RC22233 (W0111-018) November 5, 2001
Electrical Engineering

IBM Research Report

Video Summarization and Personalization for Pervasive Mobile Devices

Belle L. Tseng, Ching-Yung Lin, John R. Smith
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Video Summarization and Personalization for Pervasive Mobile Devices

Belle L. Tseng, Ching-Yung Lin, and John R. Smith

IBM T. J. Watson Research Center, 30 Saw Mill River Rd., Hawthorne, NY 10532
{belle, cylin, jrsmith}@watson.ibm.com

ABSTRACT

We have designed and implemented a video semantic summarization system, which includes an MPEG-7 compliant annotation interface, a semantic summarization middleware, a real-time MPEG-1/2 video transcoder on PCs, and an application interface on color/black-and-white Palm-OS PDAs. We designed a video annotation tool, VideoAnn, to annotate semantic labels associated with video shots. Videos are first segmentated into shots based on their visual-audio characteristics. They are played back using an interactive interface, which facilitate and fasten the annotation process. Users can annotate the video content with the units of temporal shots or spatial regions. The annotated results are stored in the MPEG-7 XML format. We also designed and implemented a video transmission system, *Universal Tuner*, for wireless video streaming. This system transcodes MPEG-1/2 videos or live TV broadcasting videos to the BW or indexed color Palm OS devices. In our system, the complexity of multimedia compression and decompression algorithms is adaptively partitioned between the encoder and decoder. In the client end, users can access the summarized video based on their preferences, time, keywords, as well as the transmission bandwidth and the remaining battery power on the pervasive devices.

Keywords: video summarization, personalization, mobile phone, PDA, wireless, MPEG-7, annotation, semantic transcoding, middleware, video streaming.

1. INTRODUCTION

With the growing popularity and capability of Personal Digital Assistants (PDA) and mobile phones, users have become much enthusiastic in watching videos through their pervasive mobile devices. These devices vary widely and are limited in terms of power consumption, processing speed, display constraint, and video decoding capabilities. When people use their pervasive devices, they generally restrict their viewing time on the limited displays and minimize the amount of interaction and navigation to get to the content. Therefore, summarization of video content for pervasive devices entails temporal as well as spatial considerations.

Pervasive devices usually have a smaller display resolution both in spatial domain and in color depth. Also, some devices such as Palm-OS PDAs do not have sound playing functionality. These constraints affect the key shots selection process of a video summarization system. For instance, if audio information is not available in the decoder, we may need to annotate more text information for video or extract video clips with embedded texts. Also, the limitation of remaining battery power in the devices may require the video summarization system to choose invariant stable scenes rather than high-motion video clips.

Most existing video summarization tools address their applications on the Internet environments. Browsing tools can display the summarized video using a number of key frames for each detected scene shot to generate a storyboard [20]. Some clustering techniques are used to optimize key frame selection based on their visual attributes or motion features [3, 12, 21]. Merialdo *et.al.* generate personalized TV news programs based on user preference and time constraint [13]. Gong and Liu use Singular Value Decomposition (SVD) of attribute matrix to reduce the redundancy of video segments and thus generate video summaries [5, 6]. In addition to the systems based on audio-

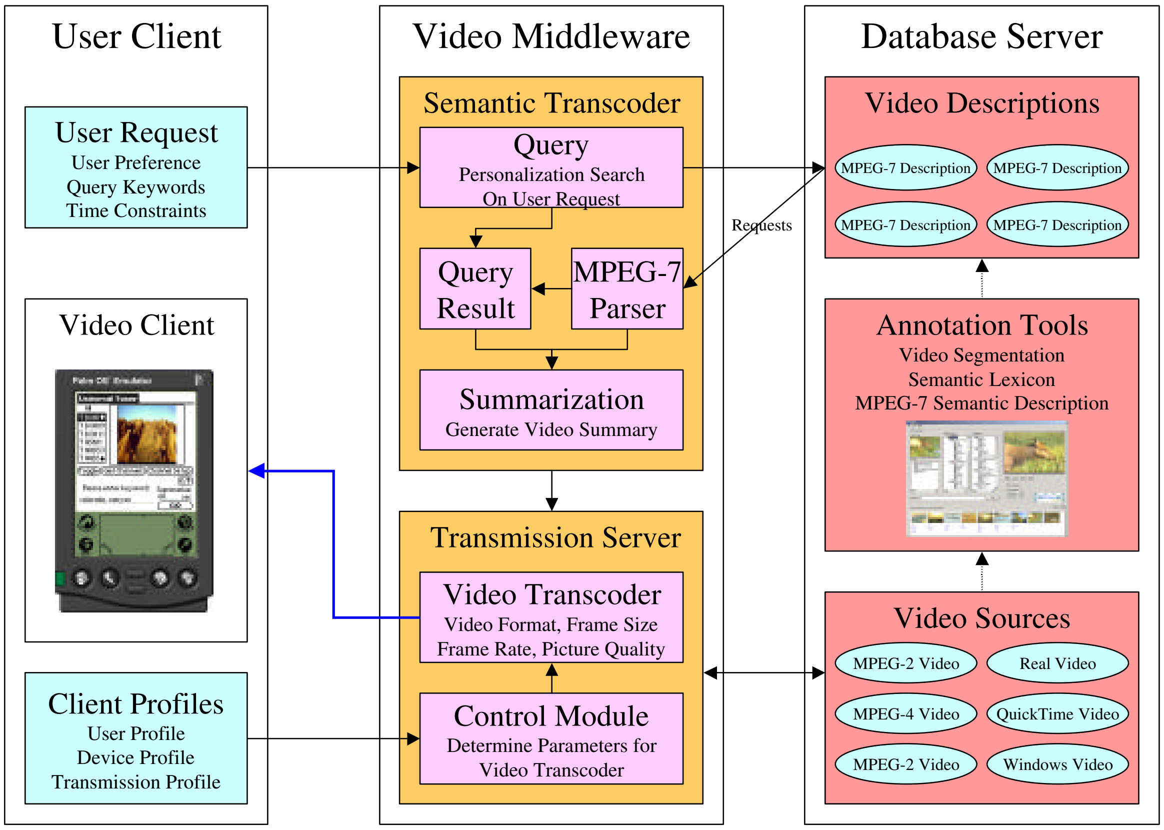


Figure 1: Block Diagram of Video Semantic Summarization System.

visual information, some researchers have proposed methods to detect semantic important events based on other resources. For instance, Aizawa *et. al.* use brain waves to detect exciting moments of the subjects [2]. Hu *et. al.* use detect similar video clips cross different source of news stations to identify interesting news events [8]. In industry, companies such as NTT DoCoMo and Virage have implemented preliminary video summarization systems. NTT DoCoMo streams video summaries to its I-mode cellular phones [17]. Virage creates video clips of NHL hockey highlights [19].

