Real Time Video Scene Detection and Classification

John M. Gauch, Susan Gauch, Sylvain Bouix, Xiaolan Zhu

Electrical Engineering and Computer Science
The University of Kansas

Abstract

The VISION (Video Indexing for Searching Over Networks) digital video library system has been developed in our laboratory as a testbed for evaluating automatic and comprehensive mechanisms for video archive creation and content-based search, filtering and retrieval of video over local and wide area networks. In order to provide access to video footage within seconds of broadcast, we have developed a new pipelined digital video processing architecture which is capable of digitizing, processing, indexing, and compressing video in real time on an inexpensive general purpose computer. These videos were automatically partitioned into short scenes using video, audio and closed-caption information. The resulting scenes are indexed based on their captions and stored in a multimedia database. A client-server-based graphical user interface was developed to enable users to remotely search this archive and view selected video segments over networks of different bandwidths. Additionally, VISION classifies the incoming videos with respect to a taxonomy of categories and will selectively send users videos which match their individual profiles.

1. Introduction

As a result of tremendous progress in video compression and transmission, the use of digital video in multimedia systems and over the Internet is becoming pervasive. In order to make intelligent use of this valuable resource, there is a need for content-based indexing and retrieval of digital video. This has motivated video library research at a number of institutions.

The VISION (Video Indexing for Searching Over Networks) digital video library system was developed in our laboratory as a testbed for evaluating automatic and comprehensive mechanisms for library creation and content-based search and retrieval of video over local and wide area networks (Gauch, et al., 1994, 1995, 1997). Our initial system was designed as an archival site for selected science and news videos from WGBH and CNN. These videos were automatically partitioned into short segments based on their content and stored in a multimedia database. A client-server-based graphical user interface was developed to enable users to remotely search this library and view selected video segments over networks of different bandwidths.

The value of any library is related to both the volume and timeliness of the information it contains. Digital video libraries are no exception. We are now able to have video footage added to the library within seconds of broadcast. In order to achieve this goal, it is necessary to perform video digitization, segmentation and compression in real time. Since VISION's original multipass architecture required video to be compressed and decompressed several times, it was not able to process video in real time. To address this problem, we have developed a new pipelined digital video processing architecture which is capable of digitizing, processing,

indexing, and compressing video in real time on an inexpensive general purpose computer.

The design and implementation of this new system is the subject of this paper. Section 2 describes related work in digital video libraries. Our pipelined video segmentation algorithm is described in section 3. Our video content classification technique is outlined in section 4. Section 5 describes our software architecture for real time video analysis. Finally, sections 6 and 7 discuss our experimental results and conclusions respectively.

2. Related Work

Digital video libraries distinguish themselves from traditional "video-on-demand" services or other similar projects in that they integrate image and video processing and understanding, speech recognition, distributed data systems, networks, and human-computer interactions in a comprehensive system. A key component of this difference is the use of content-based indexing and retrieval algorithms to enable users to interact with the video library rather than simply playing back entire movies or broadcasts. As a consequence, there has been considerable activity developing improved tools for video processing and content analysis. There has also been important progress made developing real-time multimedia systems capable of displaying video to users on different platforms and over a variety of networks. Systems which share features and goals with the VISION system are described below.

Several approaches have been proposed to decompose raw video into *shots* (a continuous roll of a camera) and *scenes* (collections of shots which occur in a single

location or are temporally unified). It is important to note that the above definitions follow usage defined by (Hampapur 1994) whereas some authors use the word *scene* to refer to a sequence of video representing continuous action and *story* to refer to a sequence of scenes. The problem of identifying *cuts* (sharp transitions between shots) has been typically approached from a bottom up perspective, looking for rapid changes in color histogram or image intensity (Arman, 1993; Nagasaka, 1992; Zhang, 1993). Model-based algorithms have also been developed to successfully detect fades, dissolves, and page translate edits (Hampapur, 1994). Once shots have been identified, keys frames which characterize the shot can be selected by considering the motion of objects within the shot. Here, we can either select frames which are as still as possible (Wolf, 1996) or identify the background and moving objects explicitly and select an image which focuses on one or the other (Sawheney, 1996). Another related approach is to combine information from multiple frames of an image sequence to create a "salient video still" which characterizes the shot in some way (Teodosio, 1993) or other forms of visual summaries (Irani and Anandan, 1998). These methods vary considerably in their computational complexity and effectiveness for different video sources, but each has its merits.

Although the problem of shot detection is essentially solved, the problem of combining shots to obtain scenes presents significant challenges. One approach used by the Princeton Deployable Video Library (PDVL) (Wolf, 1995; Yeo & Yeung, 1997) is to use identify key frames in each shot and use image-based clustering to construct a scene transition graph to visually present the relationships among shots. By browsing through a collection of graphs, users can locate scenes of interest (e.g., two person interviews). The scene transition graph can then be used to navigate through the video. The Algebraic Video System (Weiss, 1995) uses an alternative

technique where shots are organized in a hierarchical structure which allows nested stratification (subtrees may refer to overlapping portions of the raw video). This system uses the VuSystem (Lindblad, 1994) for recording and processing video but hierarchy construction is currently performed manually. A model-based approach has been proposed to parse video by an *a priori* model of the video structure (Zhang, 1995; Zhang et al, 1997). Such a model represents a strong spatial order within the individual frames of shots and/or strong temporal order across a sequence of shots. For example, it is required that all shots of the news anchorperson conform to a spatial layout. For many tasks it will be difficult or impossible to define models for the video. In their system the text description of the video contents are input by an operator. This yields high accuracy but makes production of large video collections very expensive.

Automatically identifying the content of a video segment is a particularly challenging problem. Three basic approaches have been investigated for this purpose: image understanding, speech recognition, and caption processing. Although the human visual system is very effective, research in computer vision over the past 20 years has had success in only limited domains (Haralick and Shapiro, 1992). For this reason, many approaches for image-based content identification have focused on feature-based classification schemes. For example, images can be indexed using color histograms (Swain, 1991) or combinations of shape and color features (Smoliar, 1994). The QBIC (Query By Image Content) project (Faloutsos, 1994) investigated methods to query large on-line image databases using the image contents, such as color, texture, shape, and size. Although feature-based classification is quite fast, one drawback is that very different objects may have the same features (e.g., a red car and a red apple).

