IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4489

Rapid Video Browsing on a VCR using a TV Set-top Box

Anand Balachandran and P. Venkat Rangan
Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0114, Tel: (858) 534-8173 Fax: (858) 534-7029, (anandb, venkat)@cs.ucsd.edu

ABSTRACT

The era of interactive TV will be characterized by value-added services wherein users can shop, play games and receive video-on-demand - all using their television. In this paper, we describe the design and implementation of a rapid video browser that can easily be deployed on a TV set-top box to enhance the viewing of a recorded TV program. The vast penetration of VCRs and the proliferation of set-top boxes suggest that this application can be widely deployed in viewers' homes giving them a taste of future interactive services.

1. INTRODUCTION

The synergy between the areas of Internet computing, high-speed communication and entertainment promises a new marketplace that will directly affect the world of consumers. This convergence is evident from advances in compression technologies for digital media, high-speed fiber optic connections to the home, and the introduction of set-top boxes with high computing power and networking support. Interactive TV is ushering in a new epoch where consumers can retrieve information, shop, play interactive games, and order video-on-demand through simple interaction with their television sets. Set-top box systems with high-speed multimedia processing hardware and broadband network interfaces are transforming today's television sets into interactive multimedia entertainment systems.

Analog media and broadcast transmission have thus far characterized the TV industry. Such transmission involves passive viewing where users can neither control the programs they view, nor schedule the viewing time of programs to suit their preferences [1]. Even programs that are recorded, when played back on a VCR can at best be viewed in a linear fashion. The user does not have the flexibility of exercising fine-grained control to seek to a desired position in the video sequence. On the other hand, interactive TV will help users to subscribe to channels *on-demand*, and shop and retrieve information as well, i.e. *get what they desire when they desire it*. Interactive TV services will work akin to a client-server model where there is a logical connection between the content provider and the user over which to download services and exchange information.

In this paper we describe one such application that uses the power of current set-top boxes to address a critical problem in interactive TV, i.e. video browsing. We have developed a rapid scene-browser application to enhance the viewing of a recorded TV program. The scene browser is resident on the user's set-top and retrieves a sequence of thumbnail images that correspond to the individual scenes in the TV broadcast. A user who sets up his VCR to record the broadcast now also sees the individual scenes recorded on the videotape. Such an application provides an easy way for the user to view a scene of his choice by positioning the video playback to the start of the scene using the timestamp provided in the index, as opposed to playing back the videotape in a linear fashion. Given the state of the art in interactive TV services, the scene browser is a definite step into the new era of interactive TV.

The paper is structured as follows. Section 2 gives an overview of the set-top box architecture and a high level description of the PowerTV operating system. We describe the scene browser application in Section 3, followed, in Section 4 by a description of the architecture of our system. Section 5 details the implementation including protocols used in transmission of the scenes from the broadcast station to the user and the user interface of the scene browser. We discuss related work in Section 6.

2. HARDWARE AND SYSTEM ARCHITECTURE OF A SET_TOP BOX

2.1 The Interactive Digital Set-top

The interactive TV market is now characterized by the ubiquity of the TV *set-top box* (STB). The interactive TV market is now characterized by the ubiquity of the TV *set-top box* (STB) - a terminal device that receives and decodes the content from the provider and delivers it to the user's television along with some interactive capabilities [2]. STB manufacturers like Scientific Atlanta, WebTV, GI, Pioneer, and Toshiba, to name a few, are developing devices that can be deployed with the arrival of digital television. The gradual transition from analog to digital and broadcast to interactive transmission can be seen from the evolution of three different STB architectures, viz. broadcast analog, broadcast digital and interactive digital. The anatomy of an interactive digital STB is shown in Figure 1.