This paper addresses issues of designing a video semantic summarization system in the wireless/mobile environments. We have designed and implemented a video summarization system, which includes an MPEG-7 compliant annotation interface, a semantic summarization middleware, a real-time MPEG-1/2 video transcoder for Palm-OS devices, and an application interface on color/black-and-white Palm-OS PDAs. We will first describe the system architecture of our video summarization system, and then focus on video summarization of annotated contents, which are segmented and classified. The video contents are annotated using the ISO standardized Multimedia Content Description Interface, also known as MPEG-7. MPEG-7 addresses specific requirements for Description Schemes to describe different abstraction levels and variations of multimedia content.

This paper is organized as follows. In Section 2, we describe our video summarization system, which includes the user client, the video transcoding and summarization middleware, and the database server. In Section 3, we describe the annotation tool for the summarization system. Section 4 shows the detailed hierarchical techniques we use in the system. In Section 5, we conclude this paper with demonstrated applications and identify further directions.

2. SYSTEM OVERVIEW

Our Video Semantic Summarization System comprises of three major components, *the user client*, *the video middleware*, and *the database server*. Figure 1 illustrates the block diagram of these components. The user client component, which is shown in the left module of the figure, allows the user to specify his or her preferences along with client profiles, and receives the personalized summaries on the video client. The database server component, which is displayed in the right module of Figure 1, stores all the video contents as well as their corresponding MPEG-7 descriptions. The video middleware represents the intermediate component of the figure, and processes the user preference with the video descriptions to generate a personalized summarization, which is appropriately transcoded and transmitted to the user. The following subsections describe the three system modules in further detail.

2.1 User Client

The user client allows a user to ask for some video by specifying a user request and sharing his/her client profile. In response, the video client receives and displays the personalized video to the user. The user client module of Figure 1 illustrates these three components and their data exchange with the video middleware module. The user initiates a user request for some specific content to the query block of the video middleware. The user request can take the form of preference topics, certain keywords, and the user's time constraint for watching the video.

At the same time, several client profiles are also sent over to the control module of the video middleware, including the user profile [i.e., English speaking American, user residing in New York City], the device profile [i.e., Palm IIIc with limited color display, no audio capabilities, and memory capacity], and the transmission profile [i.e., internet access through 56K modem.] These client profiles allow the middleware to dynamically perform the appropriate video transcoding on the requested video content. Thus the delivered multimedia content can be perfectly customized and optimized for the current user environment.

Following, a personalized video is delivered to the user and displayed on the video client. The video client receives the multimedia stream from the video transcoder of the video middleware in accordance with the client profile and the user request. The video client application is built based on our Universal Tuner system [7]. This system includes two parts -- a software video transcoder at the server end, which transcodes MPEG-1/2 video or live broadcast video into client dependent formats, and the client application software on the color or black-and-white Palm OS PDA.

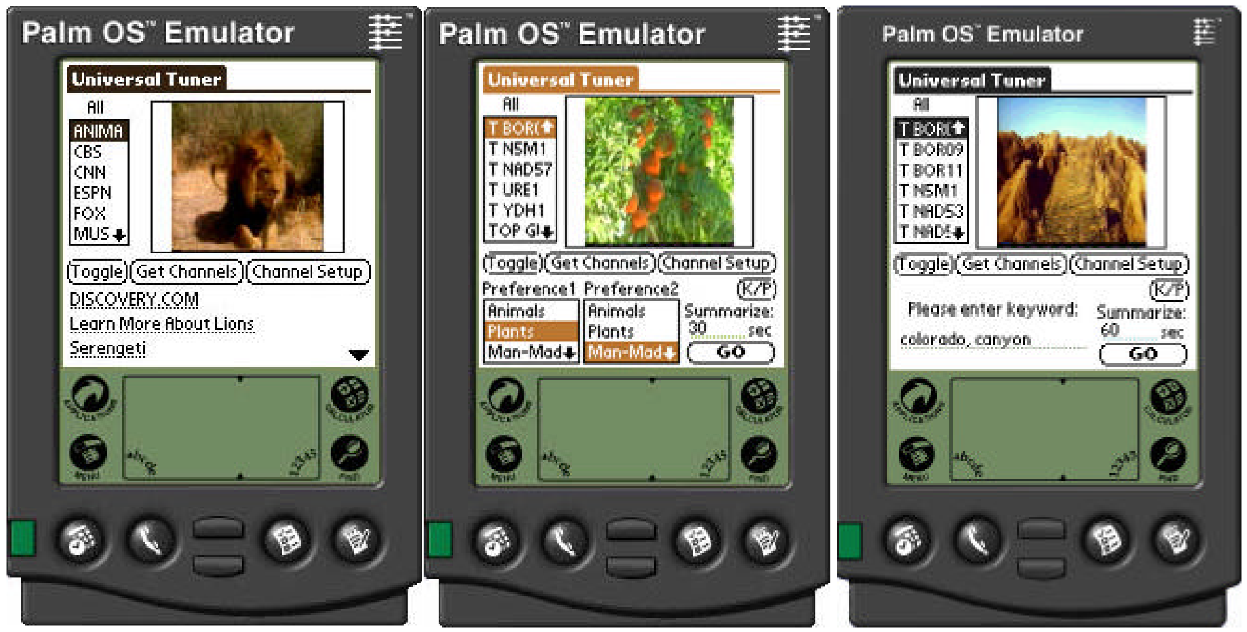


Figure 2: The User Client Interface on the Palm OS PDA for 3 Personalized Video Scenarios: (a) Video-on-demand with interactive hyperlinks, (b) Summarized video based on preference topics and time constraint, and (c) Summarized video based on query keywords and time constraint.

