The Development of an Online Video Browsing System

Jesse Jin[1] and Ruiyi Wang[2]

[1]Biomedical and Multimedia Information Technology (BMIT) Group
School of Information Technologies
University of Sydney, NSW 2006, Australia
[2]School of Computer Science & Engineering
University of New South Wales, Sydney 2052, Australia

ruiyi@cse.unsw.edu.au

Abstract

This paper presents a system designed to allow efficient retrieving, browsing and real time playing of videos through the Internet using a web browser. The system consists of a web site as well as tools to facilitate browsing, searching and video streaming. At the website, users can view video structures and clip details, search for segments in a video by key words and play video clips in real time according to their own connection speeds.

In this paper, various approaches to each component of the system are discussed. Issues encountered in the design and implementations of the system are described. Testing results on resource usage and system usability are also evaluated.

Keywords: Content Based Video Retrieval, Webcasting, Video Encoding

1 Introduction

In recent years, much interest in the development of digital video systems has been fuelled by the advancement in digital signal processing, VLSI and rapid increases in communication bandwidth. The availability of digital videos has led to a new set of communication means with applications such as video-on-demand, video-conferencing and video-aided distance learning. However, for videos to be used as commonly as text in information exchange, effective methods in retrieving videos' rich content are yet to be developed. For example, searching for a word or phrase in a page of text is trivial and has been implemented in almost all text editors and web browsers. On the contrary, searching for a character or an event in a video is still quite impractical if not impossible. The use of videos through the Internet is also limited by the fact that video files are of much bigger size than text files and thus take a significant amount of bandwidth to transfer. The difference in user bandwidths imposes even more difficulties in real time video viewing through the Internet. The current method used in organizing textual information is as the following:

"Volume → Book → Chapter → Section → Sub-section → Paragraph → Word

Furthermore, a table of contents and indices are usually made available to facilitate search of a particular topic and a word in a book."[1]

Under this organization textual information can be effectively searched and retrieved. Thus it is natural for us to apply the same technique to the organization of videos. However pixels (raw video data), unlike words in a book, do not convey much useful information by themselves. To overcome this, metadata information needs to be added to facilitate the indexing and searching of videos. The incorporation of annotations allows a video to carry textual information similar to that in a book and thus increases the applicability of the above hierarchical organization to videos.

When viewed online, a book or a long document sometimes is divided into small chapters so that to read any one chapter, a user only needs to download the specific chapter but not the rest of the book. A similar idea can also be applied to videos. And due to the larger size of videos, minimizing the size of download for browsing is crucial in online video viewing. Furthermore, the size of a video stream varies significantly with its quality and a compromise between the two is always needed. However users have varied emphasis and connection speeds, so this balance needs to be flexible as to cater for different users. Some past systems related to this paper are described below.

The **Vimix** [2] (Video Metadata in XML) system defines a hierarchical video metadata structure in XML and provides an environment for segmenting, organizing and annotating videos. It also integrates tools "to extract static frame objects via image segmentation techniques" [2] and associate objects with corresponding video segments. The metadata structure developed in Vimix is used to organize videos for browsing in this system.

"The **Berkeley Internet Broadcasting System** (BIBS) is a lecture Webcasting system developed and operated by the Berkeley Multimedia Research Center."[3] The BIBS system organizes a video in a linear structure and synchronizes the presentation of lecture notes with video segments played. It also offers lecture index browsing and text-based searching.

Real Network is used in BIBS for encoding and playing video clips. Each video is encoded in real time into 3 streams of bit rates: 50Kbps, 128Kbps and 200Kbps and stored in memory for play-on-demand.

The BIBS system inspired the streaming and closed captioning section of this project development.

In this paper, techniques used in the effective retrieving and viewing of videos through the internet are explored and the design and implement of an online web browsing site is presented.

