Processing Digital Television Streams

by

Eng Keat Khor

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Author [illegible] ....
Department of Electrical Engineering and Computer Science
May 16, 1995

Certified by [illegible] ....
Andrew Lippman
Associate Director, MIT Media Laboratory
Thesis Supervisor

Accepted by [illegible] ....
Frederic. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Processing Digital Television Streams

by

Eng Keat Khor

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 1995, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, I designed and implemented a program which simulates a television receiver on a workstation. The television receiver is connected to a network called the Media Bank. The Media Bank is a distributed database which is being developed at the MIT Media Laboratory. The program is written in C and runs on the OSF3.2 UNIX operating system. This software package simulates a future television environment where home users are able to browse through information with an easy-to-use interface. The data comes from media servers, news servers, weather servers, content owners, service providers, and the Media Bank. The television receiver will be equipped with MPEG and JPEG decoder chips and a multi-purpose microprocessor which is used to run this software package. This simulation involves real time delivery of video/audio data and exploits today's digital network to exhibit the potential of the Media Bank as a platform to provide home entertainment and personalized information in the future.

Thesis Supervisor: Andrew Lippman
Title: Associate Director, MIT Media Laboratory

Acknowledgments

First and foremost, I wish to thank my thesis advisor, Andrew Lippman at the MIT Media Laboratory, for his guidance and support throughout this study.

I would also like to thank the faculty, staff, and students in the garden for providing a positive working atmosphere. In particular, I am indebted to Henry Holtzman, who gave me the opportunity to work and learn from him as a UROP student.

Special appreciation also to the many people who provided technical assistance: Klee Dienes, Daniel Gruhl, Michelle McDonald, Constantine Sapuntzakis (author of DispImg widget), and Theodore Tonchev.

The support and understanding from my friends at MIT, in particular from Marilyn Chen, Leo Chun, Prodyut Dutt, Frank Kao, Yee Chuan Koh, Yulan Liao, and Eva Tsai, have encouraged me to stay focused and motivated in my work.

I wish to thank my family for their continued support over these last few months.

Last but not least, I dedicate this thesis to the loving memories of my sister and my grandmother.

Contents

List of Figures

Chapter 1

Introduction

Construction of the information superhighway has already begun. Telephone companies are spending billions of dollars to lay down new fiber optic cables while cable companies are rushing to upgrade their coaxial cables. In addition to these technological advancements, the government is relaxing rules on how companies can compete in this industry. As a result, we are living in an era where major changes are occurring in how information is brought into our living room.

At present, we treat televisions, VCRs, CD players, newspapers, and personal computers (PCs) as separate entities. In the future, with technological advancements, these entities will merge into a single system. We may be able to watch movies, obtain weather forecasts, and read newspapers — all on demand. Furthermore, the convenience of scanning stock prices, reading personalized news and electronic mail, and shopping for food or clothing will be available at our fingertips.[1] This means we will be experiencing a major change in how information is obtained: we do not have to watch what is fed by the cable or newspaper company.

The concept of a Media Bank fits into this new environment. The Media Bank is a generalization of a distributed media database which enables applications to store and access media objects on demand. The infrastructure available is assumed to have variable bandwidth, distributed and diverse channels [1]. The Media Bank is

[1]For example, in the context of a TV controlled from a remote.

the main source of digital data and is currently a research topic at the MIT Media Laboratory. The application built for this thesis is used to demonstrate a potential interface between home users and a Media Bank.

The simulation work described herein can be supported by workstations, personal computers or programmable settop boxes. These equipment include a few decompression chips (eg. Motion JPEG[2] or MJPG) and a general microprocessor. The decompression chips are used to decode bitstreams where the compression format must adhere to international standards. The decoded images are then displayed in real time. The general microprocessor runs a downloaded software which interfaces with the human user.

The progress in the digital realm is aided by the availability of compression standards. For example, the Moving Pictures Expert Group (MPEG) has taken steps towards significant cost reduction in the development of telecommunication applications such as teleconferencing, digital broadcast codec, and video telephony through standardization of video and audio compression technology. Today, the industry and the academia can concentrate on the development of such applications without having to worry about incompatibility among various MPEG products. As a result, analog applications are rapidly being transformed into digital applications.

1.1 Thesis Overview

This thesis demonstrates the capability of acquiring a movie on demand from the Media Bank. In addition, the program upon which this thesis is based simulates a powerful television receiver capable of by-passing news stations to obtain the most recent weather forecasts. Together with the basic functions of a television receiver and a VCR, it is envisioned that home users can obtain most services on demand, under the roof of one system: the augmented television receiver.

Chapter 2 introduces the infrastructure of the Media Bank and its role in the MIT Media Lab. This section also describes the characteristics of the Media Bank.

[2] Joint Photographic Expert Group

Chapter 3 describes the specification of the Media Bank and the implementation of the augmented television receiver. Data structures and important modules in the implementation are discussed.

Chapter 4 describes the layout and user interface of the augmented television receiver. It maps out in detail how to operate the television receiver from the stand point of a home user. Examples from the simulation are illustrated.

Chapter 5 tabulates the results of this project. It describes system integration, functionality, and performance of the augmented television receiver. The details of additional functionality such as scene segmentation are also discussed.

Chapter 6 discusses future work and possible improvements to the simulation program.

Chapter 2

Context

The traditional home entertainment paradigm involves watching television programs from station broadcasts, renting a movie (VHS tape or laser disc) from a local rental store, listening to compact discs on a CD player, playing video games on a game machine, reading daily news from a local gazette, checking electronic mail from a personal computer, and shopping at stores. This paradigm is rapidly changing with the advent of digital telecommunication. Today we can read news from the World Wide Web, play video games, order pizzas, and book flights using personal computers which have access to the Internet. With the development of the National Information Infrastructure (NII) under way, we can obtain all these and more under one system, be it a personal computer or a television receiver with a sophisticated settop box, in the near future.

A new home entertainment paradigm is to make use of various communication media to deliver a product when requested. For example, a movie is produced from video sequences delivered via satellite, audio data downloaded via telephone lines, and commercials inserted[1] by a local supermarket [2]. With this application in mind, the idea of a Media Bank is conceived at the MIT Media Laboratory and has been the major research vehicle to test the distributed media storage/access environment.

[1]A movie is made up of many segments and is not put together until demanded. Based on this model, firms are able to insert commercials in between segments before the movie reaches home users.

2.1 The Media Bank

The Media Bank satisfies the demands of many data intensive applications such as video on demand, video teleconferencing, and personalized newspapers. It is built on the interconnected web of fiber optics, coaxial cables, and telephone lines in conjunction with devices and switches such as ATM, satellites and cellular phone. The Media Bank is also an information powerhouse, where huge amount of data are stored away in many access/retrieval points in the MIT Media Laboratory.

The theme of the Media Bank focuses on these four topics [3]:

- Cross network operation (2.1.1),
- Dispersed services (2.1.2),
- Access to primitives instead of programs (2.1.3), and
- Programs on demand (2.1.4).

These characteristics enable the Media Bank to be flexible enough to accommodate new as well as old applications on existing network. Entertainment television, classroom communications, purchasing, accessing medical records, paying tolls or using credit cards are all potential applications. Furthermore, the Media Bank is ready for the NII[2] which encompasses the bitways, the services, and the applications that will support commerce, healthcare administration, and entertainment [4].

2.1.1 Cross Network

The Media Bank will grow such that it can be accessed throughout the MIT Media Laboratory and may be extended to the non-laboratory community. Fiber optics, coaxial cables, telephone lines, and ATM switches within the laboratory are currently used to demonstrate the cross network capability. The quality or format of the data which is accessed will vary with the bandwidth of the delivery medium from which it is obtained. For example, a high quality video sequence with a high bitrate can be delivered through fiber optics as opposed to the low quality video sequences delivered

[2]It is important to point out that the NII is not just a highway but a complete transportation system

through telephone lines.

2.1.2 Dispersed services

The Media Bank can support a variety of services ranging from newswire feed to real time video/audio delivery on demand. This requires many servers and services such as authentication servers and audio servers to support the needs of a home user. These supporting services as well as client applications are distributed throughout the laboratory. A unique list is needed to indicate the services or elements that are needed for a certain program demanded by the client program. This list or elements can be stored in one machine or in many.

2.1.3 Access to primitives instead of programs

The primitives are media-objects which describe the image and the coded representation of the frame. With these primitives, an application program can reconstruct the video sequence depending on some parameters such as bandwidth, screen space, and image size [1]. As a result, the video sequence can be displayed full frame in a cinema or cropped and centered on the actors and actresses in a small, portable television where the screen size and resolution are limited.

2.1.4 Programs on demand

Movies do not exist until demanded by the home user. A requested movie is formed when primitives are assembled while the movie is being played concurrently. Video and audio can be rendered through coaxial cables from one location, commercials downloaded before the beginning of the movie from a second location and billing and authentication performed on a third. Essentially, the movie is packaged together only when demanded [1]. Other products can also be generated in a similar manner.

2.2 Media Servers

The distributed Media Bank can be accessed through media servers. Media servers are programs dedicated to organizing, storing, managing, and distributing multimedia content such as text, audio and video. These servers are installed in many machines in the MIT Media Laboratory. Generally, they contain information about movie objects/segments which are stored locally. For specific queries, for example, they can provide us with information about the categories of available movies (eg. action, horror, and romance) and their ratings (eg. G, PG, and R). A unique list of movie objects is also available when requested. If not, a new list is generated to suit the needs of the client program.

Besides media servers, there are other servers such as news servers and weather servers. These servers provide services to client programs (home users) which interact with the Media Bank. The news server provides personalized news to home users. It recognizes the user, searches through the database for the user's interests, processes the request, and distributes news article of interests to the user. It must be emphasized that the personalized newspaper contains not only text and images but also video clips from TV stations, movie previews from local cinemas, and up-to-the-minute stock prices.

The weather server provides weather forecasts for any major city in the world. A potential application may access weather information for a traveller's itinerary. The information gathered will list weather forecasts for the cities outlined in the itinerary. The traveler can also monitor traffic and scheduled flight information from other service providers. These services can be bundled effortlessly by the user.

Chapter 3

Specification

The focus of this thesis is to simulate the capabilities of television receivers as an interface between home users and the Media Bank. The simulation work is done on the DEC Alpha running OSF3.2 (Open Software Foundation). This simulation involves processing live audio/video feed and accessing/storing media objects in the Media Bank. In addition, the simulated television receiver is capable of accessing/displaying weather information/forecast from a weather server run by the Center for Meteorology and Physical Oceanography.

