To appear in *SPIE Conference on Storage and Retrieval for Image and Video Database*, *Vol. 2670*, San Jose, CA, Feb. 1996.

Tools for Compressed-Domain Video Indexing and Editing[1]

Jianhao Meng and Shih-Fu Chang
Department of Electrical Engineering & Center for Telecommunications Research
Columbia University, New York, NY 10027, USA
{jmeng,sfchang}@ctr.columbia.edu

Abstract

Indexing and editing digital video directly in the compressed domain offer many advantages in terms of storage efficiency and processing speed. We have designed automatic tools in the compressed domain for extracting key visual features such as scene cut, dissolve, camera operations (zoom, pan), and moving object detection and tracking. In addition, we have developed algorithms to solve the decoder buffer control problems and allow users to "cut, copy and paste" arbitrary compressed video segments directly in the compressed domain. The compressed-domain approach does not require full decoding. Thus fast software implementations can be achieved. Our compressed video editing techniques will enhance the reusability of existing compressed videos.

Keywords: compressed-domain video manipulation, video indexing, video editing, motion recovery, object tracking

1. Introduction

Video compression is a key technology in many new media applications such as video on demand and desktop video studio. How to effectively index and manipulate the compressed video is one of the keys to the usability of massive video servers. A successful video server shall process content based queries and retrieve video clips fast and reliably based on user's input, which can be a sample picture, a hand-drawn outline of a desired object, a trajectory of a moving object, or just a few key words.

The problem of compressed video indexing and manipulation will be much easier if there were a series of automatic tools to track what the camera operations were; to identify who were and what happened in the scene[2]; to record where the shot[3] cutting boundaries and what the special effects were; and finally to pass these attributes to the encoder. Then those attributes would be good not only for indexing but also for model-based coding to achieve higher compression. Unfortunately, today's compression techniques [9][10] do not exploit the above issues.

Another critical technology for new media applications is the real-time compressed video editing. Professional video editing has started to change from linear editing, or mechanical tape based editing to digital non-linear editing, or random accessing of digital storage based editing [14]. Most of today's high end digital non-linear editing systems use the motion JPEG compression standard, which provides high video quality at low compression rate (3-15:1). Since there is no motion compensation between the neighboring

---

1. This work was supported in part by Intel Research Council, the National Science Foundation under a CAREER award (IRI-9501266), IBM under a 1995 Research Partnership (Faculty Development) Award, and sponsors of the ADVENT project of Columbia University.

2. Scene: a scene is confined to one or more shots which are taken in the same space [5].

3. Shot: a shot is any contiguous, unedited sequence of video frames from the moment when a camera is turned on until it is turned off [5].

image frames, JPEG based sequence can be easily edited. However, when storing or transmitting the final production, other compression techniques, such as MPEG, are more desirable due to the need of high compression rate. To the end users, they may want to edit video in the existing compressed form which is used in the video encoder or video server.

Our objective is to develop innovative, efficient tools for editing and automatically indexing the contents of compressed video. Compressed video indexing is a reverse engineering process, reciprocal to the production process. First, the compressed sequence is broken down to the shot segment level, e.g., using the motion vectors and DCT DC coefficients in MPEG sequences [12]. Then, the shot segments are organized into meaningful scenes by using scene clustering techniques we proposed in [20]. We further detect camera pan/zoom and track the objects within a shot segment. We also provide tools for users to freely compose new compressed sequences with the aid of scene cut information. Our on-going work also includes moving object detection and segmentation which enables object-based video manipulation.

2. Related Work

Video index refers to the video search key. Due to the spatial and temporal complexity of video data, video search key can be any attributes of the video sequence and it is application dependent. Video indexing using finite state models for parsing and retrieval of specific domain video, such as news video, was discussed by Smoliar and Zhang [17]. Hampapur *et al* [8] proposed feature based video indexing scheme, which uses low level machine derivable indices to map into the set of application specific desired video indices. Our goal is to provide tools to extract low level features from the compressed video data. Once the low level feature sets such as the camera motion, object motion, and object shape/color are extracted, many high level processing methods can be employed to extract the semantic context of the video. And a variety of image database retrieval techniques [16] can also be applied in comparing and retrieving video clips from the derived object database.

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* [15]; the searching for object appearance and using them in video indexing was proposed by Nagasaka *et al* [13]. In the compressed domain, research on detecting extended camera operations (zoom, pan) using motion vectors had been discussed in [19] and [1]. In [19], an algorithm was given to estimate the global zooming/panning parameter based on the motion vectors generated from the block matching method. However, it assumed low local object motion, thus cannot be effectively extended to general videos. Object motion tracking in MPEG video was also discussed by Dimitrova *et al* [6], however, camera operations were not taken in consideration for object motion recovery. We propose an innovative method which combines histogram segmentation, discrete search and least squares method to estimate and detect camera operation parameters as well as the object motion trajectories.

