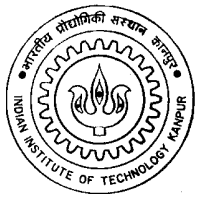
An Approach towards Real-time Distance Education using Web Server Streaming

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Ashwini Damle



to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

August, 2001

Certificate

This is to certify that the work contained in the thesis entitled "*An Approach towards Real-time Distance Education using Web Server Streaming*", by *Ashwini Damle*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

August, 2001

(Dr. Rajat Moona)
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

Abstract

In the context of distance education, a remote student may expect quick playback start-up and uninterrupted playback of a classroom lecture on his desktop. With the streaming media, the quick playback start-up can be achieved where the player starts playback without waiting for the whole media file to be downloaded to local storage. The issue of the uninterrupted playback can be resolved by low bit-rate audio-video coding. However, the simplistic approach to the low bit-rate video coding degrades the quality of the lecture video, especially of the slides being explained by the instructor.

We focus on the issue of the uninterrupted playback while maintaining the excellent quality of the slides, on the student's desktop. In our approach, we see two semantically different visual entities in the lecture video, i.e., the presentation slides and the instructor's body overlapping the slides. Normally the instructor occupies small region of the frame, so the video of the instructor's body movements is a natural choice for low bit-rate coding. We use chroma-key based coding of the instructor's video where background surface is represented by a chroma-key say green color. We then interleave the low bit-rate voice of the instructor with the coded video stream. The coded audio-video of the instructor can further be converted to the streaming media. We assume the presence of the Microsoft Powerpoint slides along with the timing information, recorded during the lecture. The presentation file and audio-visuals of the instructor are made available through a web server.

For the playback of the lecture contents, a player, on the student's desktop, first downloads the presentation file from the web server. The download time is less because of the small size of the presentation file. The player extracts the slides of the original quality. The low bit-rate coded (i.e. highly compressed) audio-visuals of the instructor are streamed through the player and are simultaneously decoded. The player makes the chroma-key background, in the video frames, transparent and superimposes the instructor's body over the slide. The player displays the visual output and simultaneously plays the instructor's voice for the whole lecture. The

overall synchronization between the instructor's audio-video and the running slides is achieved by maintaining the recorded timing for each slide in the presentation file.

Our approach surpasses the real learning experienced in a classroom, because the quality of the slides is excellent on the student's desktop as compared with the quality appeared from a certain distance in a classroom.

Acknowledgements

I have absolutely no role, but to accept His blessings!

I express my sincere gratitude towards my guide, Dr. Rajat Moona, for allowing me to work on this particular topic and offering me the freedom and the necessary resources to try out my ideas and learn on my own. I must also express that his insightful guidelines and timely suggestions have immensely helped me to proceed with the proper pace. I enjoyed working under him.

I am grateful to the staff of the television center as well as the Robotics lab at I.I.T. Kanpur for providing the facilities required for my work. I am thankful to all of the teaching, non-teaching elders and the young learners, of the CSE department, for providing me with such an exciting and encouraging work place.

I express my deep respect to all of the caretakers of the institute for maintaining the peace and security of this beautiful and industrious place.

I have no words to express my gratitude towards Dr. T.V. Prabhakar and Mrs. Archana Prabhakar for their words of encouragement, whenever I needed it.

I very much appreciate the useful criticism and thought provoking discussion that I enjoyed in the company of Souvik, my friend who kept me aware of various branches of knowledge, apart from the thesis work. I also respect his active support and encouragement that I keenly felt throughout my journey here.

I express my warm feelings towards my friends Padmaja, Tanuja, Swarna, Malabika, Aamrapali, Trishla and a lot more for creating a dream-home for me in this campus. I acknowledge the caring and support that I received from my guardians, Bapat family and Pandit family, throughout my stay here.

Finally, I note down the deepest feelings of my heart for my parents and my Master, for being with me unconditionally.

Abbreviations used in this document

- **ACM:** Audio Compression Manager
- **AD:** Audio Decoder
- **AMOS:** Active MPEG-4 Object Segmentation
- **ANSI:** American National Standards Institute
- **API:** Application Programming Interface
- **AVI:** Audio Video Interleave
- **CBDM:** Chroma-key Based Display Mixer
- **codec:** COder DECoder
- **COM:** Component Object Model
- **DCI:** Display Control Interface
- **DIB:** Device Independent Bitmap
- **fps:** frames per second
- **GIF:** Graphics Interchange Format
- **IEC:** International Electrotechnical Commission
- **IP:** Internet Protocol
- **ISDN:** Integrated Services Digital Network

- **ISO:** International Standards Organization
- **JDK:** Java Development Kit
- **JIT:** Just In Time
- **KB:** Kilo Bytes
- **kbps:** kilo bits per second
- **KHz:** Kilo Hertz
- **LBAE:** Low Bit-rate Audio Encoder
- **LBVE:** Low Bit-rate Video Encoder
- **MB:** Mega Bytes
- **NTSC:** National Television System Committee (a video standard used in United States, Japan)
- **PAL:** Phase Alternating Line (a video standard used in United Kingdom)
- **PCM:** Pulse Code Modulation
- **RFC:** Request For Comments
- **SDK:** Software Development Kit
- **SDO:** Source Data Object
- **SE:** Slides Extractor
- **TCP:** Transmission Control Protocol
- **UDP:** User Datagram Protocol
- **URL:** Uniform Resource Locator
- **VD:** Video Decoder

Terminology used in this document

- **ASF:** Microsoft's Advanced Streaming Format [6]. This audio-video format supports low bit-rates such as 28.8 kbps, 56 kbps, 64 kbps and 128 kbps and allows "progressive playback" using web server streaming. The AVI, MPEG-1 and .wav files can be encoded to ASF format using Windows Media Encoder version 6.x. ASF format is based on MPEG-4 compression technology. This is Microsoft's proprietary format and not too many ASF editing/conversion tools are freely available.

- **Audio interleaving:** Very few applications deal with digital video only. Normally digital video is accompanied by digital audio. Audio interleaving suggests the way to integrate a video stream and an audio stream in order to achieve synchronization between the two.

- **Bit-rate:** The rate at which the encoded bitstream is delivered from the storage medium to the input of a decoder.

- **DCI:** The Diplay Control Interface (DCI) for the Microsoft Windows operating system is a driver-level software interface which provides access to display devices. DCI provides access to the display device-independent benefits such as improved video playback quality. DCI is the predecessor of the Microsoft DirectDraw.

- **GOP (Group Of Pictures):** In a coded video stream, a sequence of encoded pictures from one intra-coded picture to a picture just before next intra-coded picture is called a GOP.

- **HTTP:** The HyperText Transfer Protocol [16] is an application-level protocol designed for distribution of hypertext and multimedia documents over the World Wide Web.

- **IP multicast:** IP multicast [11] allows very efficient delivery of streaming content to large number of users. However, it only works on networks with multicast-enabled routers.

- **Jitter:** The variable network latency, normally caused by the queuing of packets at each router between source and destination, is called jitter.

- **JPEG:** JPEG [20] is a standardized image compression mechanism. JPEG stands for Joint Photographic Experts Group, the original name of the committee that wrote the standard. JPEG is designed for compression of either full-color or gray-scale images of natural, real-world scenes. JPEG is "lossy", meaning that the uncompressed image isn't quite the same as the original one. JPEG is designed to exploit known limitations of the human eye, notably the fact that small color changes are perceived less accurately than small changes in brightness. The encoder can trade off file size against output image quality. The decoder can trade off decoding speed against image quality.

- **Low bit-rate video coding:** The video compression technique which outputs a coded video stream of not greater than 64 kbps bit-rate.

- **Microsoft DirectX:** DirectX [13] is an API supported by the Microsoft Windows that enhances the multimedia capabilities of the computer. DirectX includes accelerated video card and sound card drivers. These drivers provide better playback for different types of multimedia, such as full-color graphics, video, 3-D animation, immersive music, and theater sound. DirectX enables these advanced functions without requiring the programmer to identify the hardware components in the computer and ensures that most software runs on most hardware systems. DirectX is comprised of APIs that are grouped into two classes. One is the DirectX Foundation layer, and other is the DirectX Media layer.

The DirectX Foundation layer automatically determines the hardware capabilities of the computer and then sets the program's parameters to match those capabilities. These APIs control low-level functions, including 2-D graphics acceleration and control of sound mixing and sound output. The low-level functions are supported by the components that make up the DirectX Foundation layer. Some of these components are **DirectDraw** and **DirectSound**.

The DirectX Media layer works with the DirectX Foundation layer to provide high-level services that support animation, media streaming and interactivity. Like the DirectX Foundation layer, the DirectX Media layer is comprised of several integrated components. One of the components is **DirectShow**.

- **Microsoft DirectDraw:** The Microsoft DirectDraw API [13] supports extremely fast and direct access to the accelerated hardware capabilities of a computer's video adapter. It supports standard methods of displaying graphics on all video adapters, and faster, more direct access when using accelerated drivers. DirectDraw provides a device-independent way for programs, such as digital video codecs, to gain access to the features of specific display devices without requiring any additional information from the user about the device's capabilities.

- **Microsoft DirectSound:** The Microsoft DirectSound API [13] provides a link between programs and an audio adapter's sound mixing and playback capabilities. DirectSound provides multimedia applications with low-latency mixing, hardware acceleration, and direct access to the sound device. It provides these features while maintaining compatibility with existing device drivers.

- **Microsoft ActiveMovie/DirectShow SDK: [12]**

  The Microsoft ActiveMovie SDK lets developers access ActiveMovie services, which provide playback of multimedia streams from local files or Internet servers. Specifically, this allows playback of video and audio content, compressed in various formats including MPEG, QuickTime and AVI.

The heart of the ActiveMovie services is a modular system of pluggable components called filters arranged in a configuration called a filter graph. Normally one filter handles a basic function in dealing with media files such as reading from a media file, writing to a media file, transforming (e.g. compressing, decompressing etc.) a media stream, rendering media on a surface etc. A component called the filter graph manager oversees the connection of these filters and controls the data flow of the stream. Figure 1 depicts a filter graph that can render an Audio-Video Interleaved (AVI) file.

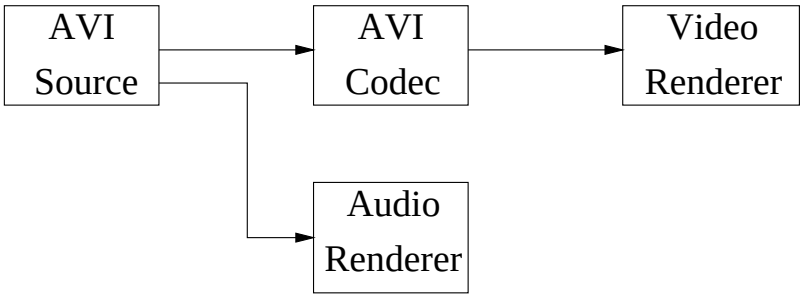


Figure 1: Generic filter graph used to render an AVI file

Applications control the activities of the filter graph by communicating with the filter graph manager. This can be done indirectly using the ActiveMovie ActiveX control or by calling COM interface methods directly. The SDK allows developers to create their own filters using the ActiveMovie class library. The base classes in the library implement the required COM interfaces on the filters and provide the basic filter framework.

The ActiveMovie version 2.0 has been renamed as DirectShow.

- **MMX technology:** MMX stands for MultiMedia eXtension. The high performance multimedia processing instructions are provided as extension to Pentium's general purpose instruction set.

- **Network congestion:** When the load offered to any network is more than it can handle (when there are too many packets present in a network), the performance degrades. This situation is called network congestion.

- **Real-time distance education:** If the distance education experience matches the time-bound learning experience of a classroom lecture, we call it real-time distance education.

- **Real-time transmission of data:** If the actual rate of data transmission matches the rate of data consumption, then it is called real-time transmission of data.

- **RTSP:** The Real Time Streaming Protocol or RTSP [3], is an application level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as video and audio from live data and stored clips. The protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP (RFC 1889).

- **RTT:** For each connection, TCP maintains a variable, RTT, that is the best current estimate of the round-trip-time to the destination in question. The round-trip-time means time required for a segment to reach its destination plus the time required for its acknowledgement to reach the source.

- **Scalability option in video compression:** This is one of the options specified while compressing a video using some codecs. When scalability is enabled, video is compressed in a manner that allows playback visual quality to vary automatically, based on the available processing power of the playback system. If processing power is inadequate, the codec degrades visual quality incrementally rather than dropping frames. This avoids jerky video playback.

