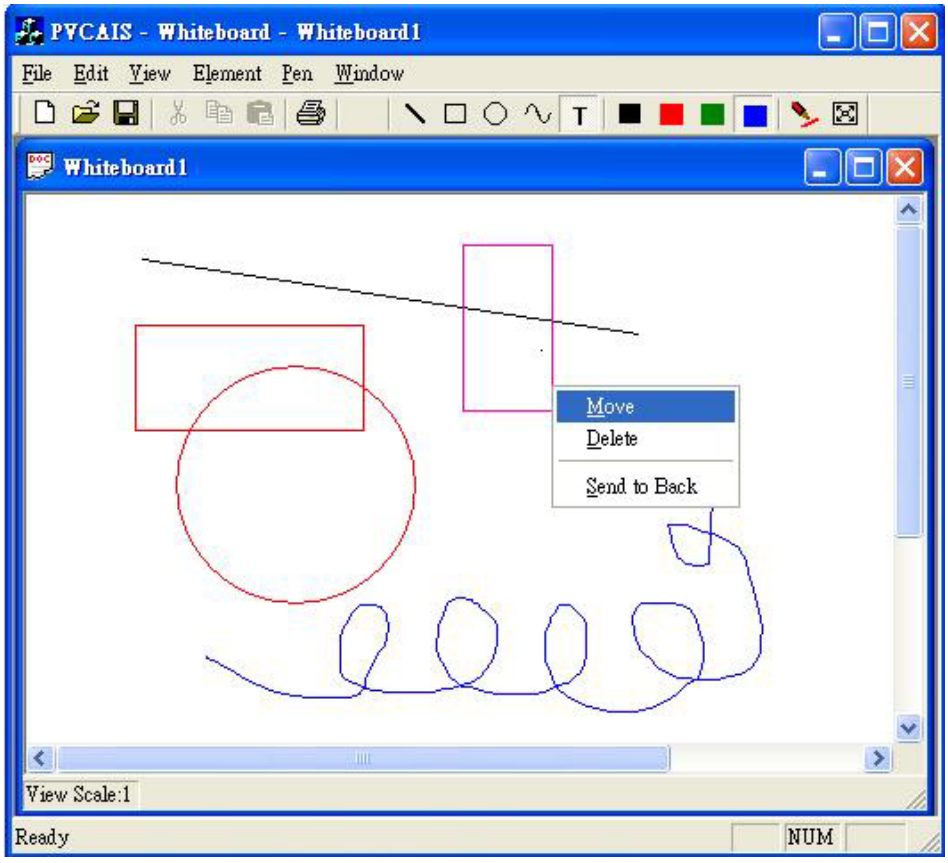

Figure 4.17 Move and delete function

4.8 Media Acquisition Module

In this module, the major data that we want to collect includes video, audio, member information, text chat, file transfer and whiteboard. Each videoconference will have a directory named by the start date and time of the conference. The above data will be stored in different files under the videoconference directory it belongs to. In this way, we can easily separate the data to do indexing later. Now let's have a description on how we obtain the conference data from our videoconference client.

4.8.1 Member Information

One of the most important data of a video conference is who have taken part in a conference. In the NetMeeting 3 SDK, there is a COM object

called INmMember which represents a participant in the videoconference. This COM object cannot be created by us and is automatically created by the NetMeeting system when a new member joins the conference. It contains information about the member it represents, the information includes:

1. The name of the member
2. A unique identifier of the member in the local system.
3. The NetMeeting Version that the member is using.
4. The address of the member, this may be IP address or machine name.

Each time when a new member joins or leaves the conference, a notification will be raised by the system.

An event sink called

```
HRESULT MemberChanged(
   [in] NM_MEMBER_NOTIFY uNotify,
   [in] INmMember *pMember
);
```

which is a method of the INmConferenceNotify interface is responsible for catching this notification. The parameter *pMember is an INmMember object pointer pointing to the member object responsible for this event. The parameter uNotify is used to state what causes this event, e.g. a member leaving or a new member joining.

So by implementing the INmConferenceNotify interface and its method MemberChanged, we can collect information of who have joined the conference, when they join and leave, and their addresses. This information will be stored in a file called MemberInfo.txt in the directory called MemberData.

4.8.2 File Transfer

During a conference, the members may want to exchange files with each other. The NetMeeting 3 SDK has provided a file transfer component to send files and an embedded component for receiving files. Programmer can simply use these two components for file exchange. However the file transfer component's source code is not available, therefore we have no way to obtain the file exchange information from it. So we implement our own file transfer function. User sends file by entering the name of the receiver or entering "Everyone" if the file is received by all the members of the conference. Each time the user sends a file, we will record the time, date, file's name and the name of the receivers.

When a file is incoming, the NetMeeting 3 SDK will automatically trigger a file receiving component to receive the file and raise a notification. An INmFt COM object will be created to access the file received. This object contains information about the file it represents, includes:

1.The name of the file

2.The size of the file

3.The sender or receiver of the file

The status of the file, weather it is incoming or outgoing

An event sink called

```
HRESULT FtUpdate(
   [in] CONFN uNotify,
   [in] INmFt *pFt
);
```

which is a method of the INmChannelFtNotify interface is responsible for catching the notification. The parameter uNotify states whether the file is incoming or sending out. The parameter pFt is an INmFt COM object pointer pointing to the INmFt COM object of the file responsible for this event. So by implementing the INmChannelFtNotify interface and its

method, we can record the information on file receiving. The information will be stored in a file called fileInfo in the directory FIleTransferData.

4.8.3 Text Chat

During a conference, the members may want to send text messages with each other. The NetMeeting 3 SDK has provided a data channel for text chat to send/receive messages. Each time the user sends/receives message, we will record the time, date, sender/receiver's name and the message content.

When a message is sent/received, the NetMeeting 3 SDK will automatically raise a notification in the sender and receiver's data channels.

The two event sinks called

```
HRESULT DataSent(
   [in] INmMember *pMember,
   [in] ULONG uSize,
   [in] byte *pvBuffer
);
```

and

```
HRESULT DataReceived(
   [in] INmMember *pMember,
   [in] ULONG uSize,
   [in] byte *pvBuffer,
   [in] ULONG dwFlags
);
```

which are methods of the INmChannelDataNotify interface are responsible for catching the notification. The parameter pMember states the receiver

for the method DataSent while it states the sender for the method DataReceived. The parameter uSize is the number of bytes of the data and the pvBuffer is a pointer pointing to the data buffer. The parameter dwFlags of method DataReceived specifies whether the message is a multicast or a private message. By implementing the INmChannelDataNotify interface and its method, we can record the information on message transfer. The information will be stored in a file called chatData in the directory ChatData.

4.8.4 Audio

The NetMeeting 3 SDK does not provide any SDK for us to record the audio conversation of the meeting. Therefore we need to use an indirect way to record the audio of the conference. The program we used to record the audio is called waveTrans. Each time when a conference starts, the program will start a new thread to record the local audio from the local microphone. When a certain amount of audio data is recorded, the program will compress the data and send it to other members of the conference. When the conference is finished, the program will combine all the audio files received and recorded to a single audio file. In this way, the audio of the conference can be recorded. The audio file will be stored in the directory called voiceData.

4.8.5 Video

The NetMeeting SDK does not provide any interface or function for us to retrieve the video data, so we have to use an indirect method to collect the data. The method we use is screen capturing. Each time when the key-frames capturing function is activated, a thread will be created to keep checking the display on the two video windows on the screen. When the change of the scene is larger than a threshold value within a small period of time, the picture on the video window will be captured and stored as still images. These images will be the key-frames of the conference and they will be stored in the directories called videoArchieve(for local video)

and videoArchieveR(for remote video).