This paper is organized as the following: camera operation parameters estimation and moving object detection, compressed video editing, and system user interface design.

3. Camera Operation Parameters Estimation and Moving Object Detection

Within a shot segment, low level visual features in a video sequence, such as camera zoom/pan and moving objects are useful information in indexing video databases. We have reported several techniques for detecting scene cut, dissolve, fade in/out in the compressed domain in [12]. In this section, we present new techniques for detecting and estimation camera zoom and pan from the motion vectors in the MPEG video.

The motion vectors in MPEG are 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 useful information for estimating the camera parameters in sequences that do not contain large dark or uniform regions.

We extract low dimension camera parameters to assist indexing of shot segment. A two dimensional model is used. When the distance between the object/background surfaces and the camera is large, it is usually sufficient to use a three parameter set $\{f, p_x, p_y\}$ to model the camera motion:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = f \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \quad (1)$$

where *(x,y)* is the coordinate of a point in the reference frame, *(x',y')* is the corresponding location in the current frame due to camera operation, f is the zooming factor, and (p_x,p_y) are the panning factor. For the convenience of estimation, we can rewrite (1) using the definition of motion vector, $\begin{bmatrix} u & v \end{bmatrix}^T = \begin{bmatrix} x & y \end{bmatrix}^T - \begin{bmatrix} x' & y' \end{bmatrix}^T$, where T denotes vector transpose,

$$\begin{bmatrix} u \\ v \end{bmatrix} = F \cdot \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} P_x \\ P_y \end{bmatrix} \quad (2)$$

where $F = (1-f)/f$ and $\begin{bmatrix} P_x & P_y \end{bmatrix}^T = -\begin{bmatrix} p_x & p_y \end{bmatrix}^T / f$. Thus, we can estimate the global camera parameter set $\{F,P_x,P_y\}$ from the macroblock coordinates and its motion vectors in the current frame.

3.1 Global Camera Parameter Estimation

Given the motion vectors for each macroblocks, one can find the global parameter by using the least squares (LS) method, that is to find a set of parameters which minimizes the squared error between the estimated motion vectors (using the estimated $\{F,P_x,P_y\}$ in (2)) and the actual motion vectors obtained from the MPEG stream [19]. In other words, find $\{F,P_x,P_y\}$, so that $S\{F,P_x,P_y)$ is minimized,

$$S(f, P_x, P_y) = \sum_{i \in \Omega} [(\hat{u}_i - u_i)^2 + (\hat{v}_i - v_i)^2] \quad (3)$$

where Ω contains all macroblocks, $\begin{bmatrix} \hat{u} & \hat{v} \end{bmatrix}^T$ is the estimated motion vector from the 2D motion model in (2).

The LS method assumes that there is no object motion. When there is a large region of object motion, the LS method is not accurate even after a few iterations. To overcome this problem, we developed a new histogram segmentation method to estimate panning and zooming parameters.

First, we check if the shot contains only pure camera panning, which corresponds to a single dominant motion direction. The histogram of motion vector angles with respect to the origin is calculated. Motion vector angles are quantized to a number of bins. If there is a peak count in one bin, then the dominant motion $\begin{bmatrix} \hat{u} & \hat{v} \end{bmatrix}^T$ is estimated by taking the average of motion vectors in that peak bin. A pure panning is declared with global parameter $\{0, \hat{u}, \hat{v}\}$, where 0 indicates zero zooming effect.

If no single direction is found to be dominant, and the average magnitude of the motion vectors is much greater than 0, then camera zooming is involved and a convergence or divergence point can be found. A convergence point corresponds to zooming in and a divergence point corresponds to zooming out. We detect the convergence or divergence point by combining a discrete search and a LS method:

1. Assign a counter for each 4 by 4 block in the image space.
2. Extend each motion vector across the whole image and increment every counter on its way.
3. Get the convergence or divergence point (x_0,y_0) by applying a LS estimation on the set of motion vectors that pass the block with the highest counts.

