To appear in *ISCAS-96*, Atlanta, Georgia, May 1996.

BUFFER CONTROL TECHNIQUES FOR COMPRESSED-DOMAIN VIDEO EDITING[1]

Jianhao Meng and Shih-Fu Chang

Department of Electrical Engineering
and Center for Telecommunications Research
Columbia University, New York, NY 10027, USA
{jmeng, sfchang}@ctr.columbia.edu
http://www.ctr.columbia.edu/{~jmeng, ~sfchang}

ABSTRACT

This paper describes algorithms to edit compressed video sequences, e.g., MPEG, directly in the compressed domain. We propose innovative methods to solve the buffer overflow/underflow issues resulted from video editing. Since full decoding of input sequences and re-encoding of the new video sequence are not required, image quality degradation and intensive computations are avoided. High quality, real-time video editing can be achieved.

1. INTRODUCTION

Professional video editing has changed from linear editing, or mechanical tape based editing to digital non-linear editing, or random accessing of digital storage based editing [6]. 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). JPEG based sequence can be easily edited, since there is no motion compensation between the neighboring image frames. However, compression techniques, such as MPEG, are more desirable when storing or transmitting the final production sequences due to the 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.

Continuing our prior work on compressed-domain image manipulation/indexing [1][5], we address a basic editing function, i.e., cut and paste (at any arbitrary frame position). Figure 1 illustrates a scenario of cutting two segments from the middle of two separate video streams and merging them to form a new compressed video stream. The cut points (C_1-C_4, two in each stream) in original sequences are arbitrary. We use the popular MPEG compression form [3] to demonstrate our techniques, although our proposed techniques can be extended to more general compression forms.

First, the neighboring frames near the cutting points need to be converted to the right frame types. Second, if the original video sequences are encoded with rate control constraints (e.g., buffer constrained constant-bit-rate video encoding), merging two segments directly may cause decoder buffer overflow/underflow problems near the merging point. We propose innovative techniques in the compressed domain to solve these problems. We have simulated these techniques using MPEG2 standard-conforming tools and demonstrated the real-time implementations and quality improvement, compared to the uncompressed-domain approach.

2. EDITING OF MPEG VIDEO

A MPEG video sequence is composed of the group of pictures (GOP) units. The size of GOP is usually fixed at 12 or 15 frames. Each GOP starts with an intracoded frame (I frame), which all later frames in the current GOP will depend on. Each GOP can be independently decoded without previous GOPs.

When cutting out a segment of MPEG video at arbitrary location, we need to decode and re-encode only the frames that are out of the GOP boundary at the beginning or ending part of the segments. The newly created GOPs at the two ends may have a different size, but the segment is still conformable to the standard format. An example of frame type conversion is illustrated in Figure 1. Next, video segments can be pasted together at any order independently to complete the initial editing process.

When editing CBR sequences, the average bitrate of the new sequence shall be kept the same as the original ones and not to violate decoder buffer constraint. Thus we need to allocate appropriate amount of bits before re-encoding the frames that need type conversion, i.e., frames in *Italic* in Figure 1. Details of bit allocation can be found in section 10 of [4]. Unfortunately, when the new GOP contains too few frames, it is impossible to maintain the correct bitrate without drastically reducing I frame's bits, which will adversely affect the quality of every picture in the GOP.

To avoid the picture quality loss, two approaches are possible. The first one is to merge the new small GOP with

1. This work was supported in part by Intel, Siemens, the National Science Foundation under a CAREER award (IRI-9501266), and the ADVENT.