4.8.6 Whiteboard

When two users use the whiteboard in their conference, any change on the whiteboard made by either side will be immediately shown on the other side. This is done by sending notify messages to the other sides. We do not send the whole image to the other side because the size of the message will be very large and the performance will be very low if there are frequent changes on the whiteboard. Also, there is very difficult to analyze the content of an image and do searching on it, therefore we have used other methods to update and save the whiteboard content.

Each line, curve, rectangle, line and text are called image objects in our system, our program will keep a list of these image objects and their properties, for example, their types, sizes, colors and positions on the whiteboard. Each time when the user drawn some things on the whiteboard, new image objects will be created and inserted into the list, and the whiteboard will be drawn and updated according to the image objects stored in the list. A message containing the properties of the newly created image objects will also be sent to the other side so that the whiteboard on the other side can be updated at the same time.

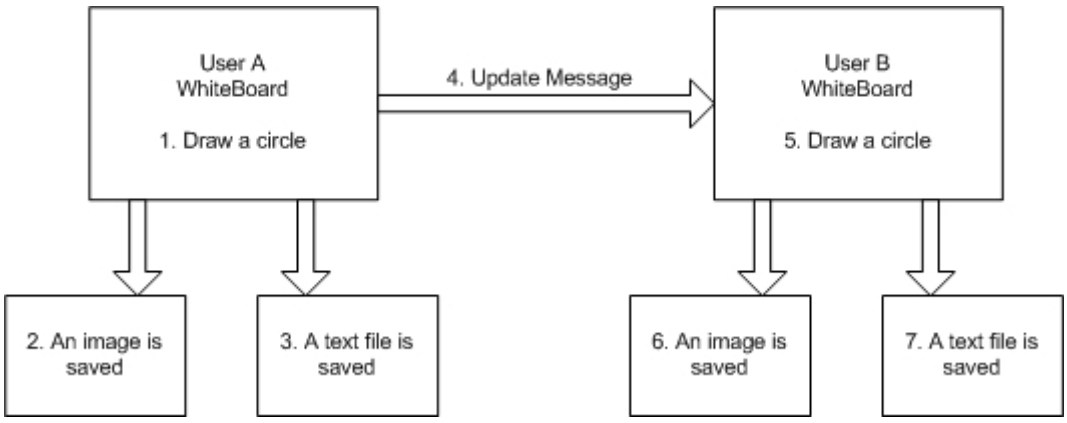

Figure 4.18 Whiteboard update flow diagram

Since we have a data structure to describe the content of the image shown on the whiteboard, each time when the whiteboard is updated, besides saving the content as an image, we can also generate a text file to describe the content of the image. For example, when user A draws a green rectangle on his whiteboard, an image of the whiteboard at this moment will be saved, a text file describing this image will be created at the same time. On the other hand, a message storing the properties of this green rectangle will be sent to user B. User B 's whiteboard will be updated after receiving this message, an image and text file of his whiteboard will also be created too.

The saved images and text files can then be used in the Videoconference Accessing Module. Users can now search the whiteboard content by using the text files, for example, he can state what type of image objects are in the whiteboard he is looking for, he can input "a red circle" and "a green rectangle" as searching criteria. After the searching, corresponding whiteboard images will be shown to the users

4.9 Archive Indexing Module

In this module, we will further process the raw data collected to produce indices for the conference. Since the member information, text chat, file transfer record is already in text format and is easy for searching, we have put our focus on the video and audio data as these two data is difficult to search and do indexing.

4.9.1 Audio

Our system will use the audio file recorded from the conference as an input for a voice recognition module. A text file containing the content of the audio will be generated. The voice engine we used in our system is the Microsoft Speech API (SAPI). The text file generated can then be used for indexing. It will be stored in the same directory as the audio file

4.9.2 Video

After each videoconference finishes, we need to change remote key-frames in BMP format into JPEG. We have integrated the CMU Face Detection Module to process on the JPEG images. If a face is found, the module will return a face region which is a 64*64 pixel BMP file. After that, the BMP file is passed to a previously trained Face Recognition module which implements the EigenFace algorithm. For details of the EigenFace algorithm, please refer to Section 4.7.1.1. Then face recognition result is displayed to users and also written to a file for later indexing and searching.

4.10 Videoconference Accessing Module

In this module, we have implemented a user-friendly interface to search and list the information of indexed videoconference archives. It also allows users to playback a selected videoconference in a synchronized way. The searching function is implemented by Visual C++ while the synchronized playback function is supported by SMIL.

4.10.1 Searching Interface

The searching user interface is divided into 3 parts, namely Conference List, Conference Information and Conference Searching. The snapshot of the user interface is shown in Figure 4.19. Initially, the Conference List will show all the past videoconference archives that the user has done before. Users can input their searching criteria in the Conference Searching Box. After they click the button “Search Now”, the system will search the index files of past videoconference archives and refresh the Conference List to show those archives which match the criteria.

Users can select any one of the videoconferences in the Conference List and its information will be displayed in the Conference Information Box. If the user is interested in that archive, he/she can browse that

videoconference by clicking the "Browse It Now" button, then the Playback Interface will be launched and the archive can be reviewed.