- **Streaming media:** Streaming media file begins playing almost immediately, while the data is being sent, without having to wait for the whole file to be downloaded to the local storage.

- **Web server streaming:** A standard web server is used (instead of a streaming server) to deliver the audio and video data, available in a "progressive

playback" format, to the client. The data received is streamed through the application. Web server streaming uses HTTP for communication between the server and the client. HTTP typically uses TCP for transport protocol.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction to the problem

With the advent of streaming media technology, "Distance Education" mechanisms have got a new dimension. Today's student expects classroom lecture contents on his desktop. However current technologies don't cope well with Internet congestion. In addition, slow player start-up forces user to wait for some time initially. It is very difficult to quickly start uninterrupted playback of a typical one hour classroom lecture video on a remote student's desktop even with the streaming media technology.

The challenge before content-developers seems to be real-time transmission and quick playback start-up on client machine while maintaining reasonable quality of the lecture contents. The present work explores 'bit-rate' as the dimension of the computer-based distance education mechanism and strives to achieve the low bit-rate (hence possibly real-time) distance education using web server streaming. In other words, we come up with a scheme to get the compressed lecture video stream which does not exceed 64 kbps bit-rate.

1.2 Motivation

- *The noble cause of education*

We typically learn about a new concept by listening to an expert's talk explaining the concept with the aid of its model. Using the technology today, it is possible to provide models, lecture material etc. in some form. But the role of the instructor seems to be essential in a complete learning experience provided the instructor is clearly seen and heard.

- *Time-tested distance education efforts*

Today's technology provides with various means to preserve the lectures in the archives and reuse them whenever needed. Video cassette and television broadcasting have been time-tested means of distance learning.

- *Future distance education using streaming media*

The Multimedia technology makes the information available in the form of graphics, images, audio, video etc. Various compression-decompression techniques solve the problem of huge storage required for audio-visual information. Internet-based streaming media technology has the potential to dominate media distribution among the currently used technologies such as broadcast TV distribution, cable TV distribution and physical video rental via cassettes and CDs.

- *Current technical issues in streaming media [2]*

Current streaming protocols don't cope well with Internet congestion at high rates. This results in annoying display quality with occasional interruptions.

The other major technical issue is the start-up time of the streaming media player. The buffering, on the player side, may result in slow start-up of the player. The reason for using a buffer is to absorb network "jitters" with time.

1.3 Our approach

In the context of distance education application, the remote student may expect the quick playback start-up and uninterrupted playback of a classroom lecture video on his desktop. With the streaming media, the first issue of the quick playback can be handled where the player can start playback without waiting for the whole media file to be downloaded to local storage. In order to resolve the issue of uninterrupted playback over today's narrow communication channels, the low bit-rate audio-video coding is a natural choice.

- *Salient features of a classroom lecture movie used in our approach*

1. The lecture is captured from a fixed point of view. Hence a fixed major portion of each video frame is occupied by the presentation slide.
2. The common presentation format is slide driven. In our approach, the slides are in the Microsoft Powerpoint format.
3. Usually the contents of the slide change slowly and at discrete time intervals.
4. During the actual recording of the lecture, the transition timing per slide can be noted and the same timing information can be saved along with the presentation.
5. The only moving object in the video is the instructor (typically the upper body of the person).
6. The instructor's body usually overlaps the slide and the region occupied by the instructor is small as compared to the size of the whole video frame.
7. Usually the only audio component is the voice of the instructor.

- *Limitations of simplistic approach to low bit-rate video coding*

The simplistic approach to the low bit-rate video coding certainly degrades the quality of the lecture contents. Figure 1.1 shows one decoded frame provided in

low bit-rate coded Microsoft's Advanced Streaming Format (ASF). In this case, the slide being explained by the instructor is not clearly visible to the remote student after decoding such a highly compressed lecture video. The decoded audio is noisy and unclear due to the high compression.

Figure 1.1: Poor quality video frame due to simplistic approach to low bit-rate video coding

- *Our approach using chroma-key based low bit-rate video coding*

In our approach, we focus on the issue of uninterrupted playback, while maintaining the excellent quality of the slides, on the remote student's desktop. The approach, if coupled with the streaming media technology, results in achieving real-time distance education with the existing web server infrastructure.

In our approach, we see two semantically different visual entities in the lecture video, i.e., the presentation slides and the instructor's body movements overlapping the slides. The instructor normally overlaps the slide and occupies small region in a video frame. Therefore the area containing the instructor's visual part is a good choice for applying low bit-rate video coding technique.

We propose an approach similar to the chroma-keying. A well-known example of chroma-keying is the television weather forecaster standing in front of a satellite weather picture. In reality, the forecaster stands in front of a blue screen and video gadgetry replaces the blue color with another video signal. Anything that is not the blue "key" is left unchanged in the final output.

During the content-development, we segment out the visuals of the instructor's body movements and replace the background with a fixed chroma key such as green color. The by-product of this segmentation, i.e., the chroma-key information, the size of the display window etc., serve as the display parameters during the playback. We apply a low bit-rate coding (i.e. high compression) technique on the segmented video of the instructor and interleave the low bit-rate coded voice of instructor with the coded video stream. We assume the presence of the slides (e.g. the Microsoft Powerpoint file) along with the transition timing per slide recorded during the lecture. The presentation file, the coded audio-visuals of the instructor and the display parameters, representing the lecture contents altogether, are made available through a web server.

For the playback of the lecture contents, the player on the remote student's desktop first downloads the presentation file and the display parameters from the web server. The download time is very small because of the relatively small size of the presentation file. The player extracts the slides of the original quality from the presentation file. The low bit-rate coded (i.e. highly compressed) audio-visuals of the instructor are also streamed through the player. The player then simultaneously decodes the audio and the video of the instructor, replaces the chroma key in the video frames with transparent color and superimposes the instructor's body movements over the slide. The overall synchronization between the instructor's audio-video and the running slides is achieved by maintaining the transition timing per slide of the presentation.

Our approach surpasses the real learning experienced in a classroom, because the quality of slides is excellent on student's desktop as compared with the quality appeared from a certain distance in a classroom.

1.4 Related work

1.4.1 "NetCast" project

One of the major attempts for distance education is the **'NetCast'** project [17] by a group of technicians with the support of University of South Florida. In this attempt, streaming audio-video and IP multicast technologies have been used. 'NetShow' (a product from Microsoft) [10] is used as the streaming server to multicast the lecture contents.

1.4.2 Live Intranet distance learning system using MPEG-4 over RTP/RTSP

A recent attempt [15] uses MPEG-4 to realize distance education application. MPEG-4 is a recent standard from ISO/IEC for the coding of natural and synthetic audio-visual data in the form of audio-visual objects that are arranged into an audio-visual scene by means of a scene description.

The scenario involves the video and audio of a classroom instructor where the overhead foils, the instructor may use, are sent as a separate data stream. The three streams are encapsulated in MPEG-4 systems, which adds synchronization, among the streams, and a combined composition (i.e. positioning and sizing) into a single multimedia presentation.

This attempt uses RTP/RTSP and HTTP as the transport mechanisms to deliver the lecture contents to the remote student's desktop.

1.4.3 Segmentation of the instructor's body using edge detection

We experimented with a simple approach for segmenting out the instructor's body (i.e., head, shoulders and arms mainly) from the lecture video. In this approach, we first apply the canny filter (an image processing algorithm that is used to detect edges), with suitable values of input parameters namely sigma and threshold, on each frame of the video. To detect the boundary of the person accurately, the person's

clothes must be in contrast with the color of the background surface. The shadow of the person should also be avoided in the frame. The next step is to replace the background, with a fixed color say green.

1.4.4 Some techniques for low bit-rate video coding

We studied two techniques for the low bit-rate coding of the video. Though they appear promising we could not use them for our approach, as no standard tool that we used had a support for the same.

- *Model-based techniques for low bit-rate video coding [7]*

Model-based video coding is a technique, which is suitable to achieve low bit-rate coding of a video that contains the repeated/similar actions of a human object in the video. The model-based approach entails to model the human body in some way and send only the model information on the other side. The human body image is reconstructed on the other side using this information. It results in a fairly low bit-rate as the model data and not the real image is sent over the network.

Various methods to create 2-d or 3-d models of human face and human body have been studied. The general description of a model-based coding system is shown in the figure 1.2.

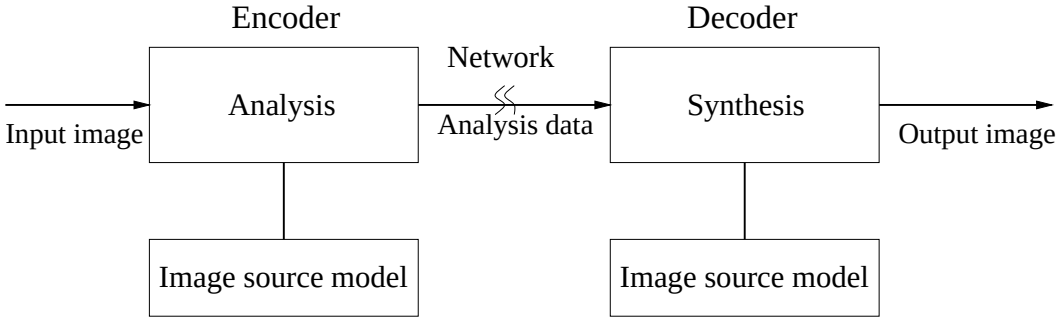


Figure 1.2: General description of a model-based coding system

MPEG-4 standard supports very low bit-rate coding of virtual human animation [18], using model-based approach, with bit-rate requirements as low as 1 kbps.

- *Low bit-rate video coding using watershed segmentation and control point tracking [1]*

Some work has been undertaken by Mr. B. Prabhakar to implement low bit-rate video coding. In his M.Tech. thesis, the author discussed about his implementation of the watershed segmentation algorithm and control point tracking. He presented the results for a sample video, which contains the upper part of the human body (face and shoulders). In this approach, the data rate achieved is between 10 and 20 kbps for a frame rate of 7.5 fps.

1.4.5 Low bit-rate speech coding

There are two approaches that we looked at for speech coding. In one approach, the coders are called waveform coders that can maintain better quality of human voice at high bit-rates. In another approach, the coders are named as vocoders (voice coders) that can achieve low bit-rate, but degrades the quality of the voice to some extent.

CELP (Code Excitation Linear Prediction) technique combines the high quality of the waveform coding with the compression efficiency of the model-based vocoders. It covers the bit-rate range from about 6 to 24 kbps. In MPEG-4, linear predictive coding (LPC) is realized by means of CELP coding technique.

We tested the CELP coding technique in its regular pulse excitation (RPE) mode using a sample audio file representing the voice of the instructor. The audio takes a storage space of 5.69 MB in .au format. The size of the CELP coded audio file (.mp4) is 307 KB that requires 21.5 kbps channel for transmission in real-time. The quality of the voice, after decoding, is acceptable for distance education application.

1.5 Organization of the thesis

The rest of the thesis is organized as follows.

In chapter 2 we discuss the overall architecture of our approach. In chapter 3 we discuss the various technologies used by us along with their programmable and

non-programmable features. In chapter 4 we discuss the scheme followed during the content-development and the implementation of the player. We discuss the results of the current work and conclude this work with some suggested future extensions in chapter 5.

Chapter 2

Overall architecture

In the context of distance education, over the existing web infrastructure, a remote student expects two things, i.e., the quick playback start-up and the smooth (i.e., uninterrupted) playback of the lecture contents, on his desktop. The streaming media technology resolves the issue of playback start-up time where the player starts the playing immediately without waiting for the whole media file to be downloaded on the client side. In our approach, we focus on the second issue of smooth playback of the lecture contents, on the remote student's desktop. Our approach, when coupled with the streaming media technology, achieves the real-time distance education over the existing narrow communication channels of bandwidth say 128 kbps.

The low bit-rate audio-video coding of the lecture contents resolves the issue of smooth playback. However, the simplistic approach to the low bit-rate video coding degrades the quality of the lecture video, especially of the slides being explained by the instructor. So our approach aims to achieve smooth playback of the low bit-rate coded lecture contents, yet maintain the excellent quality of the slides on the remote student's desktop.

In order to achieve these conflicting goals, our approach incorporates an elaborate methodology for developing the lecture contents and a dedicated player, which plays the lecture contents on the remote student's desktop (i.e., on the client side). The player executes each time a remote student accesses the lecture contents kept on a web server.

