IMPROVED MACROBLOCK-BASED REVERSE PLAY ALGORITHM FOR MPEG VIDEO STREAMING

Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu

Centre for Multimedia Signal Processing
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

ABSTRACT

Reverse playback is the most common video cassette recording (VCR) functions in many digital video players. However, the predictive processing techniques employed in MPEG severely complicate the reverse-play operation. In this paper, we enhance our previously proposed compressed-domain approach for the efficient reverse-play operation of an MPEG video streaming system. In the proposed video streaming server, a novel macroblock-based algorithm is used to adaptively select the necessary macroblocks, manipulate the macroblocks and send the processed macroblocks to the client machine. The proposed approach only manipulates macroblocks either on the VLC-domain or DCT-domain to achieve the server with low complexity. Experimental results show that, as compared to the conventional system and the previously proposed compressed-domain system, the new streaming system reduces the required network bandwidth and the decoding complexity significantly.

1. INTRODUCTION

With the rapid growth of online multimedia content, it is highly desirable that video streaming system should have the capability of providing fast and effective browsing. A key technique that enables fast and user-friendly browsing of video content is to provide full VCR functionality. The set of effective VCR functionality includes forward, backward, stop, pause, fast forward, fast backward, and random access.

Since the MPEG [1-3] video coding standards were mainly designed for forward-play operations, the predictive processing techniques employed in MPEG [4] severely complicate the reverse-play operation. One straightforward approach to implement a reverse playback of MPEG compressed video is to decode the GOP up to the current frame to be displayed, and then go back to decode the GOP again up to the next frame to be displayed. For example, consider the case with simple I-P structure of an MPEG encoded sequence. Suppose frame *n* is the starting point of the reverse-play operation. Since the next frame to be displayed is frame *n-1*, the server needs to send all P-frames from the previous and nearest I-frame to this requested frame *n-1*. At the client side, the previous and nearest I-frame to frame *n-2* do not need to be displayed and only frame *n-1* should be displayed on the user screen. Afterwards, frame *n-1* is decoded and stored into the display buffer so that this frame is displayed on the client screen. This approach requires much higher bandwidth of the network and complexity of the decoding, which is not desirable. Since the B-frames are not used as references for later frames, and are not needed to be sent over the network or decoded by the decoder, for the sake of simplicity, we focus our discussion on the cases that the video stream contains I- and P-frames only. Some works on the implementation of reverse playback for MPEG compressed video for streaming applications have recently been introduced [5-6]. These approaches require either extra decoding complexity or higher storage cost to store the local bitstream in the client.

In our previous work [7], we have developed a compressed-domain approach for an efficient implementation of the MPEG streaming video system to provide the reverse-play operation over a network. Assuming that frame *n* is the starting point of the reverse-play operation, macroblocks in the target frame (frame *n-1*) are classified into two different categories: backward macroblock (BMB) and forward macroblock (FMB). As shown in Figure 1, we assume that $MB^{n-1}_{(k,l)}$ represents the macroblock at the k^{th} row and l^{th} column of frame *n-1*. $MB^{n-1}_{(k,l)}$ is defined as a backward macroblock (BMB) if the corresponding macroblock of frame *n*, $MB^{n}_{(k,l)}$, is coded without motion compensation (non-MC macroblock). Otherwise, it is defined as a forward macroblock (FMB).

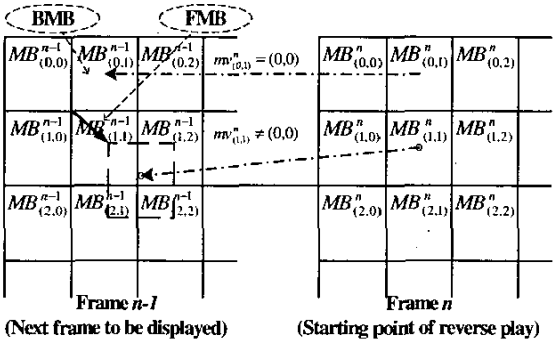


Figure 1. Definition of the forward macroblock (FMB) and the backward macroblock (BMB).