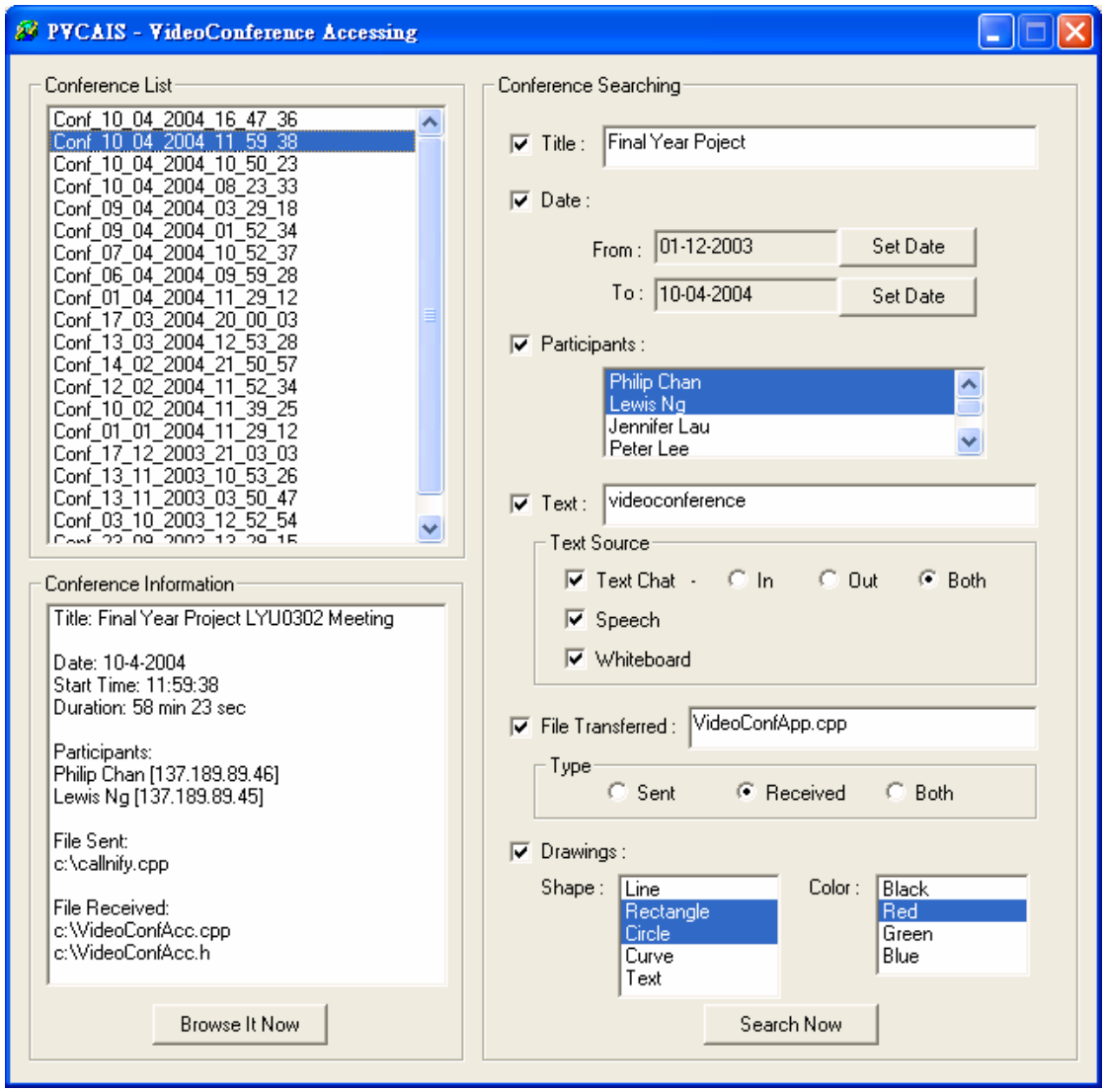

Figure 4.19 Snapshot of Searching Interface

4.10.1.1 Conference Searching

The Conference Searching Box provides six searching functions for users, namely Title, Date, Participants, Text, File Transferred and Drawings. Users can easily enable/disable a function by checking/unchecking the check box besides that function. After clicking the button "Search Now", those archives which match all the

checked functions will be updated in the Conference List.

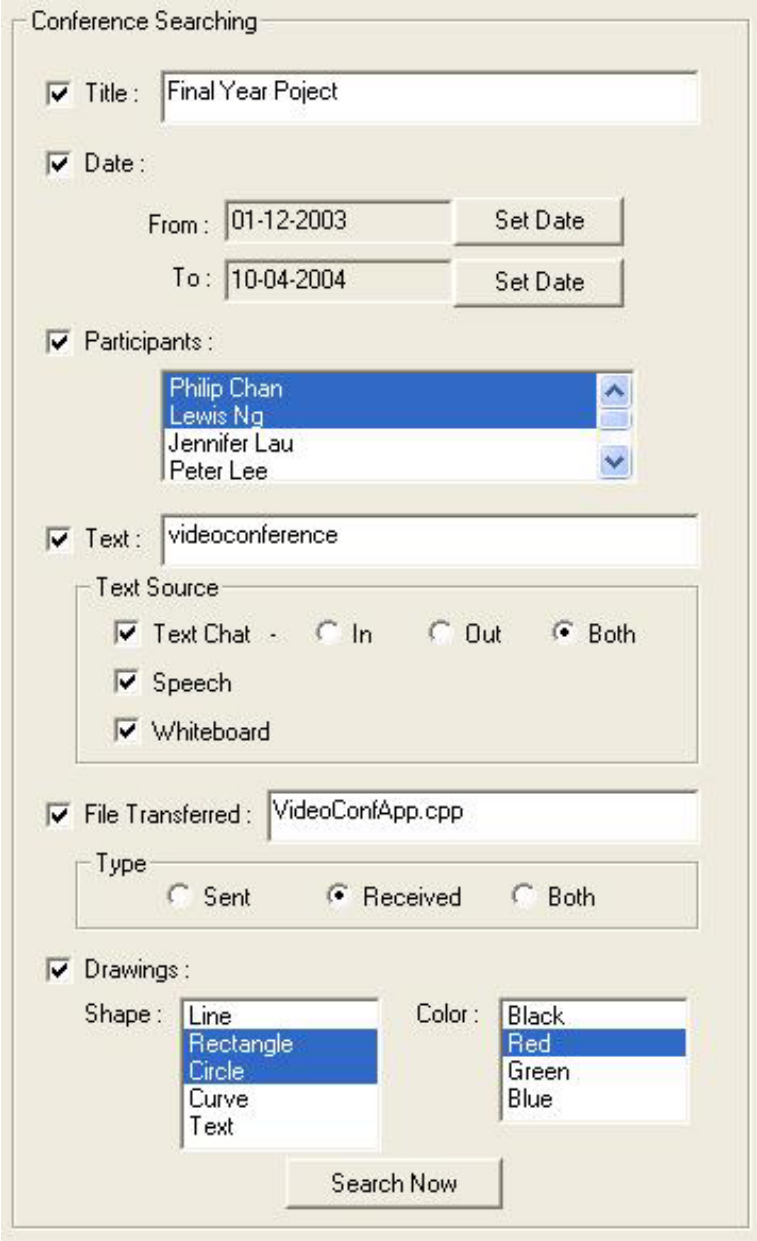

Figure 4.20 Conference Searching

4.10.1.1.1 Search by Title

The Search by Title function allows users to input text to search the title and the system will search whether the title of an archive contains the input text as a substring.

Figure 4.21 Search by Title

4.10.1.1.2 Search by Date

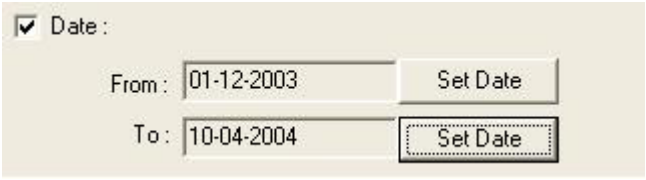

Figure 4.22 Search by Date

The Search by Date function allows users to specify a search period by inputting a Start Date and an End Date. Instead of traditional text field or combo box, the user interface uses a Calendar Box for the input. Users can click the "Set Date" button, and a window containing a calendar will be popped-up. The current date will be marked by a red circle and the selected date will be shaded in blue circle. Initially, the selected date is today. Users can easily select another date by directly clicking on the date of the calendar. They can also click the "left arrow" or "right arrow" button to change to another month. After they have confirmed their selection, they can press the "SET" button or press the "Cancel" button if they would like to withdraw their selection. All archives with start date within the selected period inclusively will be counted as matching the criteria.

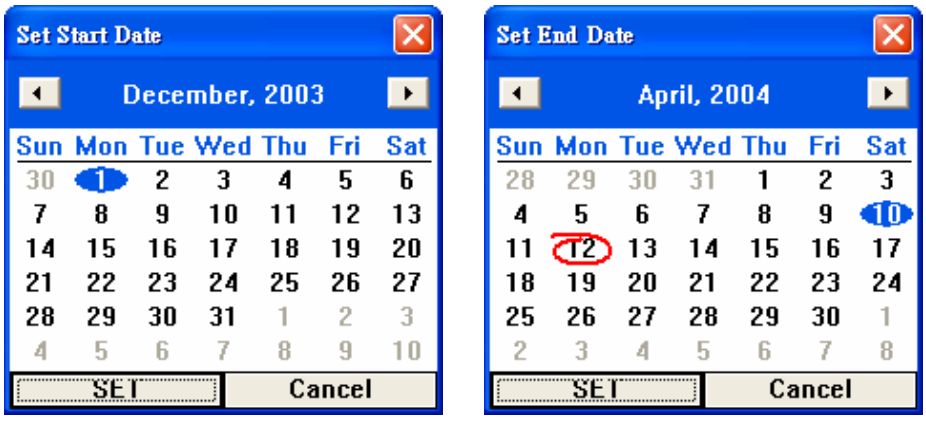

Figure 4.23 Set Start Date and Set End Date windows

4.10.1.1.3 Search by Participants

The Search by Participants function provides a multiple-selection box for users to input their interested participants. The list of members in the list contains all people who had done videoconferences with the users. Users can select one or more participant and the system will search for the archives which include all the selected users as participants in that videoconference.

