Department of Computer Science and Engineering

The Chinese University of Hong Kong

2003/2004 Final Year Project

First Term Report

LYU0302

PVCAIS – Personal VideoConference Archives Indexing System

Supervisor

Professor Michael R. Lyu

Chan, Hiu Fai Philip
Ng, Chok Ming Lewis

26 Novemeber, 2003

Table of Content

Abstract

Videoconference becomes more and more popular in business activities, personal communications and educations, because of the rapid growth of Internet bandwidth and technical advance of multimedia coding. The participant of a videoconference usually wishes to keep the video archive for the 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 video archives in his/her work, study and daily life, he/she may not be easy 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, OCR, automatic title generation, keyword selection, etc. Afterwards, an XML index file containing the summary of the videoconference is also generated. The whole indexing process is automatic and transparent except that the face of a new contact needs to be identified by the users for only once. Finally, we will develop a graphical user interface which allows the user to manage, search and browse and modify the indexed videoconference archives conveniently.

This project applies many multimedia indexing functions to personal videoconference archives to generate a searchable content summary automatically. These techniques are well integrated specific to the characteristics of videoconference. Instead of using traditional databases, XML-based repository is used to integrate archive information structurally, 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 archive 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 archives.

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 Face Detection and Recognition, Speech Recognition, OCR, Title Generation and Keyword Selection) is suitable and practical to extract secondary information from the different raw data and how to integrate these algorithm into the system to generate an XML index file for each videoconference. Consequently, we will design and implement a GUI of PVCAIS which allows users to search in 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 and OCR. By this way, more valuable information can be extracted from the same source for 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 release the 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 real media communication in addition to video or audio. They may include text chat, whiteboard, file transfer, and shared applications. Therefore, the word "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 makes and models of telephones can communicate using the telephone. That means by both applying H.323 standard, videoconferencing programs developed for different OS or by different vendors can still communicate.

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.

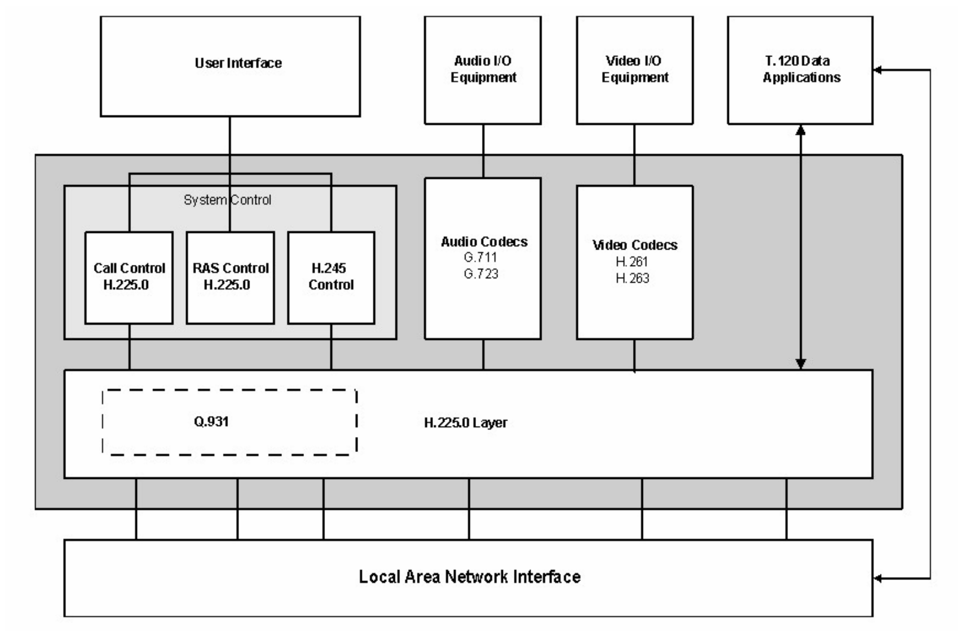


Figure 2.1 Architecture of H.323 Standard

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 XML index file for each videoconference is generated for later videoconference accessing, which includes searching, browsing and editing on past archives. The idea of PVCAIS is illustrated in Figure 7.1.

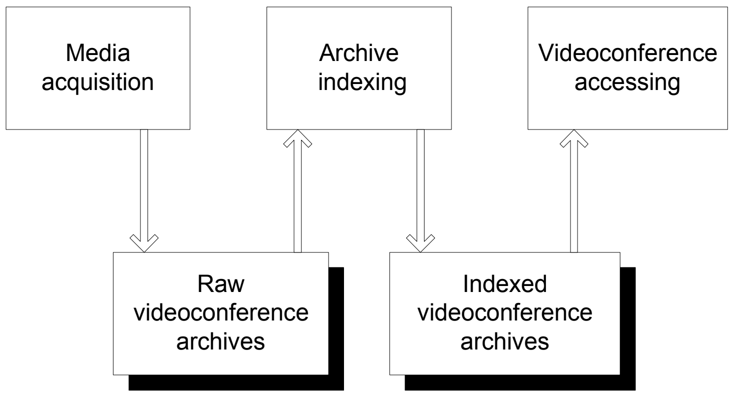

Media acquisition
Archive indexing
Videoconference accessing
Raw videoconference archives
Indexed videoconference archives

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.

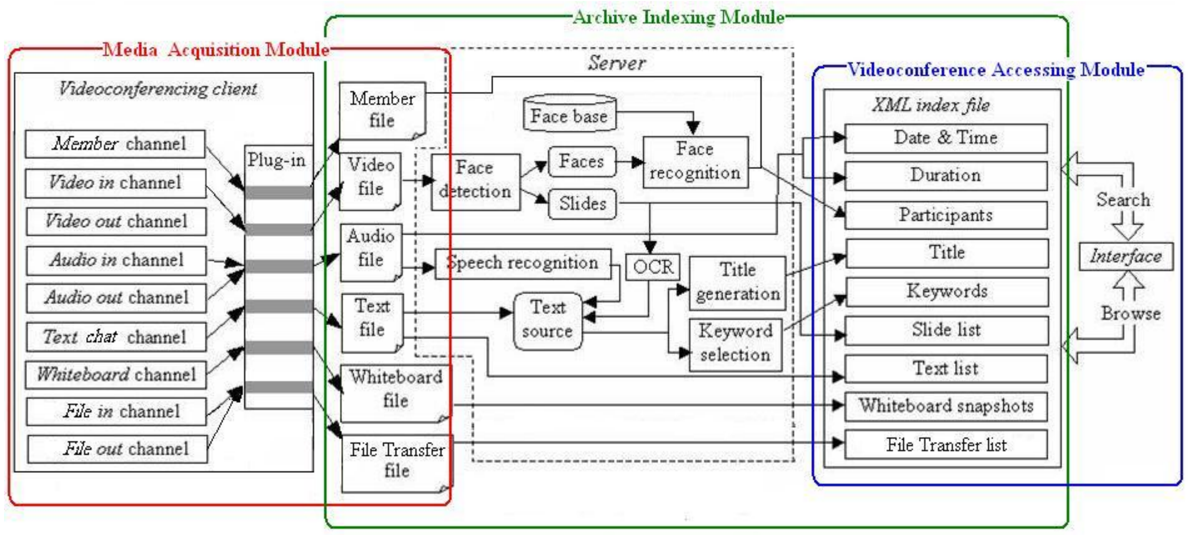


Figure 3.2 An overview of PVCAIS

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 a XML 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.