2 Video Encoding

To achieve real time viewing of the video clips through the Internet, the original video clip needs to be encoded into streams that match users' connection speeds. It was decided that in this application, every clip would be encoded into streams for 56k, 128k and 256k connections because they are the most commonly used Internet connection speeds.

2.1 Encoder SDK

Various media encoder SDKs were studied and tested. It was found that Microsoft Media Encoder was the most suitable for this application because of the features it offers. These are described below.

1. Environment and Format

Microsoft Media Encoder 7 SDK works in windows 98/2000/NT. It offers encoding methods for most of the commonly used video and audio formats including ".asf", ".avi", ".bmp", ".mpg", ".wmv", ".mp3", ".wav" and ".wma". It outputs encoded streams in windows media file format ".wmv" or ".wma" which can be played with Microsoft Media Player.

2. Encoding Profiles

The Encoder SDK provides functions to encode input video files according to thirty-three different profiles. They cover almost all common communication bandwidths including ISDN, XSDL, LAN, Cable, Dial-up, Web-Server, broadband NTSC and broadband PAL.

3. Other Functionalities

Windows Media Encoder 7 SDK also provides other useful functionalities suitable for this development.

When programming with the SDK, it is possible to encode only a section of the input video clip by specifying the start and end position of the section. It is especially useful to this project as segments of a video need to be encoded into separate streams (described in section 2.2).

Furthermore, the encoding functions provided in the SDK allow encoding for either video only, audio only or both streams. This also offers a lot of freedom for future development.

2.2 Streaming and Storage

In this system, all clip segments in a hierarchical structure need to be playable through the web browser in real time for all the three connection speeds described before. Several options in streaming and storing video clips were examined.

1. Encode at browsing time

The first method proposed was to encode the required section of the original video at real time while playing. However it was soon realized that this was not practical as encoding requires a lot of processing resources and will cause the server to be overloaded when multiple users browse the site simultaneously. (On a PentiumIII-600, with 100% CPU usage, a 16 minutes clip takes roughly 14 minutes to encode for a 256kb/s connection profile. For details see evaluation section).

2. Encode and store the top level clip only

When playing videos through Windows Media Player embedded in IE5, it is possible to play only a segment of the original video clip by specifying the start and end time of this segment. Thus one approach explored was to encode the whole video clip and specify the time frame of a sub clip when it is played during browsing.

But through experimenting, it was found that for indexing to work for a video in Windows Media Player, the complete video clip would need to be downloaded first. Indexing is essential if a sub clip is to be specified. Thus even when only a small section of the original video is to be played, the whole clip will still need to be downloaded. For example, to view a 5 seconds clip in a 5 hours long video, the whole 5 hours of video will need to be downloaded; this is clearly not practical. Therefore this approach was not used.

3. Encode and store the bottom level clips only

Another approach that was tried was to encode the bottom level clips and connect them in real time when upper level clips are requested. Using this method, the total encoding time and storage space required will be comparable to the second approach and yet only the minimum amount of data needs to be downloaded to play a segment.

Windows Media Player comes with a format –".asx" for defining a play list that consists of one or more clips to be played continuously.

```
<ASX VERSION="3.0">
  <TITLE>Example Windows Media Player
Show</TITLE>

  <ENTRY>
    <TITLE>Example Clip</TITLE>
    <REF HREF=
"D:\output\encoded9\somfinal30-436.wmv" />
  </ENTRY>

  <ENTRY>
    <TITLE>Another Clip</TITLE>
    <REF HREF=
"D:\output\encoded9\somfinal3437-1610.wmv" />
  </ENTRY>
</ASX>
```

Fig 1 ".ASX" Format Example

But there are a few difficulties hindering the implementation of this method. The structure designed in

Vimix allows overlapping of clips even at the lowest level in the hierarchy. Thus to achieve the above effect, the bottom level clips would still need to be divided into absolute partitions of the original video which could result in a video being divided into a big number of very small segments. Furthermore, when two clips are connected in a play list in Windows Media Player, there is a one second delay between them at play time. So a clip consisting of a few sub clips wouldn't look like a whole video when played. This is especially noticeable when the number of its sub clips is big and each segment is short. Thus this approach is not used in this development. However, if applications that can seal two ".wmv" streams into one in real time are developed, then this approach will become practical, as it is the most time and memory efficient.