3.1 Media Bank layers

The application program will interface mainly with media servers in order to access objects in the Media Bank. These media servers contain all the information regarding the objects. The information is stored in different layers — the content layer, the format layer, and the transport layer [1].

3.1.1 Content layer

The content layer is used to distinguish the different uses of the media objects. For example, it distinguishes between movies and their different ratings (G, PG, R, etc.) home shopping, stock prices, and weather forecasts. When a new application is needed, the new characteristics can be added to the content layer. The information can be gathered by a client program from the name server. This server contains information such as the location of the requested media objects, the size of these objects, and the quality of the objects. The media servers can then be queried with object names, object formats, and/or object qualities.

3.1.2 Format layer

The format layer encapsulates many of the standards available (MJPG, MPEG, DAT, TIFF, etc.). With this configuration, the application program can be written on various platforms. The open environment is highly desirable because this strategy avoids dependency on any particular vendor and is also transparent to the user.

3.1.3 Transport layer

Finally, the transport layer supports the format layer. The NFS is chosen as the backbone of data movement around the network. Relying on NFS has its advantages because we can use existing hardware and software systems. Current protocols such as IP/TCP running on fiber interconnected through ATM switches will be used [1]. As a result, it eliminates the necessity of building a separate transport system which requires considerable amount of work.

Figure 3-1 below is the module dependency diagram for the discussion above.

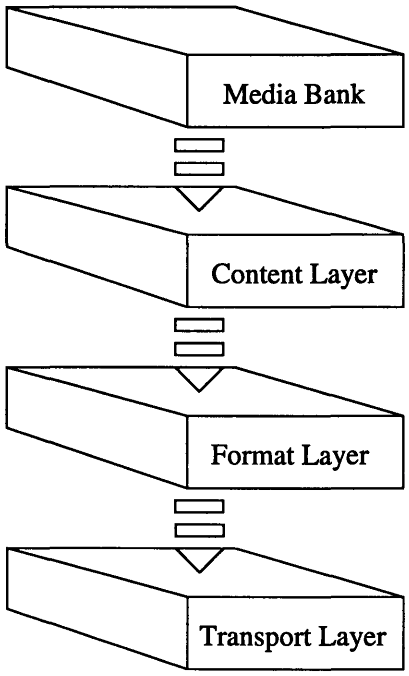


Figure 3-1: Module Dependency Diagram for Layers in the Media Bank

3.2 A network example

In addition to the modules above, the network is interconnected with content providers, service providers, and home users. Figure 3-2 below illustrates this network:

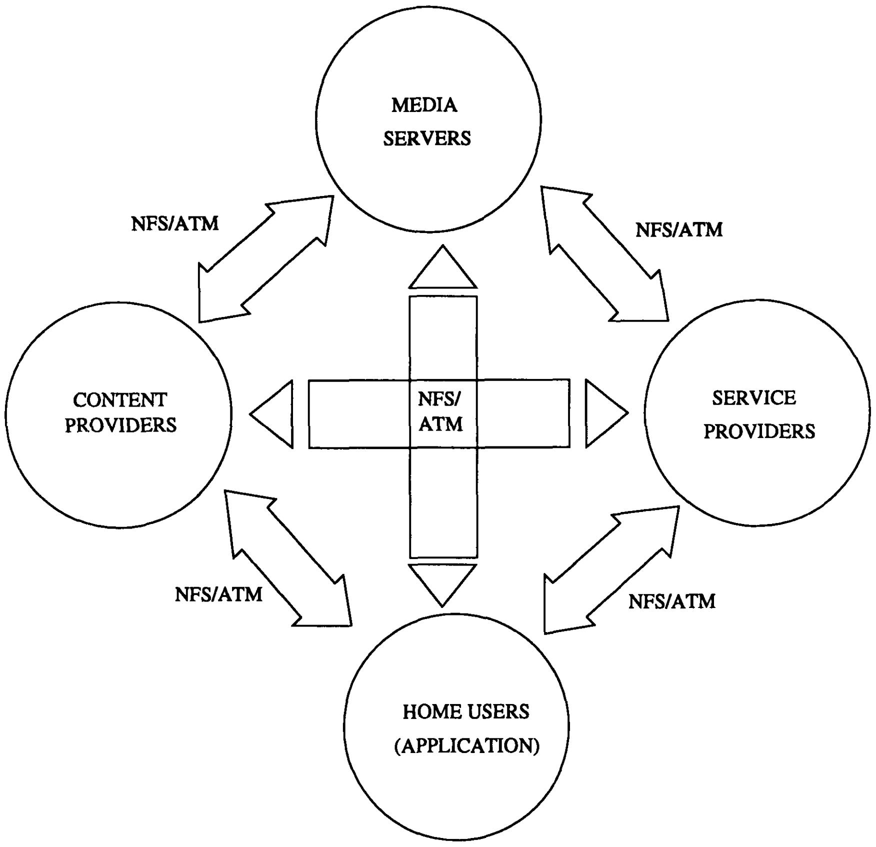


Figure 3-2: Network connection among various users/providers

Content providers gather information in a raw format and are the main source of digital information. For example, the weather observatory station in Michigan gathers weather information and makes forecasts which are made available through a weather server. The information provided by this server is not thoroughly processed.

The service providers or the home users must then process the raw data gathered from the content providers. Service providers such as newspaper companies and news stations provide value added information to the raw data and distribute their services to the home users.

In the network described above, the home users can directly access the raw data from the content providers. This is what the application program of this project demonstrates. This program pools current weather information from the Center for Meteorology and Physical Oceanography. Up-to-date hourly weathers report of all major cities in the United States are instantly available.

3.3 The Television Receiver

XToolkit and Motif are used to build the application program which binds all the various applications together, chosen because of their ease of use. When functionality must be added to the application program, a widget (an extra module) can be built and inserted into the main program without changing much of the original code.

The application built for this thesis models a TV and a VCR on an Alpha workstation which is connected to a network. This home entertainment unit of tomorrow enables the user to watch TV and/or video on demand at the same time, to read weather forecasts, to shop for food, clothes, jewelry, etc., all at the convenience of a remote control. The remote control is modeled as a mouse with user preferences inputted as mouse clicks. The simplicity of the user interface should appeal to the home user.

A widget is written for each of the application models discussed above, ie. there is a TV widget, a VCR widget, and a few movie widgets which are capable of decoding bitstreams of different formats (eg. MPEG vs. MJPG). The TV widget is capable of capturing raw video data off the coaxial cable and buffering the digitized video/audio data before displaying the images on the screen in real time. Furthermore, a second buffer can be used to encode the digitized images in JPEG if the user wishes to record the TV segment. The TV widget is built using Multimedia Services for DEC OSF/1 AXP, Digital's standard Application Programming Interface (API) to multimedia functionality on the DEC 3000 series Alpha AXP workstations running the DEC OSF/1 AXP operating system. The API provides software support for the built-in audio hardware on TURBOchannel Alpha AXP workstations and the Sound and Motion J300 option module [5].

The VCR widget is also built using the Multimedia Services for DEC OSF/1 AXP. It differs from the TV widget in its ability to capture video/audio real time data. However, it is capable of synchronous audio and video playback. In addition, it provides the user with the conventional VCR functions of fast forward, pause, etc. These functionality are supported by BufImg widget. The display capability is supported by DispImg widget. The video format is JPEG and audio format is MULAW. The VCR widget and the TV widget capture and playback video/audio using the same format.

There are two movie widgets. The first movie widget is capable of decoding MJPG bitstreams and displaying video images in real time. The second movie widget decodes MPEG (MPEG-1 and MPEG-2) bitstreams. The MPEG movie widget displays video sequences in a slower rate because the decoding is done in software. The MPEG code is copyrighted by the MPEG Software Simulation Group. The modularity of writing widgets is advantageous because the slow running MPEG software decoder can be replaced with a MPEG hardware decoder when it is available. The MJPG movie widget uses the Multimedia Services for DEC OSF/1 AXP and thus is capable of delivering synchronous audio/video data. The MPEG widget however can only deliver video images.

Figure 3-3 below illustrates the widget dependencies in the discussion above.

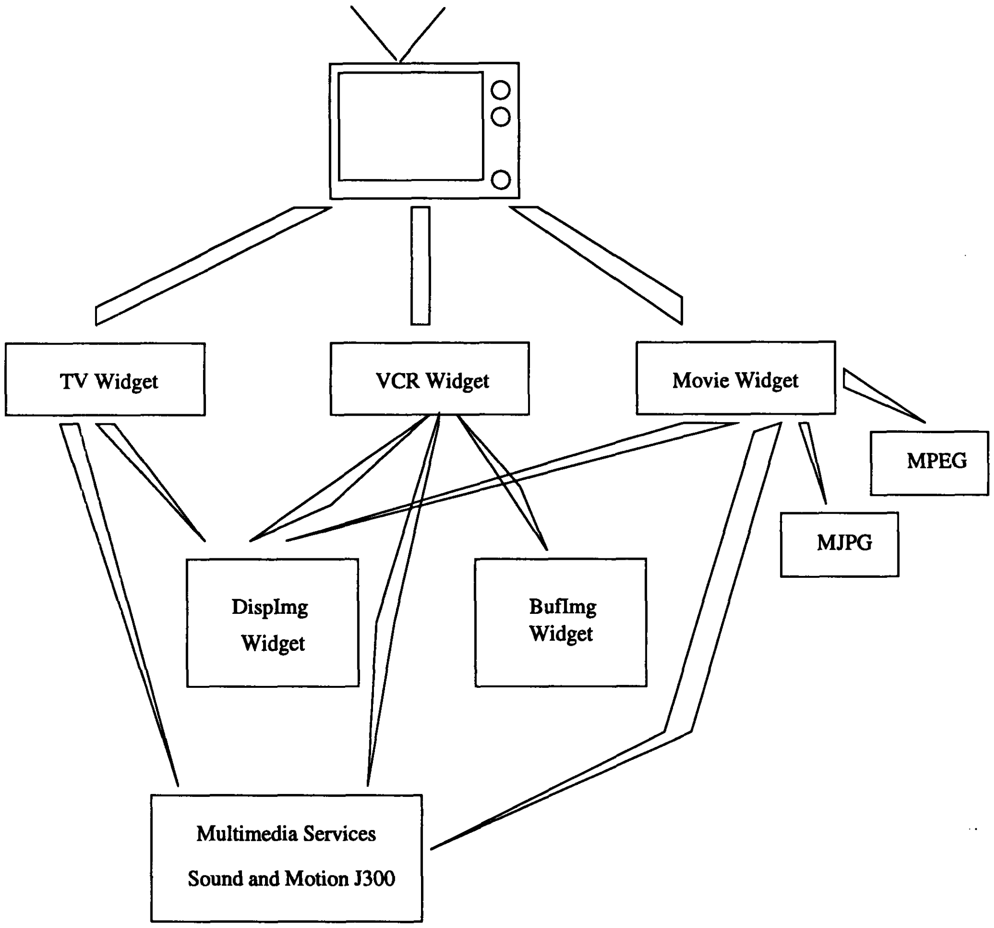


Figure 3-3: Module Dependency Diagram for the television receiver

3.3.1 The TV widget

The TV widget inherits from DispImg widget which is essentially a display widget. It has a function called *DispImgSetImage*[1] which accepts a widget and an x-image as arguments. When *DispImgSetImage* is used with proper arguments, the DispImg display widget will place the x-image at the appropriate screen.