2.1 Content-development phase

The role of the content-developer (i.e., an expert person) is to code the lecture contents such that the bit-rate of the developed lecture contents doesn't exceed 128 kbps at the same time the original quality of the slides is maintained on the remote student's desktop. The content-developer uses the digital movie of a classroom lecture for the development of the lecture contents. In the movie, a fixed major portion of each video frame is occupied by presentation slide with the instructor's body movements occupying the small region and frequently overlapping the slide. The content-developer sees two semantically different entities, in the lecture movie, i.e., the running presentation and the instructor. The presentation file (made by using the Microsoft Powerpoint), used in the lecture, contains the contents of all the slides along with the transition timing per slide rehearsed during the actual lecture event. The presentation file, when kept on the web server, directly serves as the part of the lecture contents to be used by the player. So the development of the contents is nothing but generating the encoded audio-visuals of the instructor, having bit-rate not more than 128 kbps, as the remaining part of the lecture contents.

Figure 2.1 shows the pictorial representation of the content-development procedure. Following is the high-level description of the software components used in this procedure.

1. **Splitter:** The splitting of the original lecture movie is essential because we process the audio and the video separately. The splitter takes the digital movie of the captured lecture content as input. It splits the input into separate audio and video streams and saves those streams into separate audio and video files.

2. **Segmentor:** The segmenting of the instructor's visual part is essential because it is the only visual part of the lecture which can be highly compressed without much affecting the quality of the overall playback. The segmentor takes the video file generated by the splitter as input. It goes through each frame of the video file and decides which part of the frame represents foreground object. The segmentor component executes semi-automatically, where the content-developer specifies some input parameters to decide about the boundary of

the object. The remaining part of the frame is treated non-significant. The output of the segmentation is the video with only foreground object in front of a fixed color background. The fixed background color is called chroma-key and the information about the chroma-key is saved as one of the display parameters for the player's reference. Sometimes the segmentor applies cropping on input video for separating the rectangular region in the video frame where the foreground object is most likely to be present. The segmentor discards the remaining part of each video frame. This helps the segmentor execute faster. However, this may require repositioning of foreground object on any background surface. In that case, the segmentor generates cropping parameters, i.e., top, bottom, left and right margins, as the display parameters for the player's reference.

3. **Low Bit-rate Video Encoder (LBVE):** The LBVE is required to highly compress the instructor's visual part. The LBVE takes the video file of the instructor's body movements, generated by the segmentor, as input. It also takes specifications, namely the destination frame size and the destination frame rate, from the content-developer. LBVE then generates a highly compressed video file of bit-rate not more than 64 kbps. To achieve this compression, LBVE throws away the redundant information from the input video. It tries to minimize the spatial redundancy as well as temporal redundancy from the input video.

4. **Low Bit-rate Audio Encoder (LBAE):** The LBAE is required to throw away the redundant information in the audio of the instructor and convert it into a compact format. The LBAE takes the audio file of the instructor's narration, generated by the splitter, as input. It also requires the content-developer to specify the destination sample rate, i.e., samples per second, and the destination sample precision (i.e., number of bits used to represent each sample). It generates highly compressed audio file of bit-rate not more than 64 kbps. To achieve this compression, LBAE uses some characteristics of audio specific to the human voice.

5. **Synchronizer:** This component is required by the content-developer to maintain the consistency, between the instructor's body movements and the narration, as observed in the original digital movie of the lecture. Synchronizer takes the low bit-rate encoded video file and the low bit-rate encoded audio file as input and generates the synchronized low bit-rate encoded audio-video file. The synchronization is achieved by using the implicit timing information present in the input video and audio streams.

2.2 Architecture of the player

The role of the player is to play the lecture contents, available on the web server, on the remote student's desktop each time a student accesses the lecture contents. The player takes the audio-video of the instructor, the presentation file and the display parameters saved in a separate text file as its inputs. It plays the voice of the instructor and displays the visuals of the lecture, i.e., the slides along with the instructor explaining the slides.

The player needs to fulfill the following major expectations of the remote student.

1. Ease in using the player with very little user intervention.
2. Facility to save slides to local storage for student's future reference.
3. Clarity of the slides.
4. Clarity of the instructor's narration.
5. Consistency/accuracy in the instructor's body movements with respect to the contents on the slide.
6. Uninterrupted playback of the lecture contents.

Figure 2.2 represents the architecture of the player.

The player consists of following software components.

1. **Slides Extractor (SE):** The main job of the SE is to extract the displayable information, i.e., the lecture material on slides, from the Microsoft Powerpoint file. In other words, the SE decodes the Powerpoint presentation file to get the slides, as separate images, and the transition timing per slide. The SE takes the presentation file as its input. It generates the slides as separate bitmap images and a separate file containing the transition timing per slide. The output generated by the SE serves as input to the display mixer (CBDM).

2. **Audio Decoder (AD):** The AD is required to decode the audio of the lecture, i.e., the instructor's voice before playing it on the remote student's desktop. The AD takes the audio part of synchronized audio-video of the instructor as input. It decompresses the input audio and generates the voice of the instructor as uncompressed audio. A sound device is required to play this audio.

3. **Video Decoder (VD):** The VD is required to decode the video of the instructor before playing it. The VD takes the video part of the synchronized audio-video of the instructor as its input. It decompresses the video and generates the displayable video frames as output. The generated video contains only the instructor's body movements in front of a fixed chroma-key background. The VD decides the frame rate (i.e., decoded frames per second) and the frame size in pixels of uncompressed video frames from the information present in the input video stream.

4. **Chroma-key Based Display Mixer (CBDM):** The CBDM is the core component of the player. The task of this component is to present the visual aspects of the lecture, i.e., the slides and the instructor's body movements overlapping the slides, in a realistic way on the remote student's desktop. The CBDM takes the uncompressed video frames (output of the VD), the bitmap images representing the slides (output of the SE), the stored transition timing per slide extracted by the SE and the display parameters (i.e., chroma-key and positioning information etc.), generated during the content-development,

as its input. The video representing the instructor's body movements is superimposed on top of the slide images using the positioning information where the chroma-key is replaced with the transparent color. The synchronization between the running slides and the instructor's body movements is achieved by updating the slide image at appropriate time. The output generated by this component is displayed on the screen. The voice of the instructor is played in synchronization with the instructor's video.

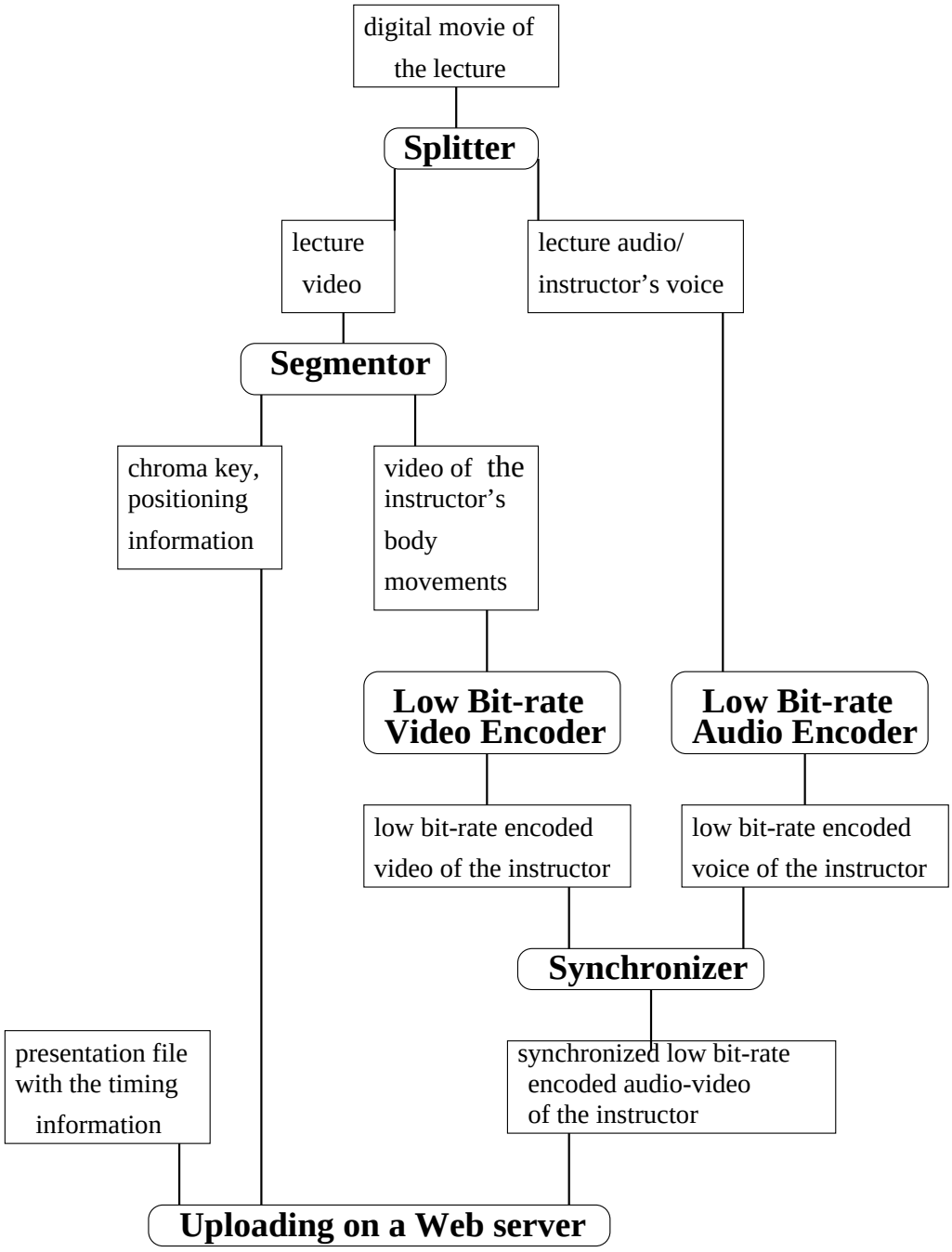


Figure 2.1: Content-development procedure

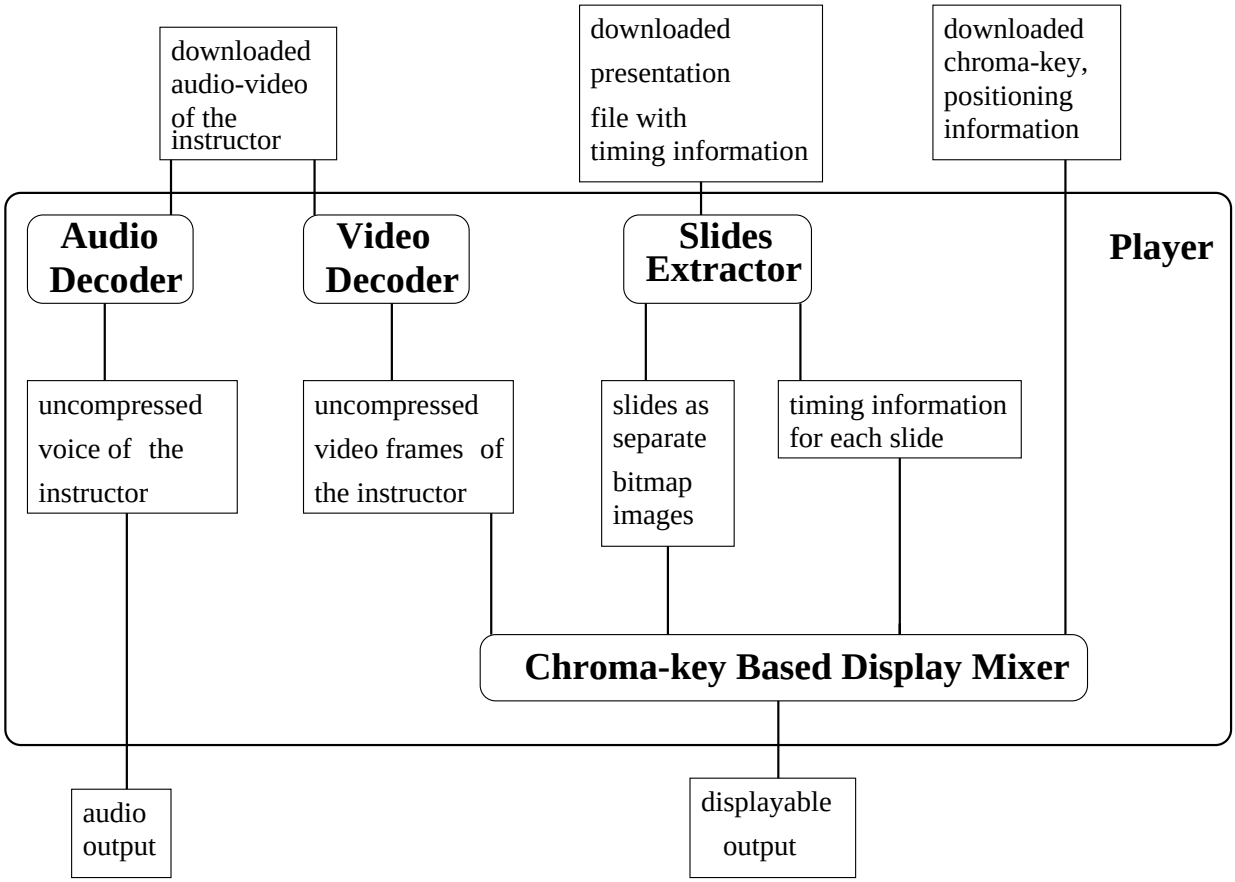


Figure 2.2: Architecture of the player

Chapter 3

Introduction to the technologies we used

3.1 Pinnacle digital video capture card

The classroom lecture video is typically available on a video cassette recorded in an analog format. To convert the audio-visual information into digital format, we use Pinnacle video capture card. The card comes with M-JPEG (Motion JPEG) compressor implemented in the hardware. It compresses each video frame separately using JPEG compression algorithm and stores the sequence of such compressed video frames, after interleaving with audio, into an AVI file. The compressor executes reasonably fast because of its hardware implementation. However, since it doesn't consider interframe difference in the video sequence, the size of the resulting AVI file is rather large. For example, for a 4 minutes lecture video, the AVI file takes 260 MB disk storage.

