COMPRESSED DOMAIN VIDEO EDITING

Ragip Kurceren and Fehmi Chebil
Nokia Research Center, Irving, TX USA

ABSTRACT

This paper addresses introduction of special video effects to the coded bitstreams in compressed domain. We propose novel compressed domain operations to achieve several video editing effects for motion-compensated family of video codecs. We then present detailed description of achieving blending of two video sequences. Results demonstrate significant computational savings over the conventional approach.

1. INTRODUCTION

Accelerated with the emergence of the latest camera equipped mobile phones, there has been explosive growth of captured image and video content. The convergence of the multimedia domains, including supported formats, and availability of easy content transfer between devices increase users' interaction with the captured information, such as sharing and content personalization. Although significant efforts have been spent on developing techniques for storage and transport of multimedia, personalization of content, in terms of manipulation of the content to achieve desired effects, still requires development of more efficient algorithms especially for power constrained devices such mobile devices and PDAs.

There are several PC-based commercial products providing video-editing functionalities, where the processing power and memory consumption have not been a constraint. Typically, the editing operations on video sequences are performed in their raw formats. More specifically, the compressed video is first decoded, the editing effects are introduced, and finally the output is re-encoded. This so called spatial domain video editing scheme cannot be applied on devices, such as mobile phones, that have limitations in processing power, storage space and battery.

This paper introduces optimized algorithms for several video editing effects, such as fading, blending sequences and logo insertion, in the compressed domain. Our approach is based on modifying the residual frames to achieve the desired effects without performing any motion estimation operation. Results demonstrate that significant speed-ups are achieved compared to the conventional spatial domain approach and to other existing methods in the literature.

In what follows, we briefly overview general video compression algorithms and highlight the motivation and challenges behind performing editing operations in the compressed bit-stream domain. We then present the techniques for applying video editing in the compressed domain. Finally, we present our experimental results, comparing the suggested techniques with the conventional approach.

2. BACKGROUND

Video editing for compressed bitstreams represent several challenges due to the dependencies in the compressed data. State-of-the art video compression standards such as MPEG-2, H.263[2], MPEG-4 v2[1], H.264, provide coding tools to exploit spatial and temporal redundancies present in the video sequences. For example, coding modes generally referred as intra-frame, exploit the spatial correlation of the pixels within the frame while coding modes referred to as inter-frame, make use of prediction from temporally other frames exploiting the temporal redundancy. Since in a typical video sequence the adjacent frames are highly correlated, higher compression efficiencies are achieved when using inter-frame coding instead of intra-frame coding. On the other hand, inter-frame coding modes introduce dependencies within the compressed bitstream. Therefore, the introduction of a visual effect on one video frame may not be confined to that particular frame but may further propagate in time due to inter-frame coding.

There are several algorithms in the literature for efficient video editing and transcoding. Chang et al. in [4] presented DCT domain algorithms for manipulating and compositing motion compensated DCT based compressed video, such as overlapping, translating, scaling, linear filtering, rotation and pixel multiplication. Smith and Rowe in [3] presented basic operations in transform domain for manipulating JPEG compressed images to implement dissolving of a sequence of images and introduce subtitles. Fernando et al [5] suggested techniques for fading in, fading out, and dissolving MPEG-2 video clips based on compressed domain modifications of estimated DCT coefficients. In [6], Meng and Chang presented a compressed video editing and parsing system, CVEPS, in which a set of post-production stage editing effects such as cutting, pasting, blending, film and temporal effects are supported.

In this paper we introduce further optimized compressed domain operations to achieve several video editing effects specifically for motion-compensated family of video codecs.

