Submitted to: Multimedia Systems, pending.

Real Time Video Analysis for Automatic Archive Creation

John M. Gauch, Susan Gauch, Sylvain Bouix

Electrical Engineering and Computer Science
415 Snow Hall
The University of Kansas
Lawrence, KS 66045
jgauch@eecs.ukans.edu

Abstract

New and emerging technologies make it possible to capture and deliver digital video information. In general, the techniques used to index video information require source manual intervention, source specific information and/or are computationally expensive. 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. Recent work has focused on efficient key frame extraction to support browsing and real-time feature extraction to support image-similarity based retrieval.

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 via keyword-based queries 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. Ideally, we would like 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. 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. VISION has also been extended to operate as an information filtering system, classifying video and sending it to users whose profiles contain the matching categories (Gauch et al, 1998). Preliminary work has been done on keyframe extraction and feature analysis to support archive browsing image based querying.

The design and implementation of this new system is the subject of this paper. Section 2 describes related work in digital video libraries. Section 3 describes our software architecture for real time video analysis. Our pipelined video segmentation algorithm is described in Section 4. Our segmentation results are discussed in Section 5. Finally, Sections 6 and 7 discuss our keyframe extraction and analysis techniques and our conclusions and future work, 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). Once selected, keyframes may themselves be analyzed to extract features to support color, texture, shape, motion or other feature-based 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) 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. 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-II 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 is 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. 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. The pipelined system runs on a dual-processor Pentium PC and produces RealMedia format videos. The real-time performance, commodity hardware and software compression all provide serious constraints on our algorithm complexity.

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 and communicate through shared memory data buffers.

3.1 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.

3.2 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 4. 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.

3.3 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 sent to the central database for indexing, and content classification is applied to select the most important words from the closed caption text for use in classifying the scene (Gauch et al, 1998). 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.

4. 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 our current application, each scene would be a news story generally consisting 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.

4.1 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 when 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 (Hampapur et al, 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.

INSERT FIGURE 1 HERE

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.

4.2 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 most 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. Incorporating speech recognition could be used to align the captions with the spoken words and thus get a more accurate estimate of the delay (Hauptmann and Witbrock, 1997), but this would require a second computer per broadcast channel and may not operate in real-time.

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 and McGill, 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

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 $D(t) > D_{threshold}$.

INSERT FIGURE 2 HERE

Figure 2. Data flow diagram for scene detection.

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

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.

4.3 Segmentation Examples

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 sequences of four frames from CNN Headline News where our system has correctly identified shot boundaries.

INSERT FIGURE 3 HERE

Figure 3. Example 1 of shot boundary detection. There is a topic change symbol, ">>>" nearby, so this becomes a scene boundary.

INSERT FIGURE 4 HERE

Figure 4. Example 2 of shot boundary detection. The audio energy is high here, so no scene boundary is detected.

INSERT FIGURE 5 HERE

Figure 5. Example 3 of shot boundary detection. The audio energy is high here, so no scene boundary is detected.

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.

5. 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, and (3) 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.

5.1 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 inspection and 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 video segmentation process, and later processing merely merges together clips to form scenes, the emphasis during video segmentation is on increasing recall (the number of cuts correctly found). Figure 6 summarizes the results found when varying $P_{threshold}$ and $C_{threshold}$. The best results (recall = 94%, precision = 11%) were achieved with $P_{threshold}$ and $C_{threshold}$ each set at 70.

INSERT FIGURE 6 HERE

Figure 6. The effects of varying the pixel difference threshold, $P_{threshold}$, and the color histogram difference threshold, $C_{threshold}$, on recall and precision.

5.2 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 evaluated the results obtained when $E_{threshold}$ was set to a wide range of constant values. However, since different video sources have very different audio properties, selecting the audio threshold dynamically based on the observed audio energy levels performed better than any fixed value for $E_{threshold}$, and dynamic audio thresholds are used for the next experiment (recall = 81.5%, precision = 24.5%).

5.3 Video/Audio/Caption Segmentation

After the audio merging phase, there are still many incorrect scene boundaries which need to be removed. We do this by considering the contents of the closed captions. Using the video and audio settings from previous experiments, we

evaluated the effect of adding closed caption information to the segmentation process. 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 (CC-delay); the size of the closed caption window used for comparison (CC-windowsize); and $T_{threshold}$, the threshold for closed caption similarity above which shots are merged. We found that a delay of 100 frames, a comparison window of 4,000 frames (corresponding to 13 seconds before and after the shot boundary) and a threshold of 7 worked best. Results with these settings are shown in Figure 7. The best results (recall = 92%, precision = 23%) were achieved with CC-delay set at and CC-windowsize set at 70.