Moving away from pure feature-based matching, it has been shown that similarity-based image retrieval can be also accomplished using Hidden Markov Models (HMM) which have been trained with representative images of outdoor scenes (rivers, trees, mountains) (Yu, 1995). Multiresolution wavelet decompositions have also been used for rapid image matching and retrieval (Jacobs, 1995). Here, a low resolution example (either hand drawn or scanned) is used as a query and multiscale matching is used to locate the most similar image in the database. More ambitious indexing based on texture, shape and appearance have been investigated within the Photobook system (Picard, 1994; Pentland, 1996). Although this system has had excellent success within a restricted domain of images (textures and faces) the computational expense associated with computing Eigenimages may limit its use for identifying video content.

Given the difficulty of image-based content analysis, processing the audio track and closed caption information is an attractive alternative. In one study [Turner, 1997], the words in closed captions were compared to index terms manually assigned by professionals. There was a strong overlap in the two sets (over 80% agreement), suggesting that the closed captions are a reasonably accurate indexing resource. However, alignment of the captions with shots must be done carefully. In the sample of Nova evaluated which, unlike news stories, is not captioned in real-time, 33% of the captions appear partially or wholly in adjacent shot(s).

The goal of the Informedia Digital Video Library project is to establish a large, on-line library featuring full content and knowledge-based search and retrieval of digital video, primarily for educational purposes (Christel, 1994; Christel, 1995; Wactlar, 1996). Informedia's News-On-Demand system (Hauptmann and Witbrock, 1997) specifically addresses the problem of providing efficient access to news videos

which requires entirely automatic segmentation and indexing. News-On-Demand uses the video, audio and closed captions for segmentation and also includes a large-vocabulary, speaker-independent, continuous speech recognizer. Speech recognition allows them to align the closed captions with the video and to provide words for indexing where closed captions are unavailable. Their archive is built using MPEG-I for the video, with one Pentium PC used for the digitization and a supplementary platform for the SPHINX-II speech recognition system.

The VISION system (Gauch, 1994, 1995, 1997) shares many of the goals of the News-On-Demand project, but we further constrain the problem by requiring a system which is capable of making all news stories available within seconds of their broadcast. In addition, we have designed the system to run continuously on a dual-processor Pentium PC. Thus, one can monitor multiple broadcast channels cost-effectively by devoting one commodity computer per channel, all of which feed the indexing information to a central database. Because of these constraints, we perform limited processing of the audio track during automated scene segmentation and content analysis. In particular, audio information is used only to combine shots. Closed captions are required in almost all television broadcasts, and they are a valuable source of information for video segmentation. The text associated with each scene is also used as input to our full-text retrieval engine to search the video library for material of interest.Our major contribution is an exploration of what can be achieved in entirely automatic archive construction running real-time on commodity hardware.

3. Video Segmentation

From a logical point of view, production video footage can be thought of as a collection of scenes which illustrate different subtopics. Each scene typically consists of several camera shots which have spliced together in some manner. For example, a news story often consists of shots of the announcer discussing the story, remote shots of reporters giving interviews, and other shots illustrating the event. The goals of video segmentation are: (1) to locate the start and end of each camera shot, and (2) to combine camera shots based on content to obtain the start and end points of each scene. In order add "live" news and information to the video library, it necessary to perform video segmentation in real time as it is broadcast.

3.1 Shot Detection

The detection of shot transitions can be trivial or complex depending on the video content being combined and the type of transition used. For example, when video from two very different sources are spliced together with zero frames of transition it is easy to detect the scene change. On the other hand, if two very similar shots are combined with a gradual cross fade, the visual changes may be much smaller than we might expect in a video with moderate object motion. Thus, it is very likely that any automated image-based shot detection algorithm will miss some fraction of the shot boundaries. Fortunately, this does not impact the quality of the scene detection greatly because shot transitions which are this gradual are often chosen by producers because the two shots are actually related and should remain in the same scene.

Several approaches to the problem of automatic location of camera motion breaks in video sequences have been investigated. Nagasaka and Tanaka (Nagasaka, 1992) have evaluated a number of image processing measures for detecting cut edits in

video sequences by detecting shot boundaries in digital video. Their conclusion is that the best measurement is the sub-window-based histogram comparison. Zhang et al (Zhang, 1993) have also presented the evaluation of different image processing routines for detection of cut edits. They tried to detect special effects having gradual transitions like fades and dissolves by using a dual threshold. Hampapur et al (Hampapur, 1994) approached the problem of digital video segmentation by proposing a model for video based on the production process and classifying video edit effects based on these models. The edit effect models are used to design feature detectors, which are used in a feature-based classification approach to segment the video. Arman, Hsu, and Chui (Arman, 1993) presented a technique operating directly on compressed video detect shot boundaries. Their technique relies on the properties of the coefficients of the discrete cosine transform used in encoding the video to detect the transitions.

Shot detection in the VISION system is performed by combining three image cues: (1) the average brightness B(t) of each video frame, (2) the change in pixel values dP(t) from frame I(t) to frame I(t+dt), and (3) the change in color distribution dC(t) from video frame I(t) to frame I(t+dt). These three quantities are compared to dynamic thresholds to identify potential shot boundaries. Specifically, we identify frame t to be a shot boundary if

$(B(t) < B_{threshold})$ or $((dC(t) > C_{threshold})$ and $(dP(t) > P_{threshold}))$

where

$$B(t) = \sum_{\forall x,y} I(x,y,t) \ ,$$

$$dP(t) = \sum_{\forall x,y} |I(x,y,t) - I(x,y,t-dt)|,$$

$$dC(t) = \sum_{\forall c} |h(c,t) - h(c,t-dt)| \ .$$

The color histogram h(c,t) for each frame can be computed using

$$h(c,t) = \sum_{\forall x,y} \begin{Bmatrix} 1 \ if \ I(x,y,t) = c \\ 0 \ otherwise \end{Bmatrix}$$

after the input image has been quantized to 256 uniformly distributed colors. If the digitizer produces an 8-bit image directly, we calculate the color histogram using the colors provided by the digitizer. When computational time is limited, the parameters above can be estimated using a sub-sampled version of the input image. We have obtained almost identical results using 640x480 and 320x240 images. Acceptable results are also possible with 160x120 images, but accuracy suffers.