Figure 2 illustrates the user client interface on the Palm OS PDA for three personalized video scenarios. The first emulator, Figure 2(a), shows video-on-demand on the Universal Tuner client application. The interface allows the user to select among different channels of video content. Furthermore, interactive hyperlinks are embedded in the video content and are shown as clickable hypertext on the device. The second emulator, Figure 2(b), illustrates the video client interface of our summarization scenario based on preference topics. The user selects a video to summarize according to two listed preference choices and a total playing time constraint. Similarly, the third emulator of Figure 2(c) depicts the video client of another summarization scenario based on query keywords. The user is only interested in multimedia content with the specified keywords, and requests the video summary to be limited to the desired time constraint.

In summary, the user client allows the user to specify his/her preference, provides critical persistent data through the client profiles, and interfaces the user input/output with a video client. The user client communicates through the video middleware in order to retrieve the appropriate personalized contents. In the following subsection, the database server, which is where the video content resides, is described.

2.2 Database Server

The database server provides the video middleware with the video descriptions and corresponding video sources in order to generate the personalized contents. In the database server, video content is stored, analyzed and annotated with MPEG-7 descriptions. Figure 1 illustrates the database server with three major components. First, the video sources identify the location and format of our content. Because our video transmission transcoding middleware accepts various kinds of video file types, bit-rates and resolutions, the database can store videos in any of the most popular formats (*e.g.,* MPEG-1/2, AVI, and QuickTime). Also, even though each output summarized video is a composition of video clips, our database does not need to pre-segment the video sequences into individual shots. Each video can be saved in only one format, because the video transcoding middleware can randomly access/decode any video clips in the video sequence at real time. Second, manual, semi-automatic, or automatic annotation tools assist to generation the descriptions for videos. The annotation can range from high-level semantic concepts to low-level feature descriptions. And third, the video descriptions are stored as MPEG-7 XML files. These MPEG-7 descriptions identify the underlying content of the videos.

Annotation tools generate semantic meanings as well as other feature descriptions of the video, and output them as MPEG-7 description schemes. Our annotation tool allows annotators to tag video segments with detail text descriptions, which is based on a defined semantic lexicon, on individual shots or individual regions/objects on the key frame of the shots. In addition, depending on the application of the semantic summarization system and the corresponding video content, the annotation tool can be easily customized. For instance, the lexicon can be changed based on the content, or the shot boundary detection method can be customized to a hierarchical structure, which represents hierarchical semantic events. Section 3 describes the functions and features of the video annotation tool in more details.

2.3 Video Middleware

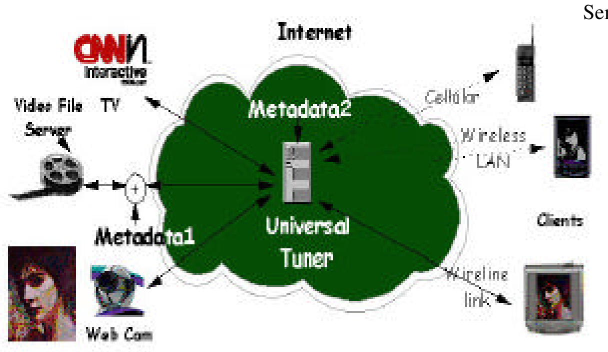
The video middleware interfaces the user client and the database server. As shown in Figure 1, the middleware consists of the semantic transcoder and the transmission server. In the semantic transcoder, a query and retrieval component matches the user preference with the MPEG-7 video descriptions to generate a video summarization. In the transmission server, the control module determines the appropriate transcoding for the desired video content to the user client, which in turn depends on the client profiles. Afterwards, the summarized videos are optimally transcoded and transmitted to the user's video client. In the following two subsections, the semantic transcoder and the transmission server of the video middleware are explained.

2.3.1 Semantic Transcoder

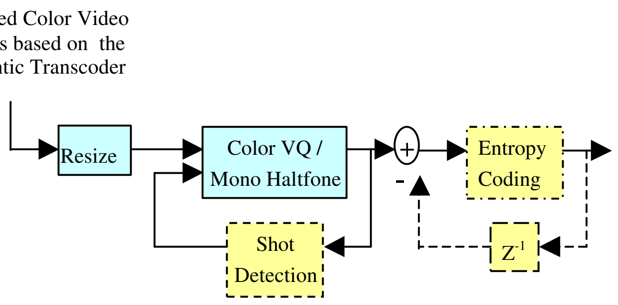
The semantic transcoder performs the matching of the user request with the video description. The user may specify his/her request in terms of preference topics, topic ranking, query keywords, and time constraint. The query module receives the user request to determine if there are contents that will fit these preferences. A search request is send to the database server, which in turn responds with the appropriate MPEG-7 video descriptions. The MPEG-7 descriptions are initially passed to the MPEG-7 parser and the desired query results are extracted. These query results identify the matched semantic descriptions and the corresponding video segments.

Having matched the user preference with the video descriptions from the MPEG-7 XML, the appropriate videos are summarized and delivered to the user. However, there are two additional issues that need to be addressed before the summarized video segments are send back to the user. First, there may be too many query results that satisfy the search request. Consequently, the resulting summarization can be refined based on the priority ranking, playing time, and the user profile. These different summarization techniques are discussed in more detail in Section 4.