3.2 MPEG-1 standard

MPEG is an international standard for the generic coding of motion pictures and associated audio. The MPEG standard specifies the coding and multiplexing of video and audio with synchronization. MPEG encoded video is made up of three

types of pictures namely I, P, B.

- I-pictures are Intra-coded, i.e., they are spatially coded like still images using information only from that frame and can be reconstructed without referring to the other frames.
- P-pictures are Predictive coded, i.e., they are coded with reference to a past picture (forward prediction) which is either an I-picture or a P-picture. The referred picture (P or I) has to be decoded first before decoding a P-picture.
- B-pictures are Bi-directionally coded, i.e., they are coded with reference to a past picture (forward prediction) and a future picture (backward prediction). The referred pictures have to be non-B pictures. In this case, both past and future referred pictures have to be decoded before decoding a B-picture. A B-picture can never be used as a reference to code any other picture.

Of the three types of coding listed above, I-pictures result in the best quality but suffer from low compression whereas B-pictures have the advantage of high compression though they result in poor quality. P-picture strike a balance between I and B-pictures in terms of picture quality, compression ratio and computational complexity.

A set of encoded pictures from one I-picture to a picture just before the next I-picture form a GOP (Group of Pictures). A sequence of GOPs form an MPEG video stream. The first picture in a GOP is always an I-picture.

We use AVItoMPEG converter to convert audio-visual information from AVI format to MPEG-1 format. For a 4 minutes sample lecture video, the output file in MPEG-1 format takes around 60 MB disk storage.

3.3 AMOS: Active MPEG-4 Object Segmentation system

AMOS [21] is an active object segmentation and tracking system for general video sources. It combines low level automatic region segmentation with an active method for defining and tracking high-level semantic video objects.

The system allows users to identify a semantic object by using mouse in the starting frame of a video object. The object is defined by an outline polygon whose vertices and edges are roughly along the desired object boundary. To tolerate user-input error, a snake algorithm [14] is used to align the user-specified polygon to the actual object boundary. The snake algorithm is based on minimizing a specific energy function associated with edge pixels. Users may also choose to skip the snake module if a relatively accurate outline is already provided. Users can then start the object tracking process by specifying a few parameters such as color threshold and motion threshold. At any frame, users may stop the tracking process to refine the object boundary, change the tracking parameters and resume the tracking process.

The tracking process is carried out in two stages. The first stage is an initial object segmentation stage where the user input is used to create a semantic object with underlying homogeneous regions. The second stage is an object tracking stage where the homogeneous regions and the object are tracked through successive frames.

The input video can be PPM picture sequence or MPEG motion picture. The system generates a binary-mask in the PGM (i.e., Portable GrayMap) format, for the tracked object, at each frame. Figure 3.3 shows the results of the AMOS system for a sample input image.

Original frame Binary mask

Figure 3.1: Sample input-output of AMOS software

3.4 Chroma-key shape coding technique

This shape coding technique was inspired from the blue screen technique used in film and TV studios. In this technique, the color of a pixel is used to distinguish between foreground object and background object. The object to be coded is placed on a static single color background. The color of the background (chroma-key) has to be the one not used by the texture of the object. Usually highly saturated colors fulfill this requirement. The image or sequence of images with the object in front of this one-colored background is then encoded using a conventional encoder in full frame mode. Chroma-key is transmitted to the decoder. The decoder decodes the images and makes transparent the pixels of color given by the chroma-key. The transparent colored pixel is one that doesn't get modified in its intensity. Figure 3.2 shows a sample video frame. Here the background is green color key (shown in gray variation) which is communicated to the decoder to treat the green pixels as transparent while decoding.

Figure 3.2: Sample frame used for chroma-key shape coding

3.5 Microsoft's AVI format

In the current work, we use the Microsoft AVI format [6] of coding the audio-video because the interactive AVI codecs are freely available. Moreover, these codecs fulfill the requirements, of our approach, namely chroma-key based transparency, reasonably good audio-video synchronization and scalability.

AVI stands for Audio Video Interleave and is defined by Microsoft. It is one of the most common format for audio and/or video data on the PC.

Web authors, planning to use AVI, must provide highly compressed video at low bit-rates because of the limited bandwidth available (from 28.8 kbps for phone line to 128 kbps for ISDN lines).

To compress the AVI video at low bit-rate, some of the options are as following.

- Use smaller frame sizes (e.g. 160x120 pixels)
- Use lower frame rates (10-15 frames per second)
- Use of the newer codecs such as Indeo 4.1/5.1 [6].

The AVI format was developed for playback of audio and video from local storage on personal computers. It is also adequate for downloading a video file from a remote site on the Internet for subsequent playback from the computer's hard drive. It is not well suited for streaming video playback over networks. The AVI file format lacks time stamps embedded in the audio and video streams. There is no mechanism to resynchronize the audio and video streams if data is lost in the network. Because of these limitations of the AVI format, it can't be used as a streaming media.

3.6 RDX: Realistic Display miXer SDK 3.02

We use RDX technology [4] to implement the Chroma-key Based Display Mixer (CBDM) in the player. The display mixer mixes the slides and the decompressed frames of the instructor's video and displays each frame of the lecture in a realistic fashion.

Intel's Realistic Display miXer SDK provides a set of basic routines for multimedia applications. The libraries are designed for use on Intel Pentium class processors with or without MMX technology. The system provides object-oriented support for multimedia objects. RDX technology is designed for Win32 API (i.e. Windows 95, 98, Windows NT 4.0 etc.) and uses the Microsoft DirectX and ActiveMovie SDK for enhanced performance while dealing with the video.

RDX architecture lets the developer access RDX technology directly or through COM-compliant interfaces, through ActiveX controls or through the Microsoft Java Virtual Machine.

Benefits of the Intel RDX system

The RDX system provides ease of use, extensibility, ease of display mixing, high performance, coordination of displayable and non-displayable activities and support for performance improvements in underlying software components.

Display Mixing: The use of display mixing had been somewhat limited because video and graphics were treated as independent entities. As a result, many multimedia Windows applications use one data type or the other but not both. However, in Intel RDX system, the display mixing is achieved whereby the programmer needs to create a set of objects, define their generic attributes, map them to a surface and then draw the surface. Each of these operations is accomplished by high-level function calls. The display mixing is software-based and is performed in real-time without requiring any special hardware.

Coordinated Activities: Multimedia applications are primarily based on displayable objects, but they also need audio objects and devices and the ability to coordinate their interplay easily and accurately. For this, Intel RDX provides 'timers and events' mechanism to support scheduling and synchronization.

Components of the Intel RDX system

Figure 3.3 shows the architecture of Intel RDX core.

In the figure, following are the programmable components of Intel RDX system.

- **Base objects:** The base objects provide base class attributes, surfaces and the display mixer. Programmable base class attributes are visibility, draw order, current image in multiple-image object, destination rectangle, position etc. Surfaces are typically used as destinations for displayable objects. The **Display Mixer** consolidates all the modifications and renders the resulting image to a surface or window using DirectDraw. Audio data is played through DirectSound.

- **2D objects:** 2-d objects are used for the creation of sprites (i.e., foreground objects such as tree, car, building etc.), backgrounds and tiles.

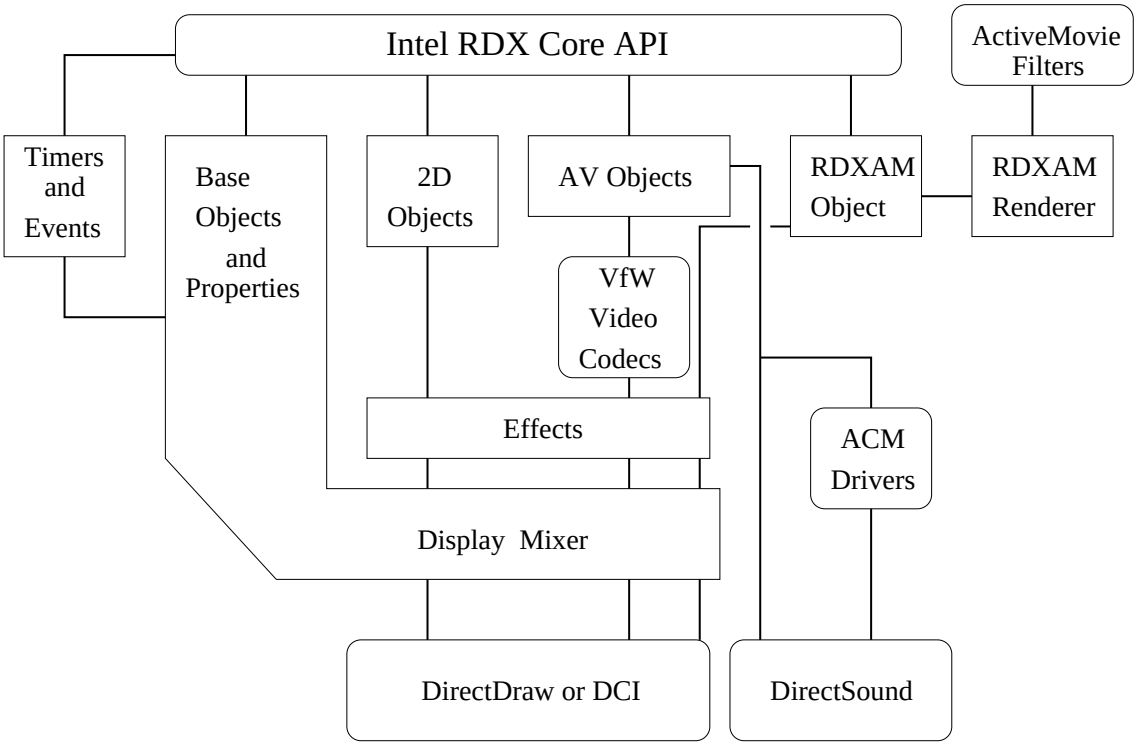


Figure 3.3: Architecture of Intel RDX core

- **RDX AM objects:** The AM objects are used for attaching the Microsoft ActiveMovie streams to an Intel RDX surface and playing video available from files, video cameras or networks.

- **RDXAM Renderer:** This component is used to terminate the filter graph built with ActiveMovie/DirectShow SDK and COM interface. The renderers in the newly built filter graph must be associated to AM objects.

- **ACM Drivers:** These components are Audio Compression Manager (ACM) drivers used to decompress audio data.

- **ActiveMovie filters:** These components are software modules provided with the Microsoft ActiveMovie. The modules include the functions such as reading from a media file, writing to a media file, transforming (e.g. compressing, decompressing etc.) a media stream, rendering video on a surface etc. These filters are used to generate ActiveMovie streams.

- **Source Data Objects:** By default, an Intel RDX object has no source data and cannot be displayed. To make source data available to an Intel RDX object, programmer must associate it with a Source Data Object (SDO). The SDO is then associated with the Intel RDX object. The system supports four kinds of SDOs. They are bitmap objects, sequence objects, file objects and filtergraph objects.