The function *TVInitialize* is called when the TV widget is created. *TVInitialize* sets up both the video and audio devices of the Multimedia Services for recording and playback. A default colormap is used and the corresponding colors are allocated and merged with the J300 device palette. The stream buffer and image buffer are initialized. A maximum of four image buffers are queued for display after decoding the incoming MJPG bitstream.

The following functions can be accessed from the application program: *TVLoop* and *TVQuit*. *TVLoop* is the main function of the TV widget. As the name suggests, it controls the loop of the main program as long as the state remains as TVState. This loop occurs when *TVLoop* adds itself to the application queue whenever the state is TVState. When a new event arrives and the state changes to another state, the loop exits (ie. *TVLoop* is no longer added to the application queue) and control is transfered to another widget.

Within the *TVLoop*, there are a few states which allows for different behavior. One such state is the Recording state which captures video and audio bits into the stream buffer. When the stream buffer is full, the bits are flushed into the hard disk. The information captured are recorded into seven different files. They are discussed in Section 3.3.3.

The *TVLoop* function also keeps track of the time taken to display all the decoded frames. This is performed in order to obtain frame rate information at the end of the program.

TVQuit is called by the application program during exit. Before the program exits, the devices of the Multimedia Services are reset and all allocated memory are

[1]Words in italic are function names.

freed. The stream buffer and image buffer are closed. Shared memory used for display are detached from the display screen. Frame rate is computed and results are tallied before the program exits.

The TV widget has the following resources: *filename*[2], *recording*, *end_recording*, *brightness*, and *contrast*. These resources are used to affect the widget's behavior. The widget records live TV onto the tape using the given *filename*. *Recording* is set to TRUE when the user requests to record live TV. *End_recording* is set to TRUE when the user requests to end recording. Hence, the TV widget also behaves like a VCR in this respect. The *brightness* and *contrast* levels are set according to inputs from the user.

The TV widget also allows for a callback function which display the current status of the display. This status is either "Live TV" or "Recording" and it may interchange depending on the user input.

3.3.2 The VCR widget

The VCR widget inherits from BufImg widget. BufImg widget is a general vehicle to display video sequences. It accepts pointer to functions for the following functionality: *initialization*, *current_image*, *next_image*, *previous_image*, and *seek_to_N_image*. It also has the following actions: play forward, play backward, fast forward, fast reverse, stop, pause, one frame forward, and one frame backward. These actions can be accessed by the user through callbacks in the application program.

In addition to the above, BufImg widget also has the following resources: *maximum_frame_number*, *current_frame_number*, *play_state*, and *state_callback*. *Maximum_frame_number* is used to detect the end of the movie segment during playback so that the movie continues at the beginning of the movie. *Play_state* is used to inform the widget which state it is in. *State_callback* is used to display the current play state at the VCR panel (eg. Play, Pause, etc.).

A callback function is needed to inform the main program of the current playback

[2]Widget resources are also in italic.

state of the VCR. This function is called in the control loops of the BufImg widget. This callback function will be ignored if it is not registered with the main program.

Similar to the TV widget, the function *VCRInitialize* is called when the VCR widget is created. *VCRInitialize* sets up both the video and audio devices of the Multimedia Services for video and audio playback. A default colormap is used and the corresponding colors are allocated and merged with the J300 device palette. The stream buffer, audio buffer, and image buffer are initialized.

Furthermore, the five pointer to functions and the filename are registered with BufImg as shown in the *MjpgImgInitialize* function below:

```
static void
MjpgImgInitialize
 (Widget req, Widget cre, ArgList args, Cardinal *num_args)
{
 MjpgImgWidget miw = (MjpgImgWidget) cre;
 MjpgImgPart *mip = (&miw->mjpgimg);

 XtVaSetValues((Widget) miw,
               XtNnextImage, (XtPointer) MjpgImgNextImage,
               XtNprevImage, (XtPointer) MjpgImgPreviousImage,
               XtNcurrImage, (XtPointer) MjpgImgCurrentImage,
               XtNseekImage, (XtPointer) MjpgImgSeekImage,
               XtNinitDecoder, (XtPointer) MjpgImgInit,
               XtNmjpgFilename, (XtPointer) args->value,
               NULL);

 MjpgImgInit ((Widget) miw);
 DispImgSetImage ((DispImgWidget) miw, MjpgImgCurrentImage ((Widget) miw));

}
```

With this set up, a different widget viewer can be written that displays pictures of other formats (eg. MPEG, TIFF, DAT, GIF) as long as the widget has the five functions: *NextImage*, *PrevImage*, *CurrImage*, *SeekImage*, and *Init*. This structure simplifies the integration of the entire program because the play, pause, stop, and other control functions only need to be written once.

Video and audio synchronous playback are controlled by a callback function called *Sync_audio_video*. Audio playback is performed at a normal rate and video frames are displayed with respect to the audio playback. If video frames are displayed faster than audio playback, the VCR widget will redisplay the same frame until audio has caught up to that particular frame. If video frames are displayed slower than audio playback, the VCR widget will drop a few frames in order to catch up with the audio playback. The callback function determines any asynchronous playback by checking the time stamps for video frames and audio blocks respectively. These checks are performed every quarter of a second.

Video playback is accompanied by audio playback only in the play forward VCR state. Audio is turned off in other VCR states in order to conform to the functionality of a real VCR. Audio playback in the reverse mode has been experimented with but without intelligible or coherent results.

Recording from the VCR has not been implemented because this simulation involves only one VCR. If two VCRs are involved, recording can be performed easily by copying the source file to the target file. If only a portion of the source file is copied, the VCR widget can mark the requested video segment and the appropriate segment is then copied to the target file.

3.3.3 File Format for TV and VCR widgets

Each recorded TV segment has the following structure: It is a directory containing seven files — the descriptor file, the video data file, the video header file, the video time stamp file, the audio data file, the audio header file, and the audio time stamp file. Figure 3-4 below gives an example of this structure.

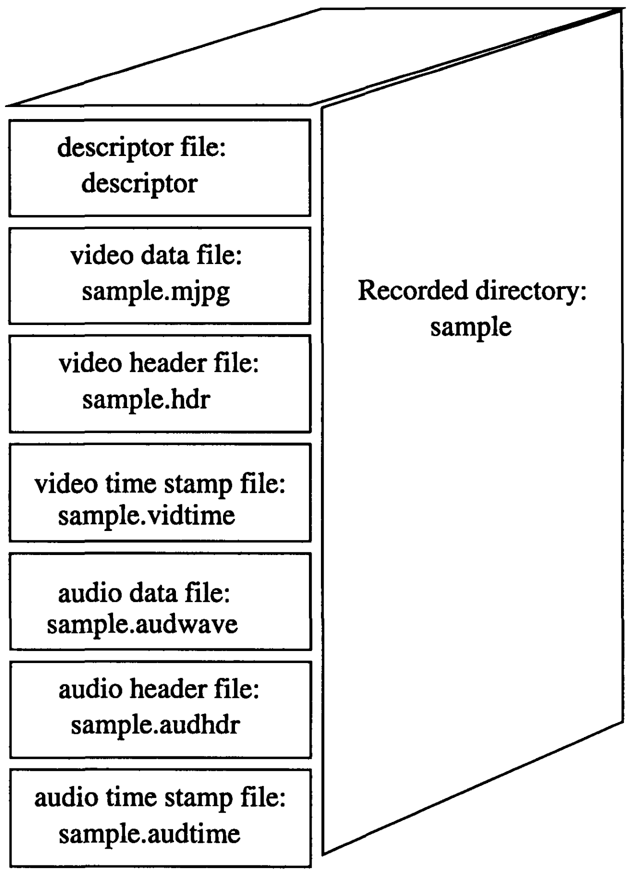


Figure 3-4: Sample file hierarchy for recorded TV segment

The descriptor file contains information about the height, width, number of audio blocks, and maximum number of frames in the video sequence. Two bytes each are used for the storage of height and width of video frame. Four bytes each are used to store audio block and maximum number of frames. If the video sequence is appended onto this directory, the descriptor file is concatenated with a new twelve bytes of information. This new information will be used in place of the old information when this file is accessed again.

The video data file consists of concatenated JPEG frames. The size of JPEG frames varies, for example, a 480x360 size image is approximately 64K bytes. Furthermore, the JPEG frames depends on the video quality — the higher the quality, the more number of bits are required to encode the frame. As a result, the frame size is proportional to the video quality.

Since each JPEG frame size varies, there is a need to store the size information.

This information is stored in the video header file. The sizes of each frame are concatenated into this file where each frame size is represented by four bytes. Therefore, in a sequence, the first four bytes correspond to the first frame, the second four bytes correspond to the second frame, and so on. These frame sizes can also be used to compute the offsets for the video segment. Random access can be achieved since we can choose to display any frame of the video segment.

Furthermore, each JPEG frame is accompanied by a time stamp. The time stamp is a positive integer and is stored in the video time stamp file. The time stamps are four bytes each with the first four bytes corresponding to the first frame, etc. These time stamps are necessary because they are used in conjunction with audio time stamps in order to synchronize video and audio data. When video frames are displayed faster than audio playback, inter-frame display will be slowed down. When video frames are displayed slower than audio playback, frames are dropped so that video can catch up with audio.

