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 involves playing back video frames in reverse order. However, it is not a straightforward task in MPEG digital video due to the use of predictive processing techniques in MPEG. In this paper, we propose a compressed-domain approach for the efficient implementation of an MPEG video streaming system to provide the reverse-play operation over a network. In the proposed video streaming server, a novel macroblock-based algorithm is used to adaptively select the necessary macroblocks, manipulate them in compressed-domain and send the processed macroblocks to the client machine. Besides, we propose a technique of mismatch error compensation such that there is only a negligibly small PSNR degradation during reverse playback of the proposed algorithm. Experimental results show that, as compared to the conventional system, the new streaming system reduces the required network bandwidth and the decoder complexity significantly in reverse playback.

Index Terms: Compressed-domain processing, digital video cassette recording (VCR), MPEG video, streaming video.

1. Introduction

To facilitate fast and effective browsing of the proliferating video content, a video streaming system should provide full VCR functionality. The set of effective VCR functionality includes forward, backward, stop, pause, fast forward, fast backward, and random access. This set of VCR functionality allows users to control the video browsing completely and it is also useful for video editing. However, the MPEG [1-7] 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 reverse playback of MPEG compressed video is to decode all the frames in the whole group-of-picture (GOP), store all frames in a large buffer of the decoder, and play the decoded frames in reverse. This approach will then lead to a huge memory in the client. Another way 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. This does not require huge memory, but it requires much higher bandwidth of the network and complexity of the decoder, which is also not desirable. The problem is more serious if the GOP is large.

Some works on the implementation of reverse playback for MPEG compressed video for streaming applications have recently been introduced [5-7]. Chen and Kandhur [7] addresses the problem of reverse playback of MPEG compressed video, which suggested an approach of converting an incoming MPEG bitstream with I-B-P structure into a local bitstream with I-B structure by performing a P-to-1 frame conversion at the client machine. This P-to-1 frame conversion was used to break the inter-frame dependencies between the P-frames and the I-frames. However, this approach requires extra decoder complexity to perform the P-to-1 conversion and higher storage cost to store the local bitstream in the client. Wee and Vasudev [7] described a reverse-play transcoder which is used to convert the I-P frames into another I-P bitstream with a reversed frame order. A method of estimating the reverse motion vectors for the new I-P bitstream based on the forward motion vectors of the original I-P bitstream as described in [6] is used to reduce the computational complexity of this transcoding process. The transcoding process, however, still requires much computation and will cause drift due to the motion vector approximation [7].

In this paper, we will explore a compressed-domain approach for an efficient implementation of the MPEG streaming video system to provide the reverse-play operation over a network with minimum requirements on the network bandwidth and the decoder complexity. The organization of this paper is as follows. Section 2 of this paper describes the proposed system with reverse-play support. A technique of mismatch error compensation is discussed in Section 3. Simulation results are presented in Section 4. Finally, some concluding remarks are given in Section 5.

Figure 1. The proposed architecture for the video streaming system with VCR functionality.
2. The Proposed Video Streaming System with VCR Functionality

The architecture of the proposed system is shown in Figure 1. In the forward-play operation, the switches $SW_1$, $SW_2$, $SW_3$, and $SW_i$ are connected to $A_1$, $A_2$, $A_3$, and $A_4$ respectively. On the other hand, in contrast to the frame-based scheme used in the conventional architecture, a macroblock-based scheme is proposed to use in the reverse-play operation. At the server side of Figure 1, motion vectors are extracted from the video stream and these motion vectors are used by a macroblock selector to identify the types of macroblocks. Two types of macroblocks are now defined in Figure 2. We assume that $MB_{n-1}^{B}$ represents the macroblock at the $k$th row and $l$th column of frame $n-1$ (the next frame to be displayed). $MB_{n-1}^{B}$ is defined as a backward macroblock (BMB) if the macroblock of frame $n$ having the same spatial position of $MB_{n-1}^{B}$, i.e. $MB_{n}^{B}$, is coded without motion compensation (non-MC macroblock). Otherwise, it is defined as a forward macroblock (FMB). For example, in Figure 2, since the motion vector in $MB_{n}^{B}$, $mv_{n}^{B}$, is zero, it means that $MB_{n}^{B}$ is a non-MC macroblock and the macroblock selector classifies $MB_{n}^{B}$ as BMB. On the other hand, since $MB_{n}^{A}$ is coded with motion compensation (MC-macroblock), $MB_{n}^{A}$ is classified as FMB. In this paper, our scheme works at the level of macroblocks and the server will process each type of macroblocks in compressed-domain such that complete decoding and encoding are not required at the server.

