Proceedings of ACM Multimedia 96, Boston, Nov. 1996

CVEPS - A Compressed Video Editing and Parsing System

Jianhao Meng and *Shih-Fu Chang*
Department of Electrical Engineering &
Center for Image Technology for New Media
Columbia University
New York, NY 10027, USA
{jmeng,sfchang}@itnm.columbia.edu
http://www.itnm.columbia.edu/{~jmeng,~sfchang}

ABSTRACT

Processing digital video directly in the compressed domain has many advantages in terms of storage efficiency, speed, and video quality. We have developed a compressed video editing and parsing system (CVEPS) with advanced video indexing and manipulation functions. The video parsing tools support automatic extraction of key visual features, e.g., scene cuts, transitional effects, camera operations (zoom/pan), shape and trajectories of prominent moving objects. These visual features are used for efficient video indexing, retrieval and browsing. The editing tools allow users to perform useful video compositing functions and special visual effects typically seen in video production studios. We contrast our compressed-domain approach with traditional decode-process-reencode approach with quantitative and/or qualitative performance comparison. We also present a client-server network based CVEPS implementation.

KEYWORDS

Compressed domain video manipulation, client-server network based video editing, video content analysis, video indexing.

1. INTRODUCTION

Digital video is an essential component of new media applications. It demands special technical support in processing, communication, and storage. This paper investigates innovative compressed-domain technologies for compressed video manipulation, indexing, and browsing, in order to support various multimedia applications such as real-time video production and video digital library.

We present a *Compressed Video Editing and Parsing System*, CVEPS, using a unique compressed-domain approach which offers many great benefits [6,7]. First, implementation of the same manipulation algorithms in the compressed domain will be much cheaper than that in the uncompressed domain because the data rate is highly reduced in the compressed domain (e.g., a typical 20:1 to 50:1 compression ratio for MPEG). Second, given most existing images and videos stored in the compressed form, the specific manipulation algorithms can be applied to the compressed streams without full decoding of the compressed images/videos. Lastly, because that full decoding and re-encoding of video are not necessary, we can avoid the extra quality degradation that usually occurs in the reencoding process. We have shown that for MPEG compressed video editing, the speed performance can be improved by more than 60 times and the video quality can be improved by about 3-4 dB if we use the compressed-domain approach rather than the traditional decode-edit-reencode approach [15].

In order to allow users to manipulate compressed video directly, two types of functionalities are required (1) key content browsing and search, (2) compressed video editing. The former allows users to efficiently browse through or search for key content of the video without decoding and viewing the entire video stream. The key content refers to the key frames in video sequences, prominent video objects and their associated visual features (motion, shape, color, and trajectory), or special reconstructed video models for representing video content in a video scene. The second type of functionalities, video editing, allow users to manipulate the object of interest in the video stream without full decoding. One example is to cut and paste any arbitrary segments from existing video streams and produce a new video stream which conforms to the valid compression format. Other examples include special visual effects typically used in video production studios.

This paper describes system components and specific proposed compressed-domain algorithms for achieving the above functionalities in CVEPS. The primary compression standard used is MPEG (MPEG1 and MPEG2). Most of our techniques are applicable to generally encoded MPEG streams with different parameter settings such as constant or variable bitrate, different frequency of I, P, B frames etc. Our scene change detection techniques assume the use of interframe coded frames (i.e. P or B). However, the underly-

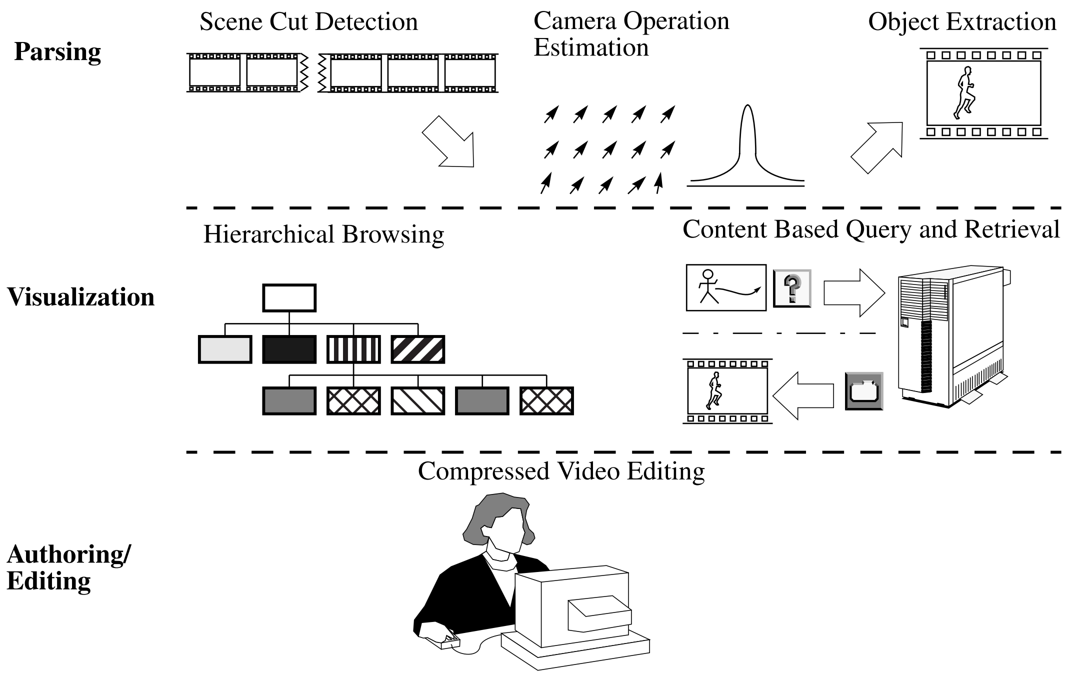


FIGURE 1. CVEPS System Overview

ing approach and techniques are general enough to be applied to other video compression standards (e.g, those using transform coding and/or interframe motion compensation). This paper is organized as the following. Section 2 discusses related work. Section 3 provides system overview for CVEPS. Section 4 presents our compressed-domain techniques for parsing MPEG video to extract visual features. Section 5 describes algorithms for compressed video editing. Section 6 discusses system design issues, followed by conclusion at the end.

2. RELATED WORK

Video indexing and manipulation has emerged as an active research area. Much work has been reported by several research groups, some of which also explored the compressed-domain approach. But there are no existing systems that provide integrated solutions for both video manipulation and video indexing. To this end, our prior work has presented techniques for manipulation of both compressed image and video [7,8], compressed image feature extraction [6], and video scene analysis using MPEG streams [15].