When a user issues a reverse-play command at frame *n*, the next frame to be display is frame *n-1*, i.e., all $MB^{n-1}_{(k,l)}$ in frame *n-1* are requested. To reconstruct each $MB^{n-1}_{(k,l)}$, all the related previous macroblocks in P-/I-frames need to be sent to the network and decoded by the decoder in the conventional streaming system. However, if $MB^{n-1}_{(k,l)}$ is classified as BMB, its corresponding macroblock in frame *n*, $MB^{n}_{(k,l)}$, is coded without motion

0-7803-8554-3/04/$20.00 ©2004 IEEE.

compensation. It means that the spatial position of $MB^{n-1}_{(k,l)}$ is the same as that of $MB^{n}_{(k,l)}$, and it can be written as

$$MB^{n-1}_{(k,l)} = MB^{n}_{(k,l)} - e^{n}_{(k,l)} \quad (1)$$

where $e^{n}_{(k,l)}$ is the prediction error between $MB^{n}_{(k,l)}$ and its motion-compensated macroblock, $MB^{n-1}_{(k,l)}$. Note that frame n is stored in FB at the client machine when a user issues the reverse-play operation at frame n. In other words, pixels of $MB^{n}_{(k,l)}$ are available at the decoder. To reconstruct $MB^{n-1}_{(k,l)}$ in the reverse-play operation, equation (1) indicates that, for BMB, the only data that the server needs to send are sign inversion of the quantized DCT coefficients of $e^{n}_{(k,l)}$.

The situation of FMB is different. The sign inversion of DCT coefficients cannot be employed since equation (1) is not valid for FMB. In other words, $MB^{n-1}_{(k,l)}$ cannot be reconstructed from $MB^{n}_{(k,l)}$. In order to reconstruct FMBs, all the related macroblocks from the previous nearest I-frame to frame n-2 need to be sent.

In this paper, we explore two techniques to improve the performance of the compressed domain approach. We perform sign inversion in VLC-domain for BMBs to reduce the encoder complexity and use direct addition in DCT-domain for FMBs to further reduce the decoding complexity and the required network bandwidth. The organization of this paper is as follows. Section 2 describes the proposed techniques. Simulation results are presented in Section 3. Finally, some concluding remarks are given in Section 4.

2. THE PROPOSED METHOD

2.1 VLC-domain technique for BMBs

The sign inversion of DCT coefficients for BMB requires additional variable length decoding and re-encoding. To reduce the computational load of the server, we propose to compute the sign inversion of DCT coefficients in VLC-domain.

In MPEG video encoding, each nonzero DCT coefficient is represented by the RUN-LEVEL symbol structure which has its corresponding variable length code (VLC). RUN refers to the number of zero coefficients before the next nonzero coefficient, LEVEL refers to the amplitude of the nonzero coefficient. The trailing bit of each VLC is the '*s*' bit that means the sign of the nonzero coefficient. If '*s*' is 0, the coefficient is positive; otherwise it is negative. For most combinations, to perform sign inversion, the server just parses the MPEG video bitstream and inverts all '*s*' bits of VLCs in BMB. However, some RUN-LEVEL combinations that are not frequently used are coded using a 6-bit "Escape" code followed by a 6-bit fixed length code (FLC) for RUN and a 12-bit FLC for LEVEL. In this case, the 12-bit FLC for LEVEL is converted into its 2's complement. The bit manipulation of VLCs in BMB is summarized in Figure 2. Since it is not necessary to perform VLC encoding, motion compensation, DCT, quantization, inverse DCT, inverse quantization and VLC decoding in the server, the loading of the server is reduced significantly.