![Figure 2. Definition of the forward macroblock (FMB) and the backward macroblock (BMB).](image)

Block motion-compensated prediction (MCP) is the type of prediction used in MPEG standards [1-3]. This prediction type is what gives the MPEG codecs the advantage over pure still-frame coding methods. In motion-compensated prediction, previously transmitted and decoded frame serves as the prediction. The difference between the prediction and the actual current frame is the prediction error. The coded prediction error is added to the prediction to obtain the final representation of the current reconstructed frame. At the client side of Figure 1, each macroblock in frame $n$, $MB_{n}^{A}$, is reconstructed according to motion-compensated prediction and it is given by

$$MB_{n}^{A} = MCMCMB^{*}(mv_{n}^{A}) + e_{n}^{A},$$

where $MCMCMB^{*}(mv_{n}^{A})$ stands for the motion-compensated macroblock of $MB_{n}^{A}$ which is translated by the motion vector $mv_{n}^{A}$ in the previous reconstructed frame $n-1$ and $e_{n}^{A}$ is the prediction error between $MB_{n}^{A}$ and its motion-compensated macroblock, $MCMCMB^{*}(mv_{n}^{A})$. The frame $n$ is then stored in FB for decoding subsequent frame $n+1$ in the forward-play operation.

When a user issues a reverse-play command at frame $n$, the next frame to be displayed is frame $n-1$, i.e., all $MB_{n}^{B}$ in frame $n-1$ are requested. To reconstruct each $MB_{n}^{B}$, all the related previous macroblocks in P-I-frames need to be sent over the network and decoded by the decoder in the conventional video streaming system. It becomes impractical when GOP is large. However, if $MB_{n}^{A}$ is classified as BMB, its corresponding macroblock in frame $n$, $MB_{n}^{A}$, is coded without motion compensation. It means that the spatial position of $MB_{n}^{B}$ is the same as that of $MB_{n}^{A}$. Hence, for this specific case, $MCMCMB^{*}(mv_{n}^{A})$ is equal to $MB_{n}^{A}$, and equation (1) can be rewritten as

$$MB_{n}^{B} = MB_{n}^{A} + e_{n}^{B},$$

where $e_{n}^{B} = e_{n}^{A}$. It becomes impractical when COP is large. However, if $MB_{n}^{A}$ is classified as BMB, its corresponding macroblock in frame $n$, $MB_{n}^{A}$, is coded without motion compensation. It means that the spatial position of $MB_{n}^{B}$ is the same as that of $MB_{n}^{A}$. Hence, for this specific case, $MCMCMB^{*}(mv_{n}^{A})$ is equal to $MB_{n}^{A}$, and equation (1) can be rewritten as

$$MB_{n}^{B} = MB_{n}^{A} + e_{n}^{B},$$

where $e_{n}^{B} = e_{n}^{A}$. It becomes impractical when COP is large. However, if $MB_{n}^{A}$ is classified as BMB, its corresponding macroblock in frame $n$, $MB_{n}^{A}$, is coded without motion compensation. It means that the spatial position of $MB_{n}^{B}$ is the same as that of $MB_{n}^{A}$. Hence, for this specific case, $MCMCMB^{*}(mv_{n}^{A})$ is equal to $MB_{n}^{A}$, and equation (1) can be rewritten as

$$MB_{n}^{B} = MB_{n}^{A} + e_{n}^{B},$$

where $e_{n}^{B} = e_{n}^{A}$. It becomes impractical when COP is large. However, if $MB_{n}^{A}$ is classified as BMB, its corresponding macroblock in frame $n$, $MB_{n}^{A}$, is coded without motion compensation. It means that the spatial position of $MB_{n}^{B}$ is the same as that of $MB_{n}^{A}$. Hence, for this specific case, $MCMCMB^{*}(mv_{n}^{A})$ is equal to $MB_{n}^{A}$, and equation (1) can be rewritten as