2.2 The PowerTV Operating System

PowerTV™ is a compact multi-tasking operating system specifically designed to address the high-performance demands of media-centric real-time applications [3]. Its highly modular structure makes it easy to be adopted across multiple hardware platforms and support a wide range of applications ranging from home shopping to video-on-demand. The PowerTV architecture consists of layers of interconnected modules designed to minimize redundancy and optimize multimedia processing. Since the operating system will reside on consumer devices, it has been designed to exist in a ROM-based system with a very small footprint. The PowerTV kernel supports a wide variety of APIs, which abstract the details pertaining to the underlying hardware. In this application we made use of the Screen Manager and TV Manger APIs provided by the kernel.

3. THE SCENE BROWSER

The scene browser, as the name implies, is an application program that allows the user to browse a video using its component scenes as an index. A scene in a video sequence is defined as

This paper appears in *Challenges of Information Technology Management in the 21st Century,* the proceedings of the Information Resources Management Association International Conference.

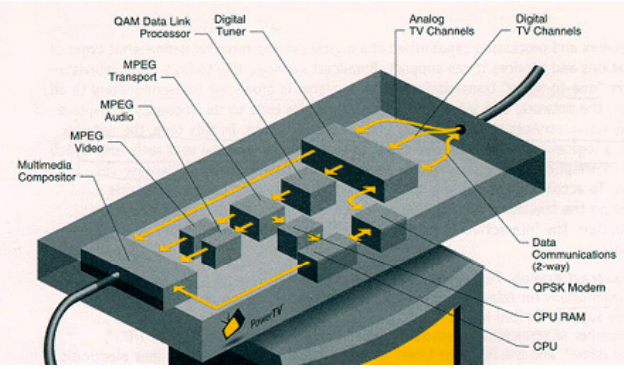

Figure 1: Anatomy of a Digital Set-top Box (Courtesy:

a continuous sequence of video frames that have no significant inter-frame difference in terms of their visual contents. A scene usually results from one continuous camera operation [4]. In this section, we give a high-level description of the application from the point of view of the end user and reserve the architectural description to the next section. The scene browser uses the features of STBs to enhance the viewing of a recorded TV program. The application, resident on the user's STB, retrieves a sequence of thumbnail images at the beginning or end of a TV broadcast that the user has programmed his VCR to record. Each of these thumbnail images corresponds to a new scene in the scheduled TV program. The thumbnails are also annotated with a timestamp representing the offset of that particular scene from the start of the featured program.

The segmentation of the video into composite scenes can be done by the content provider, the user's cable provider (head end), or even the *Internet Service Provider (ISP)* through whom the user is connected to the Internet. As illustrated in Figure 2, the architecture of the scene browser describes the data flow from the head end to the user. The process of scene segmentation can be automated or assisted by human interaction. The scene browser application provides an easy way for the user to view a scene of his choice by positioning the video playback to the start of the scene using the timestamp provided in the index, as opposed to playing back the video in a linear fashion.