  The bitmap objects encapsulate a single bitmap image. The sequence objects encapsulate multiple bitmap files and are used to create sequenced sprites and backgrounds. The file objects encapsulate the audio and/or video streams in an AVI or .wav file on local storage. The filtergraph objects represent the interconnection of active software components, called filters and are used to create audio and video objects. The main objective of filter graph mechanism is to be able to support future image/movie formats and technologies. The filter graphs provide a mechanism of attaching pluggable components called filters. A filter normally performs a basic function such as reading from a media file, writing to a media file, transforming (e.g. compressing, decompressing etc.) a media stream, rendering video on a surface etc.

Usage models for RDX ActiveMovie (AM) objects

In RDX, AM objects can be used in one of the two use models. In the first use model, the application uses high-level functions to create and manipulate streams. In the second use model the developer creates a filter graph using ActiveMovie COM interface and renders the movie onto an RDX surface.

The first model hides the complexity of the filter graph creation from the developer by providing higher-level functionality. The second model provides the developer with flexibility and more control on handling the variety of audio/video.

The first model uses the filters provides by the Microsoft ActiveMovie 1.0. This limits the audio-video formats which can be attached to the RDX AM object. For example, Indeo Video Interactive (Indeo 4.1) decompressor filter is provided by ActiveMovie 1.0. The first model can attach Indeo 4.1 compressed AVI file (either available from the local storage or from the network) to an RDX AM object. To use

Indeo Video 5.11 decompressor filter, which is a DirectShow (i.e., ActiveMovie 2.0) filter, we need to use the second model.

3.7 Indeo Video Interactive

Indeo Video Interactive, Indeo 4.1, [6], [8] is a video codec from Intel. It can be installed either as a Video for Windows (VfW) codec, or as an ActiveMovie filter.

Indeo 4.1 supports transparent pixels in video frames. These can be used to implement effects such as chroma-keying. The application can provide a bitmap image or even another video as a background in the transparent areas of the image.

Indeo 4.1 video streams are identified by the four character code 'iv41' in the video stream header for the AVI file. Indeo 4.1 claims to implement a hybrid wavelet transform.

3.8 Microsoft Powerpoint COM Object Library

The Microsoft Powerpoint COM objects provide properties and methods which are accessible in Visual Basic programming environment. Some of the classes provided by this library are *application, presentation and slide*. Using Powerpoint object model, the programmer can start Powerpoint application, close the application, maximize/minimize the application window, open a local or remote presentation file, save the slides of the active presentation as a sequence of images in GIF, BMP, JPEG formats, read the advance time of each slide etc.

The size of bitmap files supported by Powerpoint is 720 x 540 pixels. The number of bits used to represent color of a pixel (BPP) is decided separately by Powerpoint for each slide depending on the range of colors present in the respective slide. For example, one slide may be saved as a BMP file with BPP as 4 (at most 16 distinct colors), another slide with BPP as 8 (at most 256 distinct colors) or some other slide with BPP as 24 (at most 16777216 distinct colors).

Chapter 4

Implementation details

In the context of real-time distance education, a remote student expects both the quick playback start-up and the uninterrupted playback of the lecture contents on his desktop. We assume that the lecture contents are available normally in the form of a digital movie of a classroom lecture. In the movie, a fixed portion of each video frame is occupied by the presentation slide with the instructor's body movements occupying the small region and frequently overlapping the slide. If the lecture contents are delivered in the streaming media format, then the client starts showing it immediately, before downloading it completely. Thus the issue of playback start-up time is resolved by the use of streaming media technology. In order to achieve the uninterrupted playback of the lecture contents, over the existing narrow communication channels of bandwidth say 128 kbps, the low bit-rate coding (i.e. high compression) of the lecture contents is a natural choice. However, the simplistic approach to low bit-rate coding degrades the quality of the lecture contents, especially of the slides used in the lecture.

The current implementation gives the partial solution towards real-time distance education. Here we follow a scheme for the content-development such that the slides maintain their original quality at the same time the coded audio-video of the lecture contents does not exceed 128 kbps bit-rate. Even though we have not integrated the streaming media technology with the current implementation, it is rather an independent issue. Depending upon the availability of the codec, future

implementation may incorporate this technology also. Therefore to achieve the real-time playback of the lecture contents, on the remote student's desktop, the future implementation should use the streaming media for coding the audio-video of the lecture contents.

The content-developer needs to fulfill two conflicting goals. The first goal is to maintain the original quality of the slides on the remote student's desktop and the second goal is to compress the audio-video of the lecture contents to low bit-rate (i.e. not higher than 128 kbps of ISDN channels). For this, the content-developer sees two semantically different entities in the digital movie of the lecture, i.e., the slides and the instructor who is explaining the slides. In our approach, these two entities are treated separately throughout the development of the lecture contents. The developed lecture contents, namely the presentation file (made using the Microsoft Powerpoint) used in the lecture and the separate low bit-rate coded audio-video of the instructor, are made available through a web server. Additionally the presentation file contains the timing information for each slide already recorded during the lecture event.

In order to play the lecture contents on the remote student's desktop, we have implemented a dedicated player. The player treats the presentation slides and the instructor as two separate inputs and achieves the realistic synchronization between the running slides and the instructor's explanation of the slides on the remote student's desktop.

4.1 Content-development methodology

The role of the content-developer is to develop the lecture contents in a format which is supported by the player. For this, the content-developer needs to choose the suitable audio-video processing tools and integrate them to generate the lecture contents in the desired format. The main choice is that of the audio-video codec (i.e. COder DECoder pair) for compressing and decompressing the instructor's audio-visuals. We need a video codec that supports low bit-rate coding (i.e. high compression) and chroma-key based transparency. Intel's Indeo Video Interactive

(Indeo 4.1) is the only codec, supported by the player, which supports both chroma-key based transparency and low bit-rate video coding. Hence we propose to use Indeo 4.1 for low bit-rate video coding of the instructor's visuals.

Moreover, the next version of Indeo Video codec, i.e., Indeo Video 5.11 [8], supports an additional feature of progressive playback along with the low bit-rate coding and the chroma-key based transparency. This allows an easy extension to the current implementation to incorporate streaming media technology.

While authoring the media contents, usually the selection of the video codec guides the decisions, namely the selection of accompanying audio format, the audio-video synchronization format, the frame rate and the frame size in pixels etc. With the selection of Indeo 4.1, as the video codec, most of these parameters are fixed.

4.1.1 Issues resolved by the selection of Indeo 4.1

- **Selection of the media file format:** Indeo 4.1 is an AVI (described in the section 3.5) codec. So the content-developer uses AVI format for developing the instructor's audio-video.

- **Selection of the audio format:** AVI files are frequently generated with the uncompressed PCM audio [6], where the size of the audio track can be reduced by using 8-bit mono (one channel) PCM audio with 8 KHz sample rate. We found this 64 kbps format sufficient for the telephonic quality of the voice of the instructor.

- **Synchronization between audio and video:** AVI format specifies the method to integrate audio and video using implicit timing information present in audio and video streams. Hence, by using AVI format, there is no need of a separate mechanism to achieve audio-video synchronization.

- **Selection of the frame rate:** For a fixed bit-rate of coded video, the content-developer requires to make a trade-off between the frame quality and the frame rate. For higher frame rate, he has to sacrifice the frame quality and vice versa. As in a lecture, instructor's movements are typically slow, we decided

to sacrifice the frame rate. (Normally, high frame rate is used for high motion video, to avoid jerks in the motion.)

Indeo 4.1 allows reducing the frame rate to achieve low bit-rate video encoding. We assume the frame rate of original lecture movie to be either 30 fps (for NTSC) or 25 fps (for PAL). Though standard frame rate for AVI is 15 fps, with several experiments with Indeo 4.1 it was found that 5 fps gives reasonably smooth quality of the video with reasonable bit-rate. We therefore chose the frame rate to be 5 fps.

- **Selection of the frame size:** Since the instructor's visuals occupy a small region in the lecture video frame, the content-developer can choose the frame size for the instructor's video smaller than the frame size of original digital movie of the lecture. The destination frame size for the instructor's video is calculated manually in proportion to the size of the presentation slide for achieving the realistic display mixing on the remote student's desktop. However, not all frame sizes are supported by the video codec. For example, frame size of 492 x 368 pixels is not supported by Indeo 4.1. In such case, we adjust to frame size 480 x 360 in pixels, (for the instructor's video) which is acceptable by Indeo 4.1, without introducing much error.

4.1.2 System requirements for content-development

Following list specifies the system requirements for the content-developer's machine.

- Win32 (i.e. Windows 95/98/NT4.0/2000) with DirectX 3 or better
- avifil32.dll, msvfw32.dll, msacm32.dll in Windows system folder
- Display adapter that supports True-Color
- Intel's Indeo Video 4.5 (Indeo 4.1) codec [8]
- VideoMach 2.3.5 Editing tool [19]
- AMOS segmentation software [21]

- JDK 1.1.6 or above (with JIT compiler), required by AMOS software
- Java Media Framework 1.1 or above, required by AMOS software.

4.1.3 Scheme for the development of the lecture contents

The scheme, followed by the content-developer to develop the low bit-rate coded audio-video of the instructor, mainly suggests the way to integrate the existing software tools. The block diagram representing the scheme for the usage of various tools in the content-development procedure is shown in the figure 4.1. Various inputs are designated as A, B, C,..., P in this figure and are discussed below.

- A: Digital movie of the lecture in MPEG-1 format.
- B: Cropping parameters as the user input (for discarding redundant visual information).
- C: Only significant rectangular portion of the lecture video saved as separate frames in PPM (Portable PixelMap) format.
- D: Width of the display window and the positioning information of the presentation slides, with respect to display window, saved as the display parameters.
- E: Voice of the instructor saved in a .wav file of bit-rate 64 kbps
- F: Cropping parameters as the user input (for speeding up the segmentation).
- G: Positioning information of the instructor's video with respect to the display window, saved as the display parameters.
- H: Subframes of the lecture video (with frame rate 5 fps), containing the instructor's body movements, saved in PPM format.
- I: The color threshold value, the motion threshold value and the boundary information as the user input.
- J: The binary segmentation masks (one for each input frame) saved in PGM (Portable GrayMap) format.

- K: The frames representing instructor's body on the chroma-key background saved in PPM format.
- L: Chroma-key value saved as the display parameters.
- M: The destination frame rate, the destination frame size, the bit-rate, the key frame information as the user input.
- N: The low bit-rate encoded stream representing the instructor's video.
- O: Interleave duration (in seconds) as the user input.
- P: Synchronized low bit-rate encoded audio-video of the instructor saved as an AVI file.

The lecture contents are developed in the following five steps.

1. **Splitting the lecture contents and low bit-rate audio extraction:** The first step is essential to throw away unnecessary (i.e. redundant) audio-visual information from the lecture movie. This defines the scope for the further development steps.

   We use VideoMach software for this step. It takes the digital movie of captured lecture in MPEG-1 format as input, and generates the audio and the video of the lecture in two separate files. The audio containing the voice of the instructor is generated in a .wav file with 64 kbps bit-rate. The video contains only the significant rectangular portion of the lecture's visuals that is later displayed on the remote student's desktop.

   The video output is generated in the form of the cropped video frames saved as separate PPM (Portable PixelMap) files for each video frame. The PPM files generated by this step serve as the input to the next step. The content-developer manually calculates the width of the display window (for a fixed height of 540 pixels) from the cropping parameter values, i.e., top, bottom, left and right margins, which are used for cropping the video. (The player supports the display window of height 540 pixels) Also the content-developer manually calculates the positioning information of the slides for the calculated

size of the display window. The calculated information serve as the display parameters to the player.

The audio output, of bit-rate 64 kbps, is generated by tuning the parameters of VideoMach as follows.

- The sampling rate of the original audio is changed to 8 KHz.
- The number of bits per sample are changed to 8-bits.
- Only one channel (mono) is retained.

2. **Video preprocessing required before the segmentation:** This step is essential to improve the efficiency of the execution of segmentation algorithm, which is computationally intense. We further reduce the frame rate and the frame size of the input visuals in this step by using the VideoMach editing software. It selects every fifth PPM file to represent a frame rate of 5 fps and contain only the rectangular region of the video frame which shows instructor's body (i.e., arm movements, shoulder and head) during the whole video. The positioning information of the instructor's video is also extracted and saved as the display parameters for the player's reference.