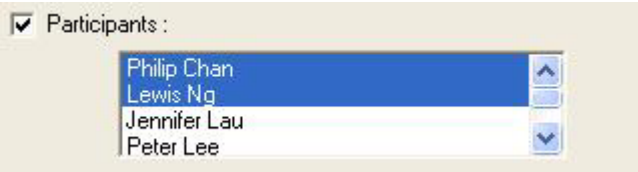

Figure 4.24 Search by Participants

4.10.1.1.4 Search by Text

The Search by Text function provides a text field for users to input their interested text. Users also have to select which Text Sources the searching are effective on. These include texts appear in Text Chat, in Speech (by the result of Speech Recognition) or in Whiteboard. If the user has checked Text Chat, he/she has to specify whether he/she want to search only the incoming message or outgoing message or both. The default setting is "Both". The system will then search whether an archive contains the input text as substring in the specified text sources.

Figure 4.25 Search by Text

4.10.1.1.5 Search by File Transferred

The Search by File Transferred function provides a text field for users to input their interested filename. Users also have to select which type of files the searching is effective on. These include files sent, files received or both. The default setting is "Both". The system will then search whether an archive contains the specified type of files transferred.

Figure 4.26 Search by File Transferred

4.10.1.1.6 Search by Drawings

The Search by Drawings function provides two multiple-selection boxes for users to input their interested shapes and colors. Users can specify one or more shapes/colors which they would like to include as the searching criteria. The system will then search whether the whiteboard content of an archive contains all the specified shapes and colors.

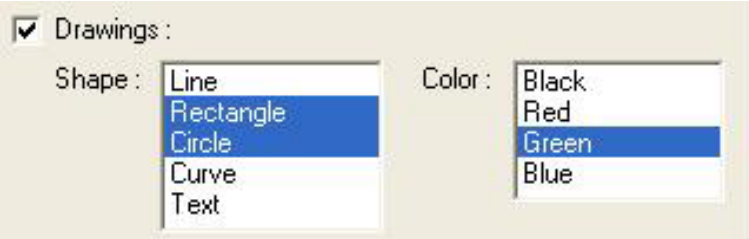

Figure 4.27 Search by Drawings

4.10.1.2 Conference List

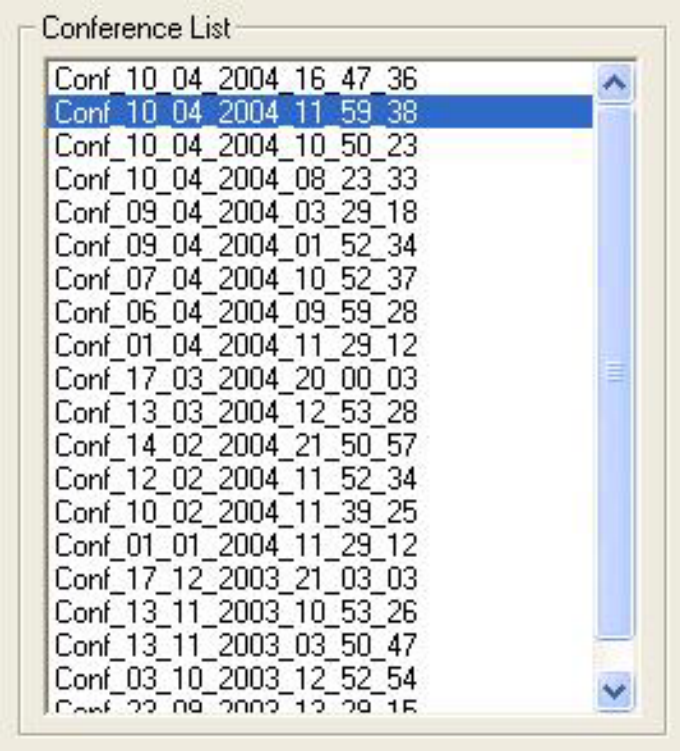

Figure 4.28 Conference List

The Conference List initially shows all the past videoconference archives. After users set the searching criteria and pressed the "Search Now" button, the search result will replace the original list. Users can resume the original list by unchecking all the selection functions and doing searching again. The conferences shown in the list are sorted by descending order of the conference date and time. In other words, the most recent one will be listed first.

4.10.1.3 Conference Information

After selected a desired conference in the Conference list, that conference will be highlighted in the list and the information of that conference will be shown automatically in the Conference Information Box. The information includes Title, Start Date, Start Time, Duration, Participants (together with their IP), Files Sent and

Files received. Users can choose whether they want to review that videoconference by clicking the "Browse It Now" button. If it is clicked, the Playback Interface will be popped-up to play the whole videoconference in which all channels are synchronized according to their timestamps specified in the index file.

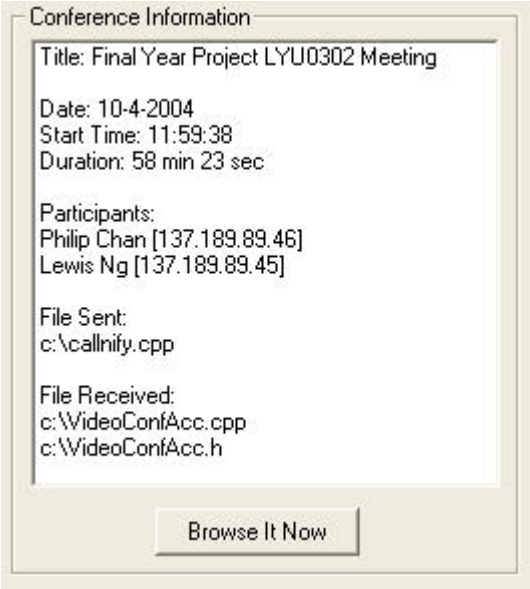

Figure 4.29 Conference Information

4.10.2 Playback Interface

After the searching part, our system can let the user to review to content of the videoconference he or she has selected. However, there are many data channels in a videoconference and the content of these channels are extracted independent of each other, if user reviews these channels content separately, then it is very difficult for the user to understand the main idea or issues discussed in the conference. For example, if two users are discussing the design of a new product, they may at the same time draw on the whiteboard and talk through the microphone. If a user just reviews the whiteboard images saved without listening to the audio record of the conference, he may not know what the exact design of the new product is and how it should be made.

Therefore, we need to find a way to let the users review all channels' content of the conference in a synchronous way. We have chosen SMIL to do this job since a SMIL presentation can embed different types of media format in it.