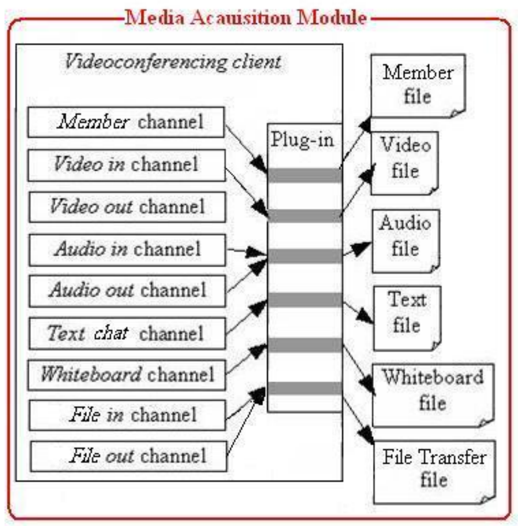


Figure 3.3 Structure of Media Acquisition Module

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

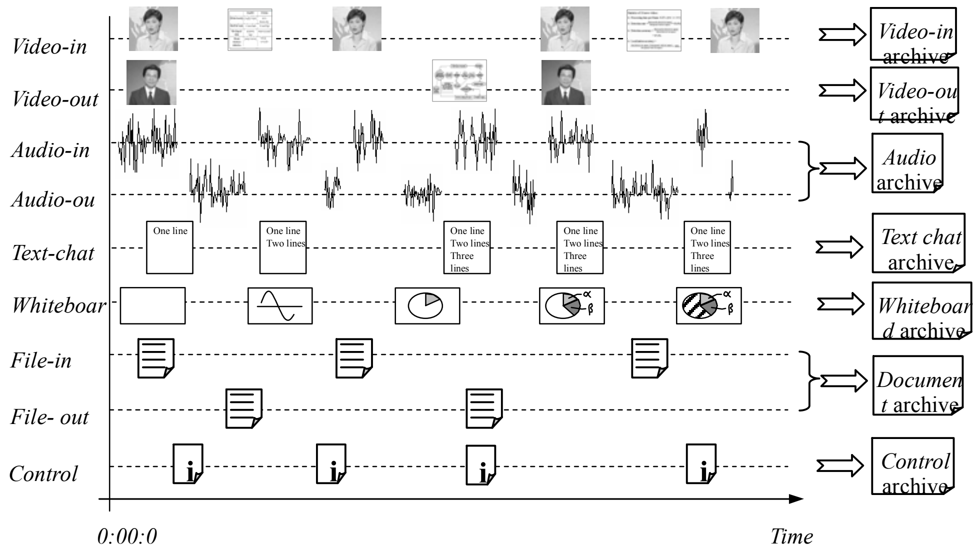


Figure 3.4 Illustration of the logical channels and media

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.

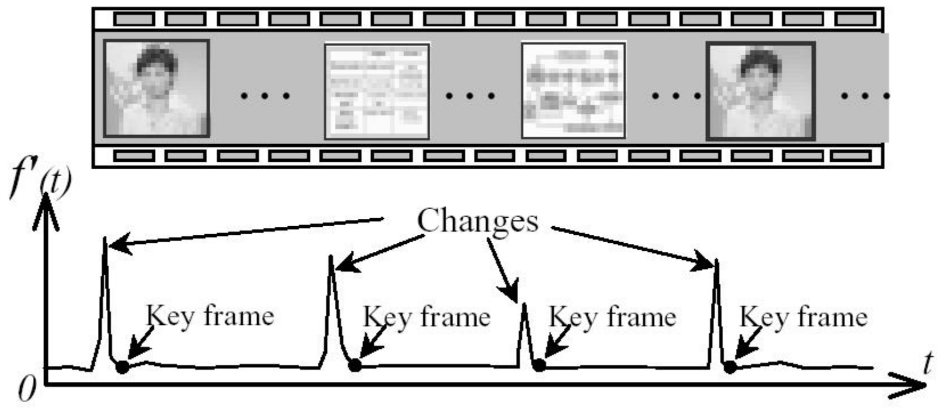


Figure 3.5 Key-frame selection algorithm on video channel

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.

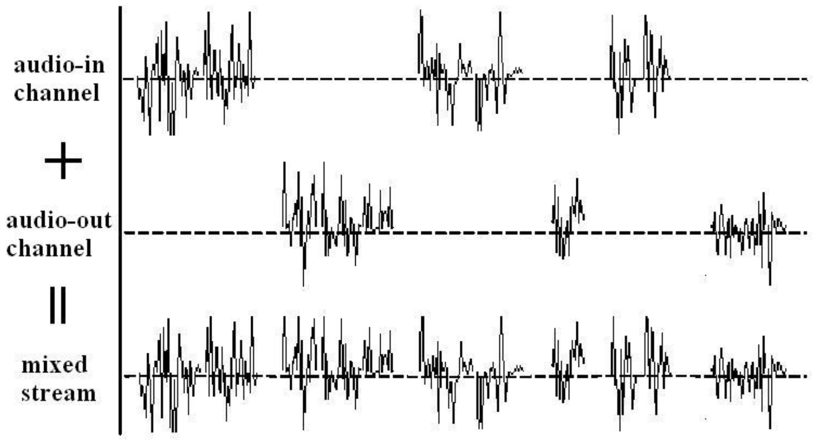


Figure 3.6 Mixing of audio-in and audio-out data

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.

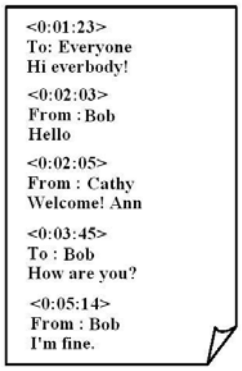

```
<0:01:23>
To: Everyone
Hi everbody!
<0:02:03>
From : Bob
Hello
<0:02:05>
From : Cathy
Welcome! Ann
<0:03:45>
To : Bob
How are you?
<0:05:14>
From : Bob
I'm fine.
```

Figure 3.7 A 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.

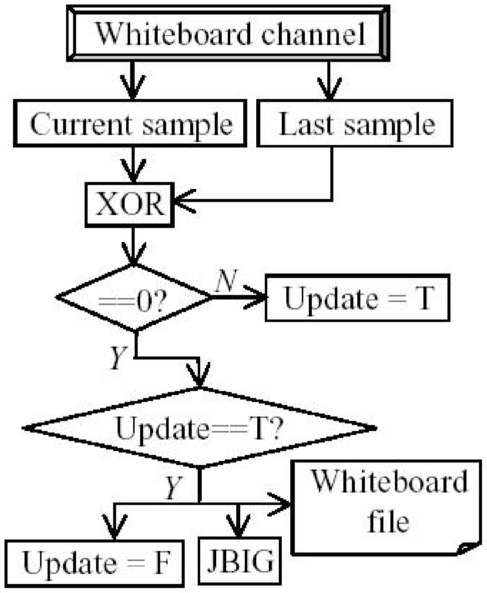
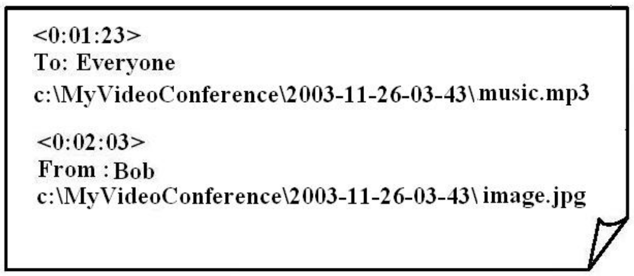


Figure 3.8 Whiteboard file generation algorithm by comparing samples

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.

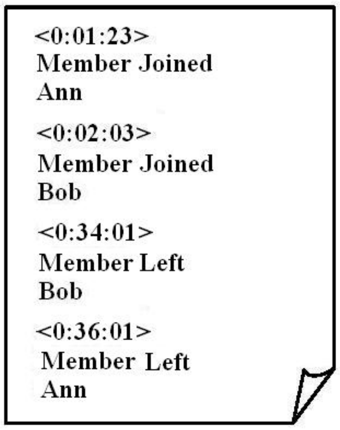


```
<0:01:23>
To: Everyone
c:\MyVideoConference\2003-11-26-03-43\music.mp3
<0:02:03>
From : Bob
c:\MyVideoConference\2003-11-26-03-43\image.jpg
```

Figure 3.9 A 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.



```
<0:01:23>
Member Joined
Ann
<0:02:03>
Member Joined
Bob
<0:34:01>
Member Left
Bob
<0:36:01>
Member Left
Ann
```

Figure 3.10 A 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 XML index file. The XML 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.