The second issue concerns the format and transmission of the video segments. If the video sources in the database server are stored in varying formats, bit rates, frame rates, image sizes, and other variations, the video segments that comprise the summarization are likely of different variations. These variations may not be compatible or optimized for the user's video client. As a result, the transmission server dynamically transcodes all the video segments and streams a compatible video summary to the user, as to be discussed in the next subsection.



(a)



(b)

Figure 3: (a) System description of the Universal Tuner video transcoding middleware; (b) Block diagram of video transcoder.

2.3.2 Transmission Server

The Transmission Server, which is called *Universal Tuner,* is designed for streaming video on various hand-held or mobile phone devices through wireless. We have implemented a prototype of a variable-complexity codec that can transcode AVI, MPEG-1/2 video or live TV broadcasting in real-time and stream it to the Palm-OS Personal Digital Assistant (PDA) devices. Figure 3(a) shows the system architecture of Universal Tuner.

Real-time compression or decompression of even simple multimedia standards can become impractical on complexity-constrained mobile devices. For example, we encountered real complexity constraints while implementing a relatively simple animated GIF decoder on a Palm OS PDA operating with a 20 MHz Dragonball processor [14]. We measured decoding times for 80x80 GIF thumbnails to be one to five seconds per frame. For 8x8 or 16x16 DCT-based multimedia standards (such as JPEG or MPEG-1/2) that need more operations, the decoding speed on the Palm OS devices is far from real time.

In our Universal Tuner project, we designed a system that eliminates the complexity at the decoder by utilizing a variable partitioned-complexity strategy. We design the encoding algorithm at the transcoding server and the decoding algorithm at the PDA client using a cascade of modules, which can be selectively disabled or enabled to the device's effective processing capability. We envision that our partitioned design will be operating in an environment in which battery life continues to serve as a primary constraint on wireless handhelds while next-generation wireless links may support greater bit rates. E.g., GPRS and 3G wireless systems intend to support bit rates on the order of 200 kbps to 1 Mbps per user. In such an environment, we feel that conserving battery power is a greater concern than conserving bandwidth.

Even though the speed of PDA's CPUs may increase in the future, power may be still a practical constraint that limits the application of multimedia on portable devices [11]. Next-generation PDA's will feature low power processors such as the Intel StrongARM that permit an operating system to conserve battery life by lowering the clock speed and the voltage [9]. Multimedia algorithms are hungry to consume processing cycles, and hence power. These complex algorithms will directly conflict with power conservation algorithms attempting to reduce the power consumed by a CPU and/or Multimedia DSP chip. Low power processors that operate at the lower end of their range in order to conserve power can effectively behave as complexity-constrained processors, thereby rendering impractical real-time processing of complex DCT-based multimedia standards such as MPEG-1/2/4 and H.26x [18].

The Universal Tuner system includes two parts -- a software video transcoder at the server end which can transcode MPEG-1/2 video or A/D converted live broadcasting video into client dependent format, and a software client application software on the color or Black-and-White Palm OS PDA. The software client part consists the key display component of the user client in this Video Semantic Summarization System.

An example of the client end has been shown in Figure 2, this application is capable of displaying transcoded MPEG1/2 color video in both 80x80 format (as shown in Figure 2) and full-screen 160x160 viewing mode. Using Palm IIIc emulator, we can show video in the 80x80 video mode at about 6 frame per second, where motion could be marginally considered as continuous in human perception, and 1.5 frame per second in the 160x160 video mode, which is perceptually similar to the slide show rather than the continuous motion video. In the client application end, we also added WML compatibility in the small video mode that can access information on the Interent through the transcoding server.

Figure 3(b) shows the block diagram of our current implementation of video transcoder. The input of the transcoder is a stream of video frames that is decoded from an AVI or MPEG-1/2 file on the database server. The transcoder randomly access and decode frames from a video (or multiple videos) based on the semantic transcoding output, which is a list of the video clips to be transmitted. Another functionality, which is not used in the Video Semantic Summarization System, of the transmission server is to convert frames from live TV signals through the frame grabber PC card or webcam inputs. In the transcoder, these RGB video frames are first resampled to either 80x80 or 160x160 image. Then, depending on whether the PDA device is Black and White or 256 colors (as in Palm IIIc), the color RGB frames are either dithered using halftoning or mapped to 256 colors using a shot-based optimal color map. In the color mode, we added a shot boundary detection functionality in order to update the color codebook for each new shot. After the color mapping or BW halftoning, then the transcoder can selectively enable an entropy coding process and an frame difference process that losslessly compress the coded bitstream.

We have tested the effectiveness of the BW mode of our Universal Tuner in the real environments through a 9600 bps wireless modem. For the color modes, we have also tested our Video Semantic Summarization System on Palm m505 devices, which support USB connection and Wireless LAN using 48kbps. Our testing results show the effectiveness of the system. In the future, we will test our system in the 3G and Bluetooth environments.

3. MPEG-7 VIDEO ANNOTATION TOOL

The effectiveness of the Video Semantic Summarization System is highly dependent on the annotation descriptions of our content. If the annotation tool generates useful and detailed MPEG-7 descriptions for an application, then the resulting summarization will be comprehensive and desirable. In this section, we outline the functionalities and feature attributes generated by our annotation tool, called IBM VideoAnn.