Audio data is stored in the audio data file in blocks of 1024 bytes each. These blocks are concatenated in the audio data file. Audio is encoded in MULAW format at the rate of 8000 samples per second.

The size of audio blocks is recorded as four bytes each in the audio header file. Since audio is encoded in blocks of similar sizes, the audio header file is not as useful as the video header file.

Similar to the video time stamp, the audio time stamp is recorded when each audio chunk is recorded. The audio time stamps are also positive integers and they are concatenated corresponding to the number of audio blocks in the audio header file. Each time stamp is represented by four bytes.

Ideally, the components of the display widgets (TV, VCR, and movies) and the file access routines are separated. In this project, the file access and the bitstream buffer system are written so that they are incorporated into the display widgets. The file access component is written in a modular way such that it can be re-implemented without affecting the other components in the display widgets.

3.3.4 The Movie widget

There are two movie widgets to play the two different bitstream formats — the MJPG movie widget and MPEG movie widget. The MJPG movie widget is essentially the VCR widget with the capability of switching from a VCR bitstream to a movie bitstream since both bitstreams are in MJPG format. The VCR widget is discussed in Section 3.3.2.

The MPEG movie widget is based on the software written by the MPEG Software Simulation Group. Some global variables of this software are internalized in order to convert the software into a widget. The functions used by the main program are *MpegViewerInitialize*, *MpegViewerPlayLoop*, *MpegViewerPause*, and *MpegViewerQuit*.

MpegViewerInitialize is called at the creation of the MPEG movie widget and initializes the widget resources. A MPEG stream buffer is set up and the first frame of the default MPEG bitstream is decoded. The image is in CCIR 601 resolution 4:2:2 format and is dithered to eight bit x-image format before display.

The MPEG bitstream does not contain information about the size of each frame. As a result, it is very slow to search for start codes which are used to jump to a particular frame in the bitstream. Hence, only the play and pause functions are implemented. These functions are *MpegViewerPlayLoop* and *MpegViewerPause* respectively. Other control functions such as fast reverse and fast forward are not implemented. If implemented, these functions need to search for proper start codes in the bitstream and thus will make the decoding process even slower. The decoding rate is between 1-5 frames per second, depending on picture size. If the picture size is not 480x360, the DispImg widget will automatically scale it to that size. With this scheme, the quality of the picture is inversely proportional its size.

MpegViewerQuit is called by the application program during exit. The stream buffer and image buffer are closed and all allocated memory are freed. Finally, shared memory used for display are detached from the display screen before exit.

Chapter 4

Layout and User Interface

The layout of this simulation is implemented using Motif. Many of the objects such as scroll bars, push buttons, and labels are provided in the Motif library. Motif is an excellent choice for graphical user interface because of its ease in manipulation of graphics. Furthermore, its event driven programming methodology is suitable for this application. The end result is a simple and easy to use interface.

4.1 General Layout

The microprocessor can be downloaded with an application program which interfaces with the home user. Once the system is switched on, the screen is initialized to the television setup as shown in Figure 4-1 below.

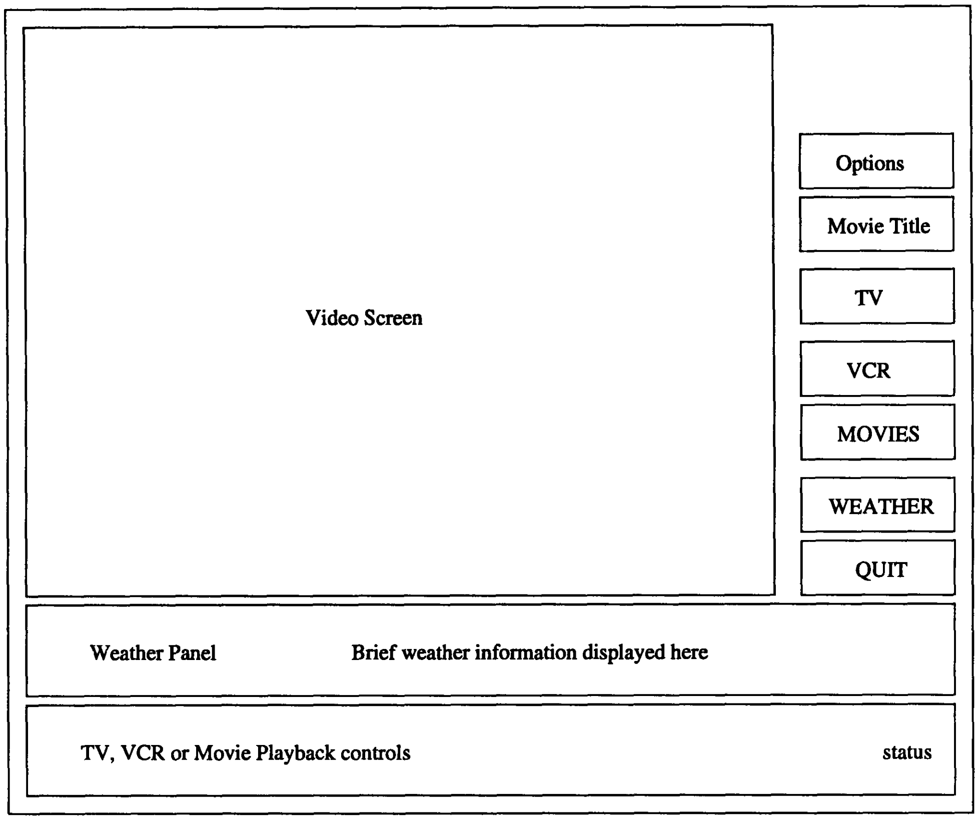

Video Screen
Options
Movie Title
TV
VCR
MOVIES
WEATHER
QUIT
Weather Panel
Brief weather information displayed here
TV, VCR or Movie Playback controls
status

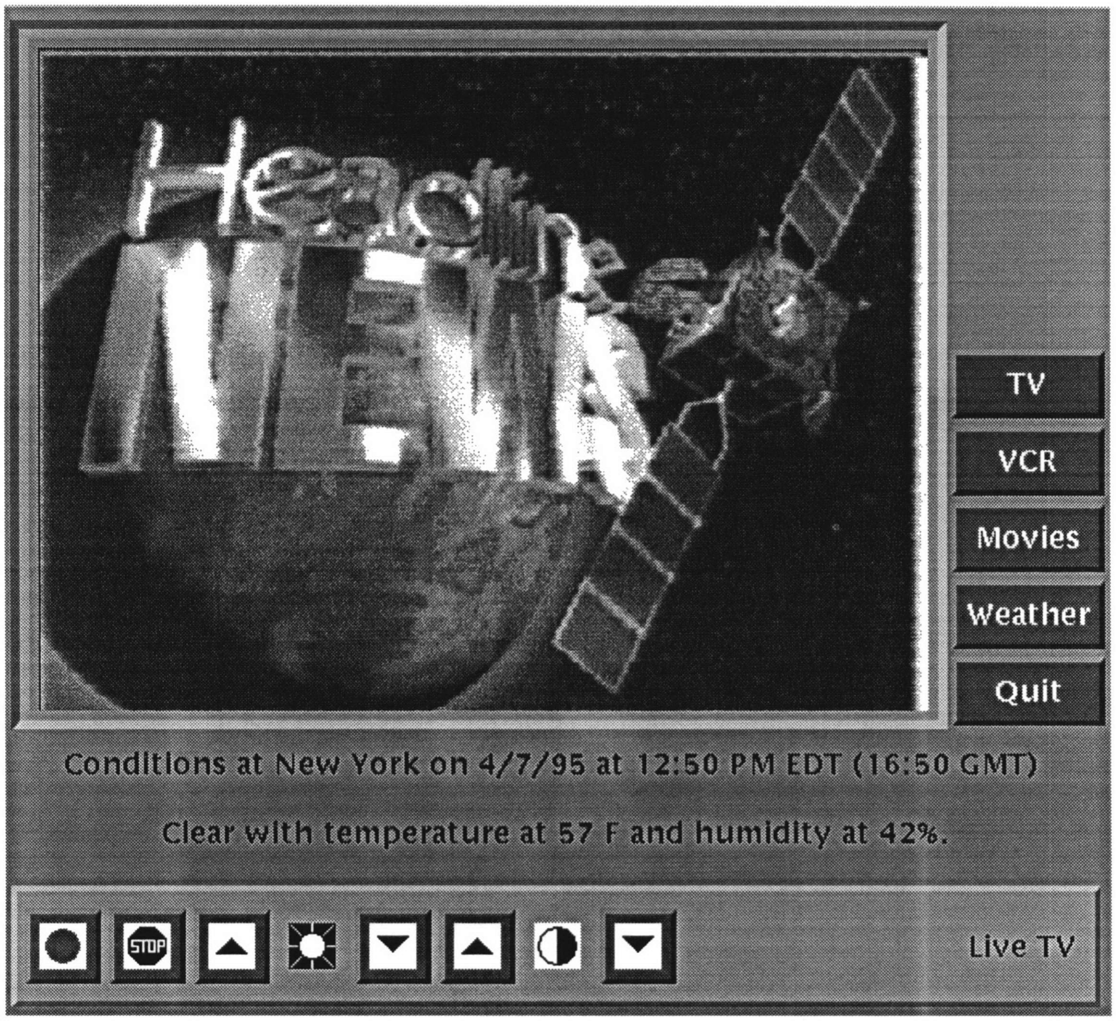
Figure 4-1: Simulated Television Receiver Layout

The mouse is used to model the remote control which has only one button. The user can select any item on the television set simply by clicking the appropriate buttons. The buttons on the right are used to select the viewing mode.

Mouse clicks are chosen as the only interface to the television receiver because it models a remote control with one button. When the user points the remote control towards the television screen, an arrow appears to indicate the corresponding position on the screen. User requests are made when the user depresses the button of the remote control while pointing at the appropriate options on the screen.

4.2 TV viewing mode

The initial viewing mode is TV. Live video feed is played on the video screen. The accompanying TV panel at the bottom of the television layout has record and stop buttons, brightness control, and contrast control. On the bottom right hand corner, the TV status will display “Live TV.” This status will be changed to “Recording” when the user has chosen to record the current video segment. The default values for brightness and contrast are 5000 and 5500 units respectively. They can be incremented or decremented by ten until the values reach the minimum (0) or the maximum (10000). All selection icons are incorporated from bitmaps. A brief weather information on a random city will also be displayed on the weather panel. This weather information are pooled from the weather server. Figure 4-2 below illustrates the initial TV viewing mode with “Live TV” status.