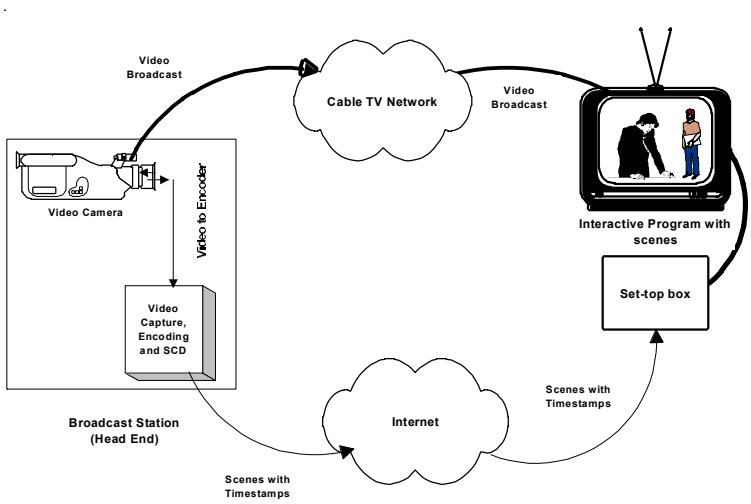

Figure 2: Architecture of the Scene Browser

3.1 Interactivity on a Set-top Box

The primary motivations for the design of the scene browser were two-fold: the need for simple interactive browsing and the penetration of VCRs. Since the early set-top market is strongly oriented towards analog broadcast owing to the high costs involved in supporting interactivity, it seems natural that an interactive application caters to users in the analog domain. We found, to our advantage, that the Explorer-2000 STB from Scientific Atlanta was designed to receive TV broadcast and in addition had networking support, over which to receive digital data like sub titles and captions to provide interactivity. In the case of the scene browser, interactivity is provided by the ability of the user to *seek* to any point of interest within the video by simple operations like fast forward or fast reverse to the desired position in time. VCRs in the market today are equipped with the capability of displaying the time as a tape is played back enabling users to exercise control at the granularity of a second.

3.2 An Example Scenario

Consider the following scenario where the scene browser can be used. The user orders a paid program from his cable provider and sets up his VCR in advance to record the program. A service like the *electronic program guide* (EPG) resident on his STB can give the user advance information about the schedule for the day's programming. The EPG is available as an installed application on the users STB when he subscribes to the cable provider for service. Upon notification by the user, the cable provider performs a scene-based segmentation (see Section 4) of the video to be delivered to the user. The segmentation results in a number of thumbnail images each corresponding to a new scene in the video. When the aired program has concluded, the scene browser requests the cable provider to digitally send the thumbnail images to the STB at the user end. These scenes are formatted as an album by the scene browser and displayed on his television or alternatively dumped onto the videotape.

4. ARCHITECTURE OF THE SCENE BROWSER

There are three components that go into the design of our prototype system - the server at the head end, the interconnecting network, and the STB at the user end. Thus far, it has been implicit in our description that the video transmitted to the user is segmented into scenes prior to transmission. Segmentation of the video into scenes is done at the head end. The software responsible for performing the scene detection also captures the time, relative to the start of the video, at which the scenes change (see Section 5 for details). This is the timestamp that is transmitted later to the user along with the scenes. The video for the program transmitted to the user is encoded here and the resultant digital feed is used for scene segmentation. Since the cable provider and the user have programmatically agreed on the viewing time, the scenes can be transmitted soon after the program has concluded. On receiving the scene images, the scene browser residing on the STB, formats them as an album to be displayed on the user's television screen.

4.1 Scene-based Segmentation of video

The process of detecting the changes in visual contents of successive video frames is referred

to as scene-based segmentation or simply *scene change detection (SCD)*. A scene usually is the result of one continuous camera operation. Hence the scene usually changes at the end of a *cut* during the video production. In this section, we present a high-level description of the process of scene segmentation. Details of our algorithm can be found in the section on implementation. Scene changes can be abrupt or gradual. Abrupt changes can come from coarse video editing where segments are cut and pasted in between the video sequence. Gradual scene changes are results of finer editing operations including special effects like zoom, camera pan, dissolve and fade in/out [5]. In general, gradual scene changes are more difficult to detect than abrupt scene changes. SCD algorithms usually operate by measuring the difference of some key properties between successive image frames in the video. Common properties include, color, spatial correlation, object shape, motion contained in the image, or DCT coefficients in the case of compressed video. The algorithm computes the mean and the variance of the pixel-wise difference between successive frames of one of these properties and looks for sharp peaks based on a predetermined threshold [6]. A point where the difference crosses the threshold would be considered a scene boundary. A *key* frame, most often the first video frame in the scene, is used to represent the entire scene.

4.2 •Design Choices for Performing SCD

The design of the scene browser is made with the primary goal of being able to efficiently distribute the functionality between the server at the head end and the STB at the user end. We were faced with the decision as to where the scene change detection would be performed. The video can be segmented into scenes by the content provider, the broadcaster, or even the user's Internet Service Provider (*ISP*). Figure 3 illustrates the setup of our choice. We list the pros and cons of the different approaches below:

- *SCD performed by the content-provider:* When a program is telecast, the producer of the video has rights to the content and therefore may want to control the processed information as well. In the case of the football game, for example, the rights may belong to ESPN who delivers its content to the various cable providers. In this case, the SCD can be performed at the post-production stage itself and the content provider may edit the scenes resulting from automated SCD using his discretion whether a particular scene change is actually valid or not.
- *SCD at the head end (server-based approach):* There are many advantages of performing the SCD at the cable provider (head-end). First, there is better support for memory and processing power in servers at the head end than on the user's set-top. STBs today are designed to keep the costs low and hence have limited memory support. A second advantage of server-based SCD is that the segmentation can be performed in advance and made available to the user at the end of the program. We have seen that interactive digital STBs connect to the Internet through one of several networks, viz., a cable modem service, an ADSL line, or an ISDN. This implies that the user is connected to a service provider who is responsible for any data that he receives over the Internet. Since the ISP is an entity in the user's data path, we can envision a scenario where the ISP undertakes the task of performing the SCD on behalf of the user. If the user gets his network service through a cable modem network, then the cable provider himself becomes the ISP.
- *SCD at the user end (client-based approach):* With the SCD performed at the client, i.e. on the user's STB, the algorithm can make use of the real-time scheduling features of the set-top's operating system. Thus it would be possible to provide efficient real-time performance to the user without having to worry about network latency that the server-based approach would entail. Interactive digital set-tops, in the near term will be designed to support more memory and processing power at an affordable cost. It is the vision of set-top manufacturers that the users will be ready to afford the differential cost for the additional interactivity and quality of service. Consequently, applications that operate on multimedia data (e.g. the SCD algorithm) can reside on the STB and can be provided as a value-addition that the user can subscribe to.

In our implementation we have chosen to perform the SCD at the server end. Herein, the scenes comprising the video are represented by thumbnail images, each representing the scene's key frame. The key frames are stored at the head end and transmitted to the user over the digital channel that connects his set top to the cable provider.

4.3 Transmission of the Scene Images

Performing the SCD and storing the scenes at the head end prompted us for another decision on the time and frequency of transmitting the thumbnail images. Given that the user would view the scene as an album when playing back the videotape, we envision three approaches for transmitting the scenes.

- *Transmission before the video broadcast:* If the scenes were detected and stored at the head-end in advance, they can just be transmitted before the program. In this case, the scene browser would be programmed to retrieve the scenes a certain time before the scheduled broadcast. The scene browser would then become operational prior to the broadcast and the formatted scene album would be displayed/recorded on tape before the program. In the case of a live broadcast however, the SCD would also be done live and the situation is different. Live broadcasts are dealt with in Section 4.4.
- *Transmission after the video broadcast:* An alternative to the first approach is to transmit the scenes at the end of the broadcast. This, in fact, is the approach that we have adopted in our implementation. Though not very different from the first, it made our implementation easier as in this case we did not have to deal with programming the STB to re-tune to the particular channel to receive the video broadcast. Further, this approach resulted in one common implementation even for live broad-

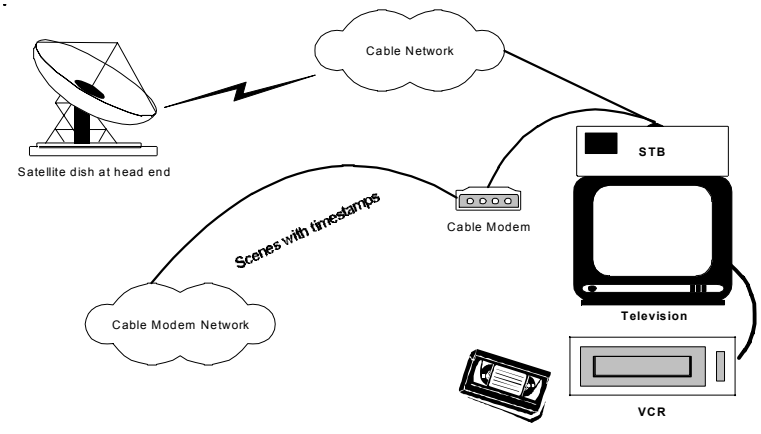

Figure 3: Set up showing the reception of scenes at the user end

casts where the scenes would not be available until the end of the program.