4. Encode and store every clip in the hi erarchy

In this method, every clip in the hierarchical structure is encoded and stored separately. When a clip is browsed by a user, the encoded version of this clip will be played as a whole and only this particular stream needs to be downloaded at the user side. This method ensures browsing quality and minimal data transferring although it sacrifices encoding effort and data storage space.

However because none of the other methods suited the currently available tools for the project, this approach was chosen due to its practicality.

3 Video Playing

3.1 Playing video in IE5

Microsoft Media Player 6 and 7 are capable of playing video/audio clips in almost all common formats. And the player control is a standard ActiveX control that uses Microsoft Component Object Model (COM). Using the SDK, the players can be programmed on a standard HTML Web page using JScript or VBScript. Furthermore, Media player 6 is installed by default with Windows 95/98/2000/NT. Therefore it is the most suitable player SDK to be used for this project.

The code shown in Fig 2 embeds Media Player 6 into IE5, loads up the specified ".wmv" file and auto-plays it.

```
<OBJECT ID="player"
        CLASSID="CLSID:22D6f312-B0F6-11D0-
94AB-0080C74C7E95">
<PARAM
    NAME="FileName"
    VALUE="http://127.0.0.1/output/encoded9/somfinal
30-24272.wmv">
<PARAM
    NAME="AutoStart"
    VALUE="1">
</OBJECT>
```

Fig 2 Embedding Media Player 6 in IE5

A Media Player 6 control object is defined in the code. "ID" is the name used to reference this instance of the control in the script. CLSID defines the 128 bit Class Identifier of the Player control, which is used by the Web browser to create the ActiveX object on the page. Parameters including FileName, AutoStart, ShowControls and so on can be defined as shown in the sample code. Buttons can also be programmed to allow users to control such parameters. [4]

Similar principle can be applied when programming Media Player 7 in IE5.

```
<OBJECT ID="player"
        CLASSID="CLSID:6BF52A52-394A-11d3-
B153-00C04F79FAA6">
<PARAM NAME="AutoStart" VALUE="1">
<PARAM NAME="URL" VALUE="D:/FF8
Dance.mpg">
</OBJECT>
```

Fig 3 Embedding Media Player 7 in IE5

3.2 Closed Captioning

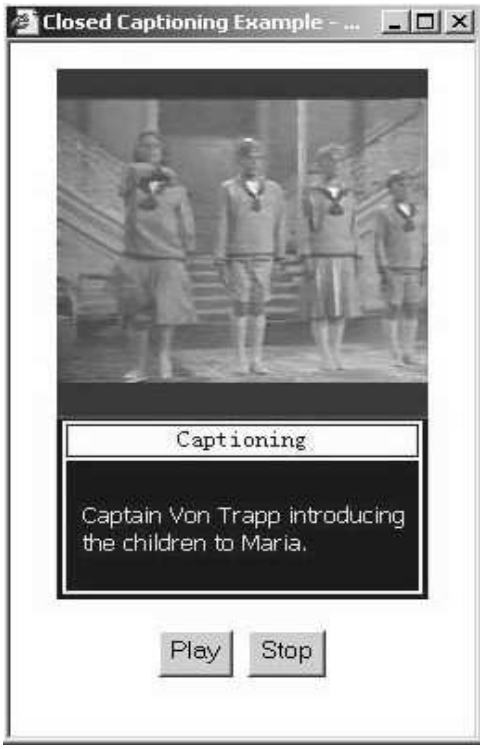

Fig 4 Closed Captioning Example

Media Player 6 and 7 SDKs also support captioning text to be played with videos through the use of Synchronized Accessible Media Interchange (SAMI)--(.smi) files containing text strings associated with specified times within the video. "The text strings appear in the Windows Media Player closed captioning display area as the clip reaches the designated times."[4] This feature provides our video browsing system the capability to display

subtitles for foreign language films or for assisting hearing impaired users.