For scene cut detection in the spatial domain, Smoliar and Zhang proposed color histogram comparison [22] and Shahraray used a block-based match and motion estimation algorithm [19]. In the compressed domain (Motion JPEG video), comparison of DCT coefficients of selected blocks from each JPEG frame was used to detect the scene cuts [4]. We detect scene cuts in motion compensated video sequences such as MPEG. Distribution of motion vectors is used for detecting direct scene cuts and the variance of DCT DC coefficients is used for detecting transitional scene cuts [14]. After the scene cuts are found, video shots can be browsed with the clustering algorithms proposed in [24].

Within each shot, camera operation and moving objects are important visual features. In spatial domain, finding parameters of an affine matrix and constructing a mosaic image from a sequence of video images was addressed by Sawhney *et al* [18]; searching for object appearance and using them in video indexing was proposed by Nagasaka *et al* [16]. In compressed domain, detecting camera operations (zoom, pan) using motion vectors had been discussed in [2,25]. Both [2,25] used a simple 3 parameter model with the assumption that the camera panning is very small and focal length is very long. The two restrictions make the algorithms not suitable for general video processing. Object motion tracking in MPEG video was also discussed by Dimitrova *et al* [9], however, camera operations were not taken into consideration for object motion recovery. We use a 6-parameter affine transform model and the least squares (LS) method to estimate camera operation parameters. With the estimated camera parameters we further recover the local object motion from the global motion.

Video indexing using finite state models for parsing and retrieval of specific domain video, such as news video, was discussed by Smoliar *et al* [22]. Hampapur *et al* [11] proposed feature based video indexing scheme, which uses low level machine derivable indices to map into the set of application specific video indices. Our goal is to extract a rich set of visual features associated with the scenes and individual objects from the compressed video to enable content based query, and allow for integration with domain knowledge for derivation of higher-level semantics.

To manipulate image and video sequences, a resolution independent video language (Rivl) was proposed by Swartz and Smith [23]. Although Rivl utilized group of pictures (GOPs) level direct copying whenever possible for “cut and

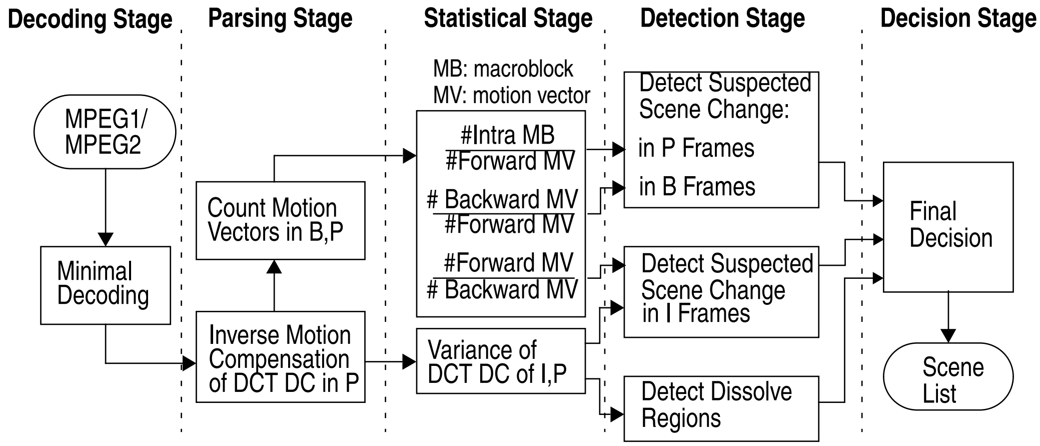


FIGURE 2. The Compressed Domain Scene Cut Detection Algorithm

paste" operations on MPEG video, Rivl did not use compressed domain approach at the frame level and macroblock level for special effects editing (see Section 5.2). Most video effects in Rivl were done by decoding each frame to pixel domain and applying image library routines. Also the rate control problems due to editing of constant bitrate video was not addressed by Rivl.

3. SYSTEM OVERVIEW

The CVEPS system consists of three major modules: Parsing, Visualization and Authoring, see Figure 1. In the Parsing module, MPEG compressed video is first broken into shot segments. Within each shot, camera operation parameters are estimated. Then moving objects are detected and their shape and trajectory features are extracted. In the Visualization module, the scene cut output list and the camera zoom/pan information are used to extract key frames for representing each video shot. The key frames can be browsed with the hierarchical video scene browser [26]. Our content-based image query system, VisualSEEk [20] and WebSEEk [21], are used to index and retrieve key frames or video objects based on their visual features and spatial layout. In the Authoring module, we provide tools for cutting/pasting of arbitrary MPEG video segments and adding special effects such as dissolve, key, masking and motion effects (described in more details later).

4. PARSING OF MPEG VIDEO

4.1 Scene Cut Detection in Compressed Domain

Within a video shot, consecutive frames have high temporal correlation. In MPEG video, this correlation can be characterized by the ratio of the number of backward motion vectors (or intracoded macroblocks) versus the number of forward motion vectors in B (or P) frames. For example, when a direct scene cut occurs on a P-frame, most macroblocks will be intracoded (i.e., no interframe prediction). We calculate the motion vector ratios for every B/P frame and use local adaptive thresholds to detect the peak values.

To detect the transitional scene cut such as dissolve, we use the fact that the variance of the pixel intensity of each frame in the dissolve region shows an approximated parabolic curve [3]. For MPEG video, we use the DCT DC values to approximate the pixel intensity. We are able to successfully detect long dissolves in sequences without high motion. Short dissolves with high motion are trickier and often treated as direct scene cuts.

Figure 2 shows the block diagram of our scene cut detection algorithm. MPEG video is minimally decoded and parsed to get the motion vector counts and DCT DC coefficients. This involves simple parsing of the MPEG streams and does not need any intensive computation. In the Statistical Stage, three ratios are calculated for detecting direct scene cuts in P, B, and I frames, respectively; variance of DCT DC coefficients are calculated from I and P frames for detecting dissolve curves. The peaks of ratios and the dissolve curve are found in the Detection Stage. Finally, duplicated cuts are eliminated before returning a list of scenes.

We have tested our algorithms on several bitstreams from classic movies and CNN news. Table 1 shows the results of a 10 minutes CNN news (unconstrained content) with 19931 frames, Group of Pictures (GOP) size 15, one I or P frame for every two B frames, and frame size 352 pixels by 240 pixels. For the direct scene cuts, we detected 54 out of 59 correctly; the 7 false alarms were mainly caused by a shot including the strobe motion special effect (refer to Section 5.2.4); the 5 missed cuts were due to similar dark background of the two shots. For transitional effects, we detected 19 out 21 correctly; the false alarms and misses in the transitional scene cut detection were mainly due to our light-weight implementation which skipped B frames.