Since the motion vector is **0** at (x_0,y_0), plug it in Eq. (2) and we will get: $\begin{bmatrix}0 & 0\end{bmatrix}^T = F \cdot \begin{bmatrix}x_0 & y_0\end{bmatrix}^T + \begin{bmatrix}P_x & P_y\end{bmatrix}^T$. Subtract this equation from Eq. (2): $F = u/(x' - x_0)$, and $F = v/(y' - y_0)$. Now we can estimate F from its maximum frequency [8]. Finally, P_x and P_y are obtained from Eq. (2) with the estimated F, $(x', y')=(x_0,y_0)$, and $[u,v]^T=[0,0]^T$.

Camera motion trajectory can be easily obtained from a series of the pan/zoom factors. One may also construct a mosaic frame from the global parameter [15]. From the zooming speed and duration, we may classify the type of a shot: long shot, medium shot, close-up shot etc. Therefore, based on the needs of the application, higher level of indices can be generated from the low level camera motion indices.

3.2 Moving Object Detection

After we found the global camera parameters $\{F,P_x,P_y\}$, we may recover the true object motion by applying global motion compensation. If an object moved $[m_x\ m_y]^T$ from location (x,y) in the reference frame, then its new location (x',y') in the current frame after camera pan and zoom is:

$$\begin{bmatrix}x' \\ y'\end{bmatrix} = f \cdot \begin{bmatrix}x + m_x \\ y + m_y\end{bmatrix} + \begin{bmatrix}p_x \\ p_y\end{bmatrix} \tag{4}$$

Therefore the object motion $[m_x,m_y]^T$ can be recovered from motion vectors in the current frame provided that $\{f,p_x,p_y\}$ is known,

$$\begin{bmatrix}m_x \\ m_y\end{bmatrix} = \left(\begin{bmatrix}x' \\ y'\end{bmatrix} - \begin{bmatrix}p_x \\ p_y\end{bmatrix}\right) \Big/ f - \begin{bmatrix}x' \\ y'\end{bmatrix} - \begin{bmatrix}u \\ v\end{bmatrix} \tag{5}$$

where $f = 1/(F+1)$, $\begin{bmatrix}p_x & p_y\end{bmatrix}^T = -\begin{bmatrix}P_x & P_y\end{bmatrix}^T/(F+1)$. This is the global motion compensation (GMC). For macroblocks of the background, GMC will give mostly 0 motion vectors. For macroblocks of the foreground moving objects, GMC will provide an estimate of the true object motion, see Figure 1 (c) and (d).

Blocks with motion vector magnitudes much greater than their neighbors are considered as noise and filtered out. By using the histogram segmentation method, motion of objects can be found. Note that this will work even when there are multiple objects moving towards different directions, as long as there are clear peaks and valleys in the histogram. We then mark the blocks, whose motion vectors fall into the peak bins in the histogram, as object blocks. Finally, a simple morphological operation is used to delete those objects that consists with only one block and merge those blocks that are surrounded by object blocks.

The estimation process is repeated on all P and B frames within a shot. Currently, we use the forward motion vectors only. The backward motion vectors can be used to estimate the global parameters using a slightly modified version of Eq. (2) with $f'=1/f$ and $\begin{bmatrix}p_x' & p_y'\end{bmatrix}^T = -\begin{bmatrix}p_x & p_y\end{bmatrix}^T/f$. The backward motion vectors may be better for the B frames that are closer to the anchor frames later in display order. Also the global parameter and object motion in the I frame can be estimated from the B frame right before it.

We can construct trajectories of the moving objects from the object motion estimated from each frame. An initial reference frame, usually an I frame is selected. We then project object motion in the later frames to the initial frame using the estimated zoom/pan factor. We plot the results of a moving object's trajectory in Figure 1(e).

Note that this technique for separating foreground/background objects may not generate perfect results for arbitrary type of video due to the following reasons. First, the motion vectors from the existing motion compensation techniques in MPEG may not represent the true object motions especially in large uniform or dark region. Second, a 2D camera model is assumed in our current system. For improved results, 3D camera models, such as planar surface flow [3], may be used.

3.3 Indexing at the Object Level

Once the object regions are found, we extract the bitmap and the DCT DC coefficients of each object. For reasonably large object, we may collect enough of DC coefficients to perform color histogram discrimination to distinguish different objects. Color histogram is powerful in object similarity comparison, since the it is invariant to object translation, rotation and non-rigid motion [18].

Every time we detect an object in a new shot, we compare its color histogram with that of previous objects. Thus, we can avoid storing duplicated objects. We are also able to record the temporal instances of each object's appearance.