Below is a sample SAMI file for the clip "Meeting the Children" in "the Sound of music".

```
<SAMI>
<HEAD><Title>Close Captioning Sample</Title>
</HEAD>
<BODY>
 <SYNC Start=1000>
  <P Class=ENUSCC>Maria Meeting the Children
 <SYNC Start=4000>
  <P Class=ENUSCC>Captain introducing the children to Maria.
 <SYNC Start=12000>
  <P Class=ENUSCC>Liesl
 <SYNC Start=14000>
  <P Class=ENUSCC>Friedrich
 …
</BODY>
</SAMI>
```

Fig 5 Sample SAMI file for clip "Meeting the Children"

4 Browsing and Navigation

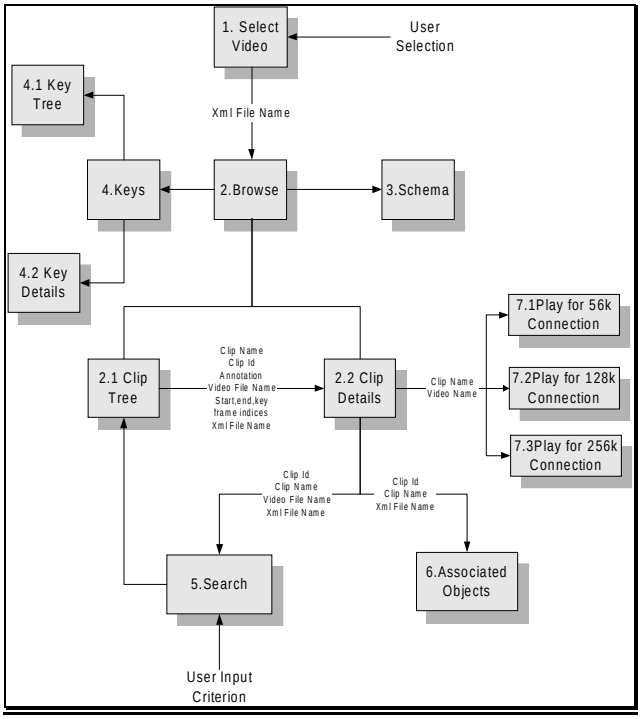

Fig 6 Data Flow Diagram

4.1 Video Structure Tree

Video structure is implemented as an interactive tree using JavaScript. The tree can be expanded or collapsed at any intermediate node (a node with children). The format of the tree is very similar to windows help files and thus is familiar to most windows users. IE5 and most other commonly used browsers support JavaScript by default and so no extra software needs to be installed for browsing which further increases usability of the system.

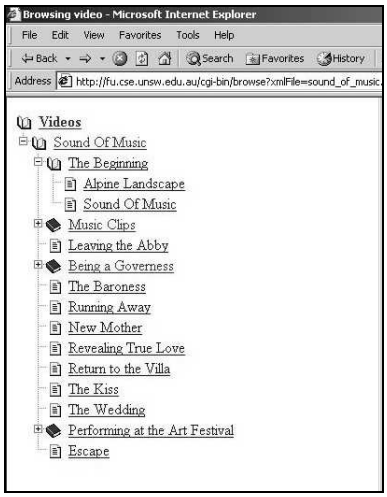

Fig 7 Video Structure Tree

4.2 Object (Key) Details

In the metadata structure used, objects (keys) can be associated with video clips. For every video structure, a hierarchical schema is defined for object types. Each type defines a different set of attributes which can be used in key word based searching. The system allows users to view the object schema tree and details of each object in the video structure.