2.2 DCT-domain technique for FMBs

To further reduce the decoding complexity and network traffics in processing FMBs, we propose to use a technique of direct addition of DCT coefficients. In Figure 3, a situation in which $MB^{n-1}_{(k,l)}$ is a non-MC FMB is illustrated. $MB^{n-1}_{(k,l)}$ is a non-MC macroblock, since its motion vector, $mv^{n-1}_{(k,l)}$, is equal to zero. And if $mv^{n}_{(k,l)}$ is also equal to zero, $MB^{n-1}_{(k,l)}$ is a BMB instead of FMB. So, in general, a macroblock in frame n-1, $MB^{n-1}_{(k,l)}$, is treated as non-MC FMB when $mv^{n}_{(k,l)} \neq (0,0)$ and $mv^{n-1}_{(k,l)} = (0,0)$.

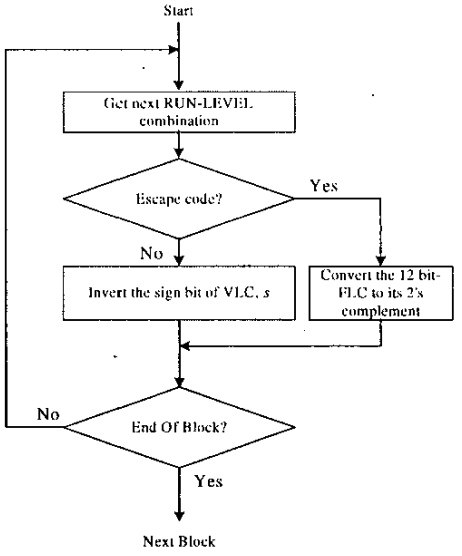


Figure 2. Execution flow of the server during bit manipulation of VLCs in BMB.

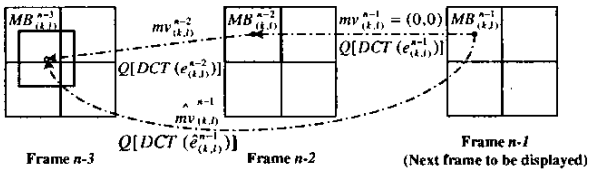


Figure 3. A situation in which there are two FMBs in frame n-1 is illustrated.

To reconstruct this non-MC FMB, the decoder needs to decode the prediction errors $e^{n-1}_{(k,l)}$ in frame n-1 and $e^{n-2}_{(k,l)}$ in frame n-2. If pixels in $MB^{n-2}_{(k,l)}$ are used as reference for $MB^{n-1}_{(k,l)}$ only, it is too wasteful that the client machine to decode $e^{n-2}_{(k,l)}$ for reconstructing $MB^{n-2}_{(k,l)}$. Thus, in the proposed scheme, the server employs the DCT-domain technique to combine $e^{n-1}_{(k,l)}$ and $e^{n-2}_{(k,l)}$ in one single macroblock for the non-MC FMB. The required number of macroblocks, which are decoded by the decoder, is then reduced.

Now, let us formulate how to efficiently combine $e^{n-1}_{(k,l)}$ and $e^{n-2}_{(k,l)}$ in the server for non-MC FMB. When pixels in $MB^{n-2}_{(k,l)}$ are

not decoded directly in the client machine, it means that the incoming quantized DCT coefficients of the prediction error from the original video stream, $Q[DCT(e_{(k,l)}^{n-1})]$, are no longer valid because they refer to the pixels which are not stored in the FB. The server needs to compute the new motion vector $\hat{mv}_{(k,l)}^{n-1}$ and prediction errors in the DCT-domain, $Q[DCT(\hat{e}_{(k,l)}^{n-1})]$, by using frame *n-3* as a reference, as shown in Figure 3. Since $mv_{(k,l)}^{n-1}$ is zero, we have

$$\hat{mv}_{(k,l)}^{n-1} = mv_{(k,l)}^{n-2} \tag{2}$$

One straightforward approach for computing $Q[DCT(\hat{e}_{(k,l)}^{n-1})]$ is to decode $MB_{(k,l)}^{n-1}$ in the pixel domain, and the decoded macroblock is re-encoded by using frame *n-3* as a reference and can be written as

