EFFICIENT ALGORITHM FOR REVERSE PLAYBACK IN MPEG VIDEO STREAMING

Chang-Hong Fu, Yui-Lam Chan, Tak-Piu Ip 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. Previously, we have proposed a compressed-domain algorithm for the MPEG video streaming system to provide efficient reverse playback. In the proposed video streaming server, it classifies macroblocks in the requested frame into two categories – a backward macroblock (BMB) and a forward macroblock (FMB). For BMB, a sign inversion technique has been proposed to reduce the number of macroblocks that need to be decoded by the decoder. However, the required number of macroblocks for reconstructing FMBs is still very high. In this paper, the server identifies the related previous macroblocks of FMBs and reduces the redundancies of those related maroblocks to further reduce the computational complexity of the client decoder. Experimental results show that, as compared to the conventional and the previously proposed algorithms, the new streaming algorithm reduces the required network bandwidth and the decoding complexity significantly.

1. INTRODUCTION

With the development of high-speed networking and compression technologies, multimedia streaming applications have become very attractive in recent years. To accelerate the marketability of these applications, it is highly desirable that the 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. 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 macroblock-based approach for an efficient implementation of the MPEG streaming video system to provide the reverse-play operation over a network. In this paper, we explore a way to further improve the performance of the macroblock-based system. The organization of this paper is as follows. Section 2 briefly introduces our video streaming server in which 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 new enhanced technique is described in Section 3. Simulation results are presented in Section 4. Finally, some concluding remarks are given in Section 5.

2. The MACROBLOCK-BASED SYSTEM

Consider the case when a user issues a reverse-play command at frame n during the normal forward play. At that moment, frame n is displayed at the client side and the next frame to be display is frame $n-1$. In the macroblock-based system, 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).

When a user issues a reverse-play command at frame n, the next displayed frame is frame $n-1$, i.e., all $MB^{n-1}_{(k,l)}$ in frame $n-1$ are requested. To reconstruct $MB^{n-1}_{(k,l)}$, 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 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

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

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 the frame buffer 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 $e^{n}_{(k,l)}$.

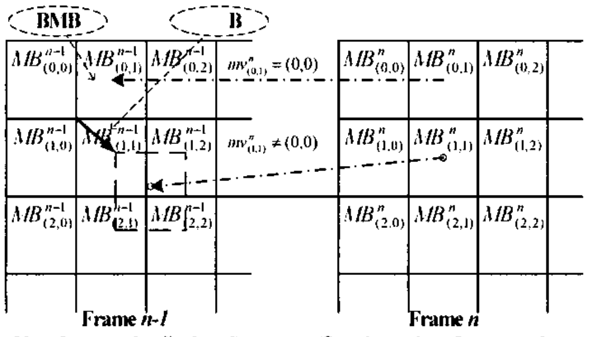

BMB
B
$MB^{n-1}_{(0,0)}$ $MB^{n-1}_{(0,1)}$ $MB^{n-1}_{(0,2)}$
$mv^{n}_{(0,1)}=(0,0)$
$MB^{n}_{(0,0)}$ $MB^{n}_{(0,1)}$ $MB^{n}_{(0,2)}$
$MB^{n-1}_{(1,0)}$ $MB^{n-1}_{(1,1)}$ $MB^{n-1}_{(1,2)}$
$mv^{n}_{(1,1)}\neq(0,0)$
$MB^{n}_{(1,0)}$ $MB^{n}_{(1,1)}$ $MB^{n}_{(1,2)}$
$MB^{n-1}_{(2,0)}$ $MB^{n-1}_{(2,1)}$ $MB^{n-1}_{(2,2)}$
$MB^{n}_{(2,0)}$ $MB^{n}_{(2,1)}$ $MB^{n}_{(2,2)}$
Frame *n-1*
Frame *n*
(Next frame to be displayed)
(Starting point of reverse play)