The selection of $B_{threshold}$, $C_{threshold}$, and $P_{threshold}$ is challenging because no set of thresholds will be effective for all sources of production video. We have addressed this problem in two ways. First, we use *a priori* knowledge of each video producer (e.g., WGBH, CNN, CNBC) to select initial values for these thresholds based on which video source is being processed. Then, we gather statistical information about B, dC, and dP during video processing to update the thresholds dynamically every few minutes.

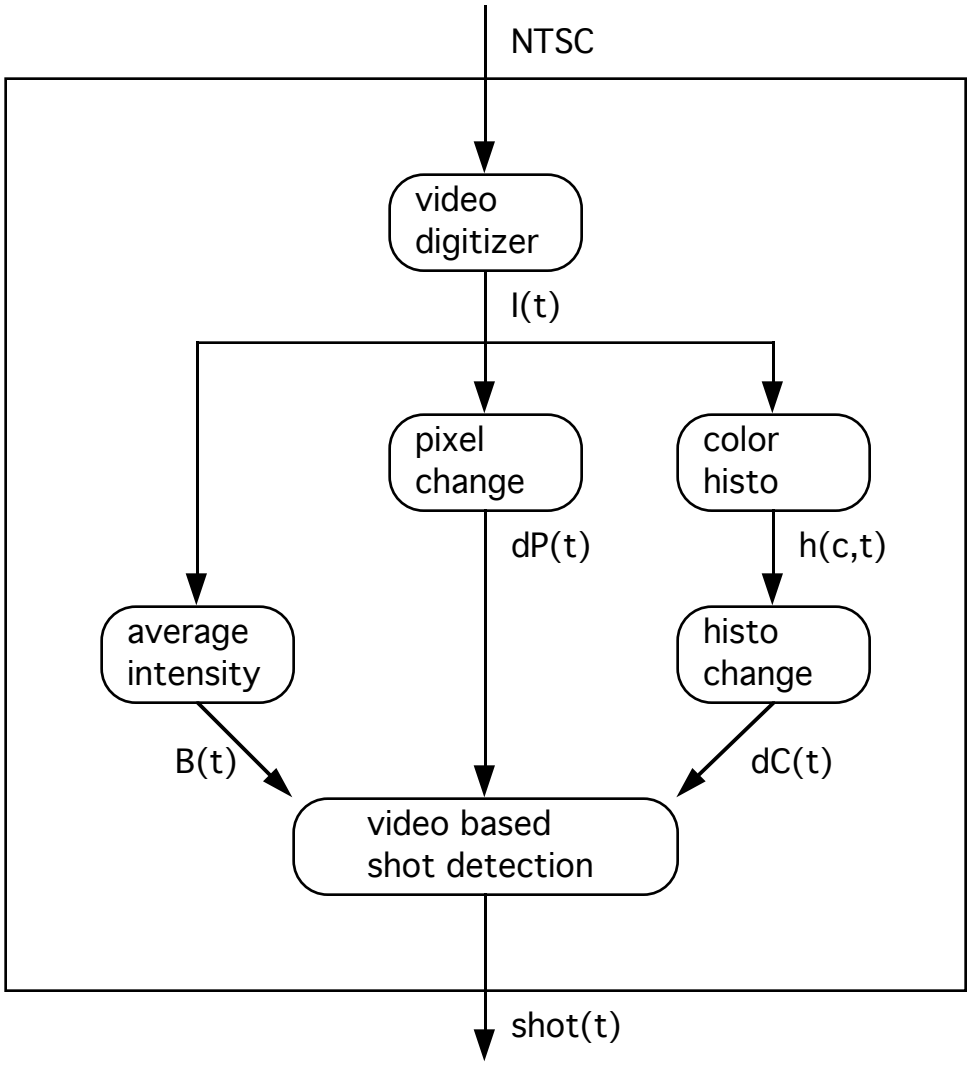

Figure 1. Data flow diagram for shot detection.

Our process of shot detection is illustrated in Figure 1. In this data flow diagram, raw NTSC video is input. The Boolean output function shot(t) is true if a shot boundary has been detected at frame t, and false otherwise.

3.2 Merging Shots to Obtain Scenes

Since many video producers use motion and shot transitions to attract and retain viewer interest, it is common to have numerous shots per scene. Merging related shots back together to identify scenes is very important to avoid excessive

fragmentation of information in the library. There are two sources of information which can be exploited for this purpose: audio cues, and closed caption cues.

Given our requirement for real time video segmentation, we focus on low level audio properties. If someone is talking while there is a shot transition, it is an indication that the two shots are related and should be merged. To determine if this situation occurs, we perform endpoints detection on the audio signal to identify the start and end of each utterance. This is done by computing the short-time energy function using n audio samples centered in time about frame t as follows:

$$E(t) = \sum_{k=0..n-1} A(t \cdot n + k - n/2).$$

We merge adjacent shots together if $E(t) > E_{threshold}$. The value of $E_{threshold}$ is again chosen using *a priori* information about the video source and updated dynamically using audio statistics.

The second source of information is actually the most important for the VISION system. Since closed captions are now embedded in almost all broadcast video, a transcription of the audio channel can be obtained by decoding caption data in line 21 of field 1 of each video frame. In our current system, we use a stand alone product TextGrabber by Unitec Inc. to capture this information. One problem with some video sources is that the closed captions are entered as the show is broadcast. This introduces a 2-3 second time delay between when words are spoken and when the transcription appears. Hence, it is necessary to estimate the time delay and realign the closed captions with the audio and video.