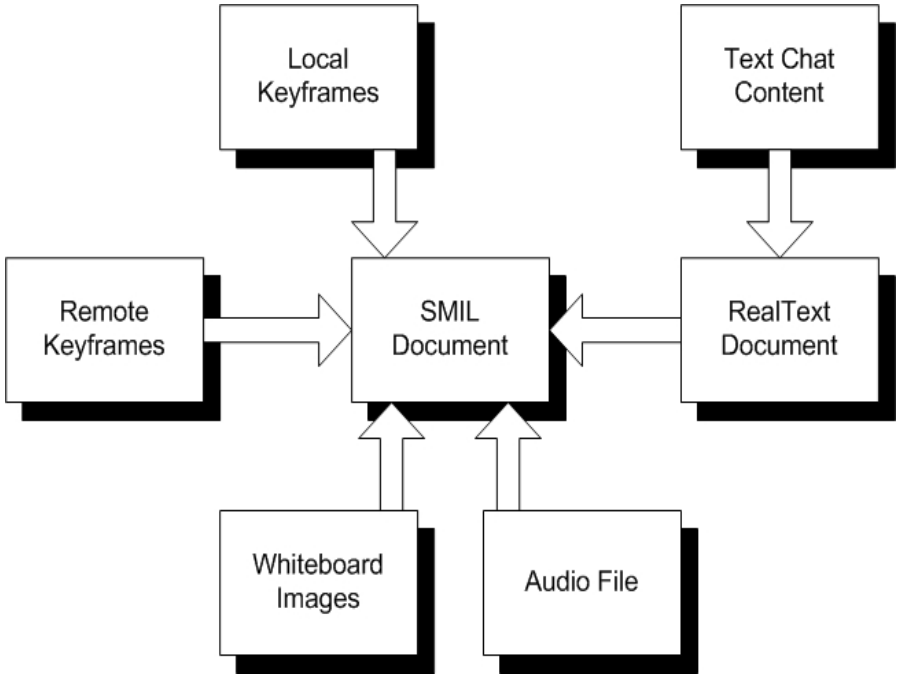

Figure 4.30 SMIL document generation process

In the Media Acquisition Module, we have collected the content of all the data channels of a videoconference. In the Archive Indexing Module, we have processed some of the raw data into more useful and searchable data. And in this module, the Videoconference Accessing Module, we have implemented a module called SMIL generator to generate the SMIL presentation of the videoconference from the data collected. All the data files will have a time stamp included and this time stamp is used to state when this data is created during the conference. During the SMIL generating process, we will use this time stamp to determine the time to display this content to the users. For example, if the users drew an image on the whiteboard at 20 seconds after the conference started, then in the SMIL presentation, we will display this image 20 seconds after the

presentation started. The text chat content will be first converted into RealText format before putting into the SMIL presentation. This is because SMIL does not have very advance control on the display of text, so we need to convert the text chat content into a streaming text format like RealText to provide a better display effect. In this way, the users can truly review the content of the videoconference.

The SMIL presentation window is divided into four regions, two small windows for displaying the local and remote key frames collected, one large window for displaying the whiteboard image and a text box for displaying the text chat content. The audio record will be played at the background at the same time.

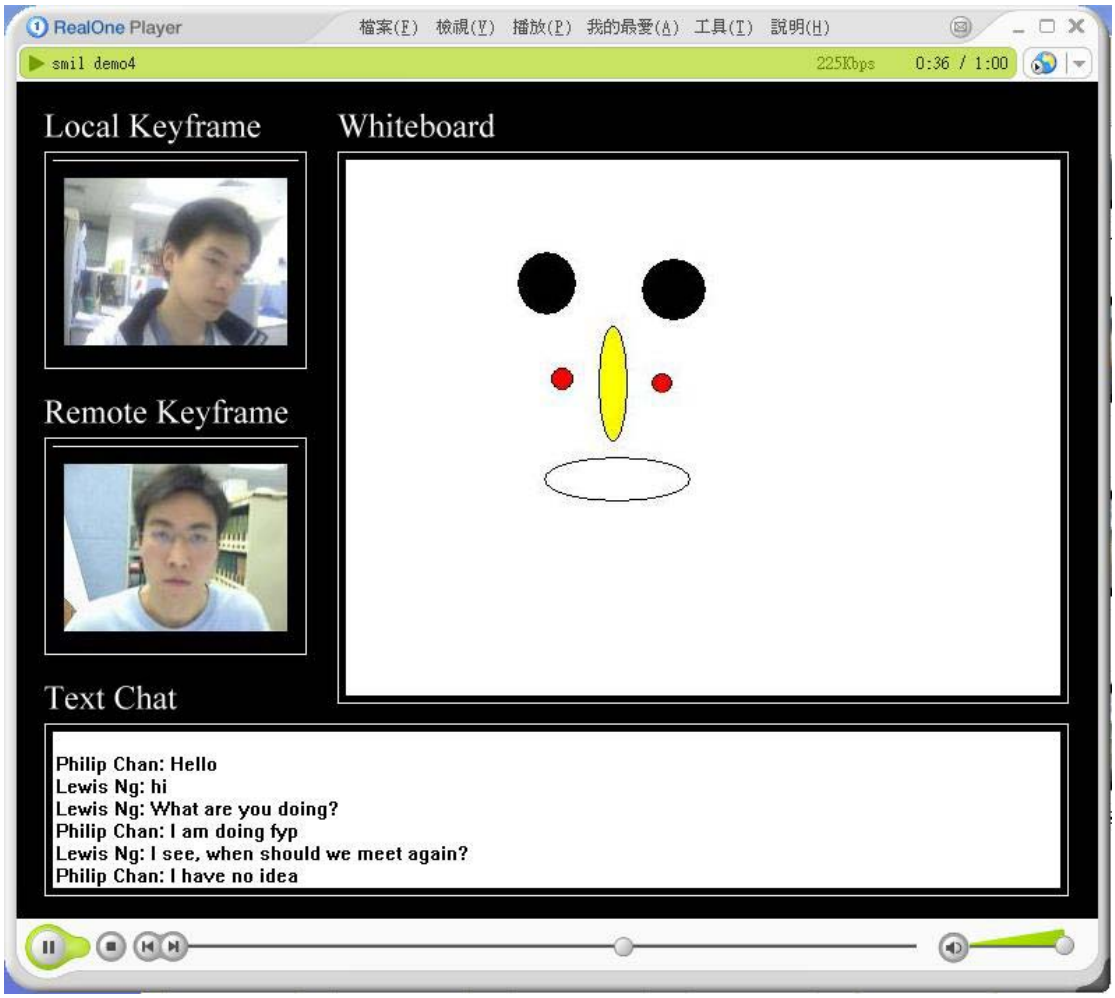

Figure 4.31 Snapshot of Playback Interface

5. Problems and Solutions

Problem 1

We cannot find any source code of the existing videoconference program and so we do not have tool to collect the raw data of videoconference.

Solution

We find out that Microsoft has provided a SDK for building application based on the NetMeeting. We therefore decided to build our own videoconference client.

Problem 2

After we decide to use the NetMeeting 3.0 SDK to build our own videoconference client, we find that the language that the SDK supports is C++ which we have not learnt before. Also the API is in the form of COM interfaces and functions which is a totally new programming concept for us. Since NetMeeting is product of Microsoft, therefore Windows will be the platform we used to build our system. However, we are not familiar with Visual C++ and Windows programming.

Solution

We have spent two months to get familiar with all these materials. We read books on Visual C++ and study some on-line Windows programming tutorials. Most of the time is used to study COM. We have read several tutorials and have tried to write some COM programs. We also study the MFC to learn how to build the graphical user interface in Windows. After we get familiar with Visual C++ and COM, we study the NetMeeting API tutorials and samples to learn how to use and implement the interfaces provided.

Problem 3

We find that the NetMeeting SDK does not provide any functions for us to retrieve video, audio and whiteboard data.

Solution

We use indirect methods that have been mentioned in previous section to collect the data. For the whiteboard, we decide to build our own in the future.

Problem 4

We find that the file transfer component of the NetMeeting SDK does not provide any information of the file being transferred.

Solution

Instead of using the file transfer component provide by the SDK, we build our own one to perform the same function.

Problem 5

We need to use face detection and recognition in our system, but we are not familiar with these two technology.

Solution

We obtain the source code of the face detection and recognition system developed by Sunny and Harvest(Graduated M. Phil. Students of Professor Lyu). We spent one month to understand and test their code.

Problem 6

We need to use voice recognition in our system, but we are not familiar with this technology.

Solution

We study the existing voice recognition engine API, the Microsoft SAPI and IBM ViaVoice. We use SAPI because it is easier to integrate into our system.

Problem 7

The accuracy of voice recognition is not very good.