Conditions at New York on 4/7/95 at 12:50 PM EDT (16:50 GMT)

Clear with temperature at 57 F and humidity at 42%.

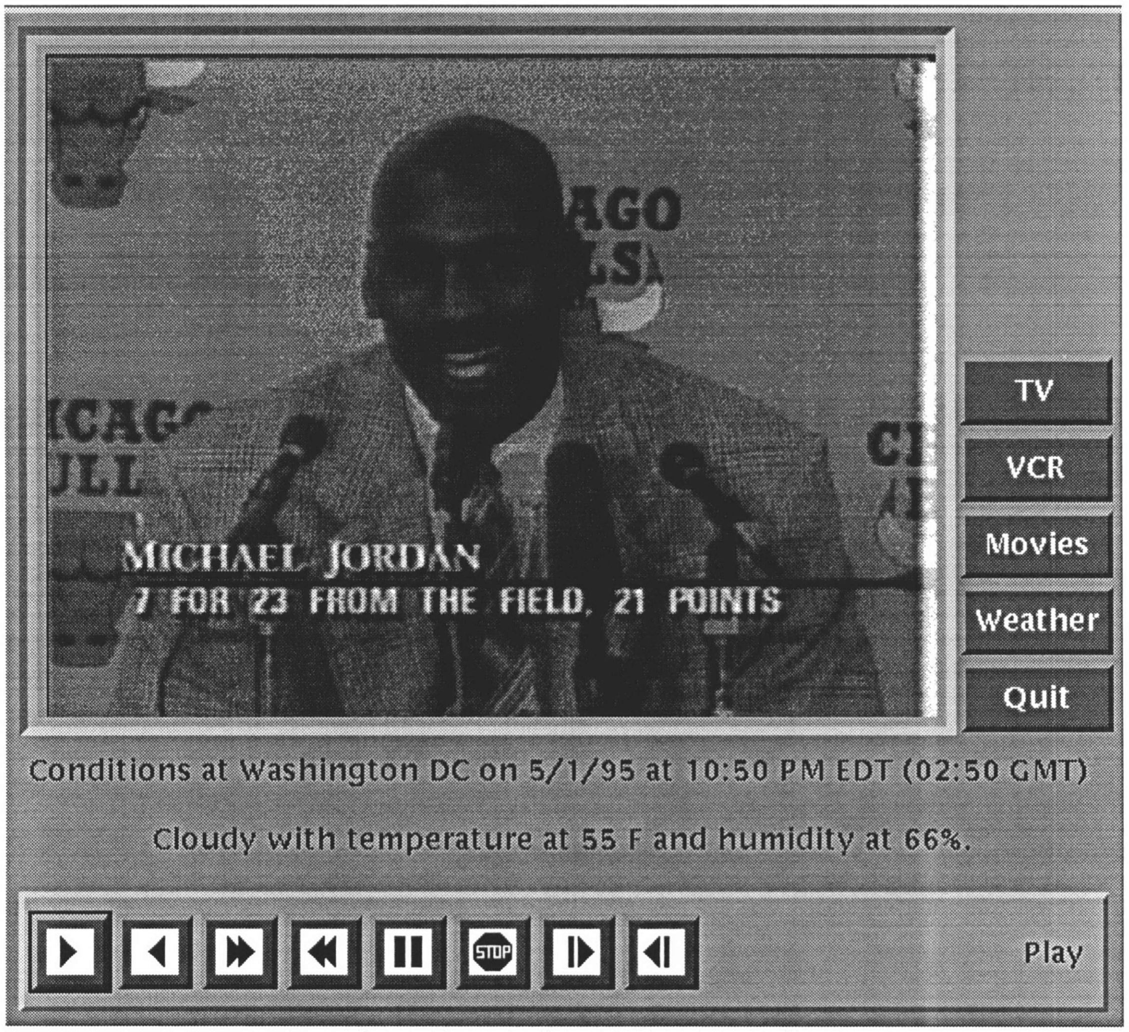
Figure 4-2: TV viewing mode with "Live TV" status

Users are able to move to other viewing modes by clicking the appropriate buttons to the right of the screen. The specific movie icon will not be available for selection before a movie is selected. A new movie can be selected from the movie menu by clicking on the movie button.

4.3 VCR viewing mode

In the VCR viewing mode, the video screen will play the video and audio content of the current VHS tape. The accompanying VCR panel has play, reverse, fast forward, fast reverse, pause, stop, one frame forward, and one frame backward. The VCR status will display one of the modes described.

Figure 4-3 below illustrates the VCR viewing mode with play status.



Conditions at Washington DC on 5/1/95 at 10:50 PM EDT (02:50 GMT)

Cloudy with temperature at 55 F and humidity at 66%.

Figure 4-3: VCR viewing mode with play status

4.4 Movies selection mode

In the Movies viewing mode, the user selects one movie from the media library. The requested movie will be shown momentarily and the current movie title is displayed at the corresponding button on the right side of the television layout.

In this simulation, short and medium length movie sequences are requested from the media servers. Figure 4-4 below illustrates the movies selection mode.

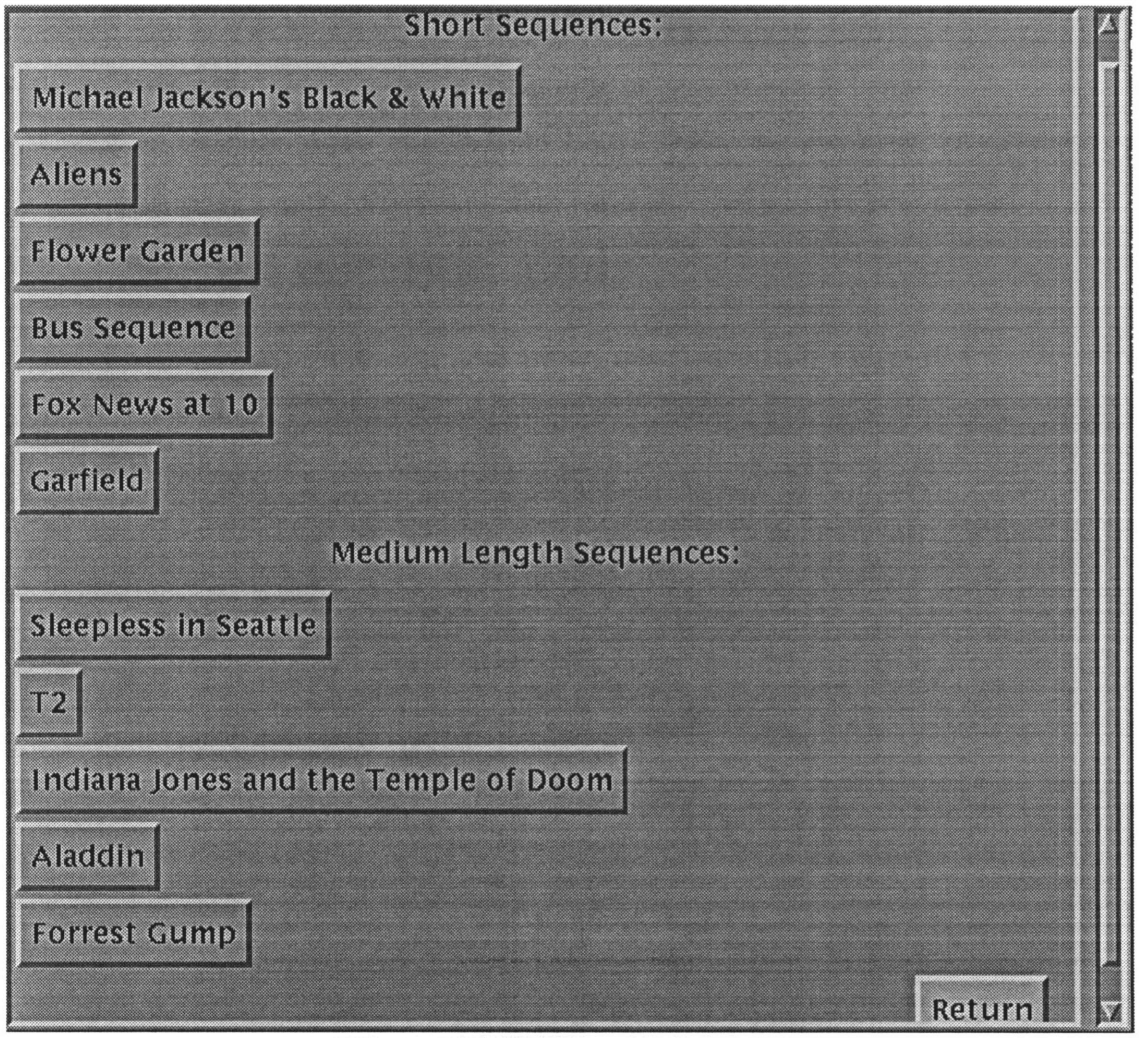


Figure 4-4: Movies selection mode

In this menu, we can select one movie by clicking on the movie name. Once a movie is selected, this menu will give way to the display of the new movie. This simulation model is simplified because billing information and authentication is ignored.

Sometimes, the movie list can be longer than the height of the screen. As a result, scroll bars are used in the menu to accomodate the long list. The return button can be clicked to return to the previous viewing mode.

4.5 Movies viewing mode

In this viewing mode, the panel below the weather panel contains different icons. The displayed icons depend on what format is the bitstream displayed. If the movie displayed is of the MJPG format, then this panel displays all the icons exactly like the VCR panel (described in Section 4.3). Otherwise, the movie is encoded with MPEG format and the panel displays only two icons: play or pause. The other icons are not made available because the MPEG software decoding is slow. As a result, the other icons are rendered impractical.

Figure 4-5 below illustrates the movie viewing mode with pause status. Note that the current display is decoded from the "flower garden" MPEG-2 bitstream.