INSERT FIGURE 7 HERE

Figure 7 The effects of varying the delay, CC-delay, and window size, CC-windowsize, on recall and precision.

5.4 Segmentation Results Discussion

From the results shown in Table 1, we can see that most scene boundaries are detected by the video segmentation (over 94%). However, only 11% of the shot boundaries detected actually correspond to scene boundaries. In other words, the average scene is chopped into ten pieces, clearly demonstrating the need for other techniques merge some shots together. By adding audio information, the percentage of boundaries produces that correctly correspond to 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%.

Segmentation Technique	Recall	Precision
Video Only	0.9425	0.1087
Video and Audio	0.8150	0.2450
Video/Audio/Captions	0.9200	0.2313

Table 1 Recall and precision values for various scene segmentation techniques evaluated on two hours of CNN Headline News. $P_{threshold}$ = 70, $C_{threshold}$=70, automatic audio thresholding, $T_{threshold}$=7, CC-delay=100 and CC-windowsize=4,000.

When we add closed caption information during segmentation, we achieve precision percentages comparable to those resulting from video and audio segmentation (23% versus 24.5%) while greatly improving recall (92% versus 81.5%). 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 94% recall after the video phase which decreased only 2% while precision more than doubled. On average, 92% of all scene boundaries are detected and the average scene is split into four pieces rather than ten pieces which was the case before the merging process began. Although there is obviously work remaining to further increase precision, the results are quite encouraging.

6. Keyframe Extraction and Analysis

We wish to perform keyframe extraction for two reasons. First, the keyframe is used as the iconic representation of the video shot when the shot is retrieved by the search engine. Second, the keyframes for a shot can be analyzed to form the basis of feature-based shot matching and retrieval. We developed an efficient keyframe extraction algorithm and evaluated a series of image analysis techniques for keyframe indexing and retrieval.

6.1 Keyframe Extraction

Due to efficiency constraints, we concentrated on identifying a single representative keyframe for each shot rather than attempting to combine frames to produce a shot summary. Most keyframe extraction techniques rely on motion estimation. Since motion information is stored as part of the MPEG format, this information is readily available. However, we use a RealMedia or AVI format for our video and must calculate our motion estimates, which can be computationally expensive. To address this, we defined a new *stillness* criteria based on a local energy function $E_p(t)$.

$$E_p(t) = 1/n \times \left(\sum_{k=-n/2}^{n/2} dP(t+k, 1) \right)$$

where

t = time, measured in frame numbers

n = comparison window around frame t, measured in number of frames

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

where

I = pixel intensity

This function computes the sum of pixel intensity differences over a sequence of n frames around a specified frame t. For each shot, we first identify t_0 the frame which minimizes $E_p(t_0)$. To identify the most representative frame from the n candidate frames in the window around t_0, we select the frame whose brightness is closest to the average brightness for frames in the window. We experimented with different window sizes, varying n between 3 and 20, and found our best results with values in the range of 5 to 10. Larger values tended to produce too much smoothing, missing important, shorter still sequences whereas smaller values were unable to discriminate between brief periods of stillness and longer, more important low-motion sequences.

We evaluated our keyframe selection algorithm by having users view 80 minutes of video which contained roughly 1500 shots. For each shot, they were asked to rate quality of the chosen keyframe. They rated the keyframes as Very Satisfactory (60%), Satisfactory (35%), or Irrelevant (5%), indicating a high-level of satisfaction with the keyframe selections (Bouix, 1998). An example video sequence and the resulting keyframe (rated as Satisfactory) are shown in figures 8 and 9. The keyframe selected focuses on the two main participants, with little extraneous background motion.

INSERT FIGURE 8 HERE

Figure 8. A video sequence of 12 frames from a news broadcast.

INSERT FIGURE 9 HERE

Figure 9. The 9th frame in the sequence above was selected as the keyframe.