4. Compressed Video Editing

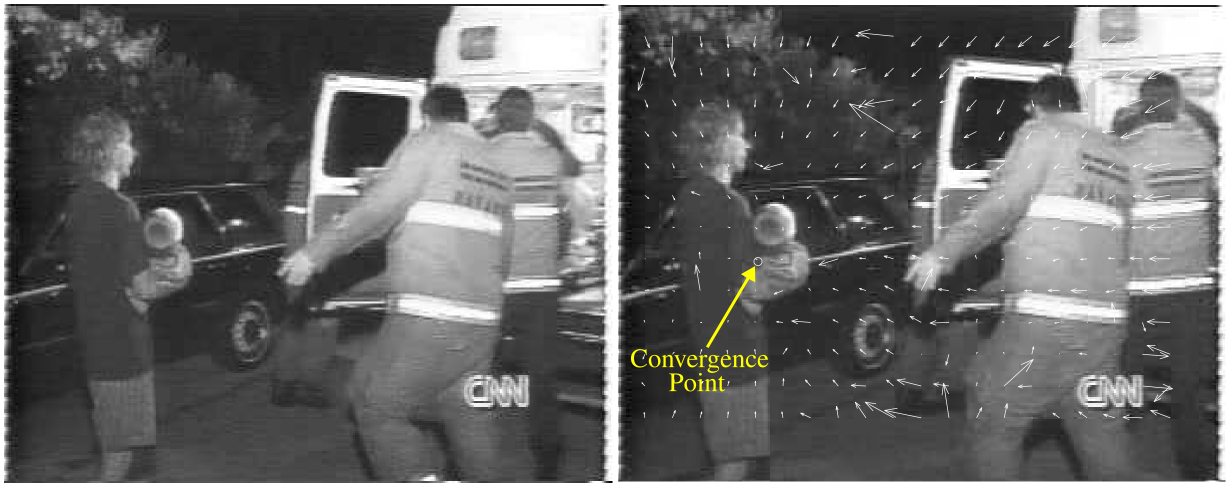
After the scene cut detection and object extraction, the user may want to compose new MPEG sequences based on the segment information provided by the system. A straightforward way is to decode the segments that the user specifies and re-encode the final composed sequence. This process is simple, but it demands expensive computations. Also, the final output pictures will suffer from quantization loss twice.

We apply the basic editing functions, i.e., video cut and paste (at arbitrary frame position), directly in the compressed domain. Though the function is simple in the uncompressed domain, it causes interesting technical issues in the compressed domain. Figure 2 illustrates a scenario of cutting two segments from the middle of two separate video streams and merging them to form a newly compressed video stream. The cut points (C1-C4, two in each stream) in original sequences are arbitrary.

First, the neighboring frames near the cutting points need to be processed and converted to the right frame types. Second, if the original video sequences are encoded with certain rate control constraints, merging two segments directly may cause decoder overflow or underflow problems near the merging point. We propose innovative techniques in the compressed domain to solve these problems. We have simulated these techniques using MPEG-2 standard-conforming tools and demonstrated the real-time implementations and quality improvement, compared with the uncompressed-domain approach.

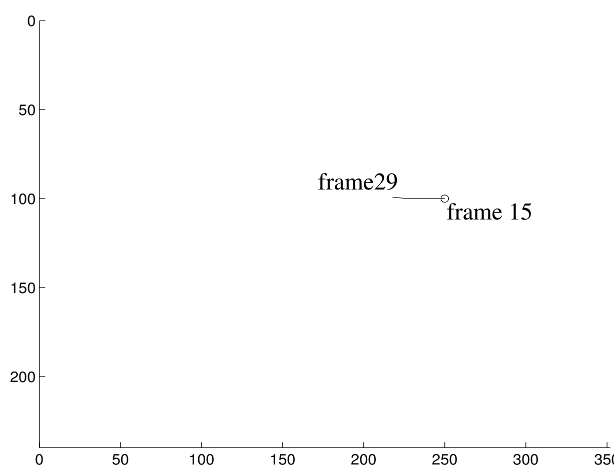
4.1 Issue I — Frame Type Conversion

The MPEG compression format requires the integrity of frame structures, which are composed of the group of pictures (GOP) units. Each GOP starts with an I frame. We only decode and re-encode certain 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. An example of frame type conversion is illustrated in Figure 2.