$$MB_{n}^{B} = MB_{n}^{A} + e_{n}^{B},$$

where $e_{n}^{B} = e_{n}^{A}$. It becomes impractical when COP is large. However, if $MB_{n}^{A}$ is classified as BMB, its corresponding macroblock in frame $n$, $MB_{n}^{A}$, is coded without motion compensation. It means that the spatial position of $MB_{n}^{B}$ is the same as that of $MB_{n}^{A}$. Hence, for this specific case, $MCMCMB^{*}(mv_{n}^{A})$ is equal to $MB_{n}^{A}$, and equation (1) can be rewritten as

$$MB_{n}^{B} = MB_{n}^{A} + e_{n}^{B},$$

where $e_{n}^{B} = e_{n}^{A}$. It becomes impractical when COP is large. However, if $MB_{n}^{A}$ is classified as BMB, its corresponding macroblock in frame $n$, $MB_{n}^{A}$, is coded without motion compensation. It means that the spatial position of $MB_{n}^{B}$ is the same as that of $MB_{n}^{A}$. Hence, for this specific case, $MCMCMB^{*}(mv_{n}^{A})$ is equal to $MB_{n}^{A}$, and equation (1) can be rewritten as

$$MB_{n}^{B} = MB_{n}^{A} + e_{n}^{B},$$

where $e_{n}^{B} = e_{n}^{A}$.
the sign of all DCT coefficients in the server, the sequence containing more BMBs can alleviate the decoder complexity and network traffic more significantly.

Table 1. Percentage of BMB for various sequences.

<table>
<thead>
<tr>
<th></th>
<th>Claire</th>
<th>Grandma</th>
<th>Salesman</th>
<th>Carphone</th>
<th>Table Tennis</th>
<th>Football</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>89.75</td>
<td>81.57</td>
<td>61.26</td>
<td>52.37</td>
<td>49.30</td>
<td>27.51</td>
</tr>
</tbody>
</table>

The situation of FMB is different. The sign inversion of DCT coefficients cannot be employed since $MCMB^{n-1}(m_{n-1})$ is no longer equal to $MB^{n-1}$ and then equation (2) is not valid for FMB. In other words, $MCMB^{n-1}$ cannot be reconstructed from $MB^{n-1}$. All related macroblocks in frame $n-2$ which are the motion-compensated macroblocks of FMBs in frame $n-1$ needs to be sent. Those macroblocks will act as a reference for performing motion compensation of FMBs in frame $n-1$. In other words, the decoder complexity is decreased by 1. This process will cause the problem of decoding BMBs and it is illustrated in the following example. In the original MPEG-2 bitstream, assume that there is a BMB and the last coefficient of one of its 8x8 blocks is “3”. In the forward playback, mismatch control at the decoder adjusts the last coefficient to “2”. In contrast, this coefficient is sign-inversed in the compressed domain during reverse playback, that is, the last coefficient becomes “-3”. After the operation of mismatch control, we get “-4” instead of “-2”. To solve this problem, compensation of mismatch control at the decoder has been performed. When mismatch control is activated for BMBs at the decoder, the last coefficient of this block after mismatch control is examined. If this coefficient after mismatch control is even, it is further increased by 2, otherwise it is decreased by 2. In the above example, the last coefficient after mismatch control is “-4”. By using the proposed compensation technique, “-2”, which is the desired coefficient, is generated again. In this way, the quality drift introduced by mismatch control in BMBs can be fully compensated.

4. Simulation Results

A series of computer simulations were conducted to evaluate the performance of the proposed sign inversion technique when applied to the video streaming system with VCR support. MPEG-2 encoders were 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 video sequences in QCIF (176x144) format, which were encoded at 64 Kbps. “Salesman” (CIF 352x288), “Table Tennis” and “Football” (SIF 352x240) were encoded at 1.5 Mbs. We have simulated the situation of the I-P structure for $L=15$. The starting point of the reverse-play operation was at the end of the sequence. For all testing sequences, the frame-rate of the video stream was 30 frames/s.

To verify the performances of the proposed system, it is compared to conventional video streaming system. The saving in terms of average number of macroblocks to be decoded and bits to be sent are tabulated in