This problem cannot be solved yet. We have tried IBM ViaVoice to see if it can give better result or not, however the result is not good too. We find that in order to get better result for voice recognition, we need to do a series of training on the system. For example if we want to recognize the voice of user A, then we need to ask user A to speak some pre-recognized words to the system to let the system "accustom" itself to the voice of user A. However, we think that it is not practical in a videoconference program, because each time you talk with a new friend, you will need to ask him to do the training process. Also, as the audio file records the conversation of the members of the videoconference, there will be more than one human voice, and sometime the members will be talking at the same time, the voices will be overlapped. All these are great challenges to the voice recognition engine and so the accuracy cannot be easily improved. More research work has to be done on it.

Problem 8

Our videoconference client is not very stable, it may crash suddenly or cannot create connection with the other sides.

This problem cannot be solved yet. We have checked our code to find out the problem, but we find that it maybe a bug of the NetMeeting SDK. This is because the problem seldom occurs, and we find that the sample codes provided by the official NetMeeting website have the same problem too.

Problem 9

Sometimes when we start our videoconference client or the original NetMeeting program, the screen will flash.

This problem cannot be solved yet. At first we suspect this problem will only occur in Window XP, however after trying our system in Windows 2000, the same problem will occur too. Also we find that the standard NetMeeting program that shipped with Microsoft Windows will have the same problem too, but like our program, it seldom occur.

Problem 10

The accuracy of face recognition is not accurate enough.

Solution

We find that the image captured by our web cam is not very clear, so we have bought new web cam which use CCD as sensor, the image captured is much clear and the accuracy is significantly improved. We also find that the environment condition may also affect the result, for example the lighting on the environment will greatly affect the accuracy. Therefore, we have to tune the web cam according to the environment we run our system.

Problem 11

We have built our own whiteboard and we try to use the data channel of NetMeeting SDK to transfer the update messages to the other members of the videoconference. We succeed to send the message through the channel but fail to receive the message on the other side.

Solution

We find that the message is mixed with other messages sent by the NetMeeting SDK. We therefore create our data channel for transfer the update messages. We use Winsock to create the connection between the whiteboards of the members.

Problem 12

We want to do indexing on whiteboard images saved, but image files are difficult to index and not convenient for searching.

Solution

Each time when there is update on the whiteboard and an image is saved, we create a text file for the image to describe the content of the image. In this way, we can search the text files instead of the image files.

Problem 13

The content of the channels of the videoconference are collected independently, we need to find a way let user to review all these content at the same time and in a synchronized way.

Solution

We decided to generate a SMIL presentation document from all the raw data. Users can then review all the content of the videoconference by playing this SMIL presentation.

Problem 14

We use SMIL for playback, but SMIL does not have many advance controls on plain text and the display effect is not good.

Solution

In order to provide better display effect, we convert the text chat content into a streaming text format. The format we use is RealText.

6. Project Progress

Month	Tasks Completed
MAY 2003	Get familiar with the programming environment of Microsoft Visual C++
	Study the background information of NetMeeting and Videoconference technology
JUN 2003	Study MFC structure and COM model for Windows programming
	Study Microsoft NetMeeting API 3.0 documentation and samples
JUL 2003	Test and study the source codes of the face detection and face recognition modules
AUG 2003	Develop a basic Videoconference client by NetMeeting API with chatting, file transfer and whiteboard features.
	Study the mechanism and media format of a Videoconference client to obtain media data
SEP 2003	Integrate key-frame selection module and audio recording module into the Videoconference client
	Study different speech recognition techniques
OCT 2003	Retrieve information about data communication events : conference member information, text chat log and file transfer information from the Videoconference client
	Integrate speech recognition - SAPI into the system to generate script from audio data
	Integrate face detection and face recognition module into the system
NOV 2003	Implement simple indexing and searching algorithm
	Implement a combined GUI for Videoconference client and Conference management
	Prepare First Term FYP presentation and demonstration
	Write First Term FYP report

Month	Tentative Schedule for Future Tasks
DEC 2003	Design own whiteboard structure and standard which can be retrieved and indexed for further searching and playback purposes
	Improve stability and reliability of audio recording module and key-frame selection module
JAN 2004	Study Visual C++ drawing API (CDC) for the implantation of whiteboard
	Implement a stand-alone whiteboard program which supports drawings and texts and allows users to move and delete existing drawings
	Study Face Verification algorithms and samples and design the Face Verification Login feature of videoconference client to enhance security
FEB 2004	Integrate the whiteboard program into existing videoconference client
	Develop Face Verification Login feature for the videoconference client and implement Face Verification Training user interface
	Develop the media acquisition and indexing of whiteboard to enable searching and synchronized playback of whiteboard content with other media
	Study SMIL technology for the development of synchronized playback system
MAR 2004	Implement the SMIL File Generator which converts videoconference index file into SMIL format after each videoconference finishes
	Improve the accuracy of Face Detection, Verification and Recognition
	Redesign and rebuild the VideoConference Accessing Module, including a more detailed searching of indexed videoconference archives and synchronized playback of a selected videoconference supported by SMIL

APR 2004	Review and finalize source codes of different modules of the system
	Prepare final FYP presentation and demonstration
	Write final FYP report

Summer	Term 1	Term Break	Term 2

7. Contribution of Work

7.1 Introduction

In this part, I will state the contribution of my part to this project and what I have learnt through the work progress.

The PVCAIS is a system that consists of 3 main modules including the Media Acquisition Module, Archive Indexing Module and the Videoconferencing Accessing Module. The details of preparation work and work done for each of the three modules will be discussed below.

The three modules cannot be done in parallel, as the second module needs the output of the first module and the third module needs the output of the second module. Our first term target is to focus on the first module and the second modules: Media Acquisition Module and Archive Indexing Module. However, in order for completeness of our first term's demonstration, we have also implemented parts of the features in the Videoconferencing Accessing Module.

In the second term, our main focus is on the third module: Videoconferencing Accessing Module. However, we have also improved and added features to the first and second modules to make the system more complete and reliable.

7.2 Preparation Work

Our preparation work started in May, 2003. We have decided to use Microsoft Visual C++ and NetMeeting SDK to implement our system. As we were not familiar with the programming environment of Visual C++, COM model and MFC, both of I and my partner first needed to study references on these issues. We have searched and borrowed quite a lot of materials to read. I have tried using it to implement some simple programs to test its functionality.

Also, as NetMeeting SDK is an important tool in our project, in order to be familiar with NetMeeting SDK, we have read the NetMeeting SDK documentation and tried the samples in the summer. And I had tried out the behavior of personal videoconference by using the software NetMeeting.

As we had to use face detection and face recognition modules in our system and we were not familiar with them, so we had spent about 1 month to investigate and test the codes of the modules.

For the second term, in order to build the whiteboard, we had studied the drawing API and other related tools in Visual C++. Also, as we decided to use SMIL for the implementation of synchronized playback function of our system, we had studied the format and syntax of SMIL technology.

For the Face Verification Login Feature, I had investigated the algorithms and samples of face verification in order to provide a solution to our system to include this feature.

7.3 Media Acquisition Module

My contribution to the Media Acquisition Module in first term was mainly on the implementation of the chatting function of our videoconference client and the extraction of the test-chat channel data by using NetMeeting SDK. I had investigated into the sample codes of NetMeeting SDK which involves text chat features and read the related documentation. I had successfully extracted the text chat data from the videoconferencing client. I also designed a format to write the text chat information into the media file.

We had successfully implemented a videoconferencing client with audio, video, chatting, whiteboard and file transfer functions in this phase. Moreover, we have successfully extracted all the channel data except the whiteboard channel in the first term and all data extracted is saved for the use in the

second module.