(a) (b)

(c) (d)



(e)

(a) Frame 15 of cnn_z.mpg (I frame)
(b) Frame 18 (P frame) and its motion vectors
(c) Frame 18, after global motion compensation (zooming in this case)
(d) Frame 18, moving object is extracted
(e) Object motion trajectory traced from frame 15 through 29

FIGURE 1. Camera Zooming and Moving Object Detection

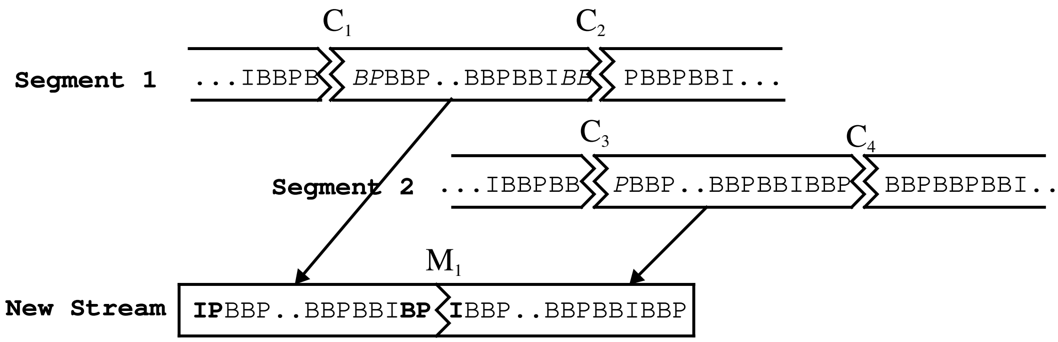


FIGURE 2. Cut and Paste MPEG bitstreams in the compressed domain.

4.2 Issue II — Decoder Video Buffer Control

Video encoders usually enforce some rate control mechanisms to ensure that the generated bitstream will not cause buffer violations at the decoders. We will discuss this issue and present our solutions as follows.

4.2.1 An Example of Rate Control — Overview of the MPEG Rate Control

For simplicity, we assume constant bit rate (CBR) compressed video here. The video buffer at the decoder is initially empty, then coded bits are placed in the buffer at a constant bitrate from the input channels, such as CBR network channels or CD-ROM interfaces. After an initial delay (specified by the *vbv_delay* field in the first I frame's picture header), the decoder will accumulate enough bits (about 80% full) in the video buffer and start to remove data from the buffer one frame at a time with a specific interval [11].

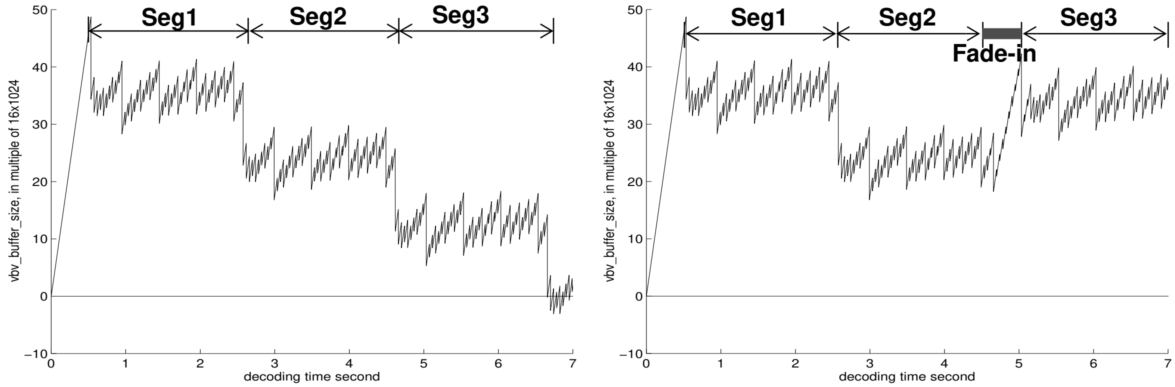
The video buffer may overflow when the bitstream has many low bitrate frames in series (such as a fade-in sequence) such that the decoder can not remove bits from the video buffer fast enough. Newly arrived bits will be lost due to lack of available buffer space. On the other hand, the video buffer may underflow if the bitstream has many large frame within a short period of time (due to scene cuts or editing). Therefore, at the supposed decoding time, the decoder cannot get a complete picture from the buffer.

The MPEG encoder solves the above rate control problem by using the "virtual buffer", 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."

The rate control mechanism usually is in effect for the entire duration of a video sequence to guarantee successful decoding of the entire sequence. 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.

4.3 Video Editing and Buffer Control Problems

After converting the boundary frames, each segment is complete and decodable which starts with an I frame and ends with an I or P frame. We may simply concatenate two or more such segments back to back to form a new compressed stream. However, as described in the previous section, this may cause problems (overflow/underflow) in the decoder buffers.



(a) Decoder video buffer underflows when pasting segments together.

(b) With the proposed synthetic fade-in connecting Seg2 and Seg3, buffer remains normal.