$$\begin{aligned} Q[DCT(\hat{e}_{(k,l)}^{n-1})] &= Q[DCT(MB_{(k,l)}^{n-1} - MCMB^{n-3}(\hat{mv}_{(k,l)}^{n-1}))] \\ &= Q[DCT(MB_{(k,l)}^{n-1} - MCMB^{n-3}(mv_{(k,l)}^{n-2}))] \end{aligned} \tag{3}$$

where MCMB stands for the motion-compensated macroblock. The decoding and re-encoding processes can create undesirable complexity on the server and also the video quality of the pixel-domain approach suffers from its intrinsic double-encoding process, which introduces additional degradation. In the proposed video server, we employ a DCT-domain technique to compute the new $Q[DCT(\hat{e}_{(k,l)}^{n-1})]$. In Figure 3, $MB_{(k,l)}^{n-1}$ can be written as

$$MB_{(k,l)}^{n-1} = MCMB^{n-2}(mv_{(k,l)}^{n-1}) + e_{(k,l)}^{n-1} \tag{4}$$

If $MB_{(k,l)}^{n-1}$ is coded without motion compensation, $mv_{(k,l)}^{n-1}$ is zero and $MCMB^{n-2}(mv_{(k,l)}^{n-1})$ is equal to $MB_{(k,l)}^{n-2}$. Equation (4) can be simplified to (5)

$$MB_{(k,l)}^{n-1} = MB_{(k,l)}^{n-2} + e_{(k,l)}^{n-1} \tag{5}$$

Similarly,

$$MB_{(k,l)}^{n-2} = MCMB^{n-3}(mv_{(k,l)}^{n-2}) + e_{(k,l)}^{n-2} \tag{6}$$

Substituting (5) into (6), we obtain

$$MB_{(k,l)}^{n-1} - MCMB^{n-3}(mv_{(k,l)}^{n-2}) = e_{(k,l)}^{n-1} + e_{(k,l)}^{n-2} \tag{7}$$

Using (3) and (7), the newly quantized DCT coefficients of the prediction error between the current non-MC FMB and its corresponding reference macroblock in frame *n-3*, $Q[DCT(\hat{e}_{(k,l)}^{n-1})]$, can be written as

$$Q[DCT(\hat{e}_{(k,l)}^{n-1})] = Q[DCT(e_{(k,l)}^{n-1} + e_{(k,l)}^{n-2})] \tag{8}$$

Taking into account the linearity of DCT, (8) becomes

$$Q[DCT(\hat{e}_{(k,l)}^{n-1})] = Q[DCT(e_{(k,l)}^{n-1}) + DCT(e_{(k,l)}^{n-2})] \tag{9}$$

Note that, in general, quantization is not a linear operation because of the integer truncation. However, $DCT(e_{(k,l)}^{n-1})$ and $DCT(e_{(k,l)}^{n-2})$ are divisible by the quantizer step-size. Thus, we obtain the final expression of $Q[DCT(\hat{e}_{(k,l)}^{n-1})]$ by using frame *n-3* as a reference,

$$Q[DCT(\hat{e}_{(k,l)}^{n-1})] = Q[DCT(e_{(k,l)}^{n-1})] + Q[DCT(e_{(k,l)}^{n-2})] \tag{10}$$

Equation (10) implies that the newly quantized DCT coefficient $Q[DCT(\hat{e}_{(k,l)}^{n-1})]$ can be computed in the DCT-domain by adding $Q[DCT(e_{(k,l)}^{n-1})]$ and $Q[DCT(e_{(k,l)}^{n-2})]$. Both of them can be directly extracted from the MPEG video stream in the server. Since it is not necessary to perform decoding and re-encoding, the computational complexity required for the server is limited. To illustrate this scheme, we refer back to Figure 3 again. Instead of transmitting and decoding $Q[DCT(e_{(k,l)}^{n-1})]$ and $Q[DCT(e_{(k,l)}^{n-2})]$ for $MB_{(k,l)}^{n-1}$, by using the direct addition of DCT coefficients, the server only needs to send $Q[DCT(\hat{e}_{(k,l)}^{n-1})]$ and only one macroblock requires to be decoded in the client machine. The DCT-domain technique can thus minimize the computational complexity of the decoding process. Furthermore, the direct addition of DCT coefficients can be computed iteratively if $mv_{(k,l)}^{n-2}$ is also equal to zero and pixels in $MB_{(k,l)}^{n-3}$ are used as reference for $MB_{(k,l)}^{n-1}$ only.