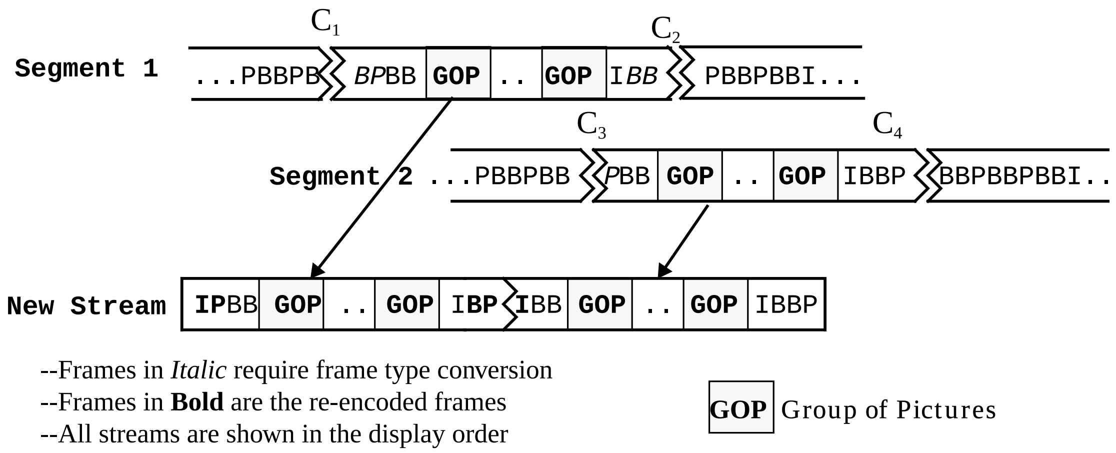


FIGURE 1. Cut and Paste MPEG bitstreams in the compressed domain.

the neighboring GOP and re-encode the second I frame to a P frame in order to save bits. The second approach is to leave this high bitrate GOP as is and apply rate modification algorithms later when pasting segments together, see the next two sections for details.

3. OVERVIEW OF THE MPEG RATE CONTROL

We assume editing of the constant bitrate compressed video here. When encoding CBR sequences, the encoder usually enforces some rate control mechanisms to ensure that the generated bitstream will not cause buffer violations at the decoders.

The decoder video buffer is initially empty, then coded data bits are placed into 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 fetch data from the buffer one frame at a time with a specific interval [3].

The decoder video buffer may overflow when the bitstream has many low bitrate frames in a 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 abrupt scene changes or video editing). The decoder will quickly drain the buffer. Then at the supposed decoding time, the decoder cannot get a complete picture from the buffer. The video/audio will be jittery.

The MPEG encoder solves the above rate control problem by using "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." When the buffer level is too high, i.e., bitrate is too low, it sets a smaller quantization parameter to generate more bits, and vice versa for buffer level is too low.

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

4. BUFFER CONTROL PROBLEMS

When cutting and pasting arbitrary segments from different compressed video streams of the same bitrate, the integrity of the original rate control mechanism may be lost. Because the new segments to be pasted may contain small size GOPs that has bitrate higher than the normal average bitrate. This may cause problems (overflow/underflow) in the decoder buffers.

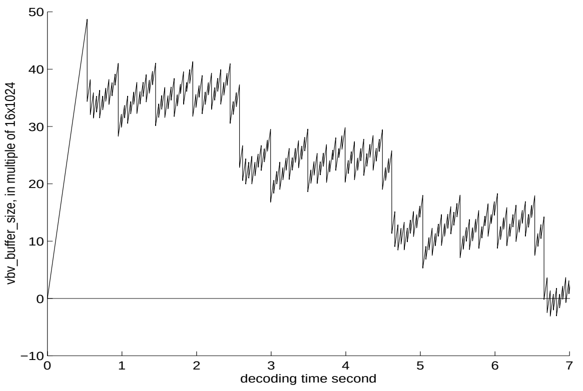
Figure 2 (a) shows the video buffer occupancy after connecting four segments. The video buffer size is 1 Mbits. Each segment consists of 49 frames, starts with an I frame and ends with a small GOP with just one I frame. The video buffer decreases to a very low level after the first I frame of Seg3. When Seg4 is pasted, the buffer starts to have the underflow problem.

4.1. Solutions for Underflow/Overflow Issues

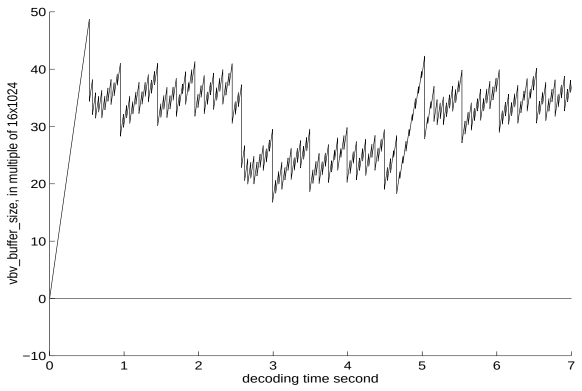
The overflow problem can be easily solved by redundant data (e.g., zeroes) stuffing whenever the buffer reaches a very high level. The underflow problem is more complex. We propose effective solutions which insert a synthetic transitional sequence between the segments or modify the bitrate before pasting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

4.2. 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 prob-



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



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

FIGURE 2. Connecting MPEG video segments in the compressed domain.