3. **Active object segmentation:** This step is essential to segment out the video containing only the instructor's body movements. We use AMOS software to implement this step. The usage of the AMOS software is described in the section 3.3. It takes as input the PPM files generated by the preprocessing step and the color and motion threshold values specified by the content-developer. The software executes semi-automatically taking manual input to decide about the boundary of the foreground object (i.e., the instructor's body). The output is the binary segmentation masks in PGM (Portable GrayMap) file format where the AMOS software chooses white color to represent the foreground object's region and black color to represent the background object's region. The AMOS software generates one PGM file for each input PPM file. An example is given in the figure 3.3.

   The effective usage of the AMOS software is possible only after gaining some

experience of its behavior. It is advised to set the low color threshold value for accurate segmentation. However, this setting results in the slow execution of the AMOS software. The motion threshold value can be low whenever the PPM sequence has slow instructor's body movements. This should be set to a high value whenever the movie sequence has fast instructor's body movements. This results in a fast execution without compromising the accuracy of the segmentation.

4. **Chroma-key screening:** This step is essential to make a fixed color (chroma-key) background for the video of instructor's body movements. We developed a small code in ANSI C to implement this step. It takes the PPM files generated by the preprocessing step and the PGM files generated by the active object segmentation step as input. It treats a pixel in a PPM file as the background pixel, if the corresponding PGM file contains the black pixel for the same position, and replaces the pixel with chroma-key color. Otherwise, the pixel in the PPM file is left unchanged. The chroma-key information is then saved as the display parameter for the player's reference.

5. **Low bit-rate video encoding and audio-video synchronization:** This is the final step essential to generate the low bit-rate encoded audio-video of the instructor as a single file. We use Indeo 4.1 AVI encoder for encoding the instructor's video. In addition, the audio-video synchronization is achieved by using the VideoMach software.

   The PPM files, generated after the chroma-key screening, and the audio (.wav) file of the instructor's voice, generated by the splitting (i.e., the first step), are used to generate the synchronized low bit-rate encoded audio-video of the instructor saved in an AVI file.

   The following parameters are used in the VideoMach software.

   (a) Frame rate = 5 fps

   (b) Key frame = (after) 5 frames (Only key frames are intra-coded)

   (c) Data rate = 8 Kbytes/second (i.e. 64 kbps)

(d) Interleave audio every 5 frames (thus interleave duration is set to 1 second.)

(e) Resolution (i.e. the destination frame size)

The resolution for the instructor's video is computed manually using the following information.

- The size (width x height in pixels) of the PPM files found during the splitting (i.e., the first step).
- The size (width x height in pixels) of the display window of the player, calculated during the splitting.
- The size (width x height in pixels) of PPM files calculated during the second step of content-development. This is the size of the rectangular region (in the digital movie of the lecture) containing the instructor's body movements throughout the lecture.

After these various parameters are set, the VideoMach software achieves the authoring of the AVI file by encoding of the video stream and interleaving the encoded video stream with the audio stream. The output of this process is the AVI file of the instructor's audio-video of the bit-rate 128 kbps.

The lecture contents are now ready comprising of these files. The low bit-rate audio-visuals of the instructor is available in the AVI file. The display parameters are saved in a text file PlayerRef.txt. The slides are in a .ppt file. All these three files are made available through a web server.

4.2 Implementation of the player

The role of the player is to play the highly compressed lecture contents, downloaded from the web server, such that the original quality of the slides is maintained on the remote student's desktop. For this, the player treats the presentation slides and the instructor's video as two separate inputs and achieves the realistic synchronization between the running slides and the instructor's explanation of the slides on the remote student's desktop. As mentioned earlier, the current implementation gives

the partial solution towards real-time distance education and provides a framework for future extension. In other words, the current implementation of the player does not support the streaming media format for the instructor's audio-visuals.

The implementation of the player follows the architecture of the player shown in the figure 2.2. We have implemented two components of the player, namely, Slides Extractor (SE), required to decode the presentation file; and Chroma-key Based Display Mixer (CBDM), required to mix the visuals of the slides with the movie of the instructor's body movements. In the current implementation, the CBDM itself chooses the suitable audio and video decoders among those provided by the ActiveMovie.

4.2.1 System requirements to execute the player

The player can run on the student's desktop with the following hardware and software.

- Pentium processor running at 90 MHz or above, with at least 16 MB of RAM.
- A sound card with DirectSound drivers.
- VGA-compatible card with DirectDraw support.
- Win32 (i.e. Windows 95/98/NT4.0/2000)
- DirectX and ActiveMovie 1.0 or better (i.e. DirectShow)
- Microsoft Powerpoint COM object library
- IrfanView image converter

4.2.2 Slides Extractor (SE)

The main job of the SE is to extract the displayable information, i.e., the lecture material on the slides, from the Microsoft Powerpoint file (i.e. .ppt file) and to make it available, on local storage of the remote student's desktop, in a format supported by the CBDM. The SE takes the Microsoft Powerpoint presentation file (.ppt) that

also contains the recorded transition timing for each slide. The SE generates the slides, in the form of 256-color bitmap images of size 720x540 and extracts the transition timing per slide, in seconds, to a separate file on the local storage, which is then used by the CBDM.

We have implemented the SE in the Microsoft Visual Basic 6.0. The implementation is done in two steps. The first step is the decoding of the .ppt file using the Microsoft Powerpoint COM object library [9]. The second step is the post processing of the slides using IrfanView image converter [5].

Decoding of the .ppt file

For the decoding of the .ppt file, the SE uses the functionality of the Powerpoint COM object library. The methods and properties of the classes available in the library are accessible using the Visual Basic programming environment. The SE extracts the bitmap of each slide using 'saveas' method of 'presentation' class. The 'saveas' method treats each bitmap file separately. For instance, some of the slides may be saved as 16-color (i.e., 4 BPP) bitmap images while the other slides may be saved as 256-color (i.e., 8 BPP) bitmap images. The SE also extracts the transition timing for each slide from the value of the property namely AdvanceTime of the class 'SlideShowTransition'. These values are saved in a temporary file, which are later, used by the CBDM.

Post processing of the slides

This step is required because the slide images, required by the CBDM, need to be all 256-color bitmap images (i.e., 8 BPP bitmap images). The SE uses IrfanView image converter in order to convert all bitmap files, generated by the previous step, to the 256-color (i.e. 8 BPP) bitmap files. The SE uses the 'Shell' function of Visual Basic that runs an executable program given its pathname as the first argument.

4.2.3 Chroma-key Based Display Mixer (CBDM)

We use Intel's RDX (Realistic Display miXer) technology [4] to implement the CBDM. The RDX provides object-oriented support for multimedia objects. We use various RDX objects, which mainly represent the visual entities in the lecture contents, i.e., the sequence of the slides and the instructor's body movements overlapping the slides.

The use model that we follow in the implementation of the CBDM is shown in the figure 4.2. The model has three phases.

1. **Building phase:** The role of the building phase is to define the required RDX objects to represent the visual input, namely the slides; the video of the instructor's body movements and a default background surface of the display window.

   RDX provides a sprite object to represent the 2-d bitmap image available on the local storage. A sequenced sprite object is used to encapsulate a sequence of bitmap images. We define a sequenced sprite object to represent the sequence of slides (in bitmap image format) and associate it with the 256-color slide images each of size 720x540, generated by the SE.

   RDX provides an AM (ActiveMovie) object to represent the audio-video data from the network. In addition, RDX requires an association of a filter graph with the AM object for rendering such audio-video data. The ActiveMovie filter graphs are the interconnection of the filters or the software components that handle the basic functions while dealing with the audio-video files. These filters include functions for reading from a file, writing to a file, compressing or decompressing a file, rendering the video etc. In our approach, we use a predefined filtergraph in RDX for handling the AVI file. We associate the AVI file of the instructor's movie with the predefined filtergraph and an AM object. This results in attaching the RDX renderer filter to RDX surface for playing the AVI file of the instructor, taken from the network.

2. **Managing phase:** The role of the managing phase is to manipulate the RDX objects, namely the surface, the sequenced sprite and the AM object, in order

to mix the visual contents represented by these RDX objects in a desired way. The manipulation is guided by the display parameters, generated by the content-developer.

Surface manipulation: The surface is manipulated by setting its attributes such as the size in pixels, the draw order, the color and the destination. We set the height of the surface to 540 pixels same as the height of the slide images generated by the SE. We set the width of the surface to the value provided by the display parameters. The color of the surface is set to black and the destination of the surface is set to the display window created by the player.

The draw order is an attribute of the surface, the sprite and the AM object. Conceptually a surface consists of multiple planes arranged one behind the other like sheets in a notepad. The planes are numbered with plane 1 in front of plane 2, plane 2 in front of plane 3 and so on. An Intel RDX object's draw order is the plane in which it is drawn. We want the surface to be behind the sprite as well as the AM object. Therefore we set the draw order of the surface to an arbitrary high value (say 100).

Sprite manipulation: The sequenced sprite represents the sequence of the slide images. The sprite is manipulated by setting its attributes such as the position and the draw order. In addition, the updating of the displayed image to the next one at an appropriate time, is achieved by using 'timer and events' mechanism provided by the RDX.

The position (i.e., X and Y coordinates) of the top-left corner of the sprite is defined with respect to the coordinate system of the surface. Since the height of the surface, in pixels, matches exactly with the height of the sprite (i.e. 540), we set the Y value to 0 and X value as obtained from the display parameters.

The draw order of the sprite is set to a value (say 50) that is less than the draw order value of the surface (in this case, 100) and greater than the draw order value of the AM object (say 30). It ensures that after rendering the surface, the instructor's body movements overlap the slides.

The slides should be displayed in the original order synchronized with the

instructor's video and audio. We achieve the realistic presentation of the slides by advancing the current image to the next one after the pre-recorded slide timing as generated by the SE. We use the 'timer and events' mechanism, provided by RDX, to perform this transition. We define an event object for each image in the sequenced sprite and program the event handler to advance the current slide to the next one. The timer object is programmed to schedule the event at the appropriate time.

AM object manipulation: The AM object represents the audio-video of the instructor. We manipulate the AM object by setting the position and the transparency color as attributes.

The position (i.e., X and Y coordinates) of the AM object refers to the position of the top-left corner of the video frame containing the instructor's body movements, with respect to the coordinate system of the surface. The X and Y are set to the values as obtained from the display parameters.

The transparency color attribute of the video of the AM object is set to the chroma-key (e.g. green color) obtained from the display parameters.

3. **Rendering phase:** In the rendering phase the surface, the sequenced sprite and the AM object are displayed with manipulations as performed in the previous phase. The rendering takes place throughout the length, of the lecture contents, in time. In this phase, the only responsibility of the programmer is to draw the surface, in a loop, so that the updated visual data on the surface gets drawn in the display window.

   When the RDX AM object is played, both of its audio and video tracks are automatically synchronized and played simultaneously. RDX communicates with Indeo 4.1 decompression filter for getting uncompressed video frames of the instructor. For each uncompressed video frame, only the non-transparent pixels are drawn on the surface, thus producing the effect of superimposing the instructor's body on the currently displayed slide. The mixed visual data is displayed using DirectDraw. Further RDX also communicates with ACM drivers to decompress the audio data and generates the voice of the instructor

using DirectSound.

The synchronization between the slides and the instructor's talk is based on an assumption. The CBDM assumes that the total duration of the instructor's audio-visuals is same as that of the total rehearsed timing of the whole presentation. Since the CBDM starts, as well as stops, the rendering of both the slides and the instructor's audio-video exactly at the same time, this automatically results in the synchronization between the running presentation and the instructor's explanation of the slides.