In the second term, I added a Face Verification Login feature to our self-written videoconference client to enhance security. I had also developed a GUI for the Face Verification Training.

For the whiteboard, as current NetMeeting API doesn't specify the data format of the whiteboard feature, we cannot decode the media data in the existing NetMeeting videoconference client, so we designed to design our own whiteboard data transfer structure and standard which can be retrieved and indexed for further searching. My partner had implemented a self-designed whiteboard in which the content can be extracted and indexed for later reference.

Also, we had improved the stability of the sound recording and key-frame selection module of the first term's program.

7.4 Archive Indexing Module

My contribution to the Archive Indexing Module in first term was mainly on integrating the Face Detection and Face Recognition multimedia-indexing functions.

I investigated and did testing on the code of the Face Detection and Face Recognition functions. I had integrated these two functions into our implementation and built the user interface for this part.

I had also tested and read the documentation of the IBM ViaVoice and Microsoft SAPI, which are two different commercial speech recognition engines.

In the second term, I have fine-toned and improved the accuracy of the Face Detection and Face Recognition modules. My partner had implemented the

indexing of the whiteboard content.

7.5 Videoconference Accessing Module

My contribution to the Videoconference Accessing Module in first term was mainly on building the searching system for text-chat information and its interface. The text-chat searching function enables users to search a substring appearing in the outgoing messages, incoming messages or both. The system will then search for all past videoconference archives and return a list of matched result, when one of them is selected, users can playback all the data from different channels separately.

In the second term, we rebuilt the whole Videoconference Accessing Module to enable more searching functions and synchronized playback of past videoconference archives.

My contribution to this part was to build the searching system with GUI which supports searching on the title, date, participants, texts (in Text Chat, Speech and Whiteboard), file transferred and drawings (in Whiteboard). Users can also view the information of each selected archive which includes the title, date, time, duration, participants and file transfer information.

If they would like to browse a selected archive, they can click the button "Browse It Now" to launch the SMIL file for the synchronized playback of different channels of the videoconference which includes text chat, local video key-frames, remote video key-frames and whiteboard contents. The synchronized playback part is done by my partner.

7.6 Conclusions

In conclusion, I am now more familiar with the programming environment of Visual C++, and the concept of COM and MFC. We have finished the implementation of a videoconference client using NetMeeting SDK. For the

whiteboard part, we have designed and implemented our own standard. And a Face Verification Login feature is added to the videoconference client. I have implemented the GUI for the Face Verification Training. I have also built the acquisition of the text-chat data and designed its storage format. All data channel can be extracted and saved as media files. I have tested on the Speech Recognition engine, investigated and integrated the face detection and recognition functions and built its user interface. I have also implemented the searching system for all media information and its user interface. We have developed a synchronized playback system of videoconference archives by using SMIL format. At last, we have reviewed and finalized the source codes of different modules of the system.

8. Conclusions

PVCAIS is a system that provides the convenient searching and browsing support for videoconferencing users on past videoconference archives. It can be divided into three main parts: the Media Acquisition Module, the Archive Indexing Module and the Videoconference Accessing Module. The Media Acquisition Module works together with the videoconferencing terminal to collect the raw data of the conference from all communications channel in real time. The Media Acquisition Module works after the conference. It analyzes the raw data to produce indices for the conference. The Videoconference Accessing Module provides an interface for user to manage the indexed conferences. User can search for a conference and review it by playing back all the stored content.

Through this project, we have studied the NetMeeting SDK and have built a videoconference client which supports all the common videoconference functions, including: video, audio, text chat, file transfer and whiteboard. A face verification login feature is also added into the videoconference client. All the conference raw data can be extracted and saved in different media files. We have studied the technique of face detection and recognition, and speech recognition to do indexing on the media data and to generate the index file for each videoconference. For the Videoconference Accessing Module, we have implemented a user-friendly interface that allows user to search for a conference by different searching criteria. Also, by using SMIL, we have developed a playback interface that allows user to review a specific conference in which all media channels are synchronized.

9. Acknowledgement

We would like to express our gratitude to Professor Michael R. Lyu, our project Supervisor, who has given us many useful suggestions and directions throughout this project.

We would also like to express our thanks to JiQiang Song, Edward Yau who gives suggestions and ideas in our project, YangFan Zhou, Sunny Tang and Harvest Jang who have offered help for the implementation of our project.

10. References

[1] A. Ginsberg, *et al*. The little web schoolhouse using virtual rooms to create a multimedia distance learning environment. In Proc. ACM Multimedia'98 Conference, pp. 89-98, 1998.

[2] Acosta, E., Torres, L., Albiol, A., & Delp, E. (2002). An automatic face detection and recognition system for video indexing applications. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'02)*, Vol.4, IV-3644-IV-3647.

[3] Albiol, A., Torres, L., & Delp, E.J.(2002a). Video preprocessing for audiovisual indexing. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'02)*, Vol.4, IV-3636 -IV-3639.

[4] Barras, C., Lamel, L., & Gauvain, J.-L. (2001). Automatic transcription of compressed broadcast audio. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'01)*, Vol.1, 265-268.

[5] Ben-Arie, J., Pandit, P., & Rajaram, S. (2001). View-based human activity recognition by indexing and sequencing. in *Proc. Intl. Conf. on Computer Vision and Pattern Recognition (CVPR'01)*, Vol.2, II-78 -II-83.

[6] Bescos, J., Menendez, J.M., Cisneros, G., Cabrera, J., & Martinez, J.M. (2000). A unified approach to gradual shot transition detection. in *Proc. Intl. Conf. on Image Processing (ICIP'00)*, Vol.3, 949-952.

[7] Calic, J., & Lzquierdo, E. (2002). A multiresolution technique for video indexing and retrieval. in *Proc. Intl. Conf. on Image Processing (ICIP'02)*, Vol.1, 952-955.

[8] Castleman, K.R. (1996). *Digital Image Processing*. Englewood Cliffs, New Jersey: Prentice Hall.

[9] Chang S.-F. (1995).Compressed-domain techniques for image/video indexing and manipulation. in *Proc. Intl. Conf. on Image Processing (ICIP'95)*, Vol.1, 314-317.

[10] Chang Y.-L., Zeng W., Kamel, I., & Alonso, R.(1996). Integrated image and speech analysis for content-based video indexing. in *Proc. the 3rd IEEE Intl. Conf. on Multimedia Computing and Systems (ICMCS'96)*, 306-313.

[11] Chiu, P., Kapuskar, A., Reitmeier, S., & Wilcox, L. (1999). NoteLook: taking notes in meeting with digital video and ink. in *Proc. ACM Intl. Conf. On Multimedia*, Vol. 1, 149-158.

[12] Del Bimbo, A. (2000). Expressive semantics for automatic annotation and retrieval of video streams. in *Proc. Intl. Conf. on Multimedia and Expo (ICME'00)*, Vol.2, 671-674.

[13] Dharanipragada, S., & Roukos, S. (1996). A fast vocabulary independent algorithm for spotting words in speech. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'96)*, Vol.1, 233-236.

[14] Diklic, D., Petkovic, D., & Danielson, R. (1998). Automatic extraction of representative keyframes based on scene content. in *Proc. the 32nd Asilomar Conf. on Signals, Systems & Computers*, Vol.1, 877-881.

[15] Doulamis, N.D., Doulamis, A.D., Avrithis, Y.S., & Kollias, S.D. (1998). Video content representation using optimal extraction of frames and scenes. in *Proc. Intl. Conf. on Image Processing (ICIP'98)*, Vol.1, 875-879.

[16] Eickeler, S., Wallhoff, F., Lurgel, U., & Rigoll, G. (2001). Content based indexing of images and video using face detection and recognition methods. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'01)*, Vol.3, 1505-1508.