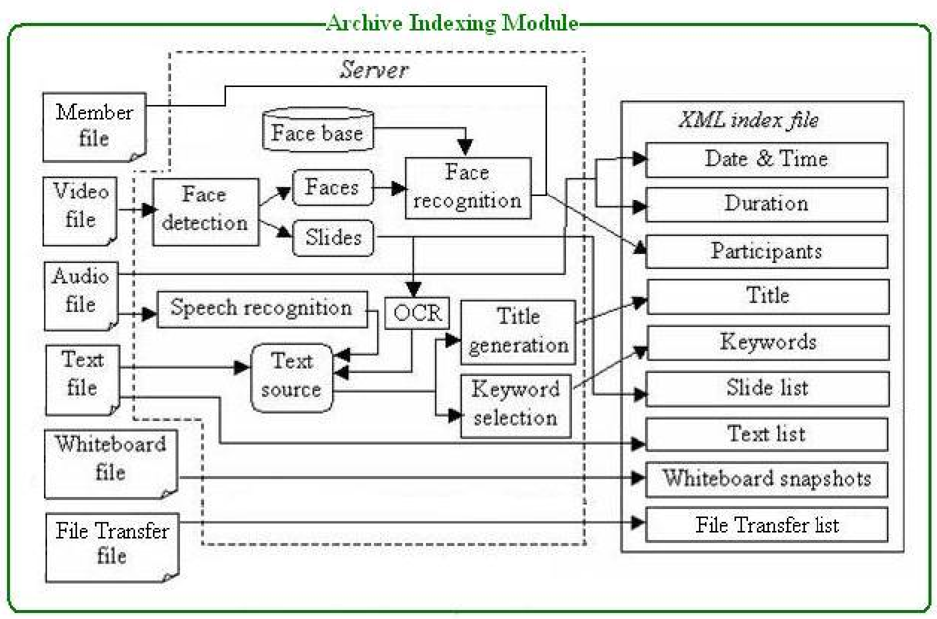


Figure 3.11 Structure of Archive Indexing Module

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.

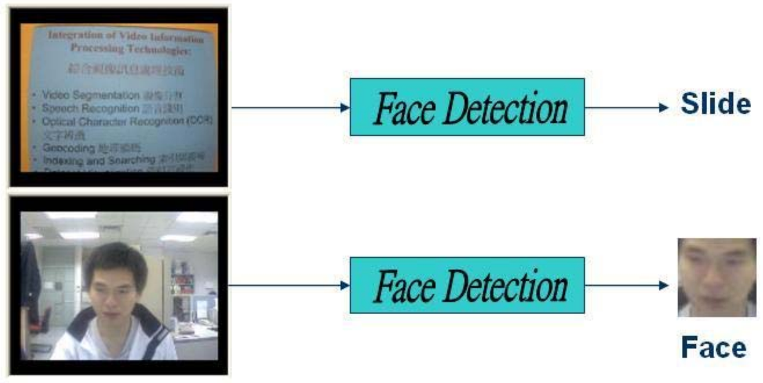


Figure 3.12 Face Detection

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.

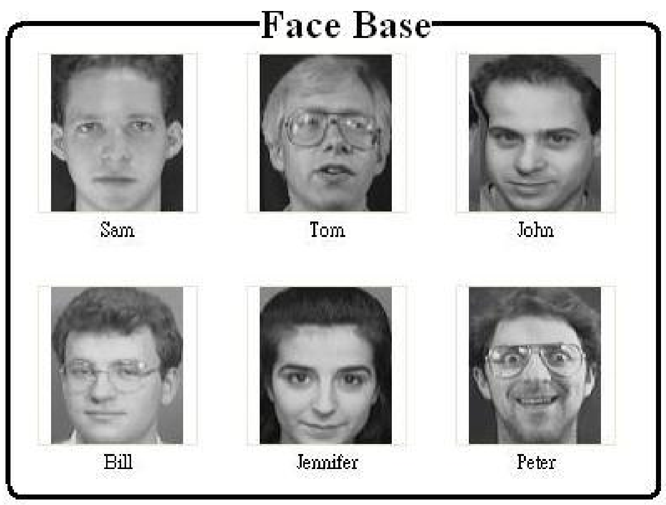


Figure 3.13 Sample face base for face recognition

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.

3.4.7 XML Index File Generation

After all multimedia-indexing functions are done, all indices and related files are integrated into an XML file. Figure 3.14 shows the file structure of PVCAIS, where the name of the directory of each videoconference is the timestamp of the meeting start time.

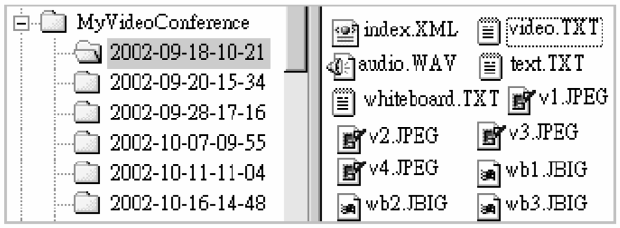


Figure 3.14 File Structure of the PVCAIS

The advantage of using XML(Extensible Markup Language) is that as not every participant have the database system, by the use of XML tag, it can defines a database-like structure which can provide efficient searching in the Videoconference Accessing Module.

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 content and modifying the information of the video archives.

It provides an interface for users to access on the XML index file. Figure 3.15 shows the structure of the Videoconference Accessing Module.

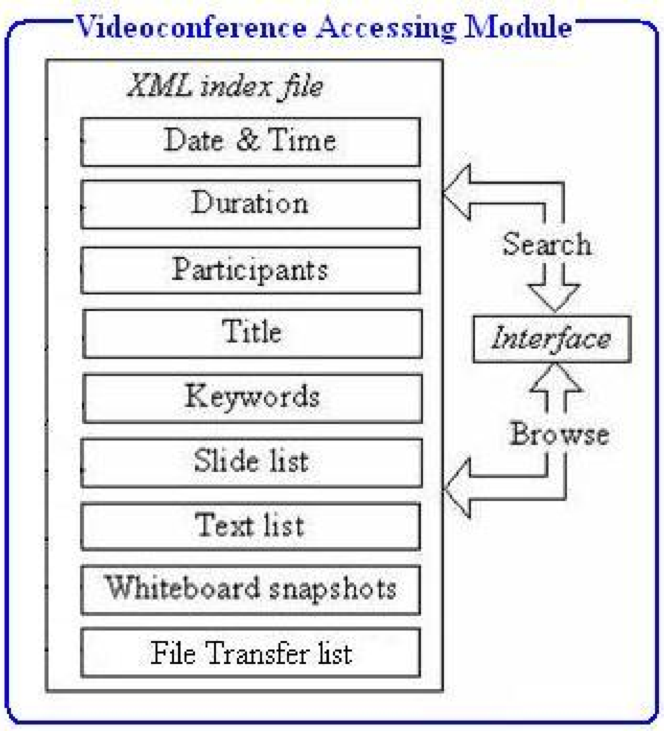


Figure 3.15 Structure of Videoconference Accessing Module

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 XML 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 video conference. Users can play the videoconference and the Videoconference Accessing Module will access the XML index file and the related files in the disk. Different type of archives, such as local video key-frames, remote video key-frames, sound, slides, 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

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. So in order to obtain the raw data of a conference, we need to build our own videoconference client first.