Once caption alignment has been performed, it is possible to consider the topics being discussed on either side of a shot boundary. If they are similar, although there is a change of shot, there is no change of scene and the shots should be merged. For each shot boundary, we consider the words used within a window of a given number of frames on either side of the boundary (adjusted by the delay factor). We tokenize the closed captions to identify words, then use the Porter stemmer (Frakes and Baeza-Yates, 1992) to remove prefixes and suffixes, and finally delete the most frequent English words (*stopwords*) from the captions in each window. The remaining terms are then weighted, where the weight for term w in video window v is calculated as follows:

$$Wt(w, v) = tf_{wv} * idf_w$$

where

f_{wv} is the frequency of term w in window v

$idf_w = \log_2 (freq_{max}/freq_w)$

$freq_{max}$ = frequency of most frequent term in related text collection[1]

$freq_w$ = frequency of term w in related text collection[1]

We then use the cosine similarity measure (Salton, 1983) to calculate the vocabulary overlap, T(t), between the windows to measure content similarity.

$$T(t) = \sum_{w \text{ in } v_1} Wt(w,v_1) * Wt(w, v_2)$$

where

v_1 is video segment from t-windowsize -> t

v_2 is video segment from t -> t+windowsize

[1]The word frequency statistics from 3 years of the Wall Street Journal are used.

We merge adjacent shots when $T(t) > T_{threshold}$. To perform this text analysis in real time, we make extensive use of special purpose hash tables for fast lookup into the stopwords list and our lexicon of 40,000 words and their frequencies in a news related corpus, three years of the Wall Street Journal. For example, if shot A contains the words "the photographer saw the elephant" and shot B has the words "three photographs of the animal were taken", frequently occurring stopwords such as "the" would be removed and after stemming we would be left with "photograph elephant" and "photograph animal". Since "photograph" occurs in both captions and this word is relatively rare in the lexicon (and thus has a high idf value), the value T(t) would be high and these clips would be merged.

Finally, there is an additional closed caption cue which is helpful in certain situations. A change in speaker is often marked by the symbol ">>" in the closed captions. Similarly, a change in topic may be indicated by the symbol ">>>". Thus, we apply the heuristic that if there is a change in topic symbol which is close to the shot transition, then we override any audio cues or word similarity cues and prevent potential shot mergers. As the caption text is processed, we calculate the distance D(t) in frames to the nearest ">>>" symbol. We do not merge adjacent shots if $D(t) > D_{threshold}$.

For video sources with very long shots, we may wish to introduce scene breaks at the change of speakers to limit the length of scenes in our video library. Here, we could use a similar approach to disable shot merger when the shot is relatively long and a ">>" symbol occurs nearby. Since we are currently using short television news stories as our video source, this feature has not been necessary.

Although most video sources have closed captions, commercials are often the exception. Hence, if the captions in both shots are all empty, this is an indication that they are part of a commercial and should be merged. In general, this approach will word for any captioned broadcast, merging the un-captioned pieces together, whether they are commercials or periods of silence while the camera is panning to show scenery. If the video is not closed captioned at all, caption-based segmentation can be turned off. Ideally, speech recognition techniques could be used in this case, but this would be difficult to achieve in real-time for a random video feed.

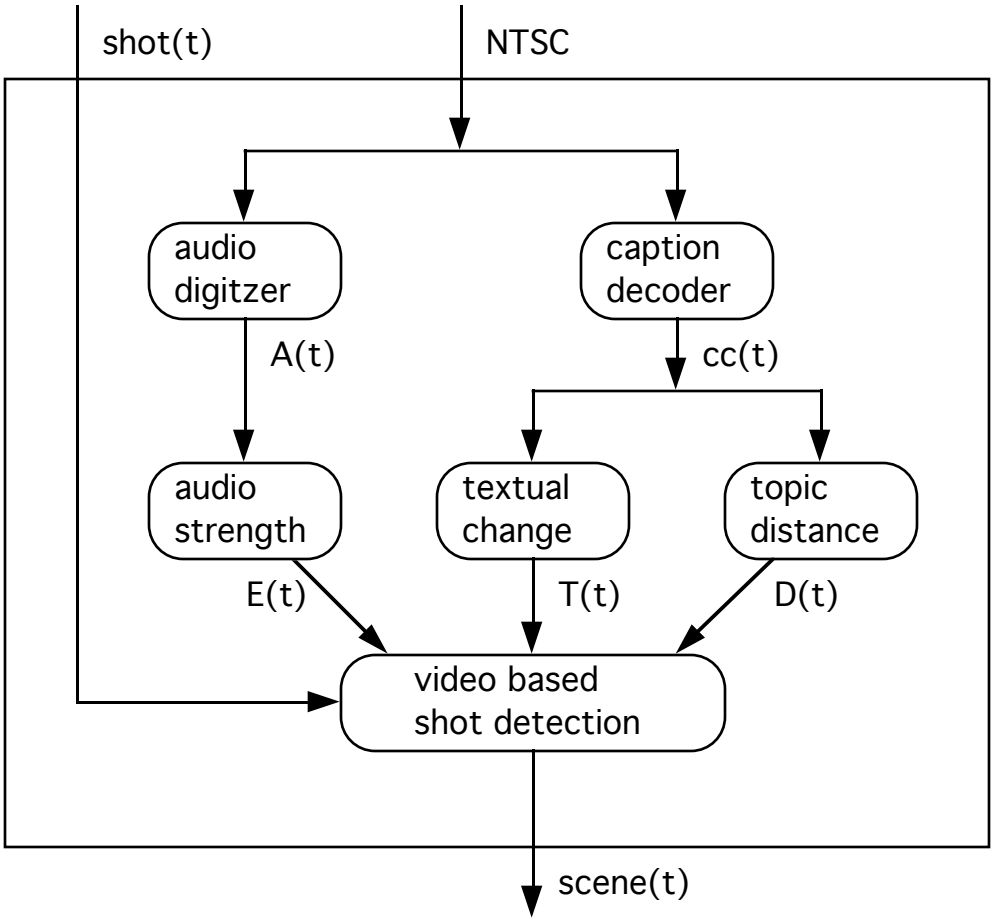

Figure 2. Data flow diagram for scene detection.

The VISION scene merger process is illustrated in Figure 2. The raw NTSC signal and the shot(t) function are inputs. The Boolean output function scene(t) is true if a scene boundary has been detected at frame t, and false otherwise.