FIGURE 3. Connecting MPEG video segments in the compressed domain.

Figure 3 (a) shows the video buffer occupancy after connecting four segments. The video buffer size is 999424 bits. 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.

4.4 Solutions for Underflow/Overflow Issues

The overflow problem can be easily solved by redundant data (e.g., zeroes) stuffing whenever the buffer reaches a very high level. The underflow problem is more complex. We propose several effective solutions which insert a synthetic transitional sequence between the segments or modify the bitrate before pasting compressed streams.

4.4.1 Insert a Synthetic Fade-in Sequence

The first technique is to insert a short synthetic fade-in sequence, whose bitrate is much lower than the average, between two original segments. Thus, the video buffer can be brought up to the normal level and the underflow problem can be solved. The fade-in effect will also introduce a graceful effect in connecting two separate video segments (supposedly with very different content).

Specifically, the algorithm works as follows. At the end of a segment, if the buffer falls below a threshold, say 20% of the full level, we insert a GOP whose bit usage satisfies the following:

$$B_{gop} = r \cdot d - (B \cdot b - B_n) \tag{6}$$

where

- B_{gop}: target bit allocation for the inserted GOP
- r: nominal *bit_rate* of sequence
- d: duration of the fade-in, a default choice is *vbv_delay* of the second segment
- B: size of the decoder video buffer
- b: the buffer occupancy when decoder starts to decode, default: 80% (of B)
- B_n: the buffer occupancy at the end of the first segment

Intuitively, the number of bits we allocate to the inserted GOP is equal to the difference between the total input bits and the increment size we want to add to the decoder buffer in this period. The number of frames inserted, *N,* equals *d* times *frame_rate* (e.g. 30 frames/sec).

To get the fade-in effect, We insert a group of frames (*0~N-1*) in storage order: $I_0P_3B_1B_2P_6.....I_NB_{N-2}B_{N-1,}$ where I_N is the first I frame of the next segment to be connected. The brightness of this sequence increases linearly from pure darkness:

$$f_k = f_N \times \frac{k}{N} \qquad k = 0, 1,.., N-1 \tag{7}$$

where f_k is the brightness of *k*th frame of the fade-in sequence. In the compressed domain, we control the brightness by setting the correct DCT DC levels. The content of each frame is generated as follows:

- I_0 is a blank frame; contains only the headers without any DCT coefficients.
- For P frames, all macroblocks are motion compensated with (0,0) motion vector.
  - P_3: the brightness is 3/N of I_N; DCT coefficients of the residue errors are directly copied from I_N with the DC coefficients(DC's) set to 3/N of that of I_N and AC coefficients(AC's) unchanged.
  - $P_{6,9..}$: the brightness is 3/N of I_N more than the previous P frame. Therefore, after motion compensation, DCT DC's of the residual errors simply equal 3/N of that of I_N and AC's are 0.
- For B frames, we can use the above linear brightness relationship to set the right DCT coefficients easily. But in order to save bits, we simply set 2/3 of the macroblocks in the 1st B frame to Forward Motion Compensated and 1/3 to Backward MC (i.e., 2/3 macroblocks copied directly from the previous reference frame and 1/3 from the latter reference frame). Latter B frames can be handled by similar methods.

The amount of bits used by an I or a B frame is just the overhead of the headers and macroblock-modes, without any bits for transmitting motion vectors and DCT coefficients. The remaining allocated bits (B_{gop}) are assigned to P frames, in which P_3 uses most of the allocated bits to copy DCT DC and AC coefficients from I_N. If the allocated bits are not enough, we reduce the amount of high frequency AC coefficients in the P frame. If there are more bits left, we may stuff zero bits to P frames. The choice of *d* in (6) is flexible, as long as $B_{gop} > N \cdot B_{min} + B_p$, where B_{min} is the minimum overhead bits required to form a frame to cover the overhead bits used for headers and macroblock-modes etc.; B_p is the estimated bit usage by P frames. For simplicity, *d* can be set to *vbv_delay* (e.g., 0.5 second) for most applications.

See Figure 3 (b) for the buffer status after applying the above algorithm. With the inserted synthetic fade-in, the video buffer level is brought back to about 70% before decoding the first I frame of Segment 3. The GOP inserted uses about only 1/3 to 1/2 of the amount of bits of a normal GOP size. Note that as mentioned above, the length of the fade-in special effect is flexible. It involves the trade-off between the transition smoothness and inserted latency. Qualitatively, smoother transition implies a longer start-up latency for the second video segment. The next section proposes another extreme approach which achieves zero latency by using abrupt transition and cutting bitrate of the first segment.