lem 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 30% of the full level, we insert a GOP whose bit usage satisfies the following:

$$B_{gop} = r \cdot d - (B \cdot b - B_n) \quad (1)$$

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:	normal 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. 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, 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 of 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 (1) is flexible, as long as

$$B_{gop} > N \cdot B_{min} + B_p , \quad (2)$$

where B_{min} is the minimum 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 2 (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. The next sub-section proposes an alternative

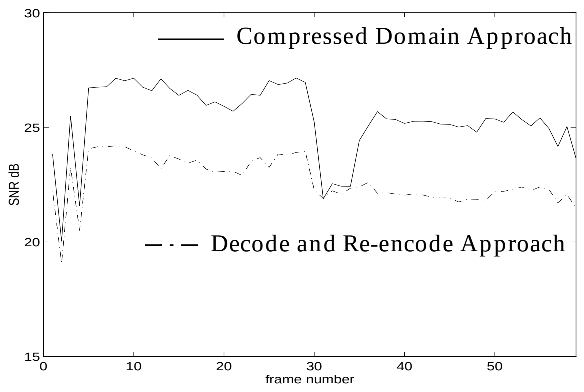


FIGURE 3. SNR Comparison

approach which achieves zero latency by using abrupt transition and cutting bitrate at the connecting point.

4.3. 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. One component will contain the most critical information, e.g., low frequency DCT coefficients, for transmitting over a reliable channel, while another will contain the less important information for transmitting over a less reliable channel.

We can also apply this technique to solve the buffer underflow problem. We may reduce the bitrate of a boundary GOP by cutting off some of its high frequency DCT coefficients of I or P frames. The optimal cutting points of the DCT coefficients can be found by using the optimal data partitioning algorithms proposed in [2].

The drawback of the data partitioning approach is that the computational complexity is higher than that of the fade-in insertion method. Reducing bits in the I frame would also cause quality degradation in all subsequent frames in the same GOP. However, this approach may still be useful for off-line applications.

Insertion of the fade-in effect between two video segments is attractive for real-time broadcasting. For example, the TV station can insert pre-compressed commercial video segments at any time. To avoid buffer underflow, a quick synthetic fade-in (other desirable special effects: fade-out, dissolve and wipe) can be inserted before the commercial.

5. SPEED-UP AND QUALITY PRESERVATION

Editing in the compressed domain saves computation and preserves picture quality. If there is a cut per 12 seconds (modem TV/movie), or about 360 frames per segment, at 30 frames/s. Using real-time codec to decode and re-encode this segment takes 24 sec. To operate in the compressed domain, the best case needs no re-encoding (i.e., cutting before an I and ending after I or P); the worst case is to cut before the B frames which require decoding of all previous I and P frames in the current GOP and then re-encode. On average, for GOP=15 and M=3 (IBBPBB..), the beginning part of the cut-out segment needs 4.6 frames of decoding and 1.9 frames of encoding; the ending part needs 3.3 and 0.67 frames respectively. The average speed-up is 68 times. For cutting only before/after I or P frames, the average speed-up is 600 times. The longer the shot segments are, the higher speed-up we gain.

Figure 3 shows the picture quality comparison of the two approaches. The edited segment has 60 frames, with a cut at frame 30 and frame 30-35 forms a shorter GOP. The video is 608x224, at 24 frames/s, and is originally encoded at 4.0 Mbps. With the spacial domain approach, the average signal to noise ratio (SNR) drops 3.6 dB. With the compressed domain approach, only few boundary frames suffer the re-encoding quality degradation, about 3-4 dB.

6. CONCLUSION

We describe efficient methods for editing MPEG video (e.g., cut/paste) in the compressed domain. The buffer overflow/underflow problems resulting from the arbitrary-position editing are solved by inserting synthetic transitional sequences between two video segments or by applying the data partitioning algorithm. We have fully implemented the compressed video editing system with enhanced graphical user interface, and automatic video feature extraction (e.g., zooming and panning).

REFERENCES

[1] 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.

[2] A. Eleftheriadis and D. Anastassiou, “Optimal Data Partitioning of MPEG-2 Coded Video,” *Proceedings, 1st International Conference on Image Processing (ICIP-94)*, Austin, Texas, November 1994.

[3] ISO/IEC 13818-2, MPEG-2, H.262, 1994.

[4] ISO-IEC/JTC1/SC29/WG11, Test Model 3, November 1992.

[5] 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.

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