6.2 Keyframe Analysis

In order to perform image-based comparison of video frames, we must extract a set of features from the images. Building on image database research (Flickner et al, 1995; Pentland et al, 1996; Smith & Chang, 1997), video databases generally index keyframes using image features such as color (brightness, color histogram, dominant colors, statistical moments), texture (Tamura features, Markov random fields, Fourier transforms, wavelet transforms) , shape (moment invariants, cumulative turning angles), spatial (localized color regions, color correlogram), or domain specific features (faces, fingerprints, image recognition).
In addition to the visual information video shares in common with image databases, video databases also incorporate temporal information. The video databases are not just collections of frames. Rather, they are collections of sequences of frames. Thus, videos can also be indexed using motion features (Chang et al, 1997; Flicker et al, 1995; Teodosio and Bender, 1993; Zhang et al, 1997) such as camera motion, brightness and/or color variation, object motion, and mosaic representations.

We implemented and evaluated real-time indexing of keyframes using color features. The video, when captured, uses the red-green-blue (RGB) color space used in television monitors. Although this color space is common and straightforward, it presents a problem when comparing colors within and between images. Colors in RGB are not perceptually uniform, i.e., the proximity of colors in RGB space does not indicate color similarity as perceived by humans. We therefore convert the images to the hue-saturation-intensity (HSI) color space which is designed to reflect the human perception of colors. Colors with similar values in HSI will be judged similar by humans and vice versa. Each pixel of an HSI image contains three values which range from 0 to 255. Thus, it can represent $2^{256} \approx 16$ million different colors. We used uniform quantization is necessary to map from the set of all possible colors to a smaller set (18 hues, 4 saturations, 4 intensities = 288 colors) which is more efficient for image color comparisons but which also adequately represents the range of colors used in the keyframes.

For each of the 1500 keyframes extracted in Section 6.1.1, we compared the use of three color features for image indexing and retrieval: the color histogram (Smith and Chang, 1997) in both RGB and HSI space and the first three color moments, mean, variance and skew, (Zhang et al, 1997) with a Euclidean distance similarity measure. The color histogram in RGB space adequately retrieved exact matches, the color histogram in HSI space gave the best results for non-identical matches. Confirming the results reported by Zhang, we found that the color moments were the method of choice. The results were nearly comparable to those produced by the HSI color histogram method, but the color moment is more efficient since it

requires only nine numbers (three moments for each of the three color dimensions, RGB or HSI) to be compared versus 288 numbers for the color histogram method.

INSERT FIGURE 10 HERE

Figure 10. The results of matching based on three color moments. The frame in the top left-hand corner is the query frame. Matches are in rank order, left-to-right, top-to-bottom.

We also implemented and tested a shape feature, invariant moments (Arman et al, 1994) which are efficiently computed. They provide the ability to match on the shape of objects contained in the images, although the rank ordering of the top matches was not as accurate as that produced by the color moments. Finally, we are working on indexing frame regions (e.g., quadrants) rather than entire frames and combining the two moment based features into a single retrieval algorithm.

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 our keyframe extraction and analysis algorithms which form the basis of our video browsing and image-based querying capabilities. 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 the Worldwide Broadcasting Network (www.wbnet.com).

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. Audio processing could also be enhanced to reduce over-segmentation through the use of speaker identification. Our current focus is on clustering the keyframes to automatically construct a browsing hierarchy for the video archive and examining quality of service issues in video delivery.

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.

Arman, F., Depommier, R., Hsu, A., Chiu, M.Y., (1994). Content-based Browsing of Video Sequences, *Proc. ACM Multimedia '94*,.

Bouix, S. (1998). VISION: Segmentation, Indexing and Retrieval of Digital Videos, *M.S. Thesis*, Electrical Engineering and Computer Science, University of Kansas.

Chang, S.F., Chen, W., Meng, H., Sundaram, H., Zhong, D., (1997). Videoq: An Automated Content-Based Video Search System Using Visual Cues, *Proc. ACM Multimedia '97*,.

Christel, M., et al, (1994). Informedia Digital Video Library, *Proc. 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.

Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., Yanker, P. (1995). Query By Image Content: The QBIC System. *IEEE Computer Magazine*, **28**(9), 23-32.

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.

Gauch, J.M., Gauch, S., Bouix, S., and Zhu, X., (1998). Real TIme Video Indexing and Classification, *Information Processing & Management*,. (pending)

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.

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.

Smith, J.R., and Chang, S.F., (1997). Querying by Color Regions Using the VisualSeek Content-Based Query System, in *Intelligent Multimedia Information Retrieval*, Mark T. Maybury (ed.), MIT Press, 159-173.

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.

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*.

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.