- *Transmission intermittent with the broadcast:* A third alternative would be to periodically provide the user with scenes corresponding to fixed sections of the program. In such a case, the scene album would be displayed at regular intervals. There are clear disadvantages with this approach. First, the user would have to seek to multiple positions on his tape to look for scene albums corresponding to the different parts of the program. Secondly, the transmission of scenes in between the program would entail synchronization with the main broadcast in order for the scenes not to interfere with the program. Synchronization of digital data with an analog signal is a topic of ongoing research and is beyond the scope of this paper.

4.4 Live Broadcasts

A special case of video broadcast is when the program is transmitted live to the user. This is a common feature in telecasts like news, sports events, and entertainment shows all of which have very large audiences. Therefore, we have made the scene browser amenable to live broadcasts too. The server that performs the SCD at the head end receives a simultaneous input of the live feed as it is being broadcast to the user. Our SCD algorithm works in *real time* and so the live video is encoded and segmented into composite scenes as it is received. The key frame marking the start of each scene is stored at the head end itself. Transmission of the scenes is delayed till the end of the broadcast so that all scenes can be delivered to the user at the same time. This is in agreement with our decision to deliver the scenes at the end of the video broadcast, no matter what the nature of the broadcast is.

5. IMPLEMENTATION

The scene browser prototype system has been developed using 300 MHz Pentium PCs as the server at the head end and the Explorer-2000 STB from Scientific Atlanta at the user end. The machine at the head end runs Windows NT and the STB runs PowerTV™. We have used the TV manager and the networking APIs provided by the PowerTV modules in order to develop the scene browser client that runs on the Explorer-2000. We chose to use Windows NT on the server because of its superior support for real-time multimedia and the availability of Microsoft's DirectShow™. For more information on the DirectShow architecture, the reader is referred to the on-line site at [7].

We have developed the SCD algorithm as a separate module that can be placed downstream from the rendering modules. The SCD algorithm is run in real time in the case of a live broadcast. However, for recorded video that is available in advance at the head end, the SCD can be performed off-line and the resulting scene images can be stored. We simulate this by running the SCD algorithm in advance and storing the thumbnail images in a Microsoft Access database. In this section, we start by describing the server side implementation (i.e. the SCD algorithm) and then detail the client side (i.e. the scene browser).

5.1 The SCD Algorithm

As mentioned in Section 4, a generic method for scene change detection in video is to sequentially measure inter-frame color differences between successive frames and study their variances to look for sharp peaks based on a predetermined threshold. The frame in which the variance crossed the threshold would be considered the first frame of a new scene. In order to compute a color-based difference between two frames, the 3-dimensional color space (RGB) is divided into bins in each dimension and a color histogram is plotted for each frame [8]. The number of bins in each dimension depends on the degree of coarseness employed in partitioning the space of each color component. The color difference between two frames is the variance of the bin-wise difference between their respective color histograms. When a scene chance occurs, the color composition between two successive frames is likely to be very different. Hence, the histogram difference between the two frames will be large. While this is true for abrupt scene changes, gradual scene changes do not exhibit this phenomenon. Hence, most SCD algorithms that identify scene boundaries in a video come up with false alarms or result in many scene changes going unidentified.

The thresholds for declaring scene changes are picked based on computing the inter-frame difference over all frames for several videos and visually examining each frame to check if it is an ideal candidate for a scene change. We also used a correction factor in computing the color histograms of each individual frame. If a pixel had its R, G and/or B value very close to a bin boundary, it was placed in both bins on either side of the boundary. We found that employing this correction significantly reduced the number of false alarms in frames detected to be scene boundaries. The frame that our algorithm picks as the scene boundary is chosen to the key frame of that scene. This frame is stored locally as a bitmap image with a thumbnail (33 x 40) resolution.

Our SCD algorithm has been implemented as a separate COM library and can be integrated as a separate module within DirectShow downstream from the rendering module. Rendering is a process where the decompressed video stream is delivered in the form of a bitmap (array of RGB values of pixels) for display. We deemed it fit to plug in the SCD module at this point because of the ease with which pixel-level operations can be performed. The rendering module outputs the same image irrespective of the compression format of the video stream.