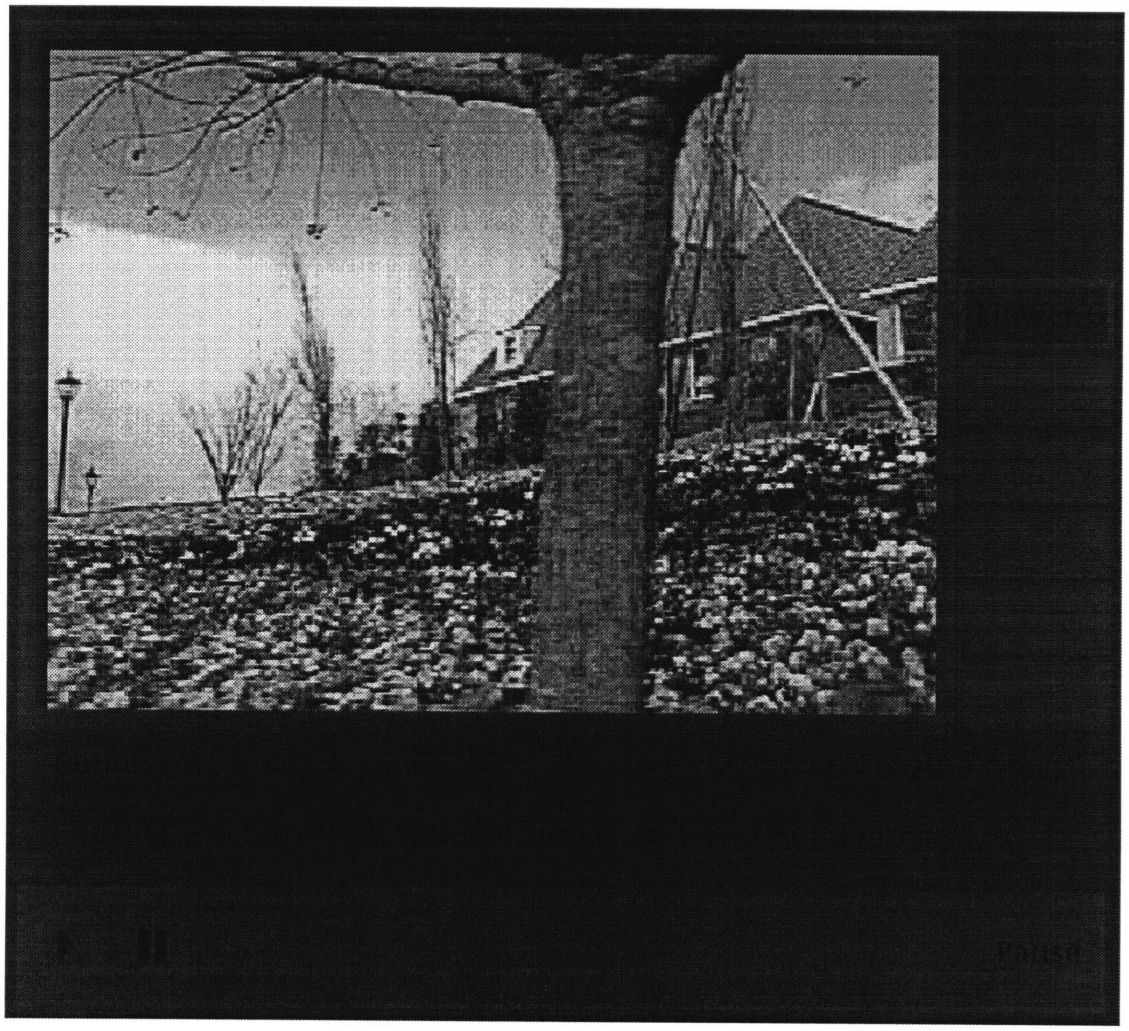
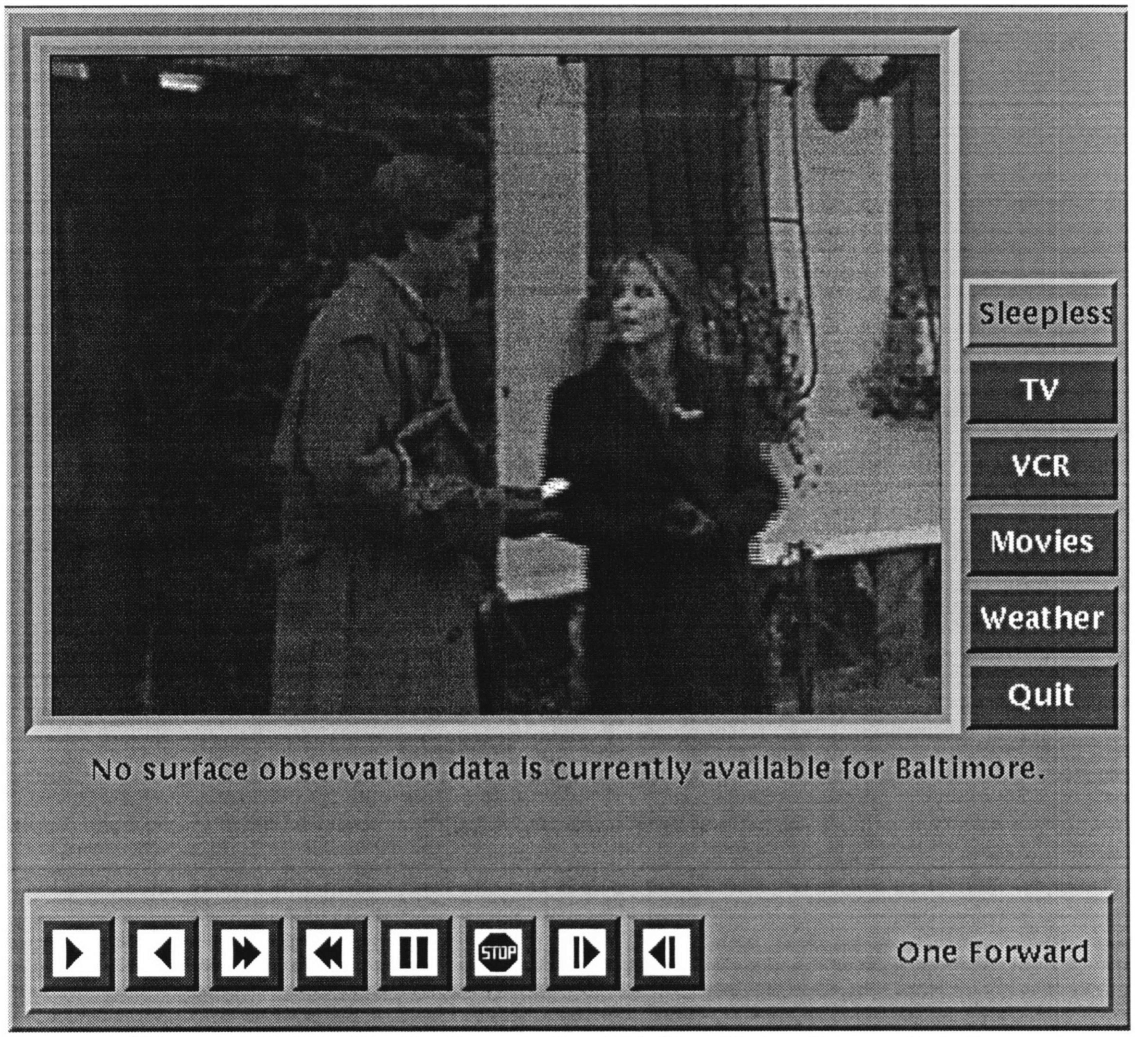


Figure 4-5: Movie viewing mode with Pause status

The colormap of the MPEG movie widget differs from the other display widgets. As a result, the color component of the screen is emphasized but its surrounding is not. Therefore, the panels and buttons are not clear in this viewing mode. This condition can be observed in Figure 4-5.

If the selected movie provides information about the location of the scene, the weather panel will display the corresponding weather information for that location.

Figure 4-6 below illustrates the movie viewing mode with one frame forward status. Note that the current display is decoded from the "Sleepless in Seattle" MJPG bitstream.



No surface observation data is currently available for Baltimore.

Figure 4-6: Movie viewing mode with One frame forward status

Baltimore is the location of the scene illustrated in Figure 4-6. The television receiver detected the geographical segmentation information in the "Sleepless in Seattle" MJPG bitstream. As a result, the weather panel displayed the weather information for Baltimore. In this particular case, the weather server does not have the weather information for Baltimore. If no geographical information is available in the bitstream, then the weather panel would continue to display weather information for cities selected randomly.

The user can switch back and forth these viewing modes. The system will return to the position where the user has left off. In effect, this television setup keeps track of three viewing channels simultaneously.

Chapter 5

Simulation Results

The TV simulation is capable of performing live audio and video, direct from the cable. The size of the TV screen is 360 by 480 pixels. Various data such as frame rate and bit rate are recorded throughout the duration of this project. Frame rate is computed from the total number of pictures displayed and its duration in seconds. Mathematically,

$$framerate = \frac{\#pictures}{\#seconds} \tag{5.1}$$

The system is expected to run at significantly below 30 frames per second due to limitation in simulation speed. As a result, the program drops a few frames a second in order to catch up. The performance of the augmented television receiver is discussed in Section 5.1. The high level functionality is discussed in Section 5.2

5.1 Performance

The 360x480 screen size is chosen over 240x320 and 480x640 because it is a compromise between frame rate and screen visibility. Although the 240x320 screen size runs at a higher frame rate (29-30 frames per second), the screen is too small for pleasant viewing. The 480x640 screen size provides comfortable viewing size but it runs too slowly (10-11 frames per second). Currently, the simulation for "Live TV" runs between 20 to 21 frames per second when the load averages of the Alpha machine is very low which is the peak performance. A sample low and high load averages (using the top command) of the Alpha machine is tabulated in Table 5.1 below:

low load avg	0.09	0.20	0.21
high load avg	4.68	4.02	3.25

Table 5.1: Sample load averages using the UNIX command: top

The performance of the simulation relies heavily on the load of an Alpha machine. When the load of the Alpha machine is high, the simulation only manages about two frames per second. At this slow frame rate, the motion picture is jerky. Furthermore, response to a user request (eg. channel switching) may be delayed. As a result, it is recommended that the simulation be performed when the load average is low.

The simulation for VCR and movie playback has a lower performance than "Live TV". The primary reason for the difference in performance is due to i/o interface. The VCR and movie playback requires the program to access bits from the hard disk. Due to the slow nature of the i/o interface, the simulation for VCR and movie playback inevitably slows down. As a result, the maximum frame rate attained is 18 frames per second. The maximum bit rate for the MJPG bitstream is between 14 to 15 Megabits per second.