[17] F. Dirfaux, "Key frame selection to represent a video,"In *Proc. Intl. Conf. on Image Processing*, vol.2, pp. 275-278, Vancouver, BC, Canada, 2000.

[18] Foote, J., Boreczsky, J., & Wilcox, L. (1999). Finding presentations in recorded meetings using audio and video features. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'99)*, Vol.6, 3029-3032.

[19] G.D. Abowd, *et al*, "Teaching and Learning as Multimedia Authoring: The Classroom 2000 Project", In Proc. ACM Multimedia'96 Conference, pp. 187-198, November 1996.

[20] Gelin, P., & Wellekens, C.J. (1996). Keyword spotting enhancement for video soundtrack indexing. in *Proc. The 4th Intl. Conf. on Spoken Language*, Vol.2, 586-589.

[21] Gross, R., Bett, M., Yu, H., Zhu, X., Pan, Y., Yang, J., & Waibel, A. (2000). Towards a multimodal meeting record. in *Proc. Intl. Conf. on Multimedia and Expo (ICME'00)*, Vol.3, 1593-1596.

[22] H. Rowley, S. Baluja, and T. Kanade, "Neural Network-Based Face Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(1): 23-38, 1998.

[23] Hidalgo, J.R., & Salembier, P. (2001). Robust segmentation and representation of foreground key regions in video sequences. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'01)*, Vol.3, 1565-1568.

[24] Hsu, C.-T., & Teng, S.-J. (2002). Motion trajectory based video indexing and retrieval. in *Proc. Intl. Conf. on Image Processing (ICIP'00)*, Vol.1, 605-608

[25] I. Bazzi, R. Schwartz, and J. Makhoul. An omnifont open-vocabulary OCR system for English and Arabic. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 21(6): 495-504, 1999.

[26] Iyengar, G., Nock, H., Neti, C., & Franz, M. (2002) Semantic indexing of multimedia using audio, text and visual cues. in *Proc. Intl. Conf. on Multimedia and Expo (ICME'02)*, Vol.2, 369-372.

[27] J.A. Sprey, "Videoconferencing as a communication tool," *IEEE Trans. on Professional Communication*, 40(1): 41-47, March 1997.

[28] J. Song, M.R. Lyu, J.-N. Hwang, and M. Cai, "PVCAIS: A Personal Videoconference Archive Indexing System," Proc. 2003 International Conference
on Multimedia & Expo (ICME2003), Baltimore, Maryland, July 6-9 2003.

[29] Jasinschi, R.S., Dimitrova, N., McGee, T., Agnihotri, L., Zimmerman, J., & Li, D. (2001b). Integrated multimedia processing for topic segmentation and classification. in *Proc. Intl. Conf. on Image Processing (ICIP'01)*, Vol.3, 366-369.

[30] Jasinschi, R.S., Dimitrova, N., McGee, T., Agnihotri, L., & Zimmerman, J. (2001a). Video scouting: an architecture and system for the integration of multimedia information in personal TV applications. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'01)*, Vol.3, 1405-1408.

[31] Jasinschi, R.S., Dimitrova, N., McGee, T., Agnihotri, L., Zimmerman, J., Li, D., & Louie, J. (2002). A probabilistic layered framework for integrating multimedia content and context information. in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'02)*, Vol.2, 2057-2060.

[32] Jennifer L.Beckham, Giuseppe Di Fabbrizio, Mils Klarlund. Towards SMIL as a Foundation for Multimodal, Multimedia Applications. W3C Multimodal Interaction Activity, 2002.

[33] K. Fukunaga, Introduction to Statistical Pattern Recognition
(Second Edition), Academic Press, New York, 1990.

[34] K. Lagus and S. Kaski, "Keyword selection method for characterizing text document maps," In *Proc. Intl. Conf. on Artificial Neural Networks*, vol.1, pp. 371-376 1999.

[35] Kang, E.K., Kim, S.J., & Choi, J.S. (1999). Video retrieval based on key frame extraction in compressed domain. in *Proc. Intl. Conf. on Image Processing (ICIP'99)*, Vol.3, 260-264.

[36] Liang, R.Z., Venkatesh, S., & Kieronska, D. (1995). Video indexing by spatial representation. in *Proc. the 3rd Australian and New Zealand Conf. on Intelligent Information Systems (ANZIIS'95)*, 99-104.

[37] M. Cai, J. Song and M.R. Lyu, "A new approach for video text detection," In *Proc. Intl. Conf. on Image Processing*, pp. 117-120, Rochester, New York, USA, September 22-25, 2002.

[38] M. Christel, T. Kanade, M. Mauldin, R. Reddy, S. Stevens, and H. Wactlar, "Techniques for the Creation and Exploration of Digital Video Libraries." *Multimedia Tools and Applications (Vol. 2)*, Borko Furht, editor. Boston, MA: Kluwer Academic Publishers, 1996.

[39] M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of human faces," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 12, no. 1, pp. 103-108, Jan.1990.

[40] M. Witbrock and A. Hauptmann, "Speech Recognition for a Digital Video Library," *Journal of the American Society for Information Science*, 49(7): 619-632, 1998.

[41] M. Turk and A. Pentland, "eigenfaces for recognition," J. of Cognitive Neuroscience, vol. 3, no. 1, 1991.

[42] M. Turk and A. Pentland "Face Recognition Using Eigenfaces," in Proc. of IEEE International Conference Computer Vision and Pattern Recognition, pp. 586-591, 1991.

[43] Madrane, N., & Goldberg, M. (1994). Towards automatic annotation of video documents. in *Proc. the 12th IAPR Intl. Conf. on Computer Vision & Image Processing*, Vol.1, 773-776.

[44] Nam, J., Enis Cetin, A., & Tewfik, A.H. (1997). Speaker identification and video analysis for hierarchical video shot classification. in *Proc. Intl. Conf. on Image Processing (ICIP'97)*, Vol.2, 550-553.

[45] R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: a survey," Proceedings of the IEEE, Vol. 83, pp. 705-741, May 1995.

[46] R. Jin and A. Hauptmann, "Learning to Select Good Title Words: A New Approach based on Reversed Information Retrieval," in *Proc. Intl. Conf. on Machine Learning (ICML'01)*, Jun 28-Jul 1, 2001.

[47] Rabiner, L., & Juang B.-H. (1993). *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey: Prentice-Hall.

[48] S.G. Deshpande and J.-N. Hwang, "A real-time interactive virtual classroom multimedia distance learning system," *IEEE Trans. on Multimedia*, 3(4): 32-444, 2001.

[49] S. Sato and T. Kanade, "NAME-IT: Association of Face and Name in Video," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 368-373, 1997.

[50] Sawhney, H.S. (1993). Motion video annotation and analysis: an overview. in *Proc. the 27th Asilomar Conf. on Signals, Systems and Computers*, Vol.1, 85-89.

[51] Tang, H.-M., Lyu M.R., & King, I. (2003). Face Recognition Committee Machine. in *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'03)*, 837-840.

[52] W3C. Synchronized Multimedia Integration Language(SMIL) 2.0 Specification. http://www.w3.org/TR/smil20/.

[53] Wei, J., Li, Z.-N., & Gertner, I. (1999). A novel motion-based active video indexing method. in *Proc. IEEE Intl. Conf. on Multimedia Computing and Systems*, Vol.2, 460-465.

[54] X. Tang, "Texture information in run-length matrices," IEEE Transactions on Image Processing, vol. 7, No. 11, pp. 1602-1609, Nov. 1998.

[55] Y. Gao and M.K.H. Leung, "Face recognition using line edge map," IEEE Trans on Pattern Analysis and Machine Intelligence, 24(6): 764-779, 2002.