TABLE 1. Scene Cut Detection Results

	Direct Scene Cuts	Transitional Scene Cuts
Manual	59	21
Detected	54	19
Missed	5	2
False Alarm	7	8

4.2 Camera Operation Parameters Estimation

Within a shot, low level visual features such as camera zoom/pan and moving objects are useful information for video indexing. We estimate the camera zoom and pan with a 6-parameter affine transform model [5] using the motion vectors from the MPEG compressed stream.

The motion vectors in MPEG are usually generated by block matching: finding a block in the reference frame so that the mean square error is minimized. Although the motion vectors do not represent the true optical flow, it is still good in most cases to estimate the camera parameters in sequences that do not contain large dark or uniform regions.

When the distance between the object/background and the camera is large, it is usually sufficient to use a 6 parameter affine transform to describe the global motion of the current frame,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \cdot \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{bmatrix}^T \qquad (1)$$

where (x,y) is the coordinate of a macroblock in the current frame, $\begin{bmatrix} u & v \end{bmatrix}^T$ is the motion vector associated with that macroblock, $\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{bmatrix}^T$ is the affine transform vector. We denote $\boldsymbol{U}$ for $\begin{bmatrix} u & v \end{bmatrix}^T$, $\boldsymbol{X}$ for $\begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix}$, and $\vec{\boldsymbol{a}}$ for $\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{bmatrix}^T$.

Given the motion vector for each macroblock, we find the global parameter using the Least Squares (LS) estimation, that is to find a set of parameter $\vec{\boldsymbol{a}}$ to minimize the error between the motion vectors estimated in (1) and the actual motion vectors obtained from the MPEG stream [25].

$$S(\vec{\boldsymbol{a}}) = \sum_x \sum_y [(\hat{u}_{xy} - u_{xy})^2 + (\hat{v}_{xy} - v_{xy})^2] \qquad (2)$$

where $\begin{bmatrix} \hat{u} & \hat{v} \end{bmatrix}^T$ is the estimated motion vector. To solve for $\vec{\boldsymbol{a}}$, set the first derivative of $S(\vec{\boldsymbol{a}})$ to 0, then we get

$$\begin{bmatrix} N & A & B \\ A & C & E \\ B & E & D \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \text{ and } \begin{bmatrix} N & A & B \\ A & C & E \\ B & E & D \end{bmatrix} \cdot \begin{bmatrix} a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \qquad (3)$$

where ,

$$N = \sum_x \sum_y 1,\ A = \sum_x \sum_y x,\ B = \sum_x \sum_y y,$$
$$C = \sum_x \sum_y x^2,\ D = \sum_x \sum_y y^2,\ E = \sum_x \sum_y x \cdot y,$$
$$U_1 = \sum_x \sum_y u_{xy},\ U_2 = \sum_x \sum_y u_{xy} \cdot x,\ U_3 = \sum_x \sum_y u_{xy} \cdot y,$$
$$V_1 = \sum_x \sum_y v_{xy},\ V_2 = \sum_x \sum_y v_{xy} \cdot x,\ V_3 = \sum_x \sum_y v_{xy} \cdot y.$$

All summations are computed over all valid macroblocks whose motion vectors survive after the nonlinear noise reduction process. After the first LS estimation, motion vectors that have large distance from the estimated ones are filtered out before a second LS estimation. The estimation process is iterated several times to refine the accuracy.

4.3 Moving Object Detection and Tracking

After the global camera parameters $\vec{\boldsymbol{a}}$ is found, we may recover the object motion by applying the global motion compensation. If an object located at (x,y) in the current frame has a local motion $\boldsymbol{M} = \begin{bmatrix} m_x & m_y \end{bmatrix}^T$ from (x_0,y_0) to (x_1,y_1) in the reference frame with motion vector $\boldsymbol{U}$, then $\boldsymbol{U} + \boldsymbol{M} = \boldsymbol{X} \cdot \vec{\boldsymbol{a}}$, see Figure 3. That means the local object motion can be recovered from motion vectors provided that $\vec{\boldsymbol{a}}$ is known,

$$\boldsymbol{M} = \boldsymbol{X} \cdot \vec{\boldsymbol{a}} - \boldsymbol{U} \qquad (4)$$

This is the global motion compensation (GMC). For motion vectors of the background, GMC will give mostly 0. For motion vectors of the foreground moving objects, GMC will reveal the local motion of objects, see Figure 4(b).

Moving objects are detected by thresholding the magnitude of the local motion followed by simple morphological operations to delete small false objects and to fill noisy spots.

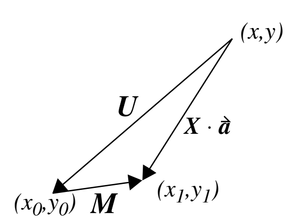
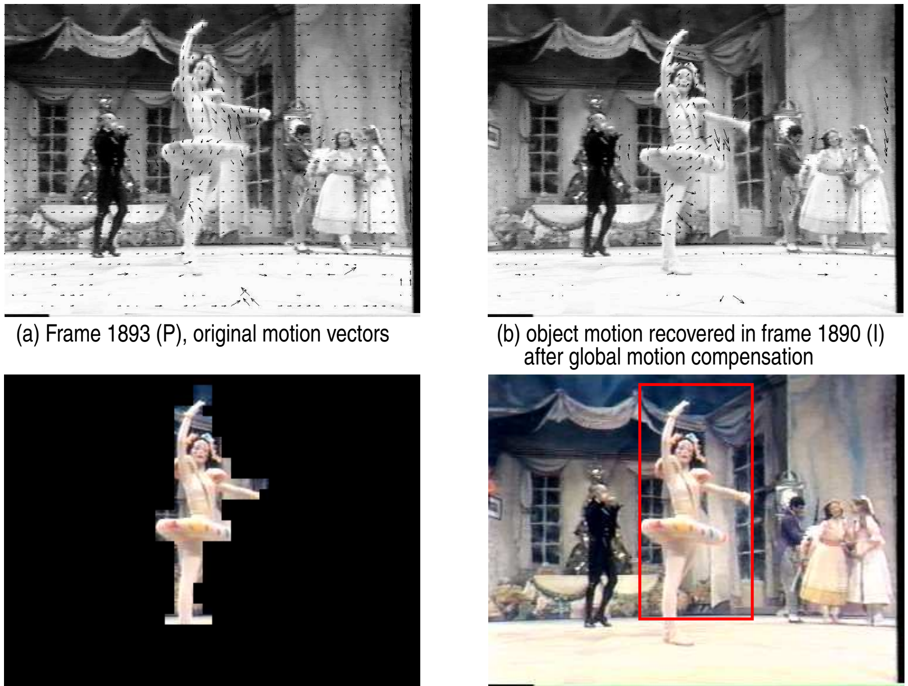


FIGURE 3. Relation among global motion $X \cdot \vec{a}$, local motion M and net displacement U



(a) Frame 1893 (P), original motion vectors

(b) object motion recovered in frame 1890 (I) after global motion compensation

(c) moving object is extracted

(d) Bounding box of the object

FIGURE 4. Camera Parameter and Moving Object Detection

See Figure 4(c) for extracted moving object. The DCT coefficients of the moving object are extracted for query purpose. The outermost points of the object are used to form a bounding box. The location and size of the bounding boxes are saved for later browsing and indexing, see Figure 4(d).