The whole system of our videoconference client is built on top of the NetMeeting 3 SDK, before looking into the implementation. Let's have a brief description of NetMeeting 3 and the NetMeeting 3 SDK.

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 this 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 fun 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 this. The greatest benefit is that 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 Chat

This feature is like an ordinary chat room on the Web, which enables user to send text to the 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. It likes a multilayered canvas which everyone may draw and write. 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 participants 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 with 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.

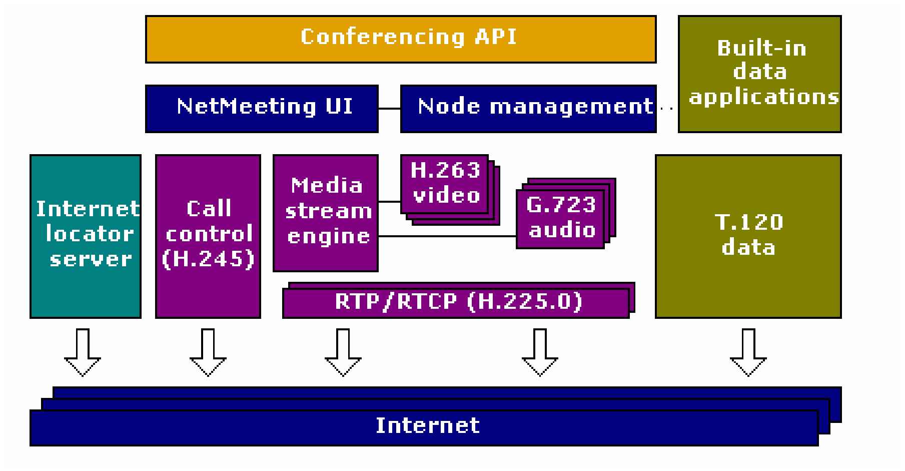


Figure 4.1 The NetMeeting Architecture

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. This includes 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.

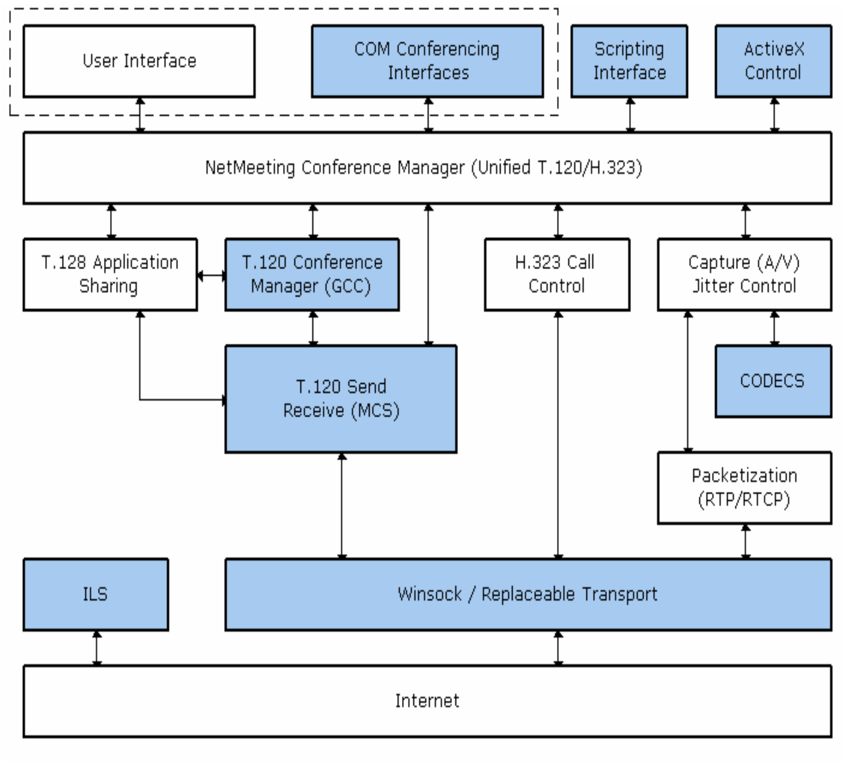

User Interface
COM Conferencing Interfaces
Scripting Interface
ActiveX Control
NetMeeting Conference Manager (Unified T.120/H.323)
T.128 Application Sharing
T.120 Conference Manager (GCC)
H.323 Call Control
Capture (A/V) Jitter Control
CODECS
T.120 Send Receive (MCS)
Packetization (RTP/RTCP)
ILS
Winsock / Replaceable Transport
Internet

Figure 4.2 The NetMeeting SDK Architecture

The COM Conference Interfaces are mainly used in our system.

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 about 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 all of the API of the NetMeeting 3 SDK are provide 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 us 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 Videoconference Client

Our videoconference client is a very simple, but it supports all the basic functions that a videoconference should have.

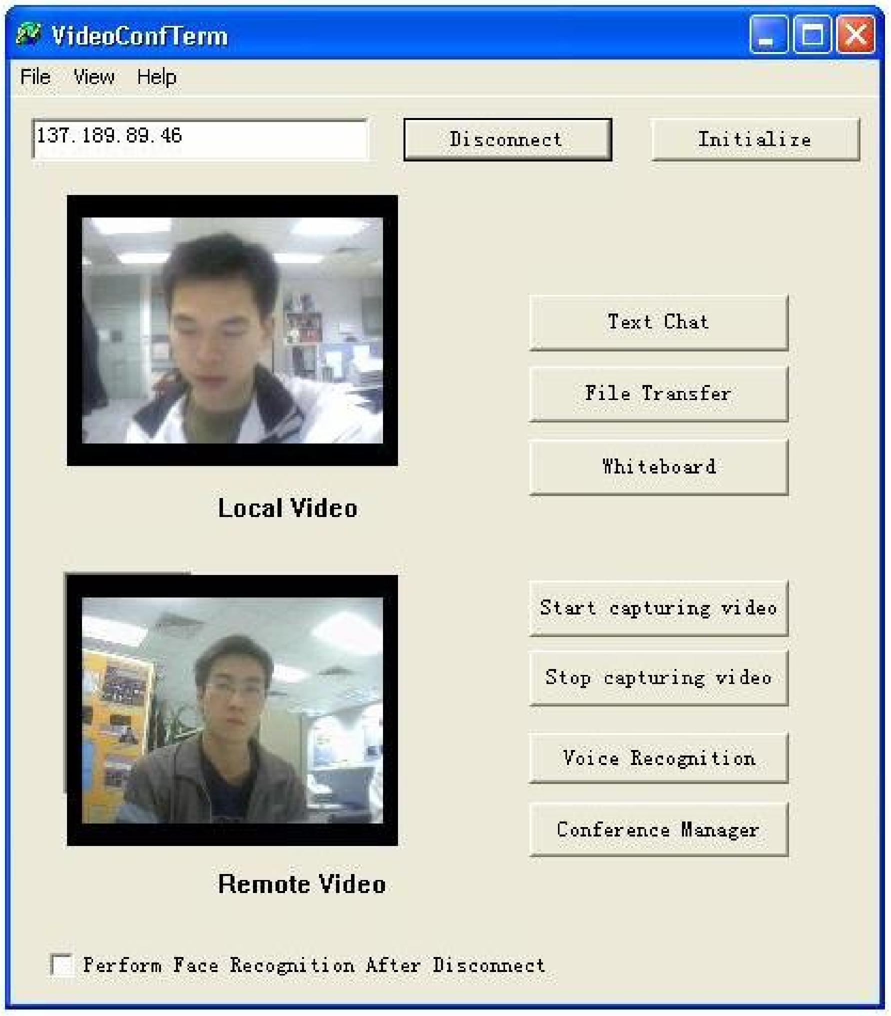
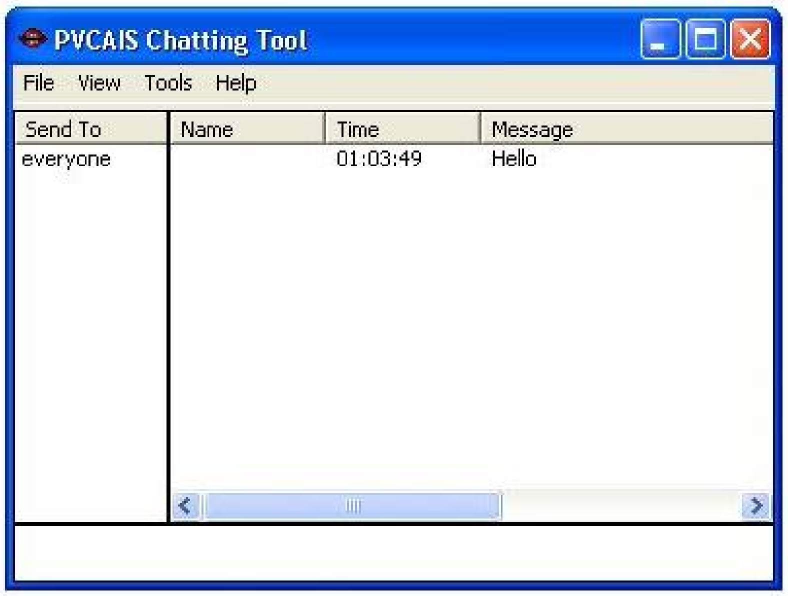