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

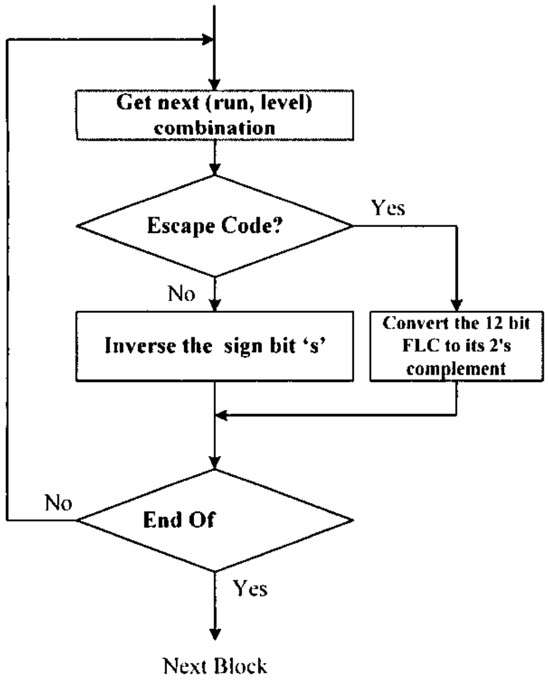

Get next (run, level) combination
Escape Code?
Yes
No
Inverse the sign bit 's'
Convert the 12 bit FLC to its 2's complement
End Of
No
Yes
Next Block

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

The sign inversion of DCT coefficients for BMB requires additional variable length decoding and re-encoding. To reduce the computational load of the server, the sign inversed DCT coefficients can be computed in VLC-domain (Variable Length Code). In MPEG video encoding, each nonzero DCT coefficient is represented by the RUN-LEVEL symbol structure which has its corresponding variable length code. RUN refers to the number of zero coefficients before the next nonzero coefficient and LEVEL refers to the amplitude of the nonzero coefficient. The trailing bit of each VLC is the '*s*' bit that indicates 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.

3. THE PROPOSED ALGORITHM

The situation of FMBs is different. The sign inversion of DCT coefficients cannot be employed since equation (1) is not valid for FMBs. 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 Figure 3, a situation in which $MB^{n-1}_{(1,1)}$ is a FMB and its corresponding motion vector is $mv^{n-1}_{(k,l)}$ is illustrated. Two macroblocks (shaded macroblocks in Figure 3) in frame *n-2* are required to act as references for performing motion compensation of $MB^{n-1}_{(1,1)}$. Those macrobocks in frame *n-2* further requires their corresponding macroblocks in frame *n-3*. This process continues until the previous nearest I-frame. In the example shown in Figure 3, the server needs to send 7 macroblocks for $MB^{n-1}_{(1,1)}$.

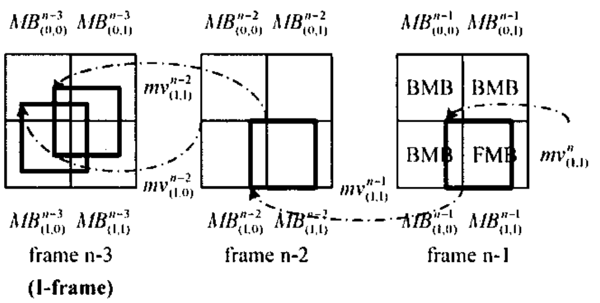

$MB^{n-3}_{(0,0)}$ $MB^{n-3}_{(0,1)}$
$MB^{n-2}_{(0,0)}$ $MB^{n-2}_{(0,1)}$
$MB^{n-1}_{(0,0)}$ $MB^{n-1}_{(0,1)}$
$mv^{n-2}_{(1,1)}$
$mv^{n-2}_{(1,0)}$
$mv^{n-1}_{(1,1)}$
$mv^{n}_{(1,1)}$
BMB
BMB
BMB
FMB
$MB^{n-3}_{(1,0)}$ $MB^{n-3}_{(1,1)}$
$MB^{n-2}_{(1,0)}$ $MB^{n-2}_{(1,1)}$
$MB^{n-1}_{(1,0)}$ $MB^{n-1}_{(1,1)}$
frame n-3
(I-frame)
frame n-2
frame n-1

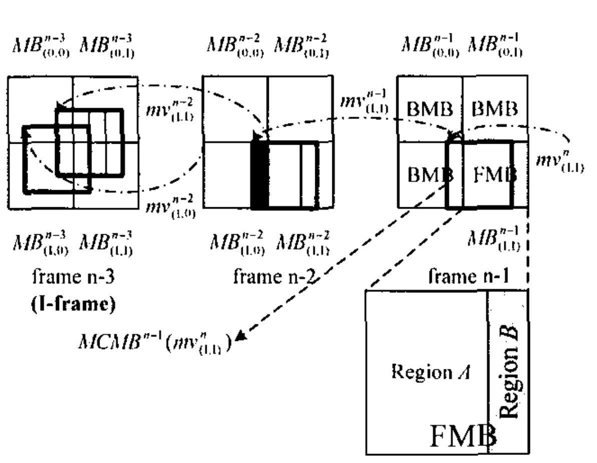
Figure 3. A situation in which there is one FMB in frame *n-1*.