3. SIMULATION RESULTS

A series of computer simulations were conducted to evaluate the performances of the proposed sign inversion (*SI*) and direct addition (*DA*) techniques when applied to the video streaming system with VCR support. MPEG-2 encoder was employed to encode various video sequences with different spatial resolutions. All the test sequences have a length of 200 frames. "Claire", "Grandma" and "Carphone" are typical videophone sequences in QCIF (*176x144*) format, which were encoded at 64 Kb/s. "Salesman"(CIF *352x288*), "Table Tennis" and "Football" (SIF *352x240*) were encoded at *1.5* Mb/s. We have simulated the situation of the I-P structure for L=15. The starting point of the reverse-play operation is at the end of the sequence. For all testing sequences, the frame-rate of the video stream was 30 fps.

Two different versions of our proposed streaming systems have been realized, and let us call them *SI* and *SI+DA*. *SI* uses the technique of sign inversion of DCT coefficients for BMB while *SI+DA* employs the technique of direction addition of DCT coefficients for FMB as well. The savings in both the average number of macroblocks to be decoded and bits to be sent are tabulated in Table 1. The average number of macroblocks to be sent for decoding is directly proportional to the decoding complexity. Note that *SI* is only to reduce the server complexity, it will not affect the decoding complexity and bandwidth requirement. That is, *SI* has the same performance in terms of the decoding complexity and bandwidth requirement as our system proposed in [7].

Table 1 shows that *SI* has outperformed the conventional system in all sequences. The results are more significant for the sequences "Claire", "Grandma", and "Salesman". The savings in both the bits to be sent over the network and macroblocks to be decoded by the decoder are from 40-75% for those sequences. It is due to the reason that those sequences contain more BMBs in which the technique of sign inversion can be employed. For sequences containing high motion activities such as "Table Tennis", "Football" and "Carphone", there still has 20-40% saving. *SI+DA* produces further 7%-17% savings in terms of both number of macroblocks to be decoded by using the technique of direction addition of DCT coefficients.

Figure 4 shows the frame-by-frame comparisons of the number of macroblocks decoded by the decoder and the number

of bits transmitted over the network of the proposed system and the conventional system for the "Salesman" sequence in the reverse-play operation. It is obvious that *SI* can achieve significant performance improvement in terms of the decoding complexity and the network traffic load and *SI+DA* can provide further improvement. Note that, as shown in Figure 4, the required number of macroblocks and bits for the last frame of each GOP of *SI* and the conventional system are the same. The reason behind is that there is no inter-frame dependency between the last frame of the current GOP and the first frame of the next GOP, which is an I-frame. In that case, no BMB exists in the last frame of the current GOP. As a result, the technique of sign inversion cannot be applied in the last frame of each GOP. However, the direct additional of DCT coefficients can achieve 50% savings in both the bits to be sent and macroblocks to be decoded by the decoder of that particular frame, as shown in Figure 4. This is another improvement of *SI+DA* over *SI*.

Table 1. Performance improvement of the proposed system over the conventional system.

Sequences	Bitrate	Saving of macroblocks to be decoded by the decoder		Saving of bits to be sent over the network	
		SI	*SI+DA*	*SI*	*SI+DA*
Salesman	1.5M	40.0%	50.7%	41.0%	50.0%
Football	1.5M	21.9%	28.0%	18.4%	22.9%
Table Tennis	1.5M	36.5%	47.1%	25.0%	31.5%
Carphone	64K	28.7%	45.8%	29.7%	32.7%
Claire	64K	71.3%	84.0%	76.5%	79.8%
Grandma	64K	59.6%	72.8%	72.4%	75.5%