Compared to a MPEG bitstream of comparable bit rate, the MJPG bitstream delivers pictures of lower quality. MJPG is essentially a MPEG bitstream with only I-frames encoded. I-frames requires the most bits to code compared to B-frames and P-frames.[1] Hence, MJPG does not provide as much compression as MPEG. As a result, a MPEG bitstream delivers higher quality pictures than a MJPG bitstream of comparable bit rate.

[1]An explanation for these terms is provided in Appendix A.

The machines which serve as media servers are all located in the Media Lab. As a result, requests for movies are granted almost immediately. The simulation runs as if the movie stream is in the local machine. Hence, the user would not notice where the movie stream comes from.

The simulation program is written based on event driven programming. Motif is an excellent tool to support this concept since it easy to build an efficient user interface. Keyboard and mouse inputs are acceptable but the inputs are limited to mouse clicks because it is simple to build a remote control which has only a button to model a mouse. Users can navigate through menus efficiently and requests are handled almost immediately.

5.2 Functionality

The TV is connected to the VCR or laser disc. The system is capable of recording live audio and video using the VCR. The recording is always recorded to the files specified by the user. However, the user is able to append multiple segments of a TV/movie session onto the same file. When the user specifies another file for recording, this translates to exchanging the old video tape with the new video tape for recording. Now, the disk space available is the limit, as the length of a VHS tape was before. Figure 5-1 below illustrates the connections between hardware components for this simulation.

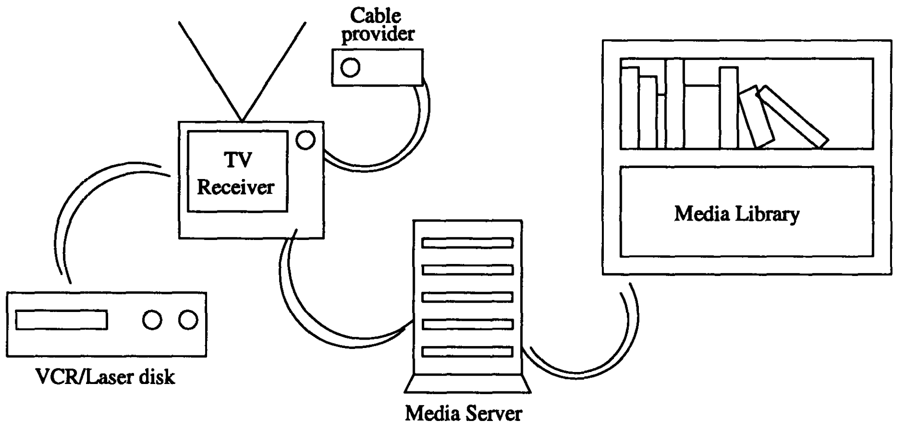


Figure 5-1: Hardware components simulated by program

Other functionalities include video playback, play reverse, fast forward, fast reverse, pause, stop, one frame forward, and one frame backward. Audio accompanies video only in the playback mode.

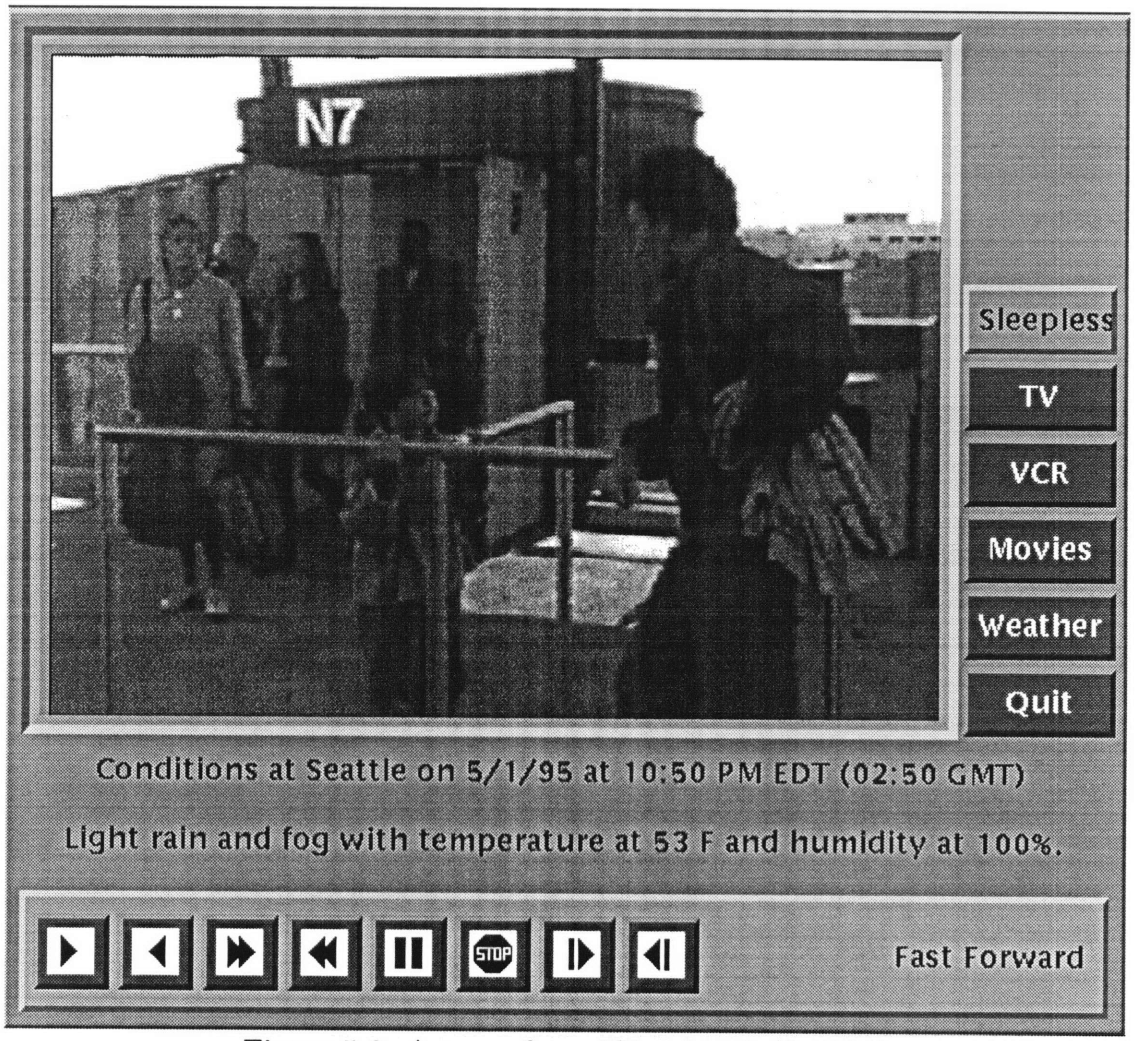
The above represent almost all the capabilities of a present day TV and VCR. A TV of the future should be capable of playing video and audio on demand. Instead of getting information from a broadcast station, the user is able to watch any movie in the media library at any time. Although there are many media servers, a movie on demand is delivered across the network from a server which is closest to the user. "Closest to the user" is one way to resolve ambiguity due to multiple media-objects residing at different sites. At present, the Media Bank is not capable of supporting live video.

In order to obtain a list of movies of interest to the user, the simulation program sends out queries to the media servers. The query categories can be movie ratings (G, PG, or R), movie length (30 minutes, one hour, or two hours), or movie content (adventure, romance, or horror). If the requested movies are available, the servers respond by returning a list of media objects to the simulation program. The user then makes a final decision as to which movie to watch. If the movie is stored at two different location, the first location which is returned by a media server will be used.

The microprocessor on board of a settop box (or a PC) can be used as an intelligent analyzer. It is able to "watch" the data stream coming into the settop box and process the information. Detection of people's location in a movie can be performed. The information about these locations comes together with the video stream. A brief weather forecast for that particular location will automatically be displayed. This action instills the sense that the television is watching together with the users. Furthermore, users have the option to view an extended weather forecast for that location.

For example, the last 30 minutes of the movie "Sleepless in Seattle" was digitized to test the program. The story revolves around the cities of Seattle, Baltimore and New York. When the scene is in Seattle, the weather of Seattle will be displayed on the weather panel. This is made possible because the movie is segmented according to geographical locations. These locations are provided as a part of the movie package. When the television receiver detects a change in location of the scene, it will query for the current weather information of that particular location or city. It is particularly interesting when the main characters are having a conversation over the telephone. The scenes change very frequently between the cities of Seattle and Baltimore. As a result, the weather panel will display the appropriate weather information accordingly.

Figure 5-2 below illustrates the screen display from "Sleepless in Seattle" and the corresponding weather information on the weather panel.