4.4.2 Apply the Rate Modification Algorithm

Data partitioning is a feature of the MPEG2 high profile which provides the segmentation of a coded bitstream into two components or partitions. One component will contain the most critical information for transmitting over a reliable channel, while another component will contain the less important information for transmitting over a less reliable channel.

We can also apply this technique to solve the above buffer underflow problem. We may reduce the bitrate of an ending I/P frame by cutting off some of its high frequency DCT coefficients. The optimal cutting points of the DCT coefficients can be found by using the optimal data partitioning algorithms proposed in [7]. By reducing the bitrate of ending I/P frames, the buffer occupancy can be kept at the right level.

4.5 Comparison of Two Approaches

The drawback of the data partitioning approach is that for every segment, if the ending frame is an I or P frame, we need to partially decode to get its DCT coefficients, find the optimal cutting point, and finally re-encode it to a new, valid compressed frame. The complexity is higher than that of the fade-in insertion method. Reducing bits in the I frame would also cause quality degradation in the B frames right after it. However, this approach may still be useful for off-line applications.

Insertion of the fade-in effect between two connected video segments is attractive for real-time broadcasting. It is only needed when the decoder buffer reaches a very low level after connecting multiple segments. For example, the TV station can insert pre-compressed commercial video segments at any time. To avoid buffer underflow, a quick synthetic fade-in can be inserted before the commercial.

Although we only discussed the use of the fade-in special effect for controlling the decoder buffer level, other desirable special effects can be used in a similar fashion. Examples include fade-out, dissolve and wipe between segments.

5. System User Interface Design

We have developed a fully functioning prototype of compressed video editing and parsing system (CVEPS) at the Image and Advanced TV Lab of Columbia University. In the compressed video editor interface, every time a MPEG video is opened, its corresponding video buffer status is plotted against a time-line. The user may run the scene cut detection to extract a list a video icons. Each icon represents for a shot segment. At any time, the user may invoke our MPEG software viewer and the interactive VCR control panel which allows the user to do random search, step forward, fast forward/reverse, etc. The user can also use the mouse to highlight the time-line to select arbitrary video segments and apply “copy, cut and paste” operations, which also provide options of inserting fade in/out between the segments to be connected. Figure 4 illustrates a snapshot of the system interface. Note that all the underlying operations are done in the compressed domain, which provides great potential in improving system performance.

6. Conclusion

Tools for extracting key visual features from the compressed video and tools for editing compressed video are presented. Compressed video is first automatically broken into shot segments. Then, camera zoom/pan associated within each shot is detected and the moving objects’ attributes (color, histogram, motion trajectories) extracted. Based on the above automatically derived low-level features, semantic indices in the higher level can be formed. A prototype system illustrating real-time editing and indexing of MPEG compressed video has be developed with enhanced graphical user interface support.