3.3 Segmentation Results

The identification of shot boundaries using video information is relatively straightforward if there is a single frame transition between one shot and the next. Gradual shot transitions and wipes are more challenging to detect. This is where the combination of pixel differences and color histogram differences assist in distinguishing shot boundaries from locations where rapid motion in the video sequence occurs. Figures 3-5 illustrate representative sequences of four frames from CNN where our system has correctly identified these types of shot boundaries.

The use of audio and closed caption information to detect scene boundaries is also demonstrated in these examples. Figure 3 demonstrates the use of the ">>>" clue in the closed caption to correctly identify this shot boundary as a scene boundary although the audio levels were below the audio threshold. The audio energy at the shot boundary in Figure 4 was found to be above the audio threshold, so these shots were merged together by our segmentation algorithm. Finally, the words in the closed captions were used to merge the two shots shown in Figure 5, although the audio levels were below the specified threshold.

INSERT FIGURE 3 HERE

Figure 3. Example of shot boundary detection where there is a single frame transition. There is a topic change symbol, ">>>" nearby, so this shot boundary is identified as a scene boundary.

INSERT FIGURE 4 HERE

Figure 4. Example of shot boundary detection where there is a gradual fading transition in the video. Audio energy is high here, so no scene boundary is detected at this location.

INSERT FIGURE 5 HERE

Figure 5. Example of shot boundary detection where there is a diagonal wipe transition in the video. Closed caption similarity is high here, so no scene boundary is detected at this location.

4. Content-Based Indexing and Classification

The overriding goal for this project is to provide content-based search and filtering of video information. The initial system (Gauch, 1997) focused on creating a growing archive of video clips indexed by their associated closed-captions. Users could search the archive using a full-text search engine to retrieve video clips relevant to their queries. While this remains a component of current system, the text processing module has been extended to operate in an information filtering mode. Rather than users seeking out information on an *ad hoc* basis, users create individual profiles against which incoming video clips are compared to determine which users should receive which clips.

Our video categorization and filtering system has the following components:

- a taxonomy of categories which will be used for video classification
- the training data which will be used to classify incoming video clips
- a content indexing algorithm
- the category index obtained by combining the taxonomy and training data

- the user profile (a selection of categories from the above taxonomy)
- the text associated with the video being classified
- a content classification algorithm

Any hand-built taxonomy of categories which matches the domain of the expected video clips would suffice. Since we are currently working with news videos, we have chosen a taxonomy designed to classify news stories on the Web as the basis for our classification system. This taxonomy contains approximately 2,000 categories arranged in a hierarchy of height three. For training data, we spidered the site over several days to download not just the category names, but also 2 - 4 Web pages hand-attached to the leaf nodes. These Web pages form the basis of our training data. Note that having accumulated a taxonomy and sufficient training data, we are unaffected by future changes to the taxonomy structure (which has, indeed, undergone a significant rearrangement). However, if new stories occur which use previously unseen words as major clues about the correct category, we will need to update the documents used as training data for the category. This could be done by adding the closed captions from correctly classified clips to the training data for the appropriate category. We then built a Java-applet which displays the top three levels of the taxonomy and allows users to select one or more leaf categories as being of interest. This list of selected categories forms their user profile.

The final component of the filtering system associates an incoming video clip with the appropriate category(s) so that it can be automatically sent to interested users. Our classification system makes use of the software built for searching the archive. Traditional text-based searching takes in a query and returns the top matches from an indexed collection of documents. Similarly, our classifications system takes in the closed-caption information for a new video clip and returns the top matching

categories from an indexed collection of categories. To accomplish this, the closed-caption information is initially processed to identify the highest weighted N words (usually 5 - 10 words per clip), where the weight for term w in video clip v is calculated as follows:

$$Wt(w, v) = \sum tf_{wv} * idf_w$$

where

f_{wv} is the frequency of term w in clip v

$idf_w = \log2\ (freq_{max} / freq_w)$

$freq_{max}$ = frequency of most frequent term in sample text collection[2]

$freq_w$ = frequency of term w in sample text collection[1]

These highly weighted words and their weights are then used as a query against the index of categories. The categories themselves are indexed by treating all of the Web pages associated with each category as a single, large document. Hauptmann and Lee [1998] also use this to classify broadcast news stories. We are currently doing extensive experimentation of our classification approach, comparing it to Bayesian approaches and studying the effect of varying amounts of training data on the quality of the results. Figure 6 shows the architecture of our classification module.

[2]The word frequency statistics from 3 years of the Wall Street Journal are used.