Four major components describe the annotation process and are depicted in Figure 4. First, video segmentation is performed to cut up the video sequence into smaller video units. Second, semantic lexicon is defined in order to regulate the video content descriptions. Third, an annotator labels the video segments with the semantic descriptions and relevance scores are also calculated to reflect the importance with respect to the labels. Fourth, the MPEG-7 descriptions of the annotation process are directly outputted from the IBM VideoAnn Annotation Tool. The goal of the video annotation is to categorize the semantic content of each video unit, assign the corresponding relevance

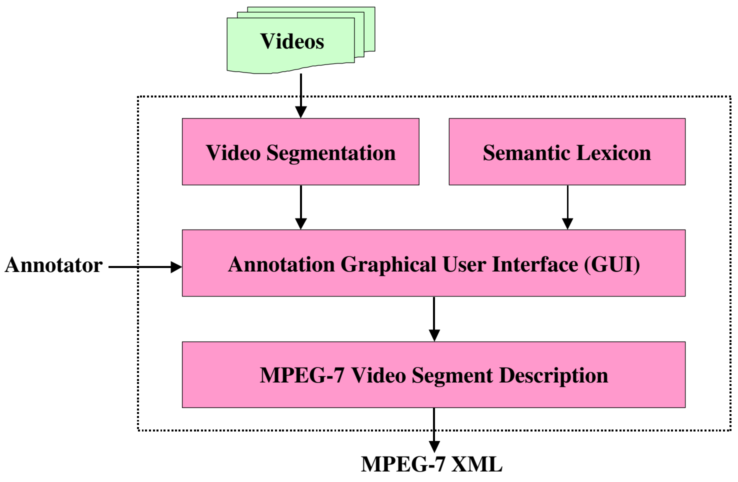

Videos
Video Segmentation
Semantic Lexicon
Annotator
Annotation Graphical User Interface (GUI)
MPEG-7 Video Segment Description
MPEG-7 XML

Figure 4: Four Major Components of the IBM VideoAnn Annotation Tool.

score, and output the MPEG-7 XML description file. The following four subsections describe these components in further detail.

3.1 Video Segmentation

A short video clip can be simply annotated by describing its content in its entirety. However when the video is longer, annotation of its content can benefit from segmenting the video into smaller units. In our IBM VideoAnn Annotation Tool, the annotation is performed on the video shot level. A video shot is defined as a continuous camera-captured segment of a scene, and is usually well defined for most video content. Given the shot boundaries, the annotations are assigned for each video shot.

For a video sequence, shot boundary detection is performed to divide the video into multiple shots. The IBM CueVideo Toolkit performs the shot detection algorithm, which is based on the multiple timescale differencing of the color histogram [1]. CueVideo segments our video content into shorter shots, where scene cuts, dissolves, and fades are effectively detected. Because each video shot can be described and retrieved independently of each other, the next step is to define our lexicon for shot descriptions.

3.2 Video Content Semantic Lexicon

Given the segmentation of video content into video shots, the second step is to define the semantic lexicon in which to label the shots. A video shot can fundamentally be described by three attributes. The first is the background surrounding of where the shot was captured by the camera, which is referred to as the *static scene*. The second attribute is the collection of significant subjects involved in the shot sequence, which is referred to as the *key object*. Lastly, the third attribute is the corresponding action taken by some of the key objects, which is referred to as the *event*. These three types of lexicon define the vocabulary for our video content.

An example of our lexicon is shown as follows. Our vocabulary for the static scenes includes “indoors”, “outdoors”, and “outer space”. Furthermore, each category is hierarchically sub-classified to comprise more specific scene descriptions. For example, “outdoors” consists of these three sub-categories: “natural scene – low level”, “natural scene – high level”, and “man-made”. Our simplified vocabulary for the key objects includes the following categories: “animals”, “human”, “man-made structures”, “man-made objects”, “nature objects”, “graphics & text”, “transportation”, and “astronomy”. In addition, each key object category is subdivided into more specific object descriptions; for instance, "rockets", "fire", "flag", "flower" and "robots." For our events vocabulary, only six events are of specific interest to our summarization work, and they are the following: "water skiing", "boat sailing", "person speaking", "landing", "take-off or launch", and "explosion."

Using the defined vocabulary for static scenes, key objects, and events, the lexicon is imported into our IBM VideoAnn Annotation Tool for describing and labeling each video shot. The shots are labeled for its content with respect to the selected lexicon. Note that the set of lexicon is dependent on the summarization application, and can be easily modified and imported into the annotation tool.

3.3 VideoAnn Graphical User Interface

The IBM VideoAnn Annotation Tool assists authors in the task of annotating video sequences. Each shot in the video sequence can be annotated with static scenes, key objects, events, and other keywords. These descriptions are labeled for each shot and are stored as MPEG-7 descriptions in the output XML file. VideoAnn can also save, open, and retrieve MPEG-7 files in order to display the annotations for corresponding video sequences.

The VideoAnn is divided into four graphical sections as illustrated in Figure 5. On the upper right-hand corner of the tool is the *Video Playback* window with shot information. On the upper left-hand corner of the tool is the *Shot Annotation* with a key frame image display. On the bottom portion of the tool is two different *Views Panel* of the annotation preview. A fourth component, not shown in Figure 5, is the *Region Annotation* pop-up window for specifying annotated regions. These four sections provide interactivity to assist authors of the annotation tool.

The *Video Playback* window displays the opened MPEG video sequence. As the video is played back in the display window, the current shot information is given as well. The *Shot Annotation* module displays the defined semantic lexicons and the key frame window. The key frame is a representative image of the video shot segment, and thus offer an instantaneous recap of the whole video shot. This is the region where the annotator selects the descriptions for the video segment. The *Views Panel* displays two different previews of representative images of the video. The *Frames in the Shot* shows all the I-frames as representative images of the current video shot, while the *Shots in the Video* view (as in the bottom of Figure 5) shows all the key frames of each shot as representative images over the entire video. As the annotator labels each shot, the descriptions are displayed below the corresponding key frames in the *Shots in the Video* view. Furthermore after the MPEG-7 descriptions are saved into an XML file, anyone can load and review these files at a later time by previewing the annotations at this views panel. The *Region Annotation* window allows the author to associate a rectangular region with a labeled text annotation. After the text