Figure 4.3 The main window of the videoconference client

There are nine buttons in the interface of our system, they are:
Connect: this button is used to create a conference. Before creating a conference, user needs to enter the IP address of his or her friend's computer into the box on the top left corner.

Initialize: after the conference is created, this button will be pressed to initialize the data collection functions of the system.

Text Chat: when this button is pressed during a conference, a chat box will be created which allows the members of the conference to chat using text messages.



Send To	Name	Time	Message
everyone		01:03:49	Hello

Figure 4.4 The chatting window

File Transfer: user can press this button to use the file transfer function of the system.

User needs to enter the name of the receiver or enter "Everyone" if he or she wants to send the file to all the members of the conference. Then user can choose the file he or she wants to send.

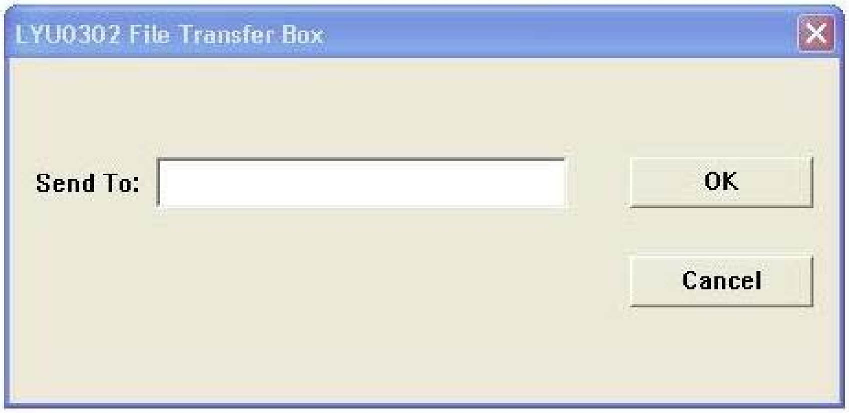


Figure 4.5 The file transfer window

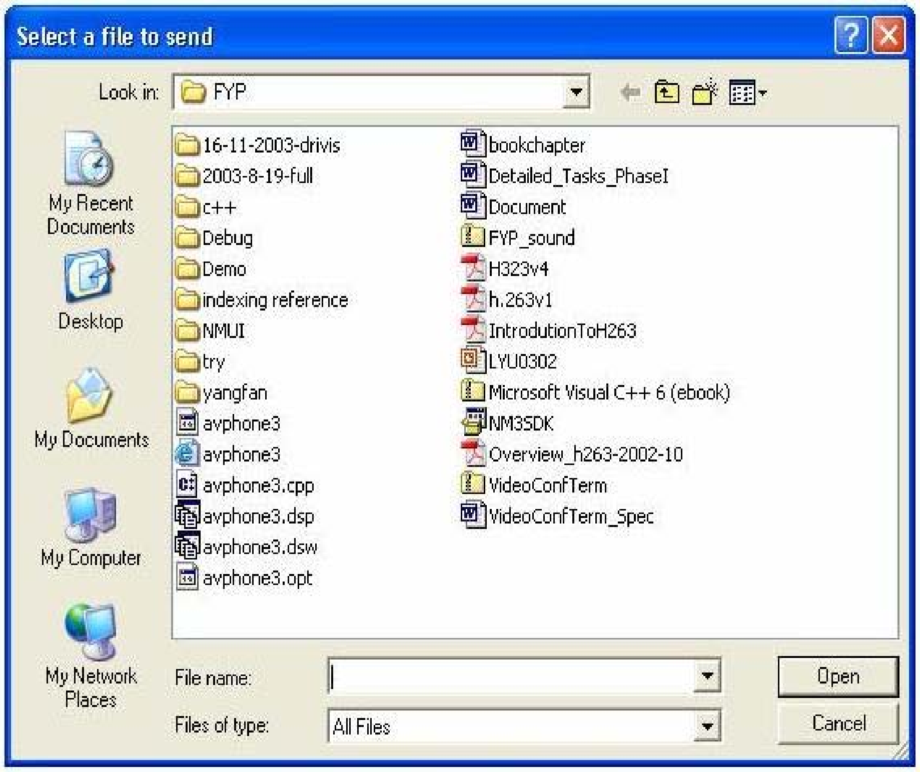
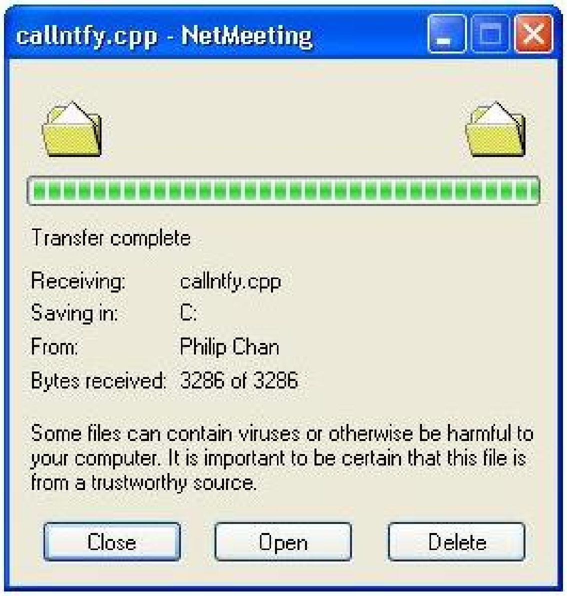


Figure 4.6 The file selection window

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.



Transfer complete

Receiving: callntfy.cpp
Saving in: C:
From: Philip Chan
Bytes received: 3286 of 3286

Some files can contain viruses or otherwise be harmful to your computer. It is important to be certain that this file is from a trustworthy source.

Figure 4.7 The file receiving window

Whiteboard: when this button is pressed, a whiteboard window will pop up to allow user to draw on it. Currently, we use the whiteboard component provided by NetMeeting SDK.

Start capturing video: when this button is pressed, the key-frames of both local and remote video will be collected.

Stop capturing video: when this button is pressed, the key-frames capturing functions will be stopped.

Voice Recognition: After the conference is finished, user can press this button to perform voice recognition on the recorded audio data of the conference. The result will be stored in a text file.

Conference Manager: When this button is pressed, the conference manager window will pop up. This interface will allow user to manage and search for the indexed conferences. We will describe more about this interface in later section.