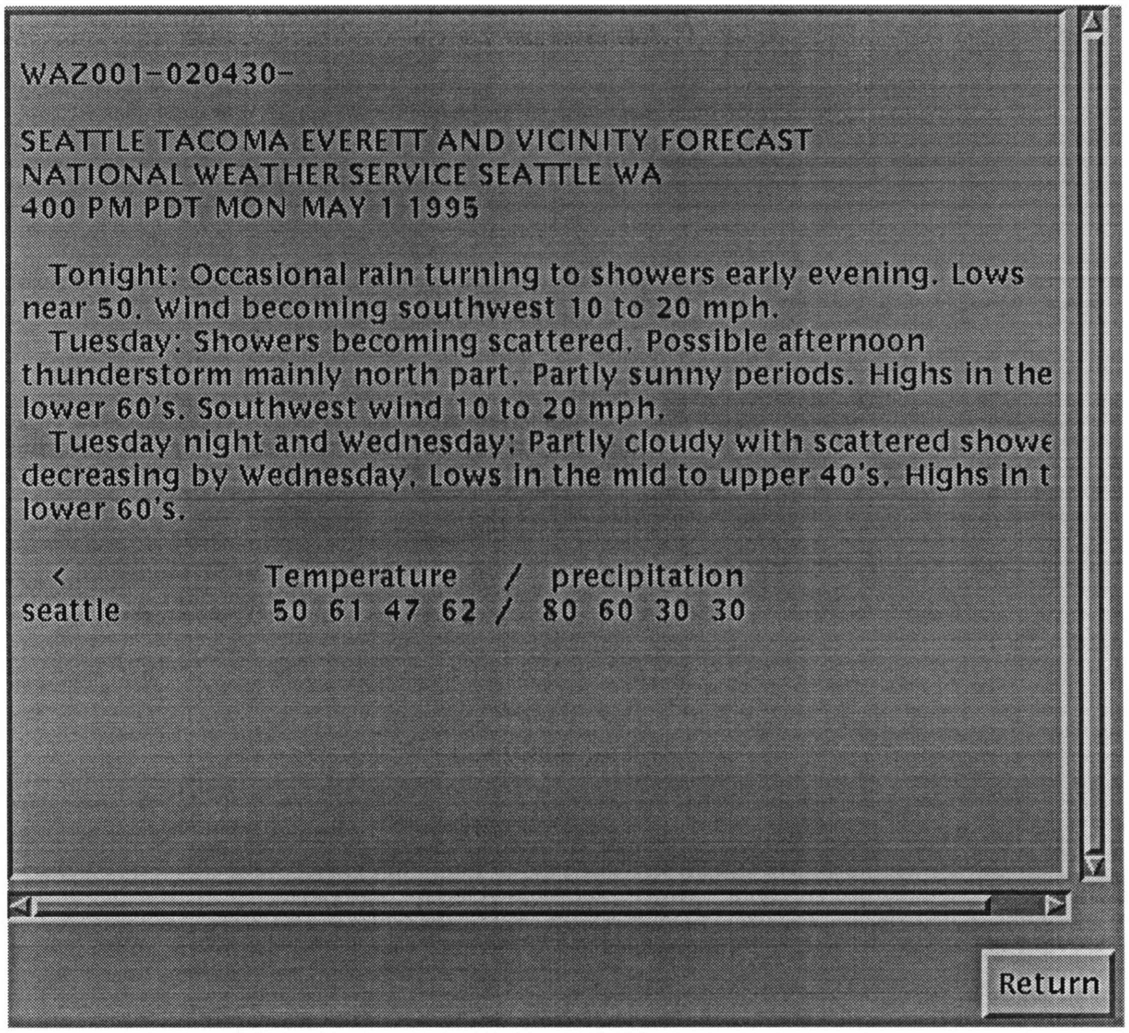
Conditions at Seattle on 5/1/95 at 10:50 PM EDT (02:50 GMT)

Light rain and fog with temperature at 53 F and humidity at 100%.

Figure 5-2: A scene from "Sleepless in Seattle"

The location of this scene is Seattle. The microprocessor of the settop box (or PC) processed this information and queried the weather server for Seattle. The corresponding weather information is displayed on the weather panel once it is obtained.

The weather button can be clicked to obtain an extended weather forecast for the city displayed on the weather panel. For example, the accompanying extended weather forecast for the weather panel in Figure 5-2 is illustrated in Figure 5-3 below.



```
WAZ001-020430-

SEATTLE TACOMA EVERETT AND VICINITY FORECAST
NATIONAL WEATHER SERVICE SEATTLE WA
400 PM PDT MON MAY 1 1995

  Tonight: Occasional rain turning to showers early evening. Lows
near 50. Wind becoming southwest 10 to 20 mph.
  Tuesday: Showers becoming scattered. Possible afternoon
thunderstorm mainly north part. Partly sunny periods. Highs in the
lower 60's. Southwest wind 10 to 20 mph.
  Tuesday night and Wednesday: Partly cloudy with scattered showe
decreasing by Wednesday. Lows in the mid to upper 40's. Highs in t
lower 60's.

  <              Temperature  /  precipitation
seattle           50 61 47 62 /  80 60 30 30
```

Figure 5-3: An extended weather forecast for Seattle.

If a movie is segmented according to geographical locations such as with "Sleepless in Seattle", the program is capable of displaying scenes from Seattle only, scenes from Baltimore only, or a combination of them such that the story will be told with a different flavor. When a movie is segmented into objects, it is possible to create many versions of the same movie. Each of them caters to specific categories of audience. Furthermore, this flexibility enables the audience to "edit" the movie to suit their taste.

Film editors welcome the object oriented video structure. The availability of objects simplifies editing like never before. Producers can produce one movie with multiple ratings because the movie content is altered to fit the ratings. For example, if the movie "T2" is segmented into violent and nonviolent scenes, and the violent scenes are edited, the ratings can change from R to PG-13. Of course, the flavor to the movie will be different. However, the flexibility of using objects enables this movie to reach a wider audience.

This simulation shows that current technology is capable of delivering movies on demand to the user. However, for simplicity, the issues of billing, privacy, authentication, etc. have been ignored. Furthermore, the simulation program interacts with the media server one to one. This is a very unlikely scenario in the real world because many potential users will try to access the same media objects at the same time. These issues can be addressed as possible future work.

Chapter 6

Future Work

This simulation introduces some aspect of the Media Bank. The Media Bank may be taken literally as a bank. This idea forces us to deal with tracking all the transactions between home users and the Media Bank. Furthermore, in order to avoid fraud and abuse of the system, security and authentication are required. Therefore, structures for authentication and billing need to be explored.

The simulation program in this thesis interacts with one server at a time. This is not a realistic reflection of possible implementations of the Media Bank because there will be many home users competing for limited resources. As a result, there is a need to experiment with multiple programs which query media servers from multiple machines in order to test the efficiency and performance of the overall system. This experiment is performed to test the durability of the servers in the face of huge demand.

Additional functionality could be added to the program. For example, shoppers can browse through an online fashion outlet to purchase a birthday gift. Service vendors such as hair stylists enable us to "try on" a new hair style online before going for the actual cut. The program may learn the shopping patterns of the user and inserts commercials at the right time. The program may also offer personalized newspaper which contains articles of interests to the home user.

A prototype network which consist of many home applications, media servers, service providers, and content providers need to be built in order to show the potential of the Media Bank. The components of this network should not comprise of fiber optics alone but also telephone lines, co-axial cable, satellites, and conventional aerial broadcast. The general model will thus be flexible enough for implementation in many environments.

Appendix A

Overview of the MPEG motion compensation

The MPEG video compression algorithm relies on two basic techniques — block-based motion compensation for temporal redundancy reduction and DCT-based compression for spatial redundancy reduction[6].Temporal redundancy reduction can be achieved by employing motion compensated interpolation. There are three types of pictures considered in MPEG — Intrapictures (I-frames), Predicted pictures (P-frames) and Interpolated pictures (B-frames).

I-frames are employed as access pointed for random access. However, they only provide moderate compression. P-frames are coded with reference to a I-frame or P-frame, and they generally will be used as a reference for future P-frames. The highest level of compression is achieved by coding B-frames. However, B-frames require both a past and a future reference for prediction. Furthermore, B-frames are never used as reference.

The relationship between I-frames, B-frames, and P-frames is shown in Figure A-1 below.

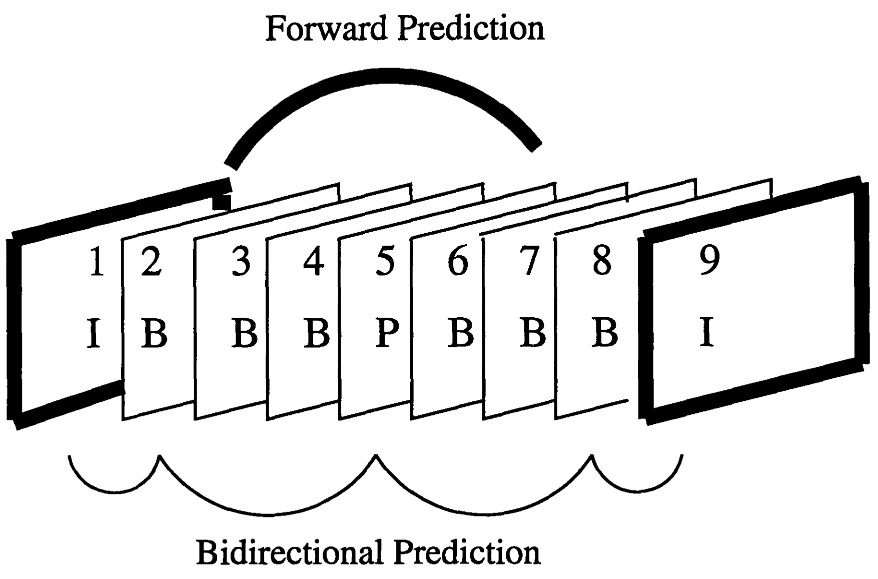


Figure A-1: Motion compensation for interframe coding

Examples of forward predictions are shown in Figure A-1 where P-frame (#5) predicts from I-frame (#1), B-frame (#2) predicts from I-frame (#1), and B-frame (#8) predicts from P-frame (#5). For backward predictions, B-frame (#2) predicts from P-frame (#5) and B-frame (#8) predicts from I-frame (#9). In the illustration above, an I-frame is inserted every eight frames, and the ratio of B-frames to I-frames or P-frames is three of four.

Glossary

ATM	Asynchronous Transfer Mode
B-frame	bidirectional prediction frame (refer to Appendix A)
CCIR	International Radio Consultative Committee
DAT	a directory containing one or more files which contains data organized in a rectangular matrix with any number of dimensions
DCT	Discrete Cosine Transform
GIF	Graphics Interchange Format
I-frame	Intra frame (refer to Appendix A)
IP/TCP	Internet protocol / transmission control protocol
MPEG	Moving Pictures Expert Group
MULAW	μ-LAW, a cellular phone encoding standard
NFS	Network File System
P-frame	prediction frame (refer to Appendix A)
TIFF	Tagged Image File Format
widget	a reusable, configurable piece of code that operates independently of the application

References

[1] Lippman, A.; "The Distributed Media Bank," IEEE First International Workshop on Community Networking, July 13-14, 1994.

[2] Lippman, A.; "The Distributed ATM Media Bank," MIT Media Laboratory, July 27, 1994.

[3] Lippman, A., Holtzman, H.; "The Distributed Media Bank: Specification," MIT Media Laboratory, April 20, 1994.

[4] Lippman, A.; "Notes on the National Information Infrastructure," MIT Media Laboratory, May 19th, 1994.

[5] "Multimedia Services for DEC OSF/1 AXP Programmer's Guide," March 1994, Digital Equipment Corporation.

[6] Le Gall, D.; "MPEG: A video compression standard for multimedia applications", Communications of the ACM, April 1991/Vol 34, No. 4

[7] Committee Draft; "Coded Representation of Picture and Audio Information", ISO/IEC JTC/SC29/WG11 (CD 11172-4 Nov 2 1993.)