3. VIDEO EDITING IN COMPRESSED DOMAIN

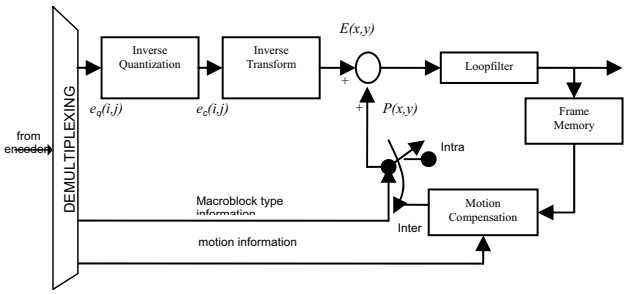

Figure 1 A Generic video decoding process.

3.1. ***Generic Video Decoding process***

This section presents a generic video decoding process, as illustrated in Figure 1, which will then be used in later sections to demonstrate the compressed domain techniques.

The decoder, as shown in Figure 1, receives the multiplexed compressed video bit-stream, decodes the variable-length coded quantized prediction error transform coefficients, the motion vectors and the macroblock type information. It then performs inverse quantization Q^{-1}, to obtain the inverse quantized transform coefficients e_q:

$$e_c(i,j,t) = Q^{-1}(e_q(i,j,t), QP) \quad (1)$$

where QP is the quantization parameter controlling the compression level. These coefficients are inverse-transformed to obtain the prediction error E_c

$$E_c(x,y,t) = T^{-1}\{e_c(i,j,t)\} \quad (2)$$

For the intra-type macroblocks, the obtained results are the pixels of the reconstructed frame. In inter-type macroblocks, the reconstructed pixels are computed by compensating the motion in the reference frame $R(x,y,t-1)$ using the motion vectors $(\Delta x, \Delta y)$. The predicted frame, $P(x,y,t)=R(x+\Delta x, y+\Delta y, t-1)$, is then corrected by adding the decoded prediction error $E_c(x,y,t)$ to obtain:

$$R(x,y,t) = R(x+\Delta x, y+\Delta y, t-1) + E(x,y) \quad (3)$$

The reconstructed values $V_c(x,y,t)$ can further be filtered to obtain decoded video frames, which are then stored in the frame buffer.

In the following sections, we develop compressed domain techniques to achieve special video effects, namely video blending and logo insertion.

3.2. ***Video Effects: Blending and Logo Insertion***

Video blending is a transitional effect between two different sequences or scenes of the same sequence by over-layering a set of frames from both, as illustrated in Figure 1. During transition part of two clips V_1 and V_2, the edited clip $V_e(x,y,t)$ can be represented by:

$$V_e(x,y,t) = \alpha_1(t)V_1(x,y,t) + \alpha_2(t)V_2(x,y,t), \quad (4)$$

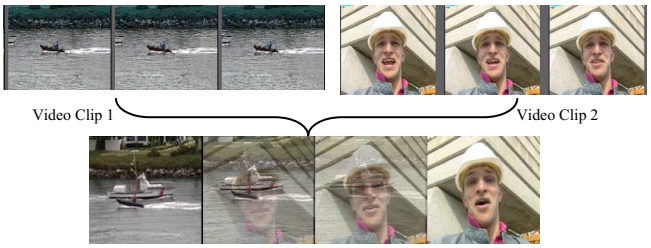

Figure 2 Blending of two video clips.

where $\alpha_1(t)$ and $\alpha_2(t)$ characterize the strength of blending for each video clip under the constraint $\alpha_1(t) + \alpha_2(t) = 1$.

In compressed domain video blending, there are four different scenarios to consider depending on the coding mode of the spatially co-located macroblocks from the two sequences: a) both macroblocks from the first video clip and the second one are intra-coded, b) macroblock from the first clip is intra-coded while the one from the second clip is inter-coded, c) macroblock from the first clip is inter-coded while the one from the second clip is inter-coded, d) both macroblocks are inter-coded. In the following each of these scenarios will be described separately.

Let us consider the case when the two macroblocks from both sequences are intra-coded, representing the simplest of the four cases defined above.