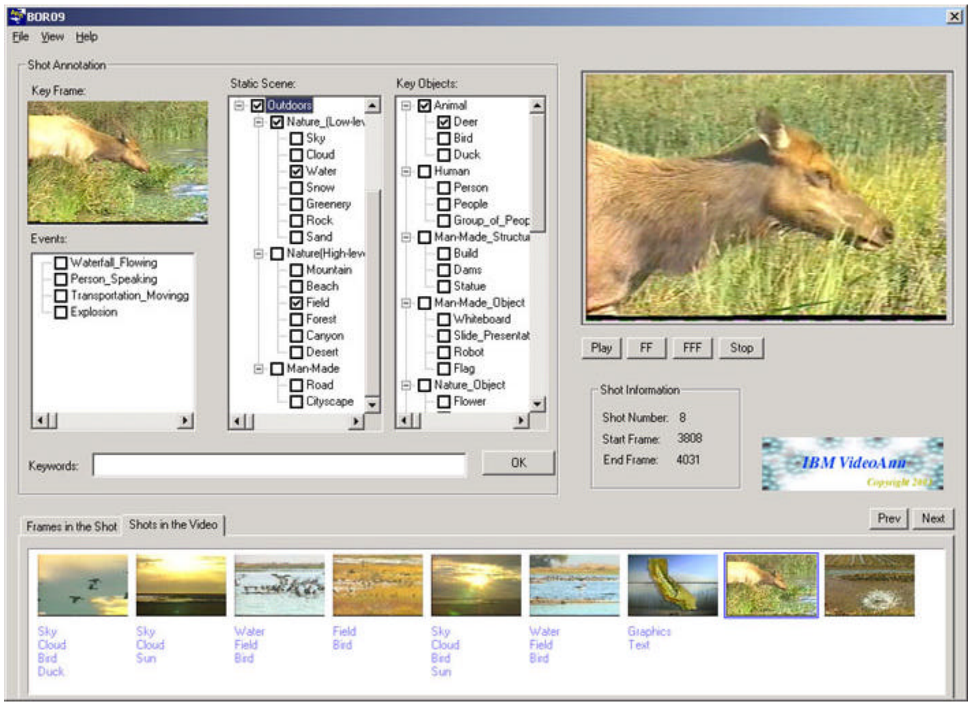


Figure 5: The Video Annotation Tool.

```
<Mpeg7 type="complete" xmlns="http://www.mpeg7.org/2001/MPEG-7_Schema"
      xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.mpeg7.org/2001/MPEG-7_Schema">
  <ContentDescription xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="VideoType">

      <Video>
       <TemporalDecomposition>
        <VideoSegment>
           <MediaTime>
               <MediaTimePoint>T00:00:00:00F30</MediaTimePoint>
               <MediaIncrDuration timeUnit="PT1N30F"> 136 </MediaIncrDuration>
           </MediaTime>
           <TextAnnotation type="scene description" relevance="1" confidence="1">
             <FreeTextAnnotation> Slide Presentation </FreeTextAnnotation>
           </TextAnnotation>
           <SpatioTemporalDecomposition>
            <StillRegion>
               <MediaIncrTimePoint timeUnit="PT1N30F"> 82 </MediaIncrTimePoint>
               <SpatialDecomposition>
                  <StillRegion>
                     <SpatialLocator>
                        <Poly>
                           <CoordsI> 41 135 290 135 290 230 41 230 </CoordsI>
                        </Poly>
                     </SpatialLocator>
                     <TextAnnotation>
                        <FreeTextAnnotation> Graphics & Text </FreeTextAnnotation>
                     </TextAnnotation>
                  </StillRegion>
               </SpatialDecomposition>
             </StillRegion>
            </SpatioTemporalDecomposition>
          </VideoSegment>
         </TemporalDecomposition>
       </Video>

    </MultimediaContent>
  </ContentDescription>
</Mpeg7>
```

Figure 6: Example of MPEG-7 Video Segment Description XML file

annotations are identified on the *Shot Annotation* window, each description can be associated with a corresponding region on the selected key frame of that shot. The region annotation is also saved in the MPEG-7 descriptions, as is described next. A more detailed description of the annotation tool as well as its active learning components are shown in [16].

3.4 MPEG-7 Video Segment Description

The IBM VideoAnn annotation tool segments the video content into shots, labels each video shot with some descriptions, identifies the associated region boundary, and generates an MPEG-7 XML description [10]. The ISO standardized MPEG-7 defines the compatible scheme and language to represent semantic meaning of multimedia content. Our MPEG-7 output is the Video Segment Description Scheme, as shown in Figure 6. In Figure 6, we annotate the first shot, which includes 136 frames, of a video as "*Slide Representation*" and annotate a rectangular region in the key frame as "*Graphics & Text*." In MPEG-7, each video shot is defined as a Video Segment, where the shot start-time (shown in the T*hh*:*mm*:*ss*:*nn*F30 format) and duration are given and the annotations are described. Furthermore, the embedded <SpatioTemporalDecomposition> tag allows us to specify the region location and the corresponding text annotation in a key frame. In Figure 6, the key frame is the 82th frame of the video sequence. The annotated region is specified by the <SpatialLocator> tag. It is identified by a polygon whose *n* vertex coordinates are recorded in the order of <x_0, y_0>, <x_1, y_1>, …, <x_{n-1}, y_{n-1}> after the <CoordsI> tag . For multiple regions in a key frame, the system needs to repeat the section between <StillRegion> and its closing tag inside the <SpatialDecomposition> section. If the annotator needs to label multiple frames in the shot, then the system needs to repeat the <StillRegion> section inside the <SpatioTemporalDecomposition> section.

In our Video Semantic Summarization System, a relevance score is automatically assigned to the video based on the confidence value of the classification. For our system, the annotation process generates a relevance score for the whole video sequence and for each attribute based on the probability of that attribute to the corresponding video unit. After these steps, we implemented an interface to allow users to manually correct the annotation as well as the scene boundaries. All these results are then saved as an MPEG-7 XML file.

4. SUMMARIZATION TECHNIQUES

Given annotation descriptions and corresponding scores for our video content, this section describes the summarization techniques to customize video delivery based on user preference and time constraint. The objective of video summarization is to show a shortened video that maintains as much semantic content within the desired time constraint based on user preference. Using shots as the basic video unit, there are four forms of video summarization based on temporal compression of the original video sequence: (1) maintain or delete each video shot depending on user preference, (2) extract temporal subsets of the original shot depending on attribute specification, (3) condense each shot in fast-forward temporal mode while maintaining comprehension, and (4) combine a weighted combination of the previous forms.