To track the moving objects throughout a video shot, we first select a reference frame where the moving object is initially detected. Secondly, we obtain the centroid of each moving object by taking the first moment of the object's shape. Thirdly, we map the centroid of each object onto the reference frame using the global camera parameters ***à***. When tracking multiple objects, color and texture of the object can be used to distinguishing them. The motion trajectory of each moving object is formed by repeatedly mapping the centroid until the object has stopped or moved out of the picture or the next scene comes. Finally, filters such as a median filter are used to smooth out the trajectories.

Visual features of the extracted objects, such as color, textures, and shape, can be used to provide content-based visual query of these and associated video scenes.

5. COMPRESSED VIDEO EDITING

Based on the source material, we classify video editing into two stages: the production stage and the post-production stage. The production stage editing are based on original analog or digital footages from cameras. At this level, sophisticated hardware is usually used to guarantee the ease of editing and the highest possible video quality. Commercially available digital video systems such as AVID, Media100 and D-Vision etc., currently use the Motion JPEG compression [17]. The compression ratio varies from 3:1 to about 10:1. With the latest technology, high bandwidth bus technology will make uncompressed video editing possible. The output video from the production stage will be eventually converted to more heavily compressed bitstreams (e.g. MPEG2) for broadcasting or storage.

At the post-production stage, the users will retrieve the MPEG bitstreams according to their needs and perform desired editing. Post-production video editing shall not be available only to users that have sophisticated video hardware. We develop the CVEPS using a pure software and compressed domain approach particularly for this purpose.

We will discuss technical issues of editing MPEG video such as frame type conversion, maintaining bitrate integrity and algorithms for creating common special effects in the compressed domain.

5.1 Basic Editing Functions: Cut and Paste MPEG Video

When cutting and pasting several MPEG video segments to create a new sequence, a straightforward way is to decode all the segments and re-encode. This method is computation intensive, and the output picture will suffer generation loss multiple times.

We apply the basic editing functions directly in the compressed domain. Figure 5 illustrates a scenario of cutting two arbitrary segments from the middle of two separate video streams and merging them to form a new compressed video stream.

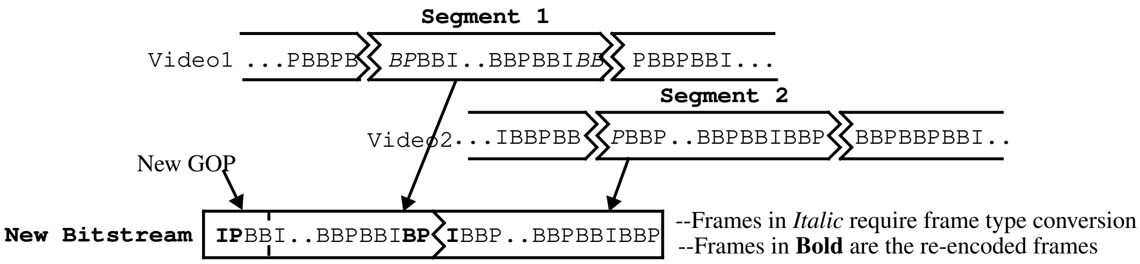


FIGURE 5. Cut and Paste MPEG bitstreams in the compressed domain.

5.1.1 Issue I — Frame Type Conversion

The MPEG video consists of GOP units. Each GOP starts with an I frame. We only need to re-encode few frames which are out of the GOP boundary at the beginning or ending part of the segments. The newly created GOP may have a different size, but it is still conformable to the MPEG format. Details of the frame type conversion may be found in [15]. After type conversion, each segment is independently decodable and can be pasted together back to back to form a new sequence. Figure 5 shows cutting out segment 1 and 2 at arbitrary location to form a new bitstream. The beginning few frames of a segment is re-encoded to form a shorter new GOP.

5.1.2 Issue II — Decoder Video Buffer Control

For constant bitrate MPEG video, the MPEG encoder solves the rate control problem with the "virtual buffer" [12,13], a simulation module of the decoder buffer. Before quantizing each macroblock, it sets the reference value of the quantization parameter based on the fullness of the "virtual buffer."

When cutting and pasting arbitrary segments from different compressed video streams of the same bitrate, the integrity of the original rate control mechanism is lost. For example, Figure 6 (a) shows the video buffer occupancy after connecting four segments. The video buffer size is 1Mbits. Each segment consists of 49 frames, starts with an I frame and ends with an I frame. The video buffer decreases to a very low level after the first I frame of Seg3. When Seg4 is pasted, the buffer starts to have the underflow problem.

The overflow problem can be easily solved by stuffing zero bits at the end of a slice or a picture whenever the buffer reaches a very high level. The underflow problem can be solved by inserting a synthetic transitional GOP [15] which has a lower average bitrate than normal GOPs or by applying rate shaping algorithm [10] to reduce the bitrate of the boundary I/P frames.