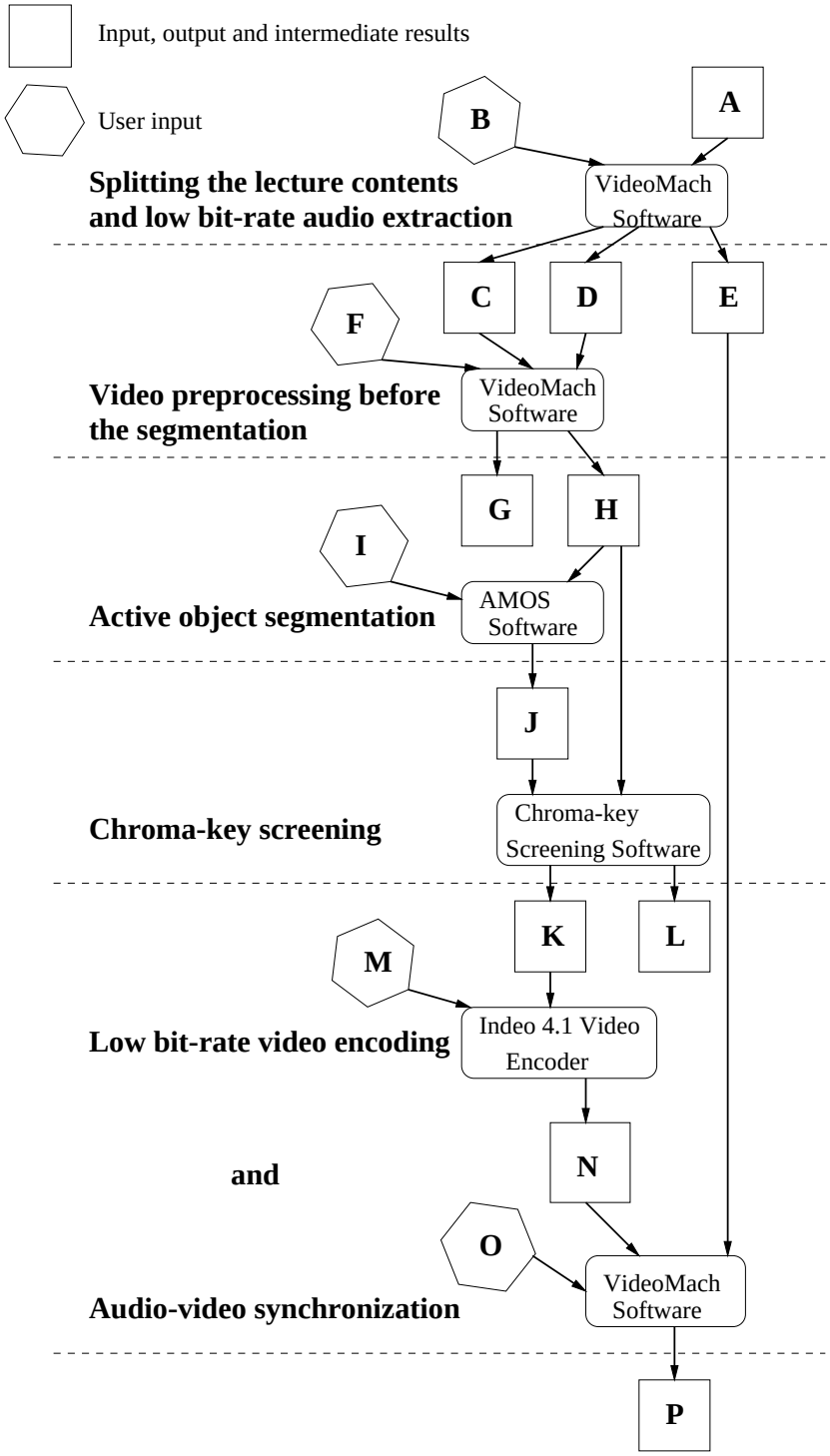


Figure 4.1: The scheme followed during the content-development

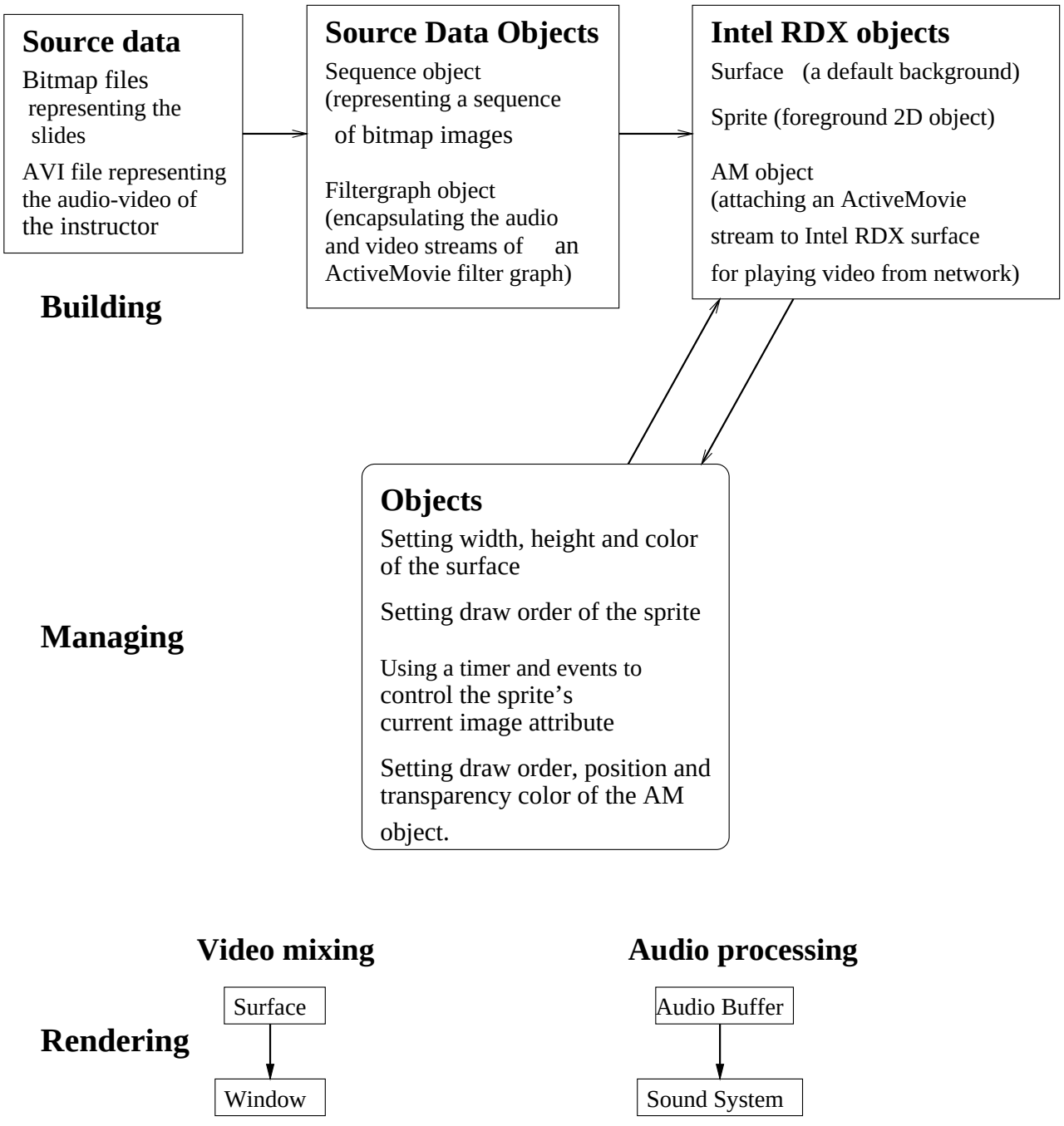


Figure 4.2: Our usage model for Intel RDX objects

Chapter 5

Results and conclusion

To test the implementation, we use a sample lecture movie (in MPEG-1 format) of 4 minutes duration. The movie is of size 57 MB, of bit-rate 1904 kbps, with the frame rate of 25 fps and the frame size of 352x288 pixels. Using this movie, we developed the lecture contents in the form of an AVI file, representing the audio-video of the instructor, and a presentation file (made using the Microsoft Powerpoint). The size of the resulting AVI file is 3.88 MB, with a bit-rate of 128 kbps, frame rate of 5 fps and the frame size of 480x360 pixels. The size of the PowerPoint file is 103 KB. Thus the compression ratio achieved for this lecture is approximately 15:1 (from MPEG-1 video).

5.1 Visual outcome at various stages of the content-development

We developed the contents on Windows 2000 platform (with Pentium III, 64 MB RAM) and generated the low bit-rate encoded audio-video of the instructor in the AVI format.

For the complete content-development task, we required 2 GB disk space. The procedure took around 5 hours including 4 and half hours, required for the execution of the AMOS software to segment out the instructor's body. The other steps of the content-development took a small time of around 10 minutes each.

Following are the intermediate results, represented in the form of one video frame of the lecture, generated by different steps of the content-development as discussed in section 4.1.3.

- Figure 5.1 shows a video frame in the original digital movie of the frame size 352x288 pixels. It represents the input to the first step of the content-development.

Figure 5.1: Frame before discarding the redundant information

- Figure 5.2 shows the video frame cropped to size 352x235 pixels containing only the significant portion of the lecture video. It represents the outcome of the first step, saved in the PPM (Portable PixelMap) format.

- Figure 5.3 shows the video frame cropped to size 200x150 pixels containing the instructor's body as the main visual object. It represents the outcome, of the second step, saved in the PPM format.

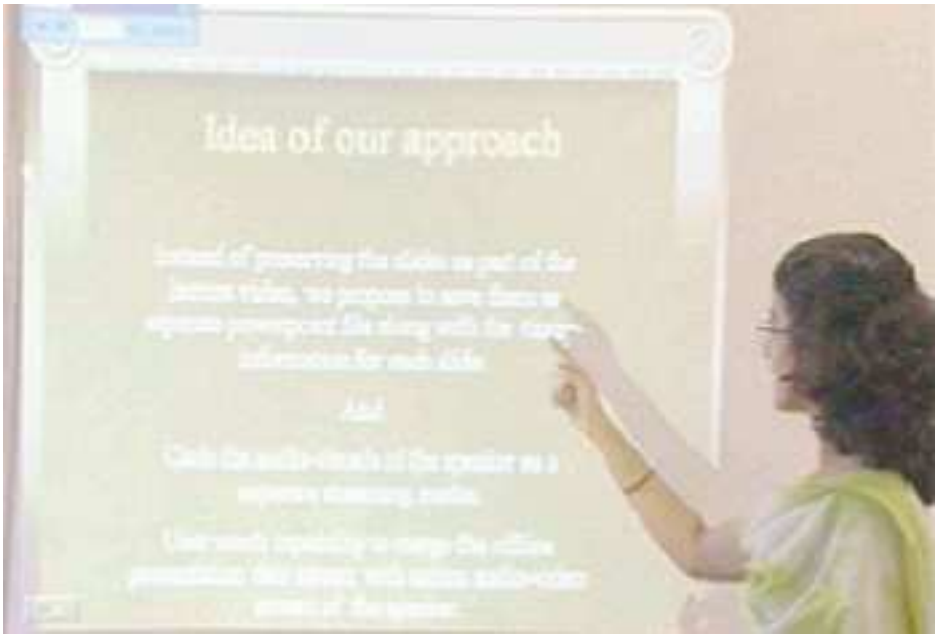


Figure 5.2: Frame after discarding redundant information

- Figure 5.4 shows the binary segmentation mask of size 200x150 pixels representing the outcome, of the third step (i.e., the active object segmentation step), saved in the PGM format.
- Figure 5.5 shows the video frame of size 200x150 pixels containing the instructor's body in front of the chroma-key background. (In this case, the chroma-key is green color shown in gray variation). It represents the outcome, of the fourth step (i.e., the chroma-key screening step), saved in the PPM format.

The sequence of the PPM files, generated by the fourth step serves as the visual input to the fifth (final) step for generating low bit-rate encoded video of the instructor.

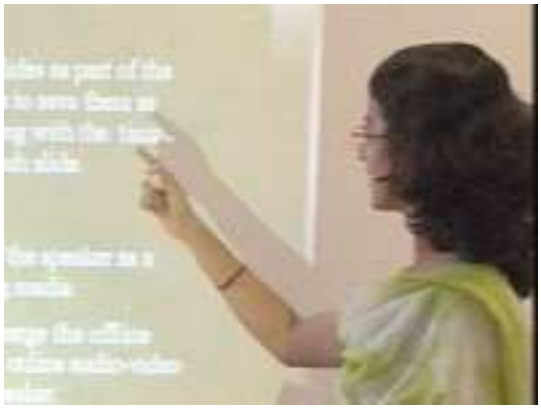


Figure 5.3: Outcome of the preprocessing, done before the actual segmentation

Figure 5.4: Outcome of the active object segmentation

5.2 Visuals generated by the components of the player

We tested the player, on Windows 2000 platform (with Pentium III, 64 MB RAM), for the lecture contents available on the web server. The inputs of the player are the low bit-rate coded AVI file that represents the instructor's audio-visuals, the .ppt file (made using the Microsoft Powerpoint) and a text file namely PlayerRef.txt that contains the display parameters generated during the content-development.