In the first formulation, each video shot is either included or excluded from the final video summary. In each shot, video annotation describes the semantic content with attributes and corresponding scores. Assume there are a total of *N* attribute categories. Let $\vec{P} = [p_1, p_2, ..., p_N]^T$ be the user preference vector, where p_i denotes the preference weighting for attribute *i*, $1 \le i \le N$. Assume there are a total of *M* shots. Let $\vec{S} = [s_1, s_2, ..., s_M]^T$ be the shot segments that comprise the original video sequence, where s_i denotes shot number *i*, $1 \le i \le M$. Subsequently,

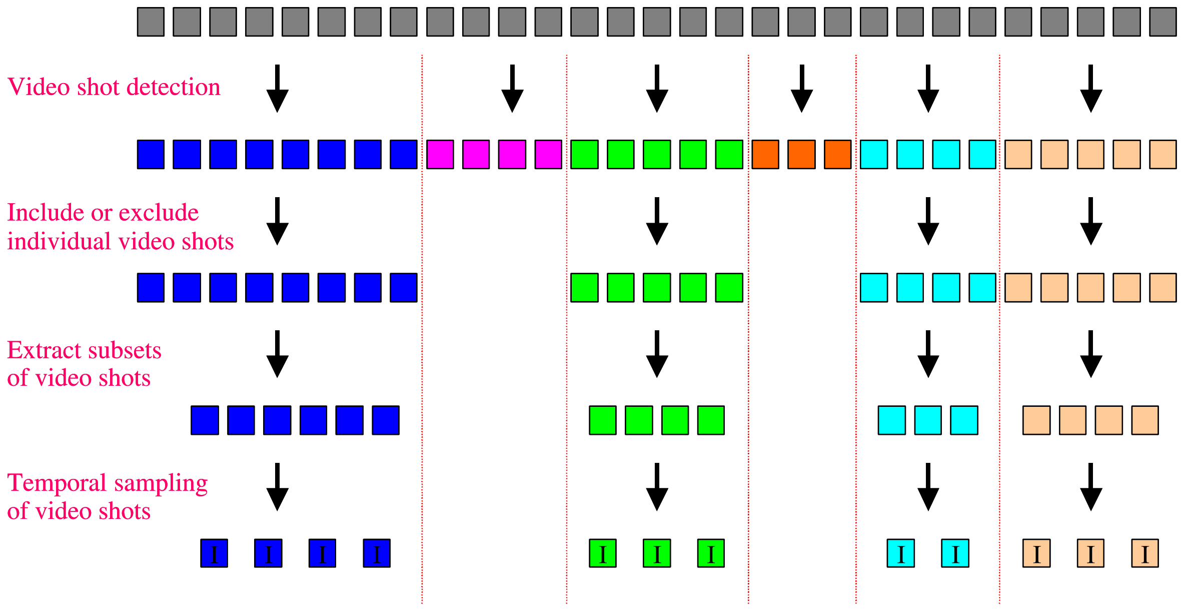

Video shot detection
Include or exclude individual video shots
Extract subsets of video shots
Temporal sampling of video shots
I

Figure 7: Three Forms of Video Summarization Techniques.

the attribute score $a_{i,j}$ is defined as the relevance of attribute *i* in shot *j*, $1 \leq i \leq M$ and $1 \leq i \leq N$. The attribute matrix ***A*** is:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & \cdots & \cdots & \cdots \\ \cdots & \cdots & a_{i,j} & \cdots \\ a_{M,1} & \cdots & \cdots & a_{M,N} \end{bmatrix}$$

It then follows that the weighted attribute w_i for shot *i* given the user preference $\vec{P}$ is calculated as:

$$\vec{W} = \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_M \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & \cdots & \cdots & \cdots \\ \cdots & \cdots & a_{i,j} & \cdots \\ a_{M,1} & \cdots & \cdots & a_{M,N} \end{bmatrix} * \begin{bmatrix} p_1 \\ p_2 \\ \cdots \\ p_N \end{bmatrix}$$

$\vec{W}$ specifies the weighted importance of each shot for the user. Assume shot s_i spans a durations of t_i, $1 \leq i \leq M$. Then this summary formulation suggests that we include shot s_i if the importance weighting w_i of this shot is greater than some threshold, $w_i > \theta$, and excluded otherwise. θ is determined such that the sum of the shot durations t_i is less than the user specified time constraint. As a result, each shot is either included or excluded in the final video summary.

In the second formulation, we extend the attribute scoring from the shot level to the time domain. In the first form, video annotation specifies a relevant score of each attribute for each shot based on semantic relevance of the shot content. Here, attribute scoring will incorporate low-level features to more precisely annotate each frame of the shot. For example in a sports clip, there should be higher relevance scoring associated with those temporal frames that demonstrate higher motion components. So if we want to include the high activity component of a shot, then only the highest score subset is extracted. Consequently, attribute score $a_{i,j}$ for attribute *i* and shot *j* is no longer a constant within a shot, but becomes a function of time based on low-level features. The attribute scoring function $a_{i,j}(t)$ can be calculated automatically as the product of the constant attribute score $a_{i,j}$ of shot *j* and a normalized low-level feature weighting $n(t)$, where $0 \leq n(t) \leq 1$ and *t* spans the duration of the shot. Thus,

$$a_{i,j}(t) = a_{i,j} * n(t)$$

Similar to the first formulation, the attribute scoring function $a_{i,j}(t)$ now determines the time interval $t \in [u,v]$ of shot subset $s_i(t \in [u,v])$ to be included in the final video summary.