5.2 Extended Editing Functions: Special Effects in the Compressed Domain

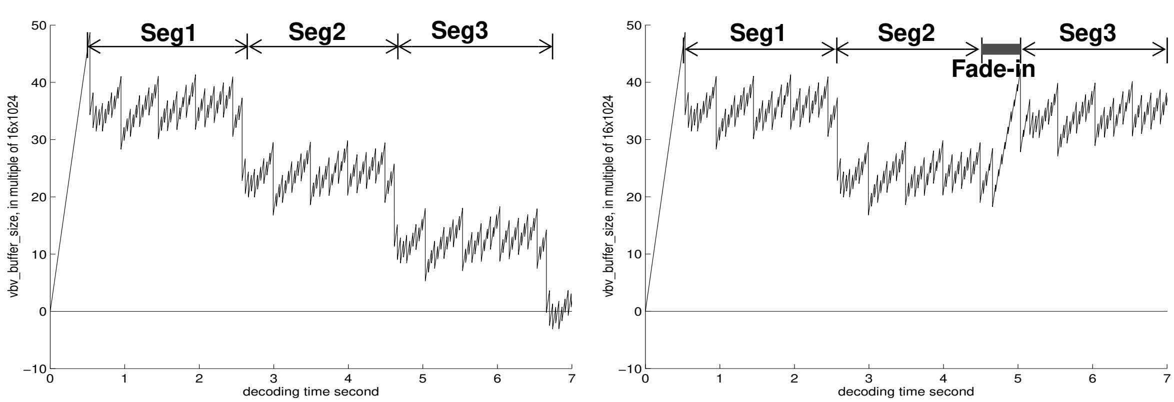
In addition to the basic editing function "cut and paste", several special visual effects can be created in the compressed domain. For I frames, the basic compression component is the Discrete Cosine Transform (DCT), which we denote as

$$F(u, v) = DCT(f(x, y)) \tag{5}$$

Basic linear operations like the intensity addition and scaling can done as follows [7],

$$DCT(f_1(x, y) + f_2(x, y)) = F_1(u, v) + F_2(u, v) \tag{6}$$

$$DCT(\alpha \cdot f(x, y)) = \alpha \cdot F(u, v) \tag{7}$$



(a) Decoder video buffer underflows when pasting segments together. (b) With the proposed synthetic fade-in connecting Seg2 and Seg3, buffer remains normal.

FIGURE 6. Connecting MPEG video segments in the compressed domain.

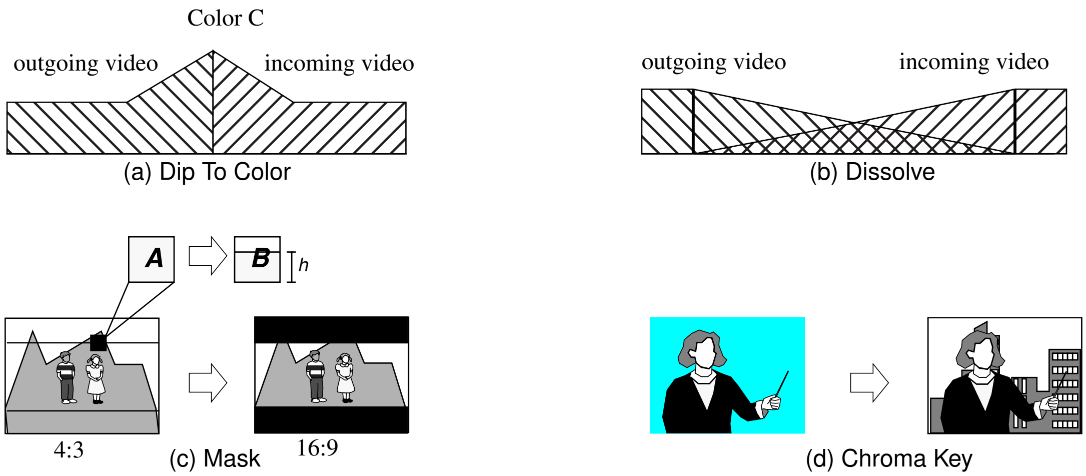


FIGURE 7. Some Special Visual Effects

Algorithms for other operations such as spatial scaling, translation, and filtering in DCT domain can be found in [7]. Usually, the DCT of the output video, Y, can be obtained by linear matrix operations of the input DCT, P_i, as follows

$$Y = \sum_i W_i \cdot P_i \cdot H_i \tag{8}$$

where H_i and W_i are special filter coefficient matrices in the DCT domain. For motion compensated B and P frames, the compressed-domain manipulation functions can be implemented in two ways. First, in [7,8], we have proposed transform-domain techniques to convert B and P frames to intraframe DCT coefficients, on which the above techniques can be readily applied. An alternative is to keep the B/P structure (i.e., DCT of residual errors and motion vectors) and develop algorithms directly utilizing these data. The following are some examples of typically used editing functions such as *Blend, Film, Key, Motion*, and *Wipe* etc. [1], some of which are illustrated in Figure 7.

5.2.1 Blend Effects

Blend effects are generally two-channel effects: to create a transitional connection between two video segments. Two commonly used ones are: dip to color and dissolve.

Dip to color

Fades from the outgoing video to black, white, or any color and then fades to the incoming video. Since the outgoing and incoming video do not overlap, this effect is achieved by modifying the DCT coefficients in outgoing and incoming video frames. The normalized color level increment ΔI_k, is added to the DCT DC of each macroblock,

$$\Delta I_k = \frac{N \cdot C_k}{n} \tag{9}$$

where *k=0,1,2* standards for luminance and two chrominance channels, C_k is the dip-to color, n is the total number of frames in this effect, and the constant N is the DCT block size (default: 8).

This operation is directly applied to the DCT coefficients in I frames or DCT coefficients of residual in B and P frames. For a typical MPEG..$I_0B_1B_2P_3B_4B_5....B_{n-1}B_n$, with I/P frequency *M=3*, the operation for each type is:

$$\textit{I frame:} \quad F'_k = F_k + i \cdot \Delta I_k \tag{10}$$

$$\textit{P frame:} \quad F'_k = F_k + M \cdot \Delta I_k \tag{11}$$

$$\textit{B frame:} \quad F'_k = F_k + mod(i, M-1) \cdot \Delta I_k \tag{12}$$

where F_k, F'_k are the original and the modified DCT DC value, and *i=0,1,...,n* is the frame number.

Dissolve

The outgoing video fades out while the incoming video fades in. When there is no or low motion in the two videos, this effects can be approximated by the linear combination of the two video:

$$F(u, v, t) = \alpha(t) \cdot F_1(u, v, t_1) + (1 - \alpha(t)) \cdot F_2(u, v, t_2) \tag{13}$$

where $\alpha(t)$ is a weighing function changing from 100% to 0%, user may modify it with any rate; $F_1(u, v, t_1)$ is the last I frame of the outgoing video and $F_2(u, v, t_2)$ is the first I frame of the incoming video. The resulting effect is a dissolve transition from a frozen frame of video 1 to another frozen frame of video 2. However, when either of the video contains high motion, re-encoding of few frames in the transitional period will be required.