5.2 Transfer of Scenes to the Client

The scene browser runs on client-side of our system (i.e., the user's STB) and is operational at the end of the broadcast from the head end. Prior to retrieval of the scenes, the scene browser set's up a connection with a known port on the head end machine by sending a *connection-request* packet. The packet holds a structure with fields indicating the type of request (*set-up or tear down*), the receiving port number of the client, and the client's network address. The requested connection is acknowledged by the server

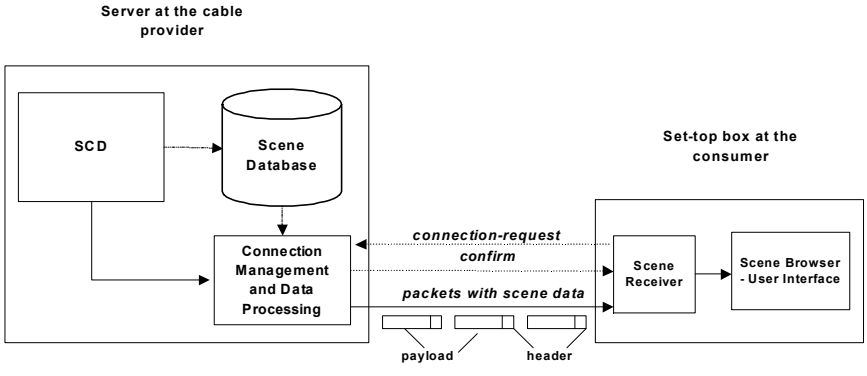

Figure 5: Communication between Server and Scene Browser Client

through a *confirm* packet. Once the server and client have agreed on the address and port number for communication, data transfer begins. The messages that form part of the communication between the server and client are illustrated in Figure 5.

5.3 User Interface of the Scene Browser

The user interfaceThe user interface of the scene browser has been designed to display the scene images retrieved from the server as an album of thumbnails. The timestamp information obtained along with the scenes is printed below each scene. We designed the user interface of the client to fit the average 21' television screen. Since each bitmap had a pixel resolution of 33 x 40, this would approximately fit about 20 bitmap images (4 rows of 5 columns) in one screen display. In the event that there are more scenes than can be displayed on the screen at one time, the scene browser refreshes the screen with an updated album containing the next 20 bitmaps. The interval between successive screen refreshes is designed to allow the user enough time to review the scene album and navigate to the scene of his choice.

The video broadcast feed for the program passes through the set-top before being displayed on the television. However, the scene information is delivered digitally and displaying it following an analog signal would mean transferring the control from the TV tuner to our scene browser. To this end, we used the Screen Manager and TV Manager APIs provided by PowerTV, which allow user-designed applications to control the output to the screen. Using the TV Manager API, the application can tune to any channel that it chooses to display the data in. The TV display unit receives one common NTSC signal whether the data is an analog video signal or digital data from an embedded application in the STB. Providing this transparency is one of the most important features that must be supported by all STBs that provide interactive services. Figure 6 illustrates the user interface of the scene browser as dumped onto a section of the videotape.

6. DISCUSSION AND RELATED WORK

There has been considerable research in the area of indexing, searching, scene boundary detection and browsing of video [9] [10]. This research, however, has not focused on deploying the technology in home entertainment systems. Although the algorithms and implementation of the scene detection have been extensively studied, we believe that their application in the context of interactive TV is new. Recently, companies like Tivo and ReplayTV have released two commercially available systems with capabilities to automatically record and replay television programs [11] [12]. The deployment of these systems involves purchasing specialized hardware devices, similar to a hard-disk drive, that need to be connected to the users cable box. These devices store the television programs in the popular MPEG-2 format. Users can program these devices, through a user-friendly interface to automate the recording of their favorite channels up to a total of 30 hours of programming and replay them on-demand. The main drawback of this system is that it requires users to purchase specialized hardware until the technology is incorporated in commercially available TV set-top boxes. Furthermore, we believe that these systems are also quite expensive and hence automatically exclude a good fraction of television viewers.