To further reduce the decoding complexity and network traffics in processing FMBs, we propose to explore the redundancies among those macroblocks that are required for the reconstruction of FMBs in this paper. Let us use Figure 4 to give a clearer account of our idea for reconstructing FMBs. Pixels in FMB of frame *n-1* are divided into two regions: *A* and *B*. All pixels in region *A* can be reconstructed according to the following formulation. For block motion-compensated prediction, each macroblock in frame *n*, $MB^n_{(k,l)}$, is reconstructed by motion-compensated prediction and it is given by

$$MB^n_{(k,l)} = MCMB^{n-1}(mv^n_{(k,l)}) + e^n_{(k,l)} \tag{2}$$

, where $MCMB^{n-1}(mv^n_{(k,l)})$ stands for the motion-compensated macroblock of $MB^{n-1}_{(k,l)}$ which is translated by the motion vector $mv^n_{(k,l)}$ in the previous reconstructed frame *n-1*. Equation (2) can be simply rewritten as

$$MCMB^{n-1}(mv^n_{(k,l)}) = MB^n_{(k,l)} - e^n_{(k,l)} \tag{3}$$



$MB^{n-3}_{(0,0)}$ $MB^{n-3}_{(0,1)}$
$MB^{n-2}_{(0,0)}$ $MB^{n-2}_{(0,1)}$
$MB^{n-1}_{(0,0)}$ $MB^{n-1}_{(0,1)}$
$mv^{n-2}_{(1,1)}$
$mv^{n-1}_{(1,1)}$
BMB
BMB
BMB
FMB
$mv^n_{(1,1)}$
$mv^{n-2}_{(1,0)}$
$MB^{n-3}_{(1,0)}$ $MB^{n-3}_{(1,1)}$
$MB^{n-2}_{(1,0)}$ $MB^{n-2}_{(1,1)}$
$MB^{n-1}_{(1,1)}$
frame n-3
(I-frame)
frame n-2
frame n-1
$MCMB^{n-1}(mv^n_{(1,1)})$
Region *A*
Region *B*
FMB

Figure 4. Redundancy reduction required for FMB

In Figure 4, the macroblock enclosed by the thick line in frame *n-1* is $MCMB^{n-1}(mv^n_{(k,l)})$. Note that pixels of $MB^n_{(k,l)}$ are available at the decoder. Equation (3) implies that the pixels in region *A* can be reconstructed by only sending the sign inversion of the quantized DCT coefficients of $e^n_{(k,l)}$. Similar to the technique used in BMB, all these sign inversed coefficients can be generated in VLC-domain. In other words, the pixels in the requested FMB of frame *n-1* overlapped with $MCMB^{n-1}(mv^n_{(k,l)})$ (region *A*) are computed from frame *n*, which has already stored in the client side.

By using the mentioned technique, we find that only region *B* of the requested FMB needs to be reconstructed from the previous reference macroblocks. This is again illustrated in the example shown in Figure 4. In this example, only the shaded region of $MB^{n-1}_{(1,1)}$ (region *B*) is reconstructed from the previous frame. In frame *n-2*, only $MB^{n-2}_{(1,1)}$ is actually required. Although the motion-compensated macroblock of $MB^{n-1}_{(1,1)}$ also covers another macroblock $MB^{n-2}_{(1,0)}$, it is no longer required. The reason behind is that region *A* of $MB^{n-1}_{(1,1)}$ can be predicted by referring from frame *n* which is available at the decoder. Similarly, in frame *n-3*, only $MB^{n-3}_{(0,1)}$ and $MB^{n-3}_{(1,1)}$ are necessary to be sent over the network and decoded by the decoder. The necessary macroblocks are used to reconstruct $MB^{n-1}_{(1,1)}$ can be reduced. Altogether, the server needs to transmit only 5 macroblocks to the decoder instead of 7 macroblocks as depicted in Figure 3. If the GOP (Group of picture) size is longer, the savings of the proposed technique will be more remarkable.