5.2.2 Film Effects

Film effects refers to masking video with 4:3 aspect ratio to different aspect ratios such as 1:1.66, 1:1.85, 1:2.35, and 16:9. For I frames, the DCT blocks outside of the desired region are set to 0, and the blocks that lie on the masking boundaries are recalculated using the simplified DCT translation algorithm described in [7].

$$DCT(B) = DCT(H) \cdot DCT(A) \text{, where } H = \begin{bmatrix} 0 & 0 \\ 0 & I_h \end{bmatrix} \quad (14)$$

where A is an original block located on the boundary, B is the new masked block, and I_h is the identity matrix with size $h \times h$, as shown in Figure 7(c).

For P and B frames, only macroblocks with motion vectors pointing outside of the masking region need to be re-encoded. Macroblocks with motion vectors pointing inside do not need any modification. Efficient algorithms for reencoding macroblocks are described in [7,8].

5.2.3 Key Effects

Key effects are often used for compositing an anchorperson with a scene, such as a weatherman in front of a satellite weather map. In spatial domain, this is done by shooting the first video with a uniform background color (usually blue), then replace every blue color pixel with the second video. In compressed domain, we segment the first video into foreground and background regions by detecting the blue color. Then we replace the macroblocks with just blue background color with corresponding macroblocks from the second video. We need to re-encode the macroblocks lying on the region boundary and the macroblocks with motion vector pointing outside their regions. The percentage of macroblocks which need re-encoding depends on the video type and MPEG encoder design. Some simulation results were reported in [7]. The complexity of the re-encoding process can be reduced by using the pre-existing motion vectors to infer new motion estimation parameters.

5.2.4 Motion Effects

Motion effects include *Freeze Frame, Variable Speed* and *Strobe Motion*.

Freeze Frame

Since the freeze effect is usually longer than 1 second, simply inserting duplicated frames (e.g. zero-energy P frames) for a long period of time is not desirable for interactive playback (e.g. random search) due to the lack of frequent I frames. We need to place an I frame at regular short interval. Therefore, the frozen frame is converted to an I frame if it were B/P frame. And the rest of the GOP is filled with duplicated P frames. All the macroblocks in the duplicated P frames are set to Motion Compensation Not Coded (i.e., 0 motion vector, and the 0 residue error blocks are not coded).

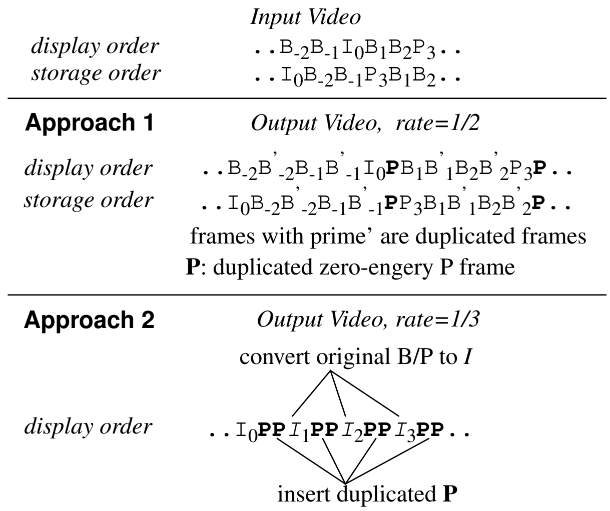


FIGURE 8. Two Approaches of Slow Motion Effect

Variable Speed

For fast motion, B, P, and I frames are subsequently dropped according to the variable speed.

For slow motion, depending on the slow motion rate, two approaches are used as shown in Figure 8. In approach 1, duplicated frames are inserted with no decoding involved. But the I/P frame delay is multiplied by the inverse of the motion rate. For example, I_0 of output video must be transmitted 4 frames earlier, rather than the original 2 frames. This approach is suitable for rate 1/2 and up.

In approach 2, original P/B frames are converted to I frames using our DCT domain techniques [7]. Then duplicated P frames will be inserted between I frames. This approach reduces the frame delay, however extra DCT domain manipulations are required.

Strobe Motion

Strobe motion is a combination of *Freeze Frame* and *Variable Speed.* It is done by dropping original B/P frame and inserting duplicated P frames.

As described in Section 5.1.2, to avoid decoder buffer to overflow (e.g., inserted frame is too small) in constant bitrate video, we may stuff redundant bits to the inserted P frames. To avoid any buffer underflow, we may apply rate adjustment techniques described in Section 5.1.2.

5.3 Advantages of Compressed Domain Approaches

For the basic editing function: cut and paste, the compressed domain approach runs at least 60 times[1] faster than the straightforward approach (decode-edit-encode). That is based on 12 second per cut on average, one P or I frame for

1. Based on analytical estimation of computation complexity as well as software simulation results.

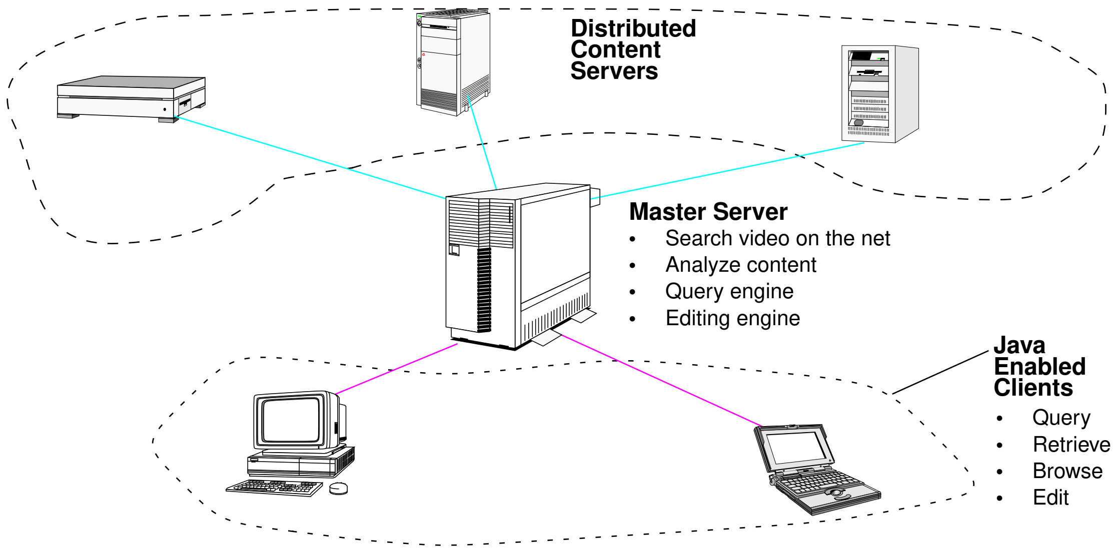


FIGURE 9. Client-Server Based CVEPS System

every two B frames, and cutting at arbitrary locations. The speedup can go over 600 times if we allow cuts only at P frames. The longer the segments are, the higher speedup we gain. The compressed domain approach also avoids quality degradation because the second quantization in the re-encoding process is avoided. For example, we observed an average 3.6 dB gain for a 60 frame segment (608x224, 4.0 Mbps). Only the re-encoded boundary GOP will suffer the 3 to 4 dB quality loss as in the straightforward approach.