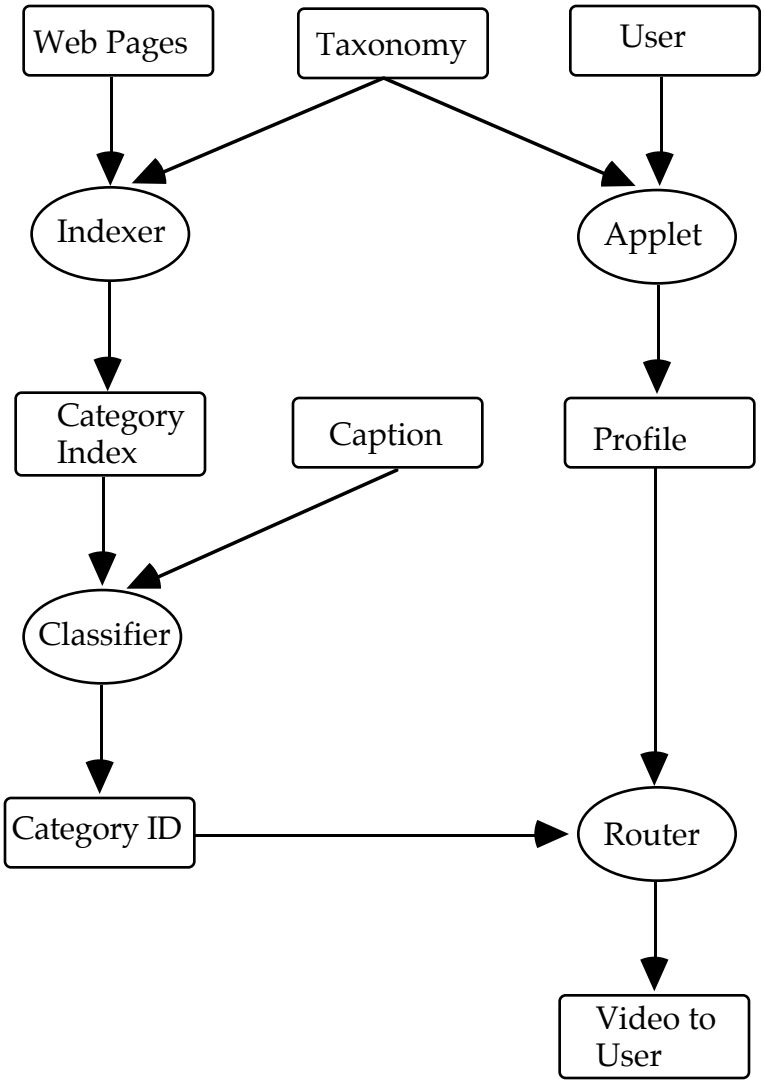

Figure 6. Data flow diagram for video scene classification.

5. System Architecture and Implementation

A wide variety of software and hardware products are available to digitize and compress full color video images at frame rates up to 30 fps. Unfortunately, once the video stream has been compressed, it is difficult to perform content-based segmentation without first decompressing the signal. Since decompression takes almost as long as compression, the time required to post-process this material to add

segmented video to the library is proportional to the length of the broadcast. This delay is unacceptable for a "live" digital video library.

Our solution to this problem is to extract video, audio and closed caption features as the video is digitized and use this information to segment the video into meaningful clips in real time. The digitized audio and video frames associated with each scene are then compressed and placed in the multimedia database within seconds of their broadcast. Given the real time constraint, and the computational platform we were targeting for this system, special care was taken to balance the computational load of feature extraction, segmentation, compression and content classification. The remainder of this section describes our software design and implementation in detail.

Our new video processing system consists of four components: (1) the video capture and feature extraction (VCFE) module, (2) the video segmentation and compression (VSC) module, (3) the content classification and indexing (CCI) module, and (4) a graphical user interface (GUI) for controlling video acquisition, segmentation, and compression parameters. These modules execute in separate threads on a dual processor PC running WindowsNT and communicate through shared memory data buffers.

5.1 The VCFE module

The Video For Windows (VFW) library from Microsoft is used to interface with the video and audio digitizer. Digitization of audio and video frames is performed by system level routines and the resulting data is stored in a set of VFW data buffers. Two callback routines are triggered, one for audio and one for video, which allow the user to access the digitized data within the VFW buffers. In order to ensure that

no audio or video frames would be lost, we do not perform any disk I/O or audio/video compression within these callbacks. Instead, we perform audio/video feature extraction to calculate E(t), B(t), dP(t), h(c,t), and dC(t) for each frame, and copy this information together with the digitized audio and video into a larger circular buffer pool in shared memory for use by other modules.

The VCFE module also captures closed captions using a third VFW callback which is invoked every 1/30th of a second (the same frame rate as the audio/video capture). In this function, we read the parallel port to obtain the two characters of caption data which has been extracted from the video frame and decoded by the TextGrabber. These characters are then processed to break the captions into tokens (words) and perform the necessary stemming and stopword removal. The remaining words are saved in a shared memory buffer together with timing and word frequency data.

5.2 The VSC module

The VSC module performs the important task of real time video segmentation, and controls the audio/video compression. Before segmentation begins, we wait until several seconds of audio/video and closed captions have been buffered. We then apply the shot and scene detection algorithms described in section 3. The data provided by the VCFE module is used to detect shot boundaries and perform audio-based shot merging. Text analysis functions are invoked at the remaining shot boundaries to calculate T(t) and D(t) and determine if caption-based merging is needed. This generates starting and ending frame numbers of scenes which in turn are used to control the software video compression process.

We perform software-based audio/video compression using a SDK produced by Real Networks (formerly Progressive Networks) which outputs digital video in

RealMedia format. This obviates the need for special purpose MPEG/JPEG video compression hardware, while also providing low bit rate digital video compression suitable for Internet broadcast. Each video scene is stored in a separate RealMedia file on disk. When the end of a scene is detected, the current file is closed and copied to the video server, and a new output file is created. A separate video library client which uses a RealMedia decoder can then be used to retrieve and play back video clips as soon as they become available.

Because compression requires file I/O and a variable amount of computational time per frame, it was important to perform this operation in a separate thread from the VCFE operations which are time critical (frames are lost if audio/video callbacks take too much time). On a dual processor system, this also enables our system to distribute the work load between the two processors.

5.3 The CCI module

The content classification and indexing phase of our video processing system is performed after the video associated with a scene has been compressed and the RealMedia file has been copied to the video server. The closed captions associated with each scene are added to the multimedia database, and the content classification procedure described in section 4 is applied to select the top words from the closed caption text and obtain additional category information to be installed in the database. This process is performed by a CCI thread which sleeps when there is no work to be done, and awakens only when there is new content to be added to the video library.

6. Evaluation of Segmentation Results

To evaluate the accuracy of our video segmentation results, we conducted a number of experiments processing videos which were hand segmented to identify the "true" scene locations. Overall, the accuracy of our results are quite good, although there is a tendency to over-segment the input video, breaking news stories and commercials into several parts. To better understand the interactions of the video, audio, and closed caption features when partitioning the video into scenes, we conducted four experiments: (1) video only segmentation, (2) video segmentation with audio merging, (3) video segmentation with caption based merging, and (4) video segmentation with audio and caption based merging. Although many values for the various thresholds were evaluated, for brevity, only the results with the best threshold settings are shown here.