Let e_{qi} $i=1,2,$ denote the quantized transform coefficients of video clips V_1 and V_2, respectively, received in the compressed bitstreams. Let e_{ci} $i=1,2$ denote the inverse-quantized transform coefficients corresponding to QP_i and e_{qi}.

It can be easily shown [3] that the transform coefficients of the edited sequence, denoted by e_{c_e}, are given by:

$$e_{c\_e}(i,j,t) = \alpha_1(t)e_{c1}(i,j,t) + \alpha_2(t)e_{c2}(i,j,t). \quad (5)$$

These coefficients would then be quantized and entropy-coded to form the edited bitstream. If QP_1 and QP_2 are identical, then the quantized coefficients of the edited sequence can be directly calculated by:

$$e_{q\_e}(i,j,t) = \alpha_1(t)e_{q1}(i,j,t) + \alpha_2(t)e_{q2}(i,j,t), \quad (6)$$

Now let us consider the case, when both macroblocks are inter-coded. The reconstruction of the macroblock for V_i is performed as given in equation (3) by:

$$V_i(x,y,t) = R_i(x+\Delta x_i, y+\Delta y_i, t-1) + E_i(x,y), \quad i=1,2. \quad (7)$$

Let us start applying the editing effects at time $t=t_0$ using the reconstructed frames R_1 and R_2 of V_1 and V_2, respectively. In order to avoid performing motion estimation for the edited clip, the motion vectors of the first video V_1, $(\Delta x_1, \Delta y_1)$, will be re-used to obtain the edited video sequence V_e represented as:

$$V_e(x,y,t_0) = R_e(x+\Delta x_1, y+\Delta y_1, t_0-1) + E_e(x,y,t_0)$$

Replacing the representation of the reconstructed macroblocks in equations (4) and (7), we obtain

$$V_e(x,y,t_0) = \alpha_1(t_0)(R_1(x+\Delta x_1, y+\Delta y_1, t_0-1) + E_1(x,y,t_0)) + \alpha_2(t_0)(R_2(x+\Delta x_2, y+\Delta y_2, t_0-1) + E_2(x,y,t_0)) \quad (8)$$

Now add and subtract the term $R_1(x+\Delta x_1, y+\Delta y_1, t_0-1) + R_2(x+\Delta x_2, y+\Delta y_2, t_0-1)$ from equation (8), we get:

$$\begin{aligned} V_e(x,y,t_0) = {} & R_1(x+\Delta x_1, y+\Delta y_1, t_0-1) + \\ & R_2(x+\Delta x_1, y+\Delta y_1, t_0-1) + \\ & \alpha_1(t_0)(R_1(x+\Delta x_1, y+\Delta y_1, t_0-1) + E_1(x,y,t_0)) + \\ & \alpha_2(t_0)(R_2(x+\Delta x_2, y+\Delta y_2, t_0-1) + E_2(x,y,t_0)) - \\ & (R_1(x+\Delta x_1, y+\Delta y_1, t_0-1) + R_2(x+\Delta x_1, y+\Delta y_1, t_0-1)) \end{aligned} \quad (9)$$

Our aim is to define parts of equation (9) in terms of R_e and E_e so that the edited effects can be achieved by simple manipulation of the residual transform coefficient. Define

$$R_e(x+\Delta x_1, y+\Delta y_1, t_0-1) = R_1(x+\Delta x_1, y+\Delta y_1, t_0-1) + R_2(x+\Delta x_1, y+\Delta y_1, t_0-1)$$

and

$$\begin{aligned} E_e(x,y,t_0) = {} & \alpha_1(t_0)(R_1(x+\Delta x_1, y+\Delta y_1, t_0-1) + E_1(x,y,t_0)) + \\ & \alpha_2(t_0)(R_2(x+\Delta x_2, y+\Delta y_2, t_0-1) + E_2(x,y,t_0)) - \\ & (R_1(x+\Delta x_1, y+\Delta y_1, t_0-1) + R_2(x+\Delta x_1, y+\Delta y_1, t_0-1)) \end{aligned} \quad (10)$$