6. SYSTEM DESIGN

The CVEPS uses a distributed client-server model as illustrated in Figure 9. The master server is linked with WebSEEk which searches for image and video files over the WWW. Once a video file is found on any other hosts or WWW distributed content servers, it will be downloaded and preprocessed by the master server to extract the keyframes and associated visual features such as camera motion, moving objects, color, texture, and temporal variance etc. The HTTP address of video and the extracted features will be stored on the master server. This client-server model gives the client much richer resources that are not constrained to the client's local environment.

The client is implemented with Java applets. The client may open any video at the server and browse the keyframes hierarchically using story structure or content clustering methods [26]. All the keyframes are hyperlinked to the WebSEEk's query engine so that the keyframes or objects may be used to form new visual queries for new videos or images over the entire master server.

To view the video, the user may simply drag the keyframe which represents for a video shot to the source monitor of the editing interface, see Figure 9. A low resolution copy of the video shot will be sent to the client by the server. The client can use the interactive MPEG2 viewer/decoder to do random access, step forward, fast forward/reverse and normal playback. The MPEG2 decoder is written in C and compiled as a run-time shared library to be called by the Java client.

The user may also turn on the *VideoMap* option of the MPEG2 player. This option will invoke the display of the bounding boxes of any moving objects detected (described in Section 4.3). By clicking the mouse inside the bounding box, the client will send a request to the server to get additional information of the object (e.g. a hyperlinked home page) or invoke content-browsed visual query using this object as a template.

To edit the video, the user may mark in/out any segment of the video shot in the source monitor to splice-in or overwrite to the new sequence in the record monitor. A separate timeline window will show the resulting video/audio tracks and the detailed information of each included video shot. The user may also insert special effects as described in Section 5.2.

During the editing, only the Edit Decision List (EDL) is created. The new sequence must be rendered before it can be displayed. There are three levels of rendering. At the first level, the client uses C routines from its shared libraries to render only the straight cuts at low resolution without showing the special effect. At the second level, the client may send the EDL to the server for generating the new low resolution video with desired special effects. Finally, when the client is done with the editing, the master server will generate a full resolution video with all the effects from the highest quality source video which is located at either the master server or the distributed remote content servers.

7. CONCLUSION

We presented a Compressed Video Editing and Parsing System with our proposed compressed-domain video manipulation and indexing techniques. The CVEPS processes the compressed video to automatically extract key visual features such as scene cut, camera operation parameters, moving objects, and then visual features (e.g., color, motion speed and trajectory). Content based queries are formed with the above visual features for retrieving new video clips. The CVEPS also provides tools for editing compressed video and creating special effects. We have shown that the compressed domain approach can achieve significant system performance improvement in speed, quality, and storage. Software implementations of the proposed algorithms have been developed in C and Java employing a client-server model over the WWW. The client-server implementation is particularly useful for users with access to regular computers or even less powerful devices (such as lightweight mobile units).

8. ACKNOWLEDGMENT

This work is supported in part by a NSF CAREER award (IRI-9501266), HP, Intel, and the ADVENT project of Columbia University. Implementation of the JAVA editing interface of the CVEPS was contributed by Jan Stanger while he was a research intern at Columbia University. The video shot browsing and clustering interface was contributed by Di Zhong.