There is a small box on the lower left corner, when this box is ticked, the system will perform face detection and recognition on the key-frames recorded from the remote video.

4.6 Media Acquisition Module

In this module, the major data that we want to collect include video, audio, member information of the conference, text chat, file transfer and the whiteboard. Each videoconference will have a directory named by the 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 for us to do indexing. Now let's have a deep description on how we obtain the video conference data from our videoconference client.

4.6.1 Member Information

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

4.6.2 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.

4.6.3 Audio

The NetMeeting 3 SDK does not provide any API 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.

4.6.4 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).

4.7 Archive Indexing Module

In this module, we will further process the raw data collected to produce indices for the conference. This module is the most difficult part of our system and is only partial done. 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.

4.7.1 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

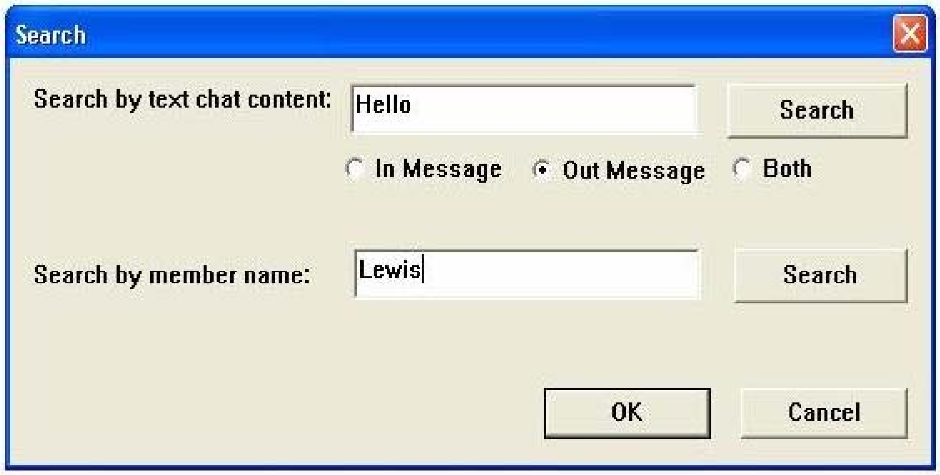
4.7.2 Video

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. Then face recognition result is displayed to users and also written to a file for later indexing and searching.

4.8 Videoconference Accessing Module

In this module, we have created a simple interface to manage, search and review all the stored conference. By pressing the "Conference Manager" button on our main interface, a searching window will pop up. We have implemented two searching functions. User can search for a conference by member name, or the text chat content.


Search

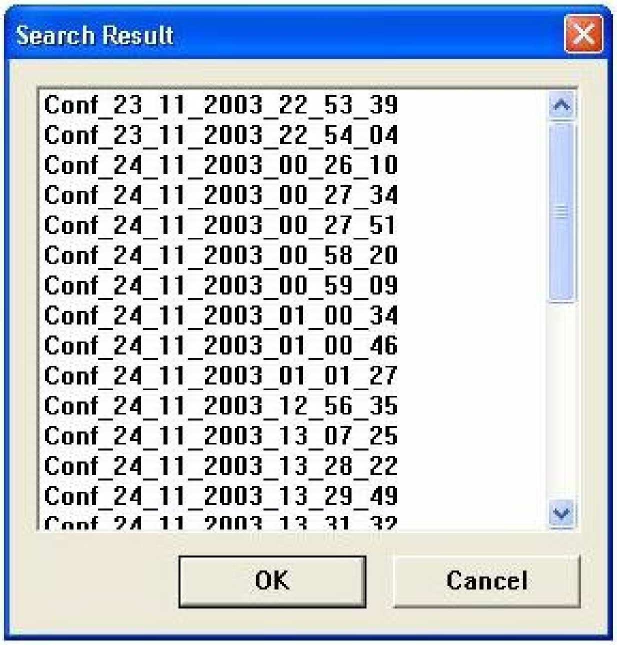
Search by text chat content: Hello

( ) In Message [✓ Out Message] ( ) Both

Search by member name: Lewis

Figure 4.8 The searching window

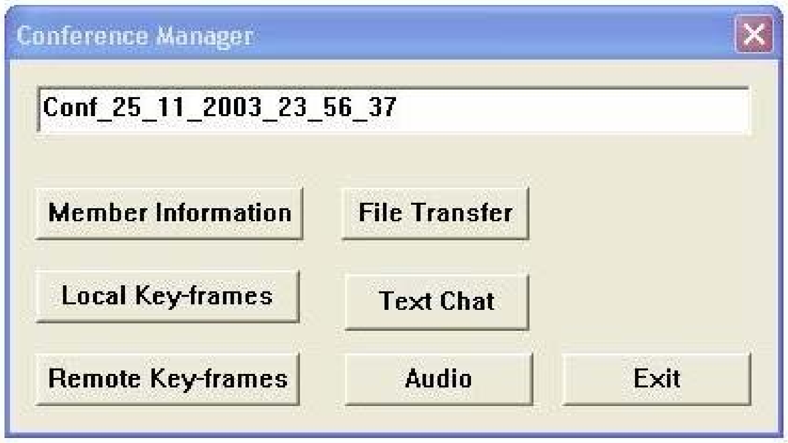
Then a list of matching conferences names will be displayed on another window, user can choose the one he or she want to review.



Conf_23_11_2003_22_53_39
Conf_23_11_2003_22_54_04
Conf_24_11_2003_00_26_10
Conf_24_11_2003_00_27_34
Conf_24_11_2003_00_27_51
Conf_24_11_2003_00_58_20
Conf_24_11_2003_00_59_09
Conf_24_11_2003_01_00_34
Conf_24_11_2003_01_00_46
Conf_24_11_2003_01_01_27
Conf_24_11_2003_12_56_35
Conf_24_11_2003_13_07_25
Conf_24_11_2003_13_28_22
Conf_24_11_2003_13_29_49
Conf_24_11_2003_13_31_32

Figure 4.9 The results displaying window

After choosing the desired conference, a new window will pop up let user review the content of the conference.

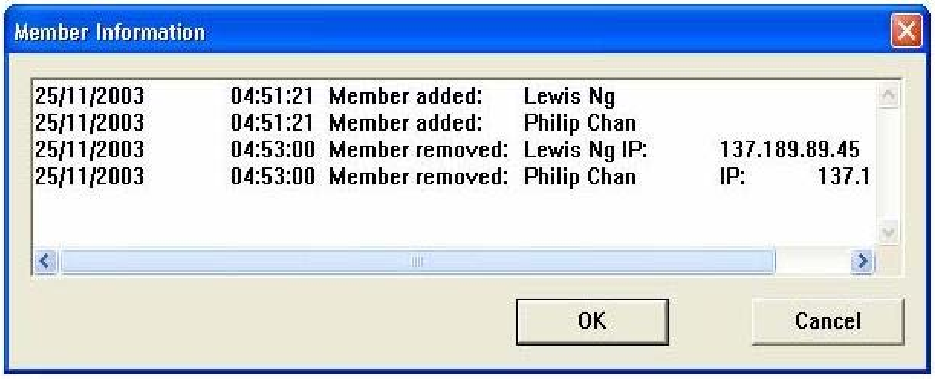


Conf_25_11_2003_23_56_37

Figure 4.10 The Conference Manager Window

The interface will show the name of the conference and there are seven buttons:

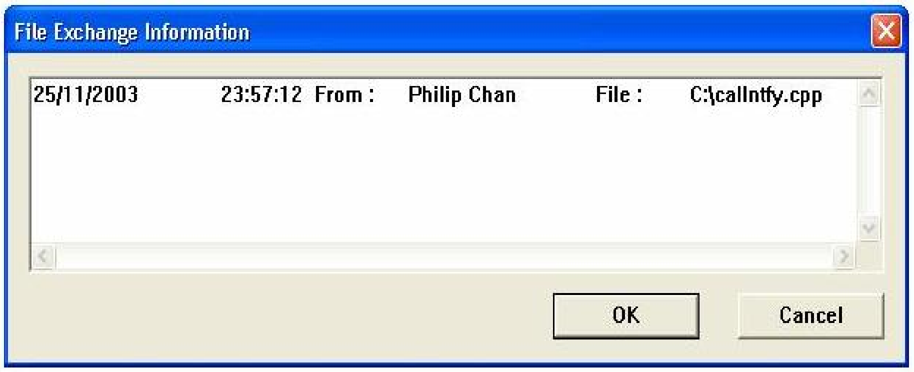
Member Information: by pressing this button, the member information of the conference will be displayed.



```
25/11/2003    04:51:21  Member added:    Lewis Ng
25/11/2003    04:51:21  Member added:    Philip Chan
25/11/2003    04:53:00  Member removed:  Lewis Ng IP:    137.189.89.45
25/11/2003    04:53:00  Member removed:  Philip Chan     IP:      137.1
```

Figure 4.11 The member information window

File Transfer: by pressing this button, the file transfer record will be displayed.



```
25/11/2003    23:57:12  From :   Philip Chan    File :   C:\callntfy.cpp
```

Figure 4.12 The file transfer information window

Remote Key-frames: by pressing this button, the key-frames of the remote video will be displayed.

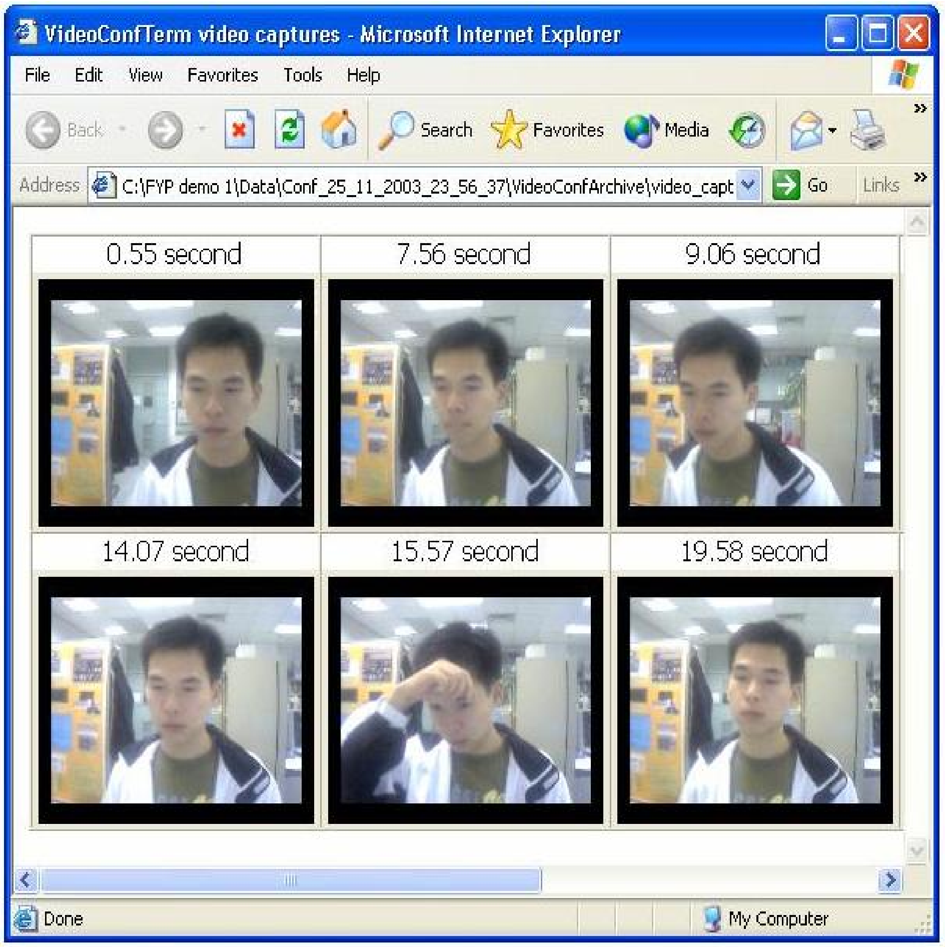
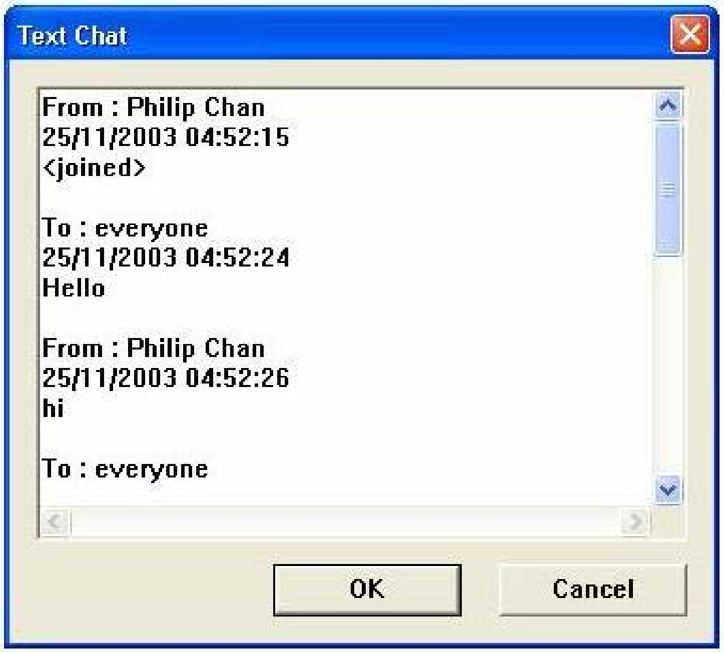

VideoConfTerm video captures - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\FYP demo 1\Data\Conf_25_11_2003_23_56_37\VideoConfArchive\video_capt
Go Links
0.55 second
7.56 second
9.06 second
14.07 second
15.57 second
19.58 second
Done
My Computer

Figure 4.13 The key-frames of remote video

Local Key-frames: by pressing this button, the key-frames of the local video will be displayed.

Text Chat: by pressing this button, all the text chat information will be displayed.



From : Philip Chan
25/11/2003 04:52:15
<joined>

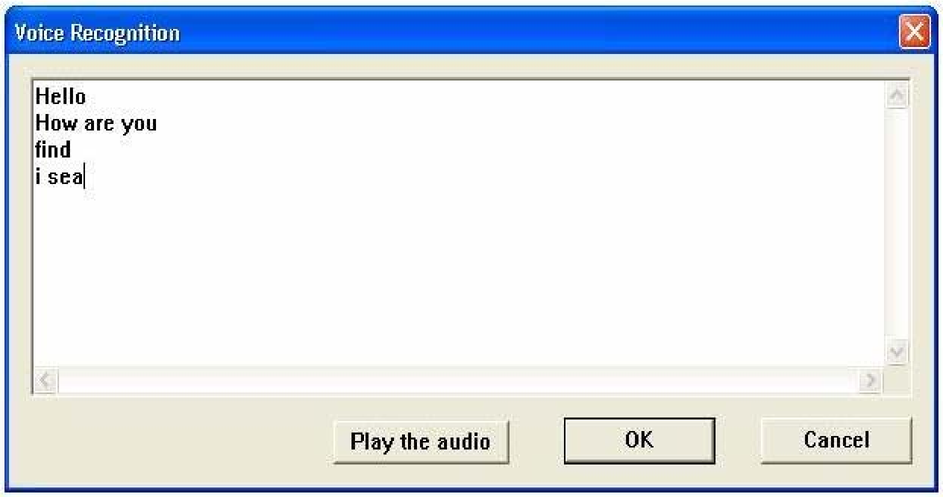
To : everyone
25/11/2003 04:52:24
Hello

From : Philip Chan
25/11/2003 04:52:26
hi

To : everyone

Figure 4.14 The text chat information

Audio: by pressing this button, the text generated by voice recognition will be displayed, user can also play back the audio.