The approach can be generalized for video editing at in time $t>t_0$ to obtain E_e:

$$\begin{aligned} E_e(x,y,t) = {} & \alpha_1(t)E_1(x,y,t) + \alpha_2(t)E_2(x,y,t) - \\ & (\alpha_1(t-1) - \alpha_1(t))R_1(x+\Delta x_1, y+\Delta y_1, t-1) - \\ & \alpha_2(t-1)R_2(x+\Delta x_1, y+\Delta y_1, t-1) + \alpha_2(t)R_2(x+\Delta x_2, y+\Delta y_2, t-1) \end{aligned} \quad (11)$$

Then the edited transform domain coefficients are given by:

$$\begin{aligned} e_{c\_e}(i,j,t) = {} & \alpha_1(t)e_{c1}(i,j,t) + \alpha_2(t)e_{c2}(i,j,t) \\ & -(\alpha_1(t-1) - \alpha_1(t))r_1(i+\Delta x_1, j+\Delta y_1, t-1) \\ & -\alpha_2(t-1)r_2(i+\Delta x_1, j+\Delta y_1, t-1) \\ & +\alpha_2(t)r_2(i, \Delta x_2, j+\Delta y_2, t-1) \end{aligned} \quad (12)$$

where $r_1(x,y)$ and $r_2(x,y)$ denote the transform coefficients of the motion compensated prediction blocks corresponding to the two input video with motion vectors $(\Delta x_1, \Delta y_1)$, $(\Delta x_2, \Delta y_2)$, respectively.

Figure 3 illustrates the block diagram to achieve the blending effect in the transform domain for two inter-coded macro blocks. The upper and the lower part of the Figure 3 represent the two decoders which operate independently on the video clips to generate the reference frames. Since we utilize the motion vector of the first video for the new sequence, note that these vectors are fed to the motion compensation of the decoder of the second video to generate the new residual data. The middle section the figure represents operations to calculate the new residual transform coefficients, Equation (12). The obtained overall data is then quantized and the coefficients are coded and sent to a multiplexer to generate the edited video.

The third case represents the scenario when the macroblock of the first video clip is inter-coded and that of the second video is intra-coded. Equation (4) can be re-written by:

$$V_e(x,y,t) = \alpha_1(t)(R_1(x+\Delta x_1, y+\Delta y_1, t-1) + E_1(x,y)) + \alpha_2(t)E_2(x,y) \quad (13)$$

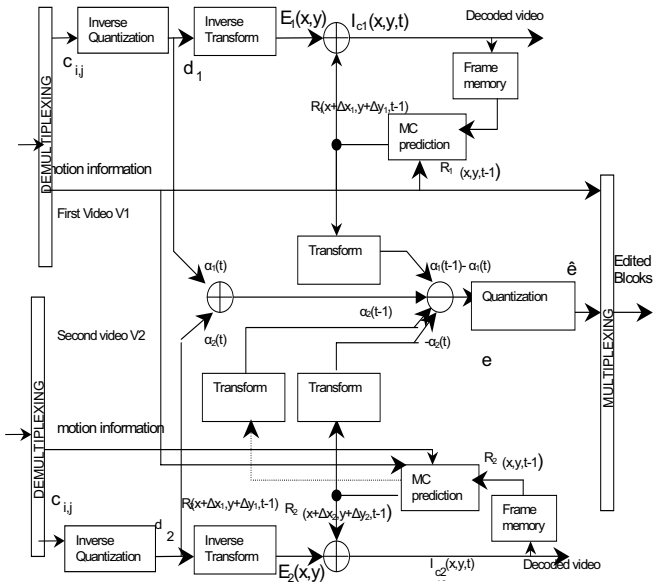

Figure 3 Video blending operations between two inter-coded macroblocks.