Users can also view objects associated with a specific clip at the clip details page.

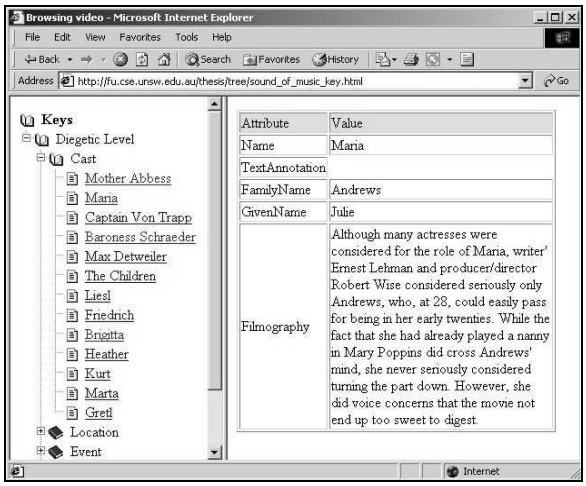

Fig 8 Object Details

4.3 Search

The search page allows users to search within the structure under the current clip. A searching criterion is formed by up to four attribute names and values that a user can input. A selection of and/or logic between the four conditions is also allowed. The server then calls the search function executable which traverses through the video structure under the current clip and selects all the clips that satisfy the criterion (which will form a tree structure). The result tree structure is displayed in the same way as video structure browsing page.

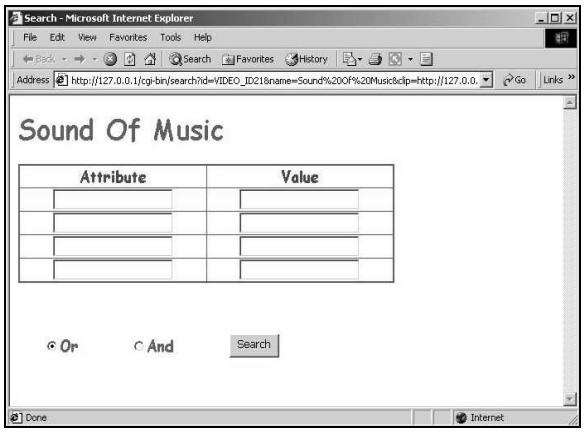

Fig 9 Web Interface of Search

5 Server Side Control

Apache HTTP Server Version 1.3 for Windows was chosen for this development because of its simplicity of use and its high performance in handling cgi-perl scripts.

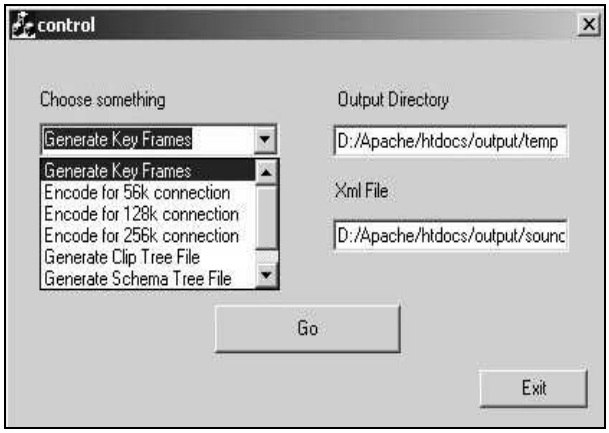

Fig 10 Control Dialogue Box

Various files including html files, encoded video clips and key frames need to be generated and stored on the server side. The overall size of the files generated for an average video clip such as "sound of music" was over 100MB. Having the control functions accessible through a web browser is therefore not practical due to security concerns. It was decided that a windows dialogue box would be used to control the generation of these files.

However in future applications it might be required to control the generation of these files via a web browser. Thus alternative solutions have also been carefully examined. Initially controlling file generations through cgi-perl scripts was tested. But unfortunately, with the current Apache server, encoding and key frame generation stopped after a certain size limit was reached. This could be due to the security settings in the server. The other approach tested was using Java Applet acting as a client that establishes a network connection with the server and then calls the executables at the server. This method was tested and proven to be viable.