6.1 Video Only Segmentation

To evaluate video only segmentation we varied the values of $P_{threshold}$ and $C_{threshold}$ while disabling audio and caption based shot merger. (i.e., $E_{threshold}$ = maxint, $T_{threshold}$ = maxint, $D_{threshold}$ = 0). The position of true scenes boundaries in two hours of broadcast news were determined by visual and auditory inspection by a single individual who was involved in the development of the segmentation algorithms. This information used to evaluate the quality of our segmentation results. We calculate the ratio of the number of correctly detected scene boundaries to the number of true scene boundaries to measure the *recall* of our system.

recall = # correctly detected boundaries / # true scene boundaries

High values of recall indicate that most of the correct scene boundaries were within the group of detected scene boundaries. We use *precision* to quantify the error due

to over segmentation by computing the ratio of the number of correctly detected scene boundaries to the total number of scene boundaries detected.

precision = # correctly detected boundaries / # detected scene boundaries

High precision indicates that few false boundaries have been detected. Ideally, we would like both recall and precision to equal one, but in practice increases in precision are at the expense of recall and vice versa.

We ran a series of experiments varying $P_{threshold}$ and $C_{threshold}$ to determine values that produced the most accurate shot detection. Since all possible boundaries are determined by the shot detection process, and later processing merely merges together shots to form scenes, the emphasis during shot detection is on increasing recall (the number of cuts correctly found). Table 1 summarizes the results found with the best threshold settings of $P_{threshold}$ and $C_{threshold}$ (each set at 70). We detected just over 94% of the scene boundaries in the two hour test data, and in half of the recorded videos we detected all scene boundaries.

Video	Recall	Precision
vid9	0.8700	0.1100
vid10	1.0000	0.1100
vid11	1.0000	0.0800
vid12	0.8000	0.1000
vid13	1.0000	0.1200
vid14	0.9300	0.1000
vid15	0.9400	0.1500
vid16	1.0000	0.1000
Average	**0.9425**	**0.1087**

Table 1. Recall and precision values for video-only segmentation of two hours of CNN Headline News captured during eight different recording sessions (vid9 - vid 16).

6.2 Video/Audio Segmentation

For the video segmentation phase, we were mostly concerned with obtaining high recall values since audio and caption based shot merger will remove false scene boundaries. Using the results from the video segmentation phase, we tested the audio merging parameter $E_{threshold}$. We compared the results obtained when $E_{threshold}$ was set to a wide range of constant values with our automatic audio selection algorithm. Not surprisingly, since different video sources have very different audio properties, our automatic selection of audio thresholds produced the best results. From the results shown in Table 2, we can see that the number of boundaries we produce that indicate scene changes has more than doubled, from roughly 11% to 24.5%. At the same time, the number of scene boundaries found has decreased approximately 13.5% from 94.5% to 81.5%.

Video	Recall	Precision
vid9	0.7300	0.1600
vid10	0.8000	0.3100
vid11	0.8600	0.2000
vid12	0.7300	0.2100
vid13	0.8400	0.2800
vid14	0.9300	0.2900
vid15	0.7800	0.2600
vid16	0.8500	0.2500
Average	**0.8150**	**0.2450**

Table 2. Recall and precision values for video and audio segmentation of two hours of CNN Headline News captured during eight different recording sessions (vid9 - vid 16).

6.3 Video/Caption Segmentation

Although we did not expect that video and closed caption analysis alone would produce accurate scene boundaries, we evaluated the performance of that

combination. From Table 3 we can see that this produced extremely high recall (in fact, not a single scene boundary was discarded based on closed caption information) but the precision was much lower than that achieved when audio information was included (see Table 4).

Video	Recall	Precision
vid9	0.8700	0.1600
vid10	1.0000	0.1700
vid11	1.0000	0.1500
vid12	0.8000	0.1200
vid13	1.0000	0.1400
vid14	0.9300	0.1700
vid15	0.9400	0.2700
vid16	1.0000	0.1300
Average	**0.9425**	**0.1537**

Table 3. Recall and precision values for video and closed caption segmentation of two hours of CNN Headline News captured during eight different recording sessions (vid9 - vid 16).

6.4 Video/Audio/Caption Segmentation

After the audio merging phase there are still some number of incorrect scene boundaries which need to be removed. We do this by considering the similarity of closed captions. Using the same video and audio settings as the previous experiment, we evaluated the effect of using closed caption information during segmentation. There were three main parameters to vary: the delay between the start of a scene and the start of the closed caption for the scene; the size of the closed caption window used for comparison; and the $T_{threshold}$, the closed caption matching threshold above which shots are merged. We found that a delay of 100 frames, a comparison window of 4,000 frames (corresponding to roughly 2 minutes) and a threshold of 7 worked best. Results with these settings are shown in Table 4.

From this experiment, we can see that using the closed caption information during segmentation produces precision numbers comparable to those resulting from video and audio segmentation (0.2313 versus 0.2450) while greatly improving recall (0.9200 versus 0.8150). In other words, with closed caption information we remove almost as many false boundaries and far fewer true ones. In fact, very few true boundaries were removed since we started with 0.9425 recall from the video phase which decreased only 2% while precision more than doubled. On average over 90% of all scene boundaries are detected and the average scene is split into 4 pieces rather than 10 pieces as was the case before the merging process began. Although there is obviously work remaining to further increase precision, the authors are unaware of empirical numbers from other real-time segmentation algorithms to be used for comparison.

Video	Recall	Precision
vid9	0.8700	0.1600
vid10	0.9500	0.1700
vid11	1.0000	0.2600
vid12	0.8000	0.1700
vid13	0.9500	0.1900
vid14	0.9300	0.5000
vid15	0.9400	0.2700
vid16	0.9200	0.1300
Average	**0.9200**	**0.2313**

Table 4. Recall and precision values for video, audio and closed caption segmentation of two hours of CNN Headline News captured during eight different recording sessions (vid9 - vid 16).

7. Conclusions and Future Work

Our real time video segmentation and classification system is now fully operational. It can continuously capture, segment, compress, classify, index and store video clips

from a live broadcast feed in real-time. In addition to the new pipeline architecture, we also presented our segmentation algorithm which fuses three sources of information: video, audio and closed-captions. This provides much higher scene detection accuracy than that realized with just video alone or video plus audio. Finally, we describe an information filtering application based upon VISION which matches incoming video to user profiles based upon a taxonomy of categories. Since all processing is completely automatic, multiple installations of the VISION system can be used to monitor multiple video feeds simultaneously with little or no burden on the archive staff. It is currently in around-the-clock commercial operation, indexing CSPAN and CSPAN-2 for FASTV (www.fastv.com).