Hello
How are you
find
i sea

Figure 4.15 The text generated by the voice recognition

Exit: pressing this button will leave this module.

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 will study the API more deeply to find out if there is way to improve the performance. We will also try IBM ViaVoice to see if it can give better result or not.

Problem 8

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

This problem cannot be solved yet. We will check our code to find out the problem.

Problem 9

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

This problem cannot be solved yet. It is suspected this problem will only occur in Window XP, we will try our system in Windows 2000 to see if the same problem will occur or not.

6. Project Progress

Month	Tasks Completed
MAY 2003	Get familiar with the programming environment of Microsoft Visual C++
	Study the background information of NetMeeting / 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
	Improve stability and reliability of audio recording module and key-frame selection module
	Study XML and different multimedia indexing algorithm
JAN 2004	Generate an XML index file for each Videoconference and implement the searching system
	Try to improve accuracy of speech recognition and face recognition and develop interface for face recognition training
FEB 2004	Add more algorithms to implement the multimedia-indexing functions such as OCR, automatic title generation and keyword selection
MAR 2004	Rebuild the GUI of the system for searching indexed videoconference archives and synchronized playback of a selected conference
APR 2004	Finalizing the source code 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 section, it is going to state the contribution of work of my part and the knowledge gained through this project.

For our final year project, it can be mainly divided into three parts, there are the Media Acquisition Module, the Archive Indexing Module and the Videoconference Accessing Module. These three modules are closely related. Without the data extracted by the Media Acquisition Module, the Archive Indexing Module can do nothing, and without the indices generated by the Archive Indexing Module, the Videoconference Accessing Module cannot do any searching and have nothing to show. Therefore, we need to finish these modules one by one.

In this semester, most of my work is on the Media Acquisition Module, the Archive Indexing Module. The detail will be stated in the following part.

7.2 Preparation Work

The first problem we met was that we could not obtain any source code of existing videoconference software. In order to collect the raw data to do indexing, we have to build our own videoconference client. After deciding to use the NetMeeting 3.0 SDK, the first thing we have to do is to get familiar with C++, especially Visual C++, because the API only supports C++ and it is provide in the form of COM interfaces and objects.

We started to work on our final year project after we finished all the examinations and projects of last semester. I have spent one month to get familiar with Visual C++ and Windows programming. I read books and studied some online tutorials. I have also written some simple Windows programs. After that I understand how to write Windows programs. I have also studied the Microsoft Foundation Classes (MFC). Now I know how to use the functions and classes provided by MFC to build applications more quickly.

After getting familiar with Visual C++ and Windows programming, I started to study the Component Object Model (COM). COM is a totally new programming concept to me. I knew how to write in object-oriented programming language. I understood the messages passing system between objects written in the same language, but I did not understand how objects written in different languages could interact with each other, especially the objects could be in different processes or even in different computers. It took me quite a lot of time to understand the operation of COM. After reading some reference books about COM, I have got a brief idea how it works and was able to program with it. Then I started to study the NetMeeting SDK.

Since we have to use face detection and recognition in our systems, we have spent one month to study these two technologies. We studied together because both of us were not familiar with them. By discussing with each other, we could learn more efficiently and faster.

7.3 Media Acquisition Module

After studying all the background materials, we started to build the videoconference client. The client was converted from a sample of the NetMeeting SDK. After we have got a simple videoconference client, we started to add in the data collecting functions. First, I implemented the member information collecting function. Then I implemented the file transfer record collecting function. However, the function could only record the information of incoming files. After several testing and tracing, I found that the standard component for file transfer would not return any message to the system. Therefore we could not collect data for outgoing files. Because of this, I implemented our own file transfer functions so that information of outgoing files can also be recorded. I have also studied the sample video capturing code given by Jiqiang. After understanding the algorithm, I applied it to the remote video window to collect the key-frames. For audio data, I first studied the sound recording code provided by Yangfan. Then I applied the algorithm into our system to collect the audio of the conference.

7.4 Archive Indexing Module

After collecting the audio, I studied the voice recognition engine, Microsoft SAPI, and implemented a function into our system to perform the voice recognition of the audio collected. The integration of face detection and recognition was done by my partner, Philip. Since audio and video data are difficult to handle, we separate the jobs, one handles the video and the other handles the audio. I have also implemented member searching function in this module, while Philip has implemented the text chat content searching function.

7.5 Videoconferencing Accessing Module

In this module, I was responsible for creating all the graphical user interface, these include the searching interface, the search result display interface and the conference content review interface (the Conference Manager).

7.6 Conclusions

In this semester, I have got familiar with Visual C++ and know how to use MFC to build Windows Program. I have learnt the concepts of COM. I am now able to build application using COM. I have finished the implementations for data extraction. A simple searching function and a simple user interface for reviewing conference content have also been implemented. I have studied the techniques of voice recognition, and face detection and recognition.

8. Conclusions

PVCAIS 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 process and analyzes the raw data to produce indices for the conference. The Videoconference Accessing Module provides an interface for user to mange the indexed conferences. User can search for a conference and review it by playing back all the stored content.

In this semester, we have learnt Visual C++ and Windows programming. We have got familiar with COM and are able to build application with it. We have studied the NetMeeting SDK and have built a videoconference client which supports all the basic functions, including: video, audio, chat, file transfer and whiteboard. All the conference raw data except whiteboard can be collected. We have studied the technique of face detection and recognition, and speech recognition. These two technologies have been integrated into our videoconference client. We have created a user interface that allows user to search for a conference by different criteria. It also allows user to review all the content of the conference.

9. Future Work

9.1 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 videoconferencing client, so we will design our own whiteboard data transfer structure and standard which can be retrieved and indexed for further searching

9.2 Improve Accuracy of Sound Recognition

The sound recognition accuracy of our current implementation is not good because of the background noise and microphone quality. Also, as the audio data contains voice for both parties, it is difficult to do training. We will try to solve this problem.

9.3 Improve Performance of Face Detection

The face detection engine of our current implementation takes quite a long time to perform face detection on a single picture. We will try to implement other different face detection algorithm to improve the performance.

9.4 OCR, Keyword Selection and Title Generation

We will study the algorithm for OCR, keyword selection and title generation and integrate them into our system.

9.5 XML index file

We will study the structure of XML and use XML index file to integrate all the extracted archive files.

9.6 Improve User Interface

The current user interface of our implementation only provides basic searching functions on past videoconference archives. We will improve the GUI of our implementation so that it can support synchronized playback of videoconference content and can provides more searching and managing functions to past conference record.

10. 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 Song JiQiang, Edward Yau who gives suggestions and ideas in our project, Yang Fan, Sunny Tang and Harvest Jang who have offered help for the implementation of our project.

10. References

[1] 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.

[2] S.G. Deshpande and J.-N. Hwang, "A real-time interactive virtual classroom multimedia distance learning system," *IEEE Trans. on Multimedia*, 3(4): 432-444, 2001.

[3] 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.

[4] vic, vat, wb: Video, audio, whiteboard conferencing tools. *Available online*: http://www-nrg.ee.lbl.gov/.

[5] rat, nte: Audio, text tools. *Available online*: http://wwwmice.cs.ucl.ac.uk/multimedia/software/.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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

[38] Ito, H., Sato, M., & Fukumura, T. (2000). Annotation and indexing in the video management system VOM. in *Proc. the 2000 IEEE International Conference on Systems, Man, and Cybernetics*, Vol.2, 834-839.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.