In the third formulation, the video is compressed temporally by subsampling the total number of frames in the original sequence. The resulting video summary resembles a fast forward playback in a shorter period of time. When video is temporally compressed in this manner, maintaining comprehension becomes the most critical issue. Comprehension depends on the compression factor, which in turn is highly dependent on the content and the amount of motion in the original video. For example, commercial videos, which consist of high motion and very short shots, cannot tolerate a high compression factor. On the other hand, interview or conferencing videos, which consist of nearly stationary people with limited motion, can withstand high compression factors. Consequently, we can assume that to maintain comprehension of the summarized video, the cumulative motion of the resulting subsampled video must be below a perceptually acceptable threshold. This requirement then determines the maximum sampling rate to adopt for the video. Note that it is desirable to use one subsampling rate for the entire video, so as not to perceptually confuse the user when changing compression rates. This restriction also limits the sampling rate over the entire video. Let $m(t)$ denote the perceived motion associated with time t. Let φ denote the human tolerance limit for perceived motion. It then follows that we are looking to find the maximum period *T* such that the perceived motion is always limited by the human tolerance limit φ, $m(t+T) - m(t) \leq \varphi$ for all *t*. After the sampling period *T* is determined, the final video summary consists of subsampled frames of the selected video shots.

5. CONCLUSION

In this paper, we describe our video semantic summarization system, which includes an MPEG-7 compliant annotation interface, a semantic summarization middleware, a real-time MPEG-1/2 video transcoder on PCs, and an application interface on color/black-and-white Palm-OS PDAs. Several issues regarding to designing a wireless compliant video summarization system have been addressed in this paper. Our video summarization system has these characteristics for the user of pervasive devices: 1.) Save time from navigating to the desired information [no clicking through a series of links]; 2.) Save time from viewing whole video clips [only show desired video segments]; 3.) Save bandwidth by downloading only desired video segments in the device-dependent formats. In the future, we will add more functionalities to this Video Semantic Summarization system such as mobile phone client applications and automatic semantic audio-visual indexing as well as event detection.

6. REFERENCES

[1] A. Amir, D. Ponceleon, B. Blanchard, D. Petkovic, S. Srinivasan, and G. Cohen, "Using Audio Time Scale Modification for Video Browsing", *Hawaii Int. Conf. on System Sciences*, HICSS-33, Maui, January 2000.

[2] K. Aizawa, K. I. Shijima, and M. Shiina, “Summarizing Wearable Video,” *Intl. Conf. on Image Processing*, Thessaloniki, Greece, Oct. 2001.

[3] N. Babaguchi, Y. Kawai, and Y. Kitahashi, “Generation of Personalized Abstract of Sports Video,” *IEEE Intl. Conf. on Multimedia and Expo*, Tokyo, Aug. 2001.

[4] R. Carlos and K. Uehara, “Video Summarization based on Semantic Representation,” AMCP, 1998.

[5] Y. Gong and X. Liu, “Generating Optimal Video Summaries,” *IEEE Intl. Conf. on Multimedia and Expo*, New York, Aug. 2000.

[6] Y. Gong and X. Liu, “Summarizing Video by Minimizing Visual Content Redundancies,” *IEEE Intl. Conf. on Multimedia and Expo*, Tokyo, Aug. 2001.

[7] R. Han, C.-Y. Lin, J. R. Smith, B. L. Tseng and V. Ha, “Universal Tuner: A Video Streaming System for CPU/Power-Constrained Mobile Devices”, *ACM Multimedia 2001*, Technical Demonstration, Ottawa, Canada, Oct. 2001.

[8] J. Hu, J. Zhong, and A. Bagga, “Combined-Media Video Tracking for Summarization,” *ACM Multimedia 2001*, Ottawa, Canada, Oct. 2001.

[9] www.intel.com, *Intel StrongARM SA-1100 Microprocessor for Portable Applications, Brief Datasheet*, Order # 278087-006, Sept. 1999.

[10] ISO/IEC JTC 1/SC 29/WG 11/N3966, Text of 15938-5 FCD Information Technology – Multimedia Content Description Interface – Part 5 Multimedia Description Schemes, Final Committee Draft (FCD) edition, March 2001.

[11] R. Kravets, P. Krishnan, “Application-driven Power Management For Mobile Communications,” *Wireless Networks*, vol. 6, no. 4, pp. 263-277.

[12] Y. Li, W. Ming, and C.-C. Kuo, “Semantic Video Content Abstraction based on Multiple Cuts,” *IEEE Intl. Conf. on Multimedia and Expo*, Tokyo, Aug. 2001.

[13] B. Merialdo, K. T. Lee, D. Luparello and J. Roudaire, “Automatic Construction of Personalized TV News Programs,” *ACM Multimedia 1999*, Orlando, FL, Sept. 1999.

[14] www.motorola.com, MC68EZ328 microprocessor

[15] K. Masumitsu and T. Echigo, “Meta-Data Framework for Constructing Individualized Video Digest,” *Intl. Conf. on Image Processing*, Thessaloniki, Greece, Oct. 2001.

[16] M. Naphade, C.-Y. Lin, J. R. Smith, B. L. Tseng, and S. Basu, “Learning to Annotate Video Databases”, *Proc of SPIE on Storage, Retrieval,for Media Database*, Vol. 4676, San Jose, Jan 2002.

[17] http://www.nttdocomo.co.jp

[18] G. J. Sullivan, T. Wiegand, and T. Stockhammer, “Using the Draft H.26L Video Coding Standard for Mobile Applications,” *IEEE Intl. Conf. on Image Processing*, Thessaloniki, Greece, Oct. 2001

[19] http://www.virage.com

[20] M. Yeung, B. Yeo, W. Wolf and B. Liu, “Video browsing using clustering and scene transitions on compressed sequences,” *Proc. SPIE on Multimedia Computing and Networking*, Vol. 2417, 1995.

[21] H.-J. Zhang, S. Y. Tan, S. W. Smoliar, and G. Y. Hone, “Automatic Parsing and Indexing of News Video,” *Multimedia Systems*, Vol. 2, No. 66, pp. 256-266, 1995.