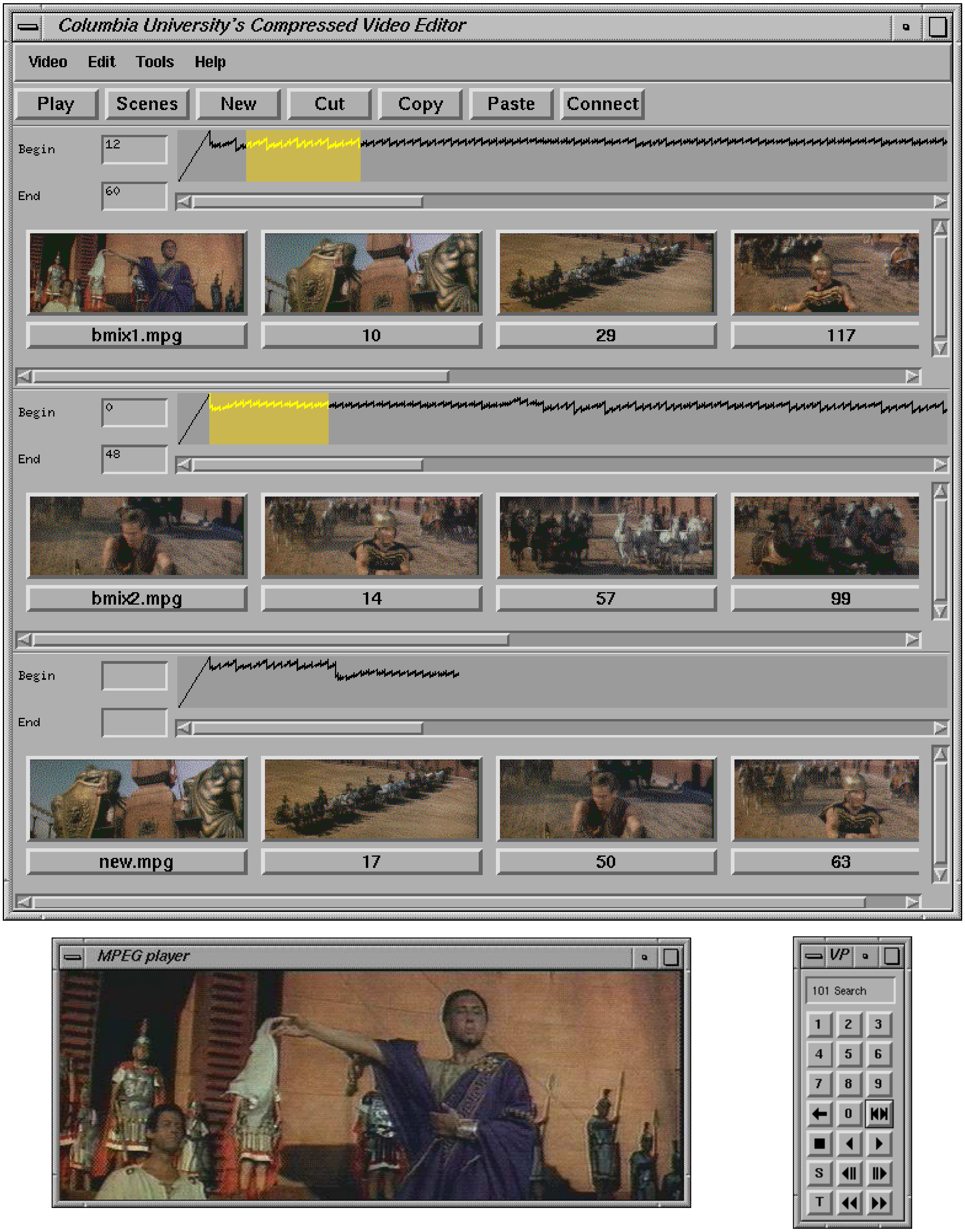


FIGURE 4. The Video Indexing/Editing System User Interface. The small panel shown on the right is the interactive control panel associated with the MPEG software viewer.

7. References

[1] 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.

[2] 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.

[3] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani, "Hierarchical Model-Based Motion estimation," *2nd ECCV*, 1992, pp. 237-252.

[4] S.-F. Chang, "Compressed-Domain Techniques for Image/Video Indexing and Manipulation", Invited Paper, *IEEE Intern. Conf. on Image Processing, ICIP 95, Special Session on Digital Image/Video Libraries and Video-on-demand*, Oct. 1995, Washington DC.

[5] T.-S. Chus, and Li.-Q. Ruan, "A Video Retrieval and Sequencing System," *ACM Transactions on Information Systems*, Vol. 13, No. 4, October 1995, pp. 373-407.

[6] 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.

[7] 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.

[8] A. Hampapur, R. Jain, and T. E. Weymouth, "Feature Based Digital Video Indexing," *IFIP2.6 Visual Database Systems, III,* Switzerland, March, 95.

[9] ISO/IEC 11172 - 2 Committee Draft (MPEG).

[10]ISO/IEC 13818 - 2 Committee Draft (MPEG-2).

[11]ISO/IEC/JTC1/SC29/WG11, MPEG Document AVC-400, Test Model 3.

[12]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.

[13]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.

[14]T. A. Ohanian, *Digital Nonlinear Editing: new approaches to editing film and video*, Focal Press, Boston, London, 1993.

[15]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; http://www.almaden.ibm.com/cs/reports/vision/dominant_motion.ps.Z.

[16]J.R. Smith, and S.-F. Chang, "Automated Image Retrieval Using Color and Texture," Technical Report No. 414-95-20, Center for Telecommunications Research, Columbia University, New York, 1995.

[17]S.W. Smoliar, and H.J. Zhang, "Content-Based Video Indexing and Retrieval," *IEEE Multimedia*, summer 1994, pp. 62-72.

[18]M.J. Swain, and D.H. Ballard, "Color Indexing," *International Journal of Computer Vision,* Vol. 7, No.1, pp.11-32, 1991.

[19]Y.T. Tse, and R. L. Baker, "Global Zoom/Pan Estimation and Compensation For Video Compression" *Proceedings of ICASSP* 1991, pp.2725-2728.

[20]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, Feb. 1996, San Jose, CA.