By applying the same steps as earlier, the transform coefficients of the edited sequence are calculated by:

$$\begin{aligned} e_{c\_e}(i,j,t) = {} & \alpha_1(t)e_{c1}(i,j,t) + \alpha_2(t)e_{c2}(i,j,t) \\ & -(\alpha_1(t-1) - \alpha_1(t))r_1(i+\Delta x_1, j+\Delta y_1, t-1) \\ & -\alpha_2(t-1)r_2(i+\Delta x_1, j+\Delta y_1, t-1) \end{aligned} \quad (14)$$

Finally, the last case represents the scenario when the macroblock of the first video clip is intra-coded and that of the second video is inter-coded. This case can be treated in two different ways. In the first approach, as before, the macroblock type of the first video clip is retained at the output. In this case, the edited video transform coefficients can be obtained from:

$$\begin{aligned} e_{c\_e}(i,j,t) = {} & \alpha_1(t)e_{c1}(i,j,t) + \alpha_2(t)e_{c2}(i,j,t) \\ & -\alpha_1(t-1)r_1(i+\Delta x_1, j+\Delta y_1, t-1) \\ & -\alpha_2(t-1)r_2(i+\Delta x_1, j+\Delta y_1, t-1) \\ & +\alpha_2(t)r_2(i, \Delta x_2, j+\Delta y_2, t-1) \end{aligned} \quad (15)$$

In the second approach, the same steps as in the 3rd case may be applied with the exception that the motion vectors of the second video clip is re-used.

As can be noted from Equations (5), (12), (14) and (15), the edited bitstream with video blending effects is achieved by manipulation of the residual transform coefficients and the rest of the compressed data, such as motion-vectors, etc would remain the same

The general video editing methods as described above can be re-applied in several other video editing operations, e.g., logo-insertion, fade-in, fade-out. Details can be found in [7].

Logo insertion is a special case of the 3rd case as described above in video blending operation, i.e., blending two video clips with inter-macroblock and an intra-macroblock. The main difference is that the logo will be

inserted in certain area of the image, i.e., the equation (14) would be applied to those macroblocks where the logo is inserted, and the macroblocks which are affected by the insertion of the logo through motion prediction.

Widely used fade-in and fade-out effects can also realized using the same approach as described above [7]. The main difference in this case is that we have a single video clip under consideration. Figure 4 illustrates fade-in operation in the compressed domain for inter-coded macroblocks. Note the similarities between Figures 3 and 4.

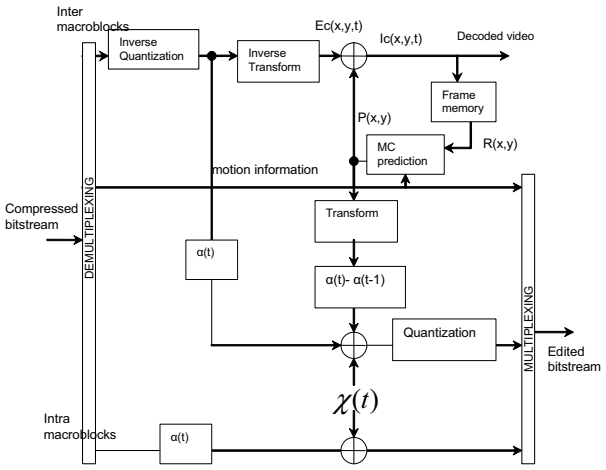

Figure 4 Video fade-in effect in compressed domain.

7. RESULTS

We developed a video-editing engine integrated with H.263 and MPEG-4 codecs. The platform, including the same codecs, was also used to perform the editing operations using the conventional spatial domain approach. We used test sequences with different resolutions (subQCIF, QCIF and CIF). To have an implementation independent and content independent evaluation of the techniques, we report the normalized execution time. Our encoder implementation takes, on average, 4 times more processing time than the decoder, so we assume that decoding 1 frame takes 1 unit of time and encoding it requires 4 units.