After segmentation, we locate keyframes within each of the scenes in our video library by examining the same video information used for segmentation. We are currently working on analysis of keyframes to provide image-based browsing of video archives. Future extensions to the VISION system may also include improvements to the video processing component of the VPS. For example, we could incorporate motion analysis to reduce the over-segmentation we experience. Or, we could incorporate image processing/computer vision techniques to attempt to do image-based indexing and retrieval. Audio processing could also be enhanced to reduce over-segmentation through the use of speaker identification. However, the over-segmentation is less of a concern since this is often corrected by the closed-caption merger phase.

8. References

Arman, F., Hsu, A., et al, (1993). Image Processing on Compressed Data for Large Video Databases, *ACM Multimedia '93*, California, USA, 267-272.

Christel, M., et al, (1994). Informedia Digital Video Library, *ACM Multimedia '94*, 480-481.

Christel, M., et al, (1995). Informedia Digital Video Library, *Communications of ACM*, **38** (4), 57-58.

Faloutsos, C., et al., (1994). Efficient and Effective Querying by Image Content, *Journal of Intelligent Information Systems*, **3**, 231-262.

Frakes, W. B., Baeza-Yates, R. (1992) in *Information Retrieval, Data Structures & Algorithms* (Verde, K., Goodwin, B., Doench, G., and Papanikolaou, S. eds), Prentice-Hall International (UK) Limited, London.

Gauch, S., et al, (1994). The Digital Video Library System: Vision and Design, *Digital Libraries '94* , College Station, Texas, 47-52.

Gauch, S., Gauch, J., Pua, K.M., (1996). VISION: A Digital Video Library, *ACM Digital Libraries '96*, Bethesda, MD, 19-27.

Gauch, S., Li, W., Gauch, J., (1997). The VISION Digital Video Library System, *Information Processing & Management*, **33**(4), 413-426.

Hampapur, A., Jain, R., Weymouth, T., (1994). Digital Video Segmentation, *ACM Multimedia '94*, San Francisco, 357-364.

Haralick, R.M. and Shapiro, L.G., (1992). *Computer and Robot Vision*, Addison Wesley.

Hauptmann, A.G. and Witbrock, M.J. (1997). Informedia: News-on-Demand Multimedia Information Acquisition and Retrieval, in *Intelligent Multimedia Information Retrieval*, Mark T. Maybury (ed.), MIT Press, 215-239.

Hauptmann, A.G. and Lee, D. (1998). Topic Labeling of Broadcast News Stories in the Informedia Digital Video Library", *ACM Digital Libraries '98*, Pittsburgh, PA, 287-288.

Irani, M. and Anandan, P., (1998) Video Indexing Based on Mosaic Representations, *Proceedings of the IEEE*, **86** (5), 905-921.

Jacobs, C.E., Finkelstein, A., Salesin, D.H., (1995). Fast Multiresolution Image Querying, ACM Computer Graphics (SIGGRAPH '95), 277-286.

Lindblad, C.J., et al, (1994). The VuSystem: A Programming System for Visual Processing for Digital Video, *ACM Multimedia '94*, San Francisco, 307-314.

Nagasaka, A., Tanaka, T., (1992). Automatic Video Indexing and Full-Video Search for Object Appearances, *Visual Database Systems, II*, E. Knuth and L.M. Wegner, Editors, North-Holland, 119-133.

Pentland, A., Picard, R.W., Sclaroff, S., (1996). Photobook: Content-Based Manipulation of Image Databases, *International Journal of Computer Vision*, **18** (3), 233-254.

Picard, R.W., Liu, F., (1994). A New Wold Ordering for Image Similarity, *Proc. ICASSP*, Adelaide, Australia.

Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval.* New York: McGraw-Hill.

Sawhney, H.S., and Ayer, S., (1996). Compact Representations of Videos Through Dominant and Multiple Motion Estimation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, 814-830.

Smoliar, S., and Zhang, H., (1994). Content-based Video Indexing and Retrieval, *IEEE Multimedia Magazine*, 1(2), 62-72.

Swain, M., and Ballard, D., (1991). Color Indexing, *International Journal of Computer Vision*, 7(1), 11-32.

Teodosio, L., Bender, W., (1993). Salient Video Stills: Content and Context Preserved, *ACM Multimedia '93*, California, 39-46.

Turner, J.M., (1997). Deriving Shot-Level Indexing from Audio Description Texts, *Association of Moving Image Archivists Annual Conference (AMIA '97)*, November 17-22, Bethesda, MD, http://esi25.ESI.UMontreal.CA:80/~turner/english/texts/amia97.html, visited: 9/16/98.

Wactlar, H.D., et al, (1995). Intelligent Access to Digital Video: Informedia Project, *IEEE Computer*, Vol. 29, No. 5, 46-52.

Weiss, R., Duda, A., Gifford, D.K., (1995). Composition and Search with a Video Algebra, *IEEE Multimedia*, 12-25.

Wolf, W., Liu, B., Wolf, W., (1995). A Digital Video Library for Classroom Use, *Proceedings of the International Symposium on Digital Libraries*.

Wolf, W., (1996). Key Frame Selection by Motion Analysis, *Proc. ICASSP*.

Yeo, B.-L., and Yeung, M.M., (1997). Retrieving and Visualizing Video, *Communications of the ACM*, **40** (12), 43-52.

Yu, H.H., and Wolf, W., (1995). Scenic Classification Methods for Image and Video Databases, *Digital Image Storage and Archiving Systems*, SPIE 2606, 363-371.

Zhang, H.J., et al, (1993). Automatic Partitioning of Video, *Multimedia Systems*, Vol. 1, 10-28.

Zhang, H.J., et al, (1995). Automatic Parsing and Indexing of News Video, *Multimedia Systems*, Springer-Verlag, **2** (6), 256-266.

Zhang, H.J., Low, C.Y., Smoliar, S.W., and Wu, J.H. (1997). Video Parsing, Retrieval and Browsing: An Integrated and Content-Based Solution, in *Intelligent Multimedia Information Retrieval*, Mark T. Maybury (ed.), MIT Press, 139-158.