4. CONCLUSION

In this paper, we have improved our previously proposed efficient reverse-play algorithm for an MPEG video streaming system. With the motion information, the video streaming server organizes the macroblocks in the requested frame into two categories: backward macroblocks (BMBs) and forward macroblocks (FMBs). Our previous algorithm has already provided the way to obtain BMBs with much less decoding complexity and network bandwidth than the conventional system. The improved system has the following main features: (1) a sign inversion technique is proposed for BMBs, which is operated in the VLC-domain, to achieve the server with low complexity; and (2) a technique of direction addition of DCT coefficients is used for those FMBs coded without motion compensation to reduce the number of macroblocks that need to be decoded by the decoder and the number of bits that need to be sent over the network in the reverse-play operation. Experimental results show that, as compared to the conventional system, the new streaming system reduces the required network bandwidth and the decoding complexity significantly.

5. ACKNOWLEDGMENT

The work described in this paper is partially supported by the Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, Hong Kong Polytechnic University and a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (PolyU 5216/03E). Chang-Hong Fu acknowledges the research studentships provided by the University.

6. REFERENCES

[1] L. Chiariglione, "The development of an integrated audiovisual coding standard: MPEG," Proceedings of IEEE, vol. 83, pp. 151-157, Feb. 1995.

[2] ISO/IEC 11172-2, "Information technology -- coding of moving pictures and associated audio for digital storage media at up to About 1,5 Mbit/s -- Part 2: Video," 1993

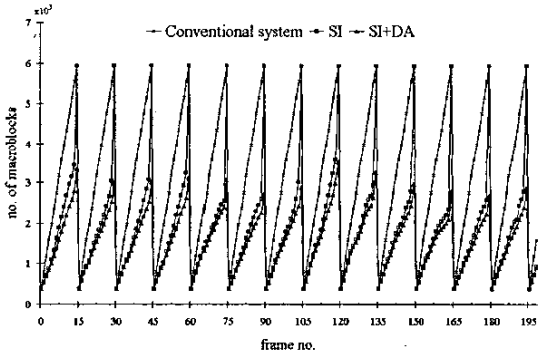
[3] ISO/IEC 13818-2, "Information technology -- generic coding of moving pictures and associated audio information: video," 1996.

[4] Y.L Chan and W.C. Siu, "An efficient search strategy for block motion estimation using image features", IEEE Trans. on Image Processing, Vol. 10, No. 8, pp. 1223-1238, Aug. 2001.

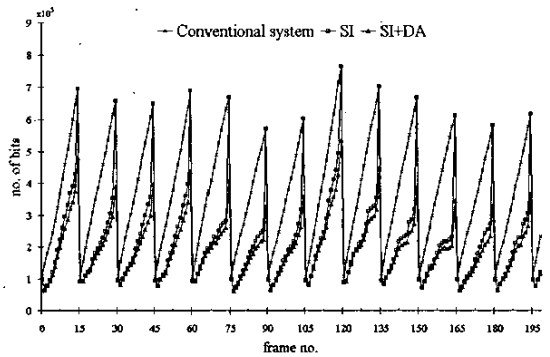
[5] M. S. Chen and D. D. Kandlur, "Downloading and stream conversion: supporting interactive playout of videos in a client station," in Proc. 2nd Int. IEEE Conf. Multimedia Computing and Systems, pp. 73-80, 1995.

[6] S. J. Wee, "Reversing motion vector fields," in Proc. IEEE International on Conference Image Processing 1998 (ICIP98), pp. 209-212, Oct. 1998.

[7] C.H. Fu, Y.L. Chan and W.C. Siu, "Efficient Reverse Play Algorithms for MPEG video streaming with VCR Functionality" in Proc. International Conf. on Information, Communications and Signal Processing, and the fourth IEEE Pacific-Rim Conf. on Multimedia, 1356-1359, Dec., 2003.



(a) Number of macroblocks to be decoded by the decoder.



(b) Number of bits transmitted over the network.

Figure 4. Performance of the conventional system SI and SI+DA for the "Salesman" sequence in the reverse-play operation.