To apply the fading effect using the proposed algorithm, our method is based on only modifying the residual of the original bitstream. The motion vectors and the block types are kept unchanged as the motion estimation, which is the most costly stage in video coding, is omitted. The operations required in our approach for each frame are a VLD, an inverse quantization, an inverse DCT, motion compensation, a DCT, a quantization and a VLC. These operations sum up to 1.7 units/frame, which is around 3 times less than the 5 units/frame needed in the conventional approach.

Similarly, for blending two sequences in a transitional effect, the proposed method utilizes first clip's motion information and updates its residual data with information from the second video. The costly motion estimation operation is not performed. In the suggested technique, for the worst case scenario (when all macroblocks in the blended frames of the sequences are coded inter), we need to perform 2 inverse quantization, 2 IDCT, 3 DCT, 2 VLD, 1 VLC, 3 motion compensation and 1 quantization operation per frame. These operations sum to around 3.4 units/frame of processing time, which is around 45% speed up compared to the spatial domain approach, which requires 2 decoding and 1 encoding per frame.

TABLE I
COMPRESSED DOMAIN VS CONVENTIONAL APPROACH NORMALIZED EXECUTION TIME

		Time per block	Fading (Number of Calls)		Blending (Number of Calls)	
			Conv.	Prop.	Conv.	Prop.
Encoder	Frame Reading	0.2	1	0	1	0
	ME + MC	2.25	1	0	1	0
	MC	0.15	1	0	1	0
	DCT	0.32	1	1	1	3
	IDCT	0.32	1	0	1	0
	Quant	0.14	1	1	1	1
	InvQuant	0.14	1	0	1	0
	VLC	0.28	1	1	1	1
Decoder	Frame Reading	0.1	1	1	2	1
	VLD	0.2	1	1	2	2
	InvQuant	0.14	1	1	2	2
	IDCT	0.32	1	0	2	2
	MC	0.15	1	1	2	3
Overall Time	(Number of calls)*(time per block)	4.7	4.7	1.7	5.62	3.25

Conv. = Conventional Approach. ME= Motion Estimation. MC= Motion Compensation.
DCT= Discrete Cosine Transform. IDCT= Inverse DCT.
Prop. = Proposed Approach. Quant= Quantization. InvQuant= Inverse Quantization.
VLC= Variable length Coding. VLD= Variable Length Decoding.

8. CONCLUSION

We have presented novel compressed domain operations to achieve several video editing effects specifically for motion-compensated family of video codecs. The algorithms could be easily extended for further video editing operations. Results indicate that significant computational savings can be accomplished against the brute force methods.

11. REFERENCES

[1] ISO/IEC JTC 1/SC 29/WG 11 N4350, "Information Techn. – Coding Of Audio-Visual Objects–Part 2: Visual", July 2001.

[2] ITU-T Recommendation H.263, "Video Coding For Low Bit Rate Communication", February 1998.

[3] Brian C. Smith and Laurence A. Rowe, "Algorithms For Manipulating Compressed Images", Computer Graphics and Applications Vol 13, No 5, pp 34-42, September 1993.

[4] S.F. Chang and D.G. Messerschmitt. "Manipulation And Compositing Of MC-DCT Compressed Video", IEEE Journal on Selected Areas in Communications: Special Issue on Intelligent Signal Processing, 13:1-11, 1995.

[5] W.A.C. Fernando, C.N. Canagarajah, D. R. Bull, "Fade, Dissolve And Wipe Production In MPEG-2 Compressed Video", IEEE Transactions On Consumer Electronics, Vol. 46, No. 3, Pp. 717-727, 2000.

[6] J. Meng and S.-F. Chang, "CVEPS: A Compressed Video Editing and Parsing System," ACM Multimedia Conference, Boston, MA, Nov. 1996.

[7] F. Chebil, R. Kurceren, A. Islam, U. Budhia, "Compressed domain editing of H.263 and MPEG-4 videos", IEEE Transactions on Consumer Electronics, Vol. 51, No. 3, pp. 947-957.