The size of the executable file for the player is around 100 KB. For the lecture of 4 minutes duration, the actual playback took around 6 minutes including 1 minute to download and process the presentation file of size 103 KB and around 1 minute to download the audio-video of the instructor, in the form of AVI file of size 3.88

Figure 5.5: Outcome of the chroma-key screening

MB. The total 5 presentation slides, when processed, took around 2 MB of the local storage.

Following are the visual outcome, represented in the form of one video frame of the lecture, generated by various components of the player.

- Figure 5.6 shows the decoded video frame of size 480x360 pixels, representing the instructor's body in front of the chroma-key background (in this case, green color shown in gray variation). The frame is generated by Indeo 4.1 video decoder. It serves as the input to the CBDM.

- Figure 5.7 shows the slide image of size 720x540 pixels. It is the outcome of the SE, saved as 256-color bitmap image, and serves as the input to the CBDM.

- Figure 5.8 shows the display window of size 805x540 pixels, showing one visual frame of the lecture displayed by the player. The display window shows the instructor's visuals superimposed on the slide. The visuals are generated by the CBDM by making use of the display parameters.

The player also plays the telephone quality voice of the instructor synchronized with the visuals of the lecture.

5.3 Analysis of the results

5.3.1 Achievements

Following is the list of achievements confirmed by the test. The test has been undertaken for the sample lecture of 4 minutes duration.

- **Low download time:** The low bit-rate encoding (i.e. high compression) of the instructor's audio-video, applied during the content-development, drastically reduces the download-time of the lecture contents over the existing bandwidth of the communication channels.
- **Excellent quality of the slides:** On the remote student's desktop, the original quality of the slides is maintained because of the separate downloading of the presentation file (.ppt) and decoding it to the slide images on the local storage. Thus the quality of the slides is independent of the (possibly poor) lighting conditions during the actual lecture event.
- **Realistic display mixing:** The CBDM in the player realistically mixes the two visual entities of a lecture, i.e., the slides and the instructor's body movements overlapping the slides.
- **Accurate synchronization of the presentation and the instructor's talk:** The RDX technology, used in the implementation of the player, accurately synchronizes the running presentation and the instructor's audio-video by using its 'timer and events' mechanism.

5.3.2 Limitations

The limitations of the current work are mainly due to the limitations of the technologies used in the current implementation.

- The decoded video frame representing the instructor's body shows some blocking artifacts at the boundary of the instructor's body, thus showing the chroma-key (in this case, green color), instead of smooth edge, at the boundary. It is

clear from the figure 5.8. This may be because of the wavelet transform used in Indeo 4.1 codec. To solve this issue, we suggest to use the sophisticated version of the codec namely Indeo Video 5.11 [8].

- Since the frame rate of the instructor's video is kept at 5 fps, the instructor's body movements, in uncompressed video, are jerky as compared to the original body movements in MPEG-1 movie of the lecture.

- Since Indeo 4.1 video encoder, used for the content-development, applies "lossy" compression algorithm for achieving low bit-rate (64 kbps), the facial expressions of the instructor are unclear on the student's desktop.

- The voice of the instructor appears different due to the telephonic quality of the voice at 64 kbps. With the audio compression, better quality voice can be integrated with the lecture content and the voice can be presented in a better quality.

- The player and the current scheme for the content-development execute only on the Win32 platform. This is because the audio-video processing tools, namely VideoMach, AMOS, Indeo 4.1 and the Intel's RDX technology, used in the implementation, are available only for the Win32 platform.

- Current implementation doesn't make use of the web server streaming. In other words, the low bit-rate audio-video of the instructor is downloaded on the remote student's machine before the actual start-up of the playback. Hence the issue of the quick start-up of the playback remains unresolved.

5.4 Future work

5.4.1 Future extension based on the current framework

A major modification proposed is to integrate the streaming media technology. The content-developer needs to develop the instructor's audio-video contents in the streaming format and the player needs to support that format.

Modifications in the content-development scheme

The major change in the content-development scheme is the use of Indeo 5.11 video codec in place of Indeo 4.1 video codec for low bit-rate encoding of the instructor's video. Indeo 5.11 shows performance enhancements over Indeo 4.1 because of the following features.

1. A new wavelet compression algorithm improves the visual quality of the decoded frames.

2. Interpolated playback, on Pentium class processors with MMX technology, allows smoother playback even with low frame rate.

3. Scalability of the video allows progressive download.

These features eliminate some of the limitations of the current implementation. With the same bit-rate, output of Indeo 5.11 decoder is less jerky and clearer.

Further the synchronized low bit-rate audio-video of the instructor is converted to the progressive playback format using Ligos' Indeo Video 5.11 Progressive Download Publisher tool. The tool takes Indeo 5.11 AVI file as input and generates an IVF (Indeo Video Format) file which can be streamed through the player. The tool uses two options, namely the progressive quality and the progressive frame rate, to create the progressive download file structure of the output IVF file.

Modifications in the implementation of the player

The player needs to be modified to support the Indeo 5.11 progressive playback format. We propose to develop a filter graph explicitly, using the DirectShow (i.e. ActiveMovie 2.0) COM interface.

In order to attach the IVF file, representing the instructor's audio-video, to the RDX surface, we propose to handle the RDX AM object in three steps.

- **Building:** In this step, the developer needs to create the filter graph. The pictorial representation of the filter graph is shown in the figure 5.9. The filter graph consists of four filters.

1. The source filter represents the URL of the IVF file containing the audio-video of the instructor.
2. The transforming filter is the Indeo 5.11 decompressor filter provided by the DirectShow SDK.
3. The renderer filter is the Intel RDX renderer filter.
4. The device filter required to play the audio is the DirectSound Device filter.

The RDX renderer filter is then associated with the RDX AM object.

- **Managing:** In this step, the associated RDX AM object can be manipulated. In this case, the transparent color in RDX AM video is set to the chroma-key, obtained from the display parameters. (Indeo 5.11 supports chroma-key based transparency.)
- **Rendering:** The RDX AM object can be played on the screen by sending the 'play' command to the filter graph.

5.4.2 Other approaches

Some work can be done to integrate the CELP-based low bit-rate audio coding and the low bit-rate video coding based on watershed segmentation and control point tracking algorithm discussed in the section 1.4.

5.5 Concluding remarks

In the current work, we put forth a framework for real-time distance education using web server streaming. We propose an approach in which the lecture contents are developed by applying high compression technique on the audio-visuals of lecture movie. The dedicated player downloads the lecture contents from a web server and plays the lecture on the remote student's desktop. The implementation is based on the existing Win32 based technologies such as Intel's RDX technology, ActiveMovie,

Indeo Video codecs, AMOS video segmentation software, Powerpoint COM object library etc. We tested the implementation using a digital movie of a classroom lecture and investigated the limitations of the current work. To remove some of the limitations, we proposed an extension of the current work, based on the Microsoft's DirectShow (i.e., ActiveMovie 2.0) filter graph architecture. We conclude to note that the integration of the streaming media technology and Intel's RDX technology can be the solution in order to fully realize our approach.

Figure 5.6: Outcome of the video decoder, i.e., a decoded frame

Figure 5.7: Outcome of the slides extractor

Figure 5.8: Outcome, of the CBDM, displayed by the player

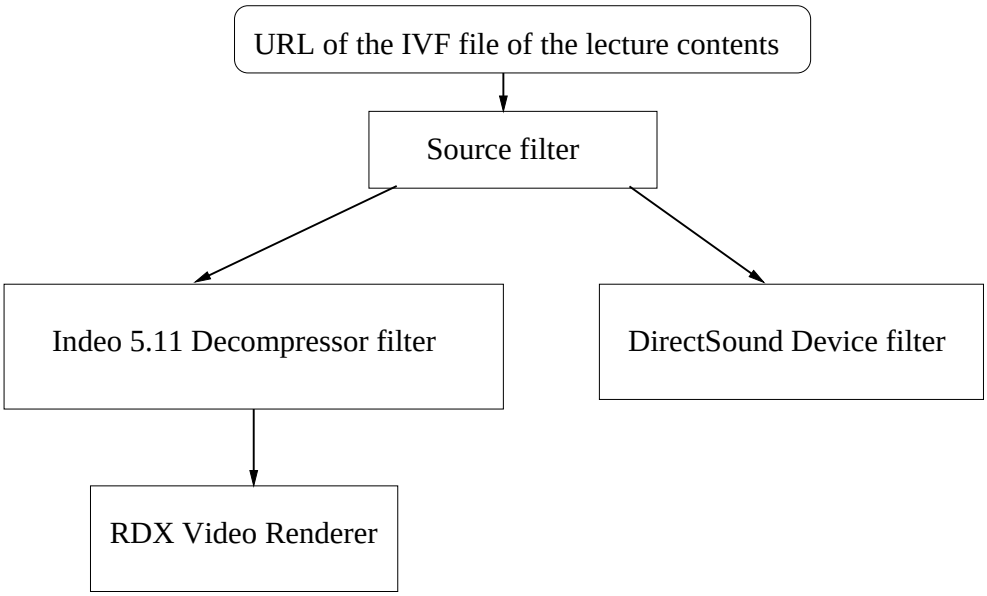


Figure 5.9: Filter graph for rendering progressive download Indeo 5.11 video on RDX surface

Appendix A

Software tools used

- **IrfanView:** IrfanView [5] is a fast freeware image viewer/converter for Win32. Some of the supported file formats are BMP/DIB, GIF, JPEG, PBM/PGM/PPM etc.
- **VideoMach:** VideoMach [19] is a powerful audio/video builder and converter. It can be used to build video clips from still images, to extract audio tracks and pictures from movies or just to convert media clips from one file format to another. It can also be used to change frame rate, frame size, color depth and other properties of a media clip.

Bibliography

[1] B.Prabhakar. "Low bit-rate video coding using watershed segmentation and control point tracking". Master's thesis, EE, I.I.T. Kanpur, Feb 2000.

[2] Fabian Meier. "Current issues in Streaming Media", Jan 2001. http://streamingmedialand.com/issues.html.

[3] H.Schulzrinne, A.Rao, R. "Real-Time Streaming Protocol(RTSP) RFC:2326", Apr 1998.

[4] Intel Architecture Labs. "Intel's Realistic Display miXer", 1997. http://developer.intel.com/ial/rdx/index.htm.

[5] IrfanView Homepage. http://www.irfanview.com A fast freeware image viewer/converter for Win32.

[6] John F. McGowan. "AVI Overview", 1999. http://www.jmcgowan.com/avi.html.

[7] K.Aizawa, T. "Model-based Image Coding: Advanced Video Coding Techniques for Very Low Bit-rate Applications". *Proceedings of The IEEE 83*, 2 (Feb 1995), 259–271.

[8] Ligos Corporation. "Ligos Indeo Video Codecs". http://www.ligos.com/index.phtml.

[9] Microsoft. *"Visual Basic Programmer's Guide: Microsoft Powerpoint Objects"*. MSDN (Microsoft Developer Network) Library Visual Studio 6.0a.

[10] Microsoft. "NetShow Theater Server", Oct 1998. http://www.microsoft.com/theater/.

[11] Microsoft. "Streaming methods: Web Server and Streaming Media Server", Feb 2000. http://www.microsoft.com/windows/windowsmedia/en/compare/-webservvstreamserv.asp.

[12] Microsoft Corporation. "DirectShow". Microsoft DirectX 8.0: Help.

[13] Microsoft Corporation. "What is DirectX?". Microsoft Windows 2000 Server: Help.

[14] M.Kass, A.Witkin, D. "Snakes: Active contour models". *International Journal of Computer Vision* (1988), 321–331.

[15] P.Westerink L.Amini S.Veliah, W. "A live intranet distance learning system using MPEG-4 over RTP/RTSP". *IEEE* (2000), 601–604.

[16] T.Berners Lee, R.Fielding, H. "HyperText Transfer Protocol/1.0 RFC:1945", May 1996.

[17] Ted Netterfield. "The Building of a Virtual Lecture Hall: Netcasting at the University of South Florida". *Proceedings of CAUSS/EFFECT 22*, 2 (1999).

[18] T.K.Capin, E.Petajan, J. O. "Very Low Bitrate Coding of Virtual Human Animation in MPEG-4". *IEEE* (2000), 1107–1110.

[19] "VideoMach Home Page". http://www.videomach.com/VideoMach.html.

[20] Wallace, G. K. "The JPEG Still Picture Compression Standard". *Communications of the ACM 34*, 4 (Apr 1991), 30–44.

[21] Zhong, and S.F.Chang. "AMOS: An Active System for MPEG-4 Video Object Segmentation". *IEEE International Conference on Image Processing* (Oct 1998).