4. SIMULATION RESULTS

A series of computer simulations were conducted to evaluate the performance of the proposed algorithm 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 196 frames. "Claire", "Grandma" and "Carphone" are typical videophone sequences in QCIF (176×144) format, which were encoded at 64 Kb/s. "Salesman"(CIF 352×288), "Table Tennis" and "Football" (SIF 352×240) 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.

Table 1. Performance improvement of the proposed algorithm 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	*Proposed algorithm*	*SI*	*Proposed algorithm*
Salesman	1.5M	40.0%	51.5%	42.0%	51.4%
Football	1.5M	22.0%	28.0%	18.8%	24.9%
Table Tennis	1.5M	36.7%	39.1%	26.8%	30.3%
Carphone	64K	28.7%	39.7%	32.1%	46.6%
Claire	64K	71.3%	72.5%	83.2%	85.1%
Grandma	64K	59.6%	63.6%	76.3%	80.6%

Results of the simulations are used to compare the performances of the conventional streaming system and our previous macroblock-based system [7] which uses the technique of sign inversion of DCT coefficients for BMB (*SI*). 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. Table 1 shows that *SI* and the proposed algorithm have 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 40-80% 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. To further reduce the number of macrblocks to be decoded and bits to be sent, the algorithm proposed in this paper can work with *SI* to reduce the redundancies among those macroblocks that are required for the reconstruction of FMBs. Table 1 shows that the proposed algorithm produces further savings in terms of both number of macroblocks to be decoded and bits to be sent as compared with that of *SI*.

Figure 5 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 algorithm, *SI* and the conventional system for the "Salesman" sequence in the reverse-play operation. It is obvious that the proposed algorithm can achieve significant performance improvement in terms of the decoding complexity and the network traffic load. Note that, as shown in Figure 5, the required number of macroblocks and bits at each last frame of GOPs of the conventional system, *SI* and the proposed system are the same. It is due to the fact 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. Thus, both the technique of sign inversion and the proposed algorithm cannot be applied in the last frame of each GOP.

5. CONCLUSION

In this paper, we have improved our previously proposed 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. In this paper, a technique that reduces the redundancies of those related maroblocks for the reconstruction of FMBs has been proposed to further alleviate the computational burden of the client decoder in 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.

6. 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.

7. 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

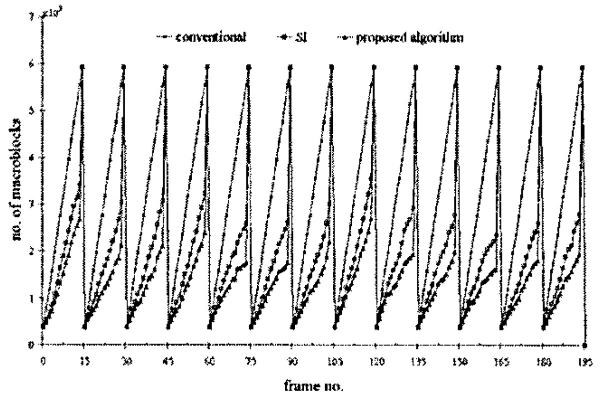
[3] ISO/IEC 13818-2, "Information technology -- generic coding of moving pictures and associated audio information: video," 1996.

[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.

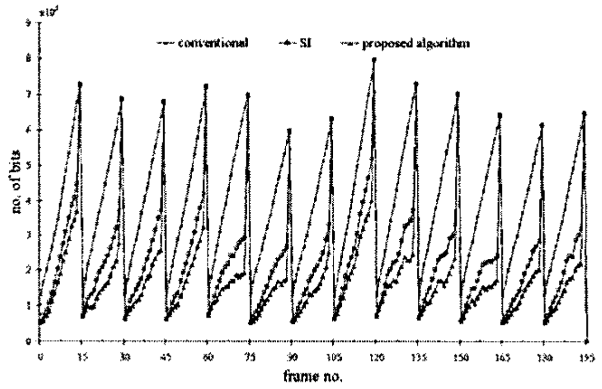
[5] M. S. Chen and D. D. Kandlur, "Downloading and stream conversion: supporting interactive playout of videos in a client station," in Proc. 2nd Int. IEEE Conf. Multimedia Computing and Systems, pp. 73-80, 1995.

[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, "Macroblock-based reverse play algorithm for MPEG Video Streaming" in Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2004), vol. III, pp. 753-756, May 23-26, 2004.



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



(b) Number of bits transmitted over the network.

Figure 5. Performance of the conventional system *SI* and the proposed algorithm for the "Salesman" sequence in the reverse-play operation.