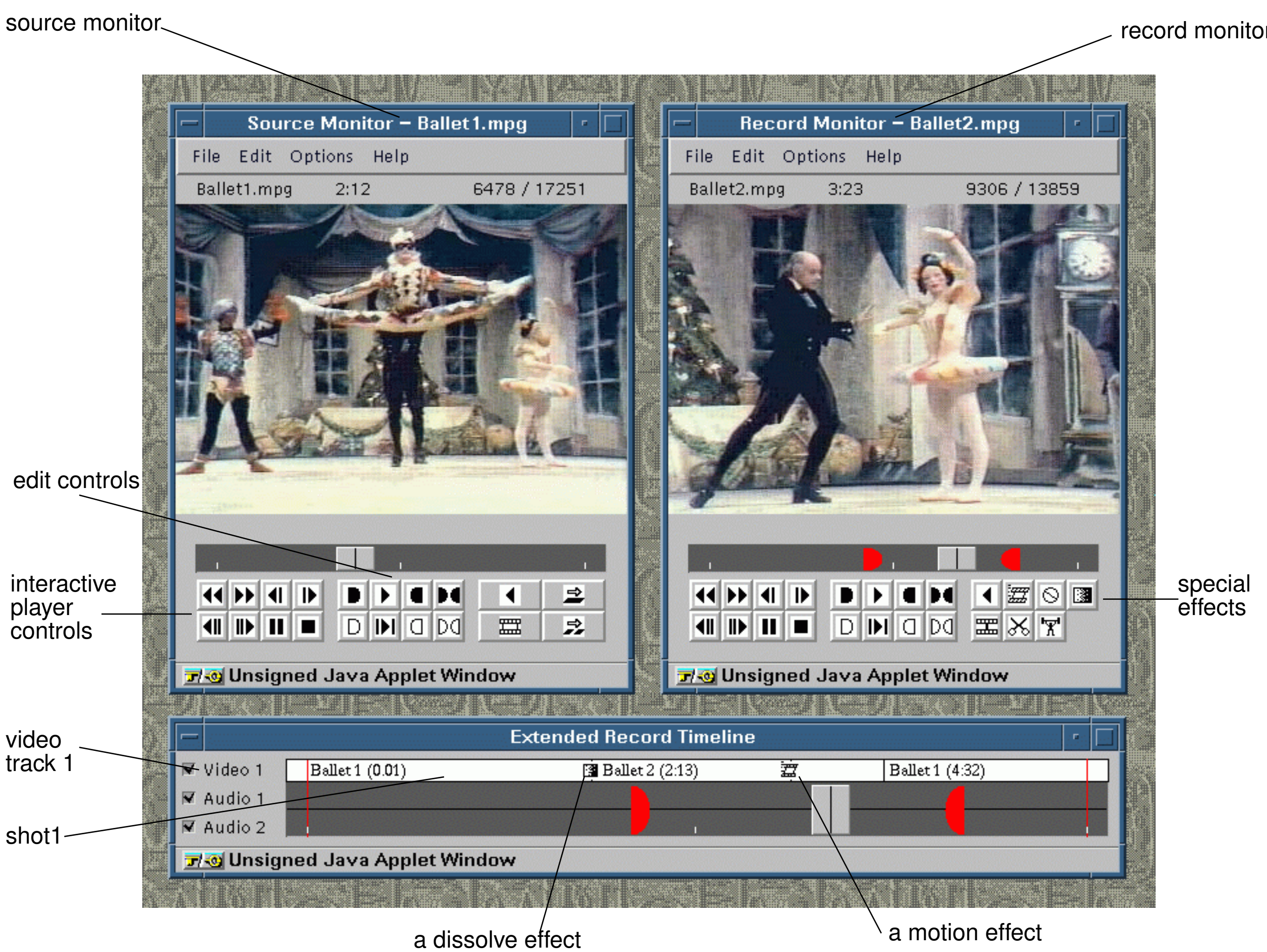


FIGURE 10. The CVEPS Video Editor Java Interface

9. REFERENCES

1. *AVID Effects Reference Guide*, Avid Media Composer and Film Composer, Release 5.50, June, 1995.
2. A. Akutsu, Y. Tonomura, H. Hashimoto, and Y. Ohba, “Video Indexing Using Motion Vectors,” *SPIE Visual Communications and Image Processing* 1992, Vol. 1818, pp. 1522-1530.
3. A. M. Alattar, “Detecting and Compressing Dissolve Regions in Video Sequences with DVI Multimedia Image Compression Algorithm,” *ISCAS* 1993, pp. 13-16.
4. F. Arman, A. Hsu, and M-Y. Chiu, “Image Processing on Compressed Data for Large Video Databases,” *Proceedings of ACM Multimedia ’93*, June 1993, pp. 267-272.
5. J. Bergen, P. Anandan, K. Hanna, and R. Hingorani, “Hierarchical Model-Based Motion estimation,” *2nd ECCV*, 1992, pp. 237-252.
6. S.-F. Chang, “Compressed-Domain Techniques for Image/Video Indexing and Manipulation,” *IEEE Intern. Conf. on Image Processing, ICIP 95, Special Session on Digital Image/Video Libraries and Video-on-demand*, Oct. 1995, Washington DC.
7. S.-F. Chang, Compositing and Manipulation of Video Signals for Multimedia Network Video Services, Ph.D. Dissertation, U.C. Berkeley, Aug., 1993. Postscript files also available from *http://www.ctr.columbia.edu/~sfchang/thesis93*.
8. S.-F. Chang and D.G. Messerschmitt, “Manipulation and Compositing of MC-DCT Compressed Video,” *IEEE Journal of Selected Areas in Communications, Special Issue on Intelligent Signal Processing*, pp. 1-11, Jan. 1995.
9. N. Dimitrova, and F. Golshani, “Motion Recovery for Video content Classification,” *ACM Transactions on Information Systems*, Vol. 13, No. 4, October 1995, pp. 408-439.
10. A. Eleftheriadis, and D. Anastassiou, “Optimal Data Partitioning of MPEG-2 Coded Video,” *Proceedings of 1st International Conference on Image Processing (ICIP-94)*, Austin, Texas, November 1994.
11. A. Hampapur, R. Jain, and T. E. Weymouth, “Feature Based Digital Video Indexing,” *IFIP2.6 Visual Database Systems, III,* Switzerland, March, 95.
12. ISO/IEC 13818 - 2 Committee Draft (MPEG-2).
13. ISO/IEC/JTC1/SC29/WG11, MPEG Document AVC-400, Test Model 3.
14. J. Meng, Y. Juan, and S-F Chang, “Scene Change Detection in a MPEG Compressed Video Sequence,” *IS&T/SPIE Symposium Proceedings,* Vol. 2419, Feb. 1995, San Jose, California.
15. J. Meng and S.-F. Chang, “Tools for Compressed-Domain Video Indexing and Editing,” *SPIE Conference on Storage and Retrieval for Image and Video Database*, Vol. 2670, San Jose, California, Feb. 1996.
16. A. Nagasaka and Y. Tanaka, “Automatic Video Indexing and Full-Video Search for Object Appearances,” In E. Knuth and L. M. Wegner, editors, *Video Database Systems, II*, Elsevier Science Publishers B.V., North-Holland, 1992, pp. 113 - 127.
17. T. A. Ohanian, *Digital Nonlinear Editing: new approaches to editing film and video*, Focal Press, Boston, London, 1993.
18. H.S. Sawhney, S. Ayer, and M. Gorkani, “Model-Based 2D & 3D Dominant Motion Estimation for Mosaicking and Video Representation,” *Proc. Fifth Int’l conf. Computer Vision*, Los Alamitos, CA., 1995, pp. 583-390.
19. B. Shahraray, “Scene Change Detection and Content-Based Sampling of Video Sequences,” *SPIE Conf. Digital Image Compression: Algorithms and Technologies 1995*, Vol. 2419.
20. J. R. Smith and S.-F. Chang. “VisualSEEk: a Fully Automated Content-based Image Query System,” *ACM Multimedia ‘96*, November, 1996.
21. J. R. Smith and S.-F. Chang, “Searching for Images and Videos on the World-Wide Web,” Technical Report # 459-96-25, Center for Telecommunications Research, Columbia University, New York, August 1996. Also submitted to *ACM Multimedia Magazine*.
22. S.W. Smoliar, and H.J. Zhang, “Content-Based Video Indexing and Retrieval,” *IEEE Multimedia*, summer 1994, pp. 62-72.
23. J. Swartz, and B.C. Smith, “A Resolution Independent Video Language,” *Proceedings of ACM Multimedia ’95*, pp. 179-188.
24. M.M. Yeung, B.-L. Yeo, W. Wolf, and Bede Liu, “Video Browsing using Clustering and Scene Transitions on Compressed Sequences,” *IS&T/SPIE Symposium Proceedings,* Feb. 1995, San Jose, California. Vol. 2417, pp. 399-413.
25. Y.T. Tse, and R. L. Baker, “Global Zoom/Pan Estimation and Compensation For Video Compression” *Proceedings of ICASSP* 1991, pp.2725-2728.
26. D. Zhong, H.J. Zhang, and S.-F. Chang, “Clustering Methods for Video Browsing and Annotation,” *Storage and Retrieval for Still Image and Video Databases IV, IS&T/SPIE’s Electronic Imaging: Science & Technology 96*, Vol. 2670, San Jose, CA., Feb. 1996.