Our solution has been implemented entirely in software and has been deployed on the PowerTV set-top box as a proof of concept. Although it provides the user a very simple indexing and browsing service, we believe that this application is compelling mainly due to its simplicity, ease of deployment, and low cost. We are confident that with advances in programming models for interactive TV systems, the applications that can be supported on next generation set-top boxes will include a much greater degree of interactivity.

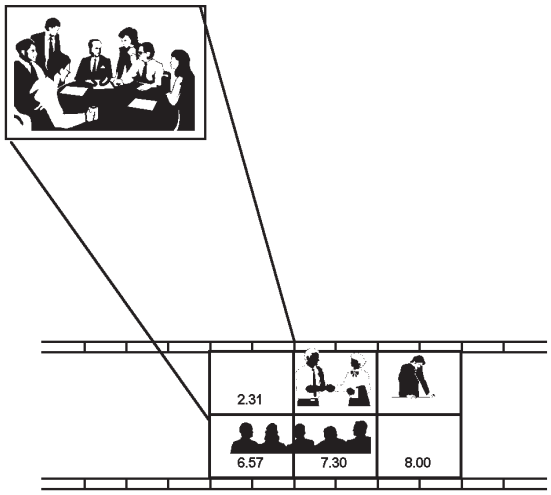

Figure 6: Section of videotape showing scenes and timestamps

7. ACKNOWLEDGEMENTS

We would like to thank Scientific Atlanta Inc. and PowerTV Inc. for the support they extended in helping us cope with the new tools for application development and in learning the PowerTV™ operating system environment. In particular, we would like to thank Brian Kitzberger for his help with the PowerTV system API. We are grateful to Kamlesh Talreja and Bhushan Chitnis who helped with the development during the critical phase of this project.

REFERENCES

1. Christos H. Papdimitriou, Srinivas Ramanathan, and P. Venkat Rangan, "Information Caching for Delivery of Personalized Video on Home Entertainment Channels", *In Proceedings of the International Conference on Multimedia Computing and Systems, Boston, May 1994, pp. 214-223.*
2. Borko Furht, Deven Kalra, Frederick L. Kitson, Arturo A. Rodriguez, and William E. Wall, "Design Issues for Interactive Television Systems", *IEEE Computer, Vol. 28, No. 5, May 1995, pp. 25-39.*
3. The PowerTV White Paper — Open-Platform Architecture for Interactive Digital Set-top Boxes, *http://www.powertv.com/product/completewhite.html*, *October 1996*.
4. Ahmed K. Elmagarmid et al, "Video Database Systems", *Kluwer Academic Publishers, 1997*.
5. Arun Hampapur, Ramesh Jain, and Terry Weymouth, "Digital Video Indexing in Multimedia Systems", *In Proceedings of the Workshop on Indexing and Reuse in Multimedia Systems, August 1994.*
6. Jinhao Meng, Yujen Juan, and Shih-Fu Chang, "Scene Change Detection in a MPEG Compressed Video Sequence", *In Proceedings of the IS&T/SPIE Symposium, Vol. 2419, San Jose, February 1995*.
7. *Microsoft Direct Show: http://www.microsoft.com/directx/pavilion/dshow/default.asp*
8. F. Arman, A. Hsu, and M-Y. Chiu, "Image Processing on Compressed Data for Large Video Databases", *In Proceedings of ACM Multimedia'93, San Diego, June 1993, pp. 267-272.*
9. H. Aoki, S. Shimotsuji and O. Hori, "A Shot Classification Method of Selecting Effective Key-frames for Video Browsing", *In Proceedings of ACM Multimedia '96, pp. 1-10*.
10. Tonomura Y., and Abe S., "Content-Oriented Visual Interface using video Icons for Visual Database Systems", *In Journal of Visual Languages and Computing, Vol. 1, 1990, pp. 183-198*.