6 System Evaluation

6.1 Resources Consumed

The system encodes all clip segments in the video structure for three different connection profiles and stores them on hard drives on the server side. Significant amount of processing and storage resources are therefore consumed in the running of the system.

6.1.1 CPU usage

On a PentiumIII-600 machine with no other application running, a 16 minutes clip takes 14 minutes to encode for any of the three connection profiles. The screenshot of CPU usage in Fig 11 was captured when the encoder was the only active process running. As shown in the figure, the encoding process takes up almost 100% of the CPU processing power. This again supports the decision made in *Section 2.2* to store encoded clips and not to encode at play time. Otherwise the amount of resource required for encoding when multiple users are accessing the web site simultaneously would be impractical.

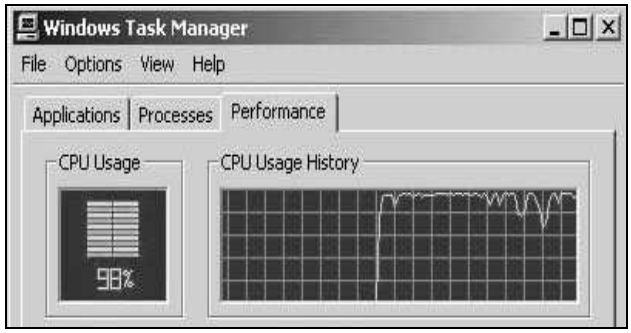

Fig 11 CPU Usage during Encoding

6.1.2 Memory Resource

The encoded video clips which need to be stored at server side take up significant amount of memory resource.

A simple summary on memory usage is done for "the sound of music":

Original "sound of music" video clip size = 149MB

Number of segments in video structure = 25

Memory used in storing encoded clips for 56k connection = 9.26MB

Memory used in storing encoded clips for 128k connection = 31.4MB

Memory used in storing encoded clips for 256k connection = 68.6MB

Total memory used in storing encoded clips = 109.26MB

Memory consumed in storing encoded clips can be effectively reduced if option 3 described in *Section 2.2* can be implemented. Currently since every clip in the structure is separately encoded and stored, there is overlapping between encoded segments and the level of overlapping depends on the video structure. However if the option of connecting multiple clips into one at play time becomes available, then only bottom level clips need to be encoded and stored and overlapping can be completely avoided. In the case of "sound of music", memory used in storing encoded clips could be reduced by more than 50 percent.

Others memory usage:

Every key frame is of size 352*288 pixels and consumes approximately 60KB of memory.

Total number of key frames = 24

Total memory used by key frames = 1.41MB

Memory used by scripts and executables ≈ 4MB

Total memory consumed ≈ 115MB

6.2 Using the System

6.2.1 Browsing

The main browsing page consists of video structure as well as clip information. By clicking a clip in the tree structure, users can view the name, key frame and annotation of the clip and also choose to search or play it.

Fig 12 Browsing Video Structure and Clip Information

6.2.2 Search

Search can be done at each clip in the video structure and results are displayed in the same format as video structure tree. Thus users can browse searching result exactly the same way as browsing the original video tree and they can further search under any clip in the searching result tree.

A searching criterion can be formed by "and" or "or" simple conditions. As can be seen in the figures below that every clip in the result tree in Fig 13 satisfies both of the two components in its composite condition, ie. it contains both "Maria" and "The Children". This result tree is in fact the intersection of the results of searching for "Name = Maria" (Fig 14) and "Name=The Children" (Fig 15) as shown.

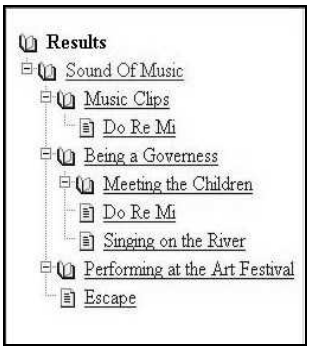

Fig 13 Search Result for "Name=Maria && Name=The Children" under "Sound of Music"

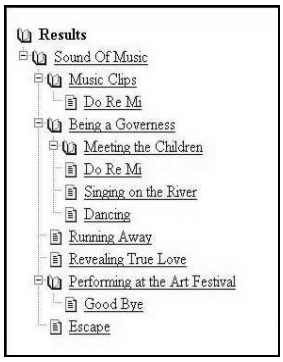

Fig 14 Search Result for "Name=Maria"

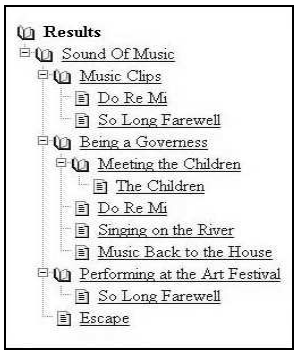

Fig 15 Search Result for "Name=The Children"

6.2.3 Associated Objects

As described in Section4.2, every clip can have objects (keys) associated with it. This association is transitive to the parent-child clip relationship. Therefore the associated objects of a clip include the clip's own objects and objects associated with its children clips.

The clip "Performing at the Art Festival" has three children clips. From Fig 16 and Fig 17, it can be seen that the parent clip's associated objects are a union of the children clips' objects and its own (XML file shows that "Performing at the Art Festival" is associated with "Salzburg Music Festival").

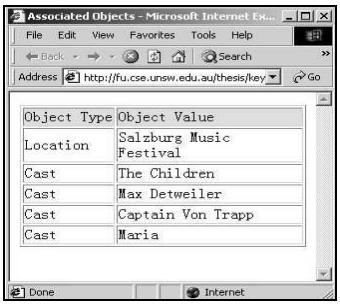

Fig 16 Associated Objects of Parent Clip "Performing at the Art Festival"

Fig 17 Associated Object of Children Clips

6.2.4 Play Encoded Streams

Every video clip is encoded into three different streams catering for connection speeds of 56k, 128k or 256k. Playing the streams in Media Player shows that the stream encoded for 56k is played at 30kb/s, stream for 128k is played at 103kb/s and stream for 256kb is played at 230kb/s. Playing the video clips through the web site using IE5 has been tested in Windows2000/98/NT using cable connection and succeeded every time with downloading time around a few seconds. If a 56kb/s dial up connection is used, for an average clip of size 370KB, the download time is less than a minute.

7 Conclusion

In this paper, an online video browsing system is presented and its design issues are discussed. The system hosts a website which allows the browsing of video metadata structures and real time playing of video segments through the internet using IE5. Furthermore, a set of tools are developed to automate the generation of files required to run the website.

Testings on both resources usage and system usability have been conducted. The system is shown to be user friendly, efficient and compatible with most web browsers in the Windows environment. Possible applications of such an online browsing system include distant education, news dissemination, ad tracking and movie archives. Webcasting with the aid of well designed metadata structure can make video communication through the internet a truly viable method.

8 References

1. S. T. Yap; J. S. Jin & D. D. Feng (1999). Automatic segmentation and semi-automatic organization in compressed video sequence, *Proc. VIP'99, November, Sydney, pp.59-68.*
2. A. Yao & J. Jin (2001). The development of a video metadata authoring and browsing system in XML, *Proc. of Pan-Sydney Workshop on Visual Information Processing (CRPIT) vol. 2. P. Eades and J. Jin Eds.*
3. BIBS: A Lecture Webcasting System by L.A. Rowe, D. Harley, P. Pletcher, and S. Lawrence. *Berkeley Multimedia Research Center, TR 2001-160, June 2001.*
4. Microsoft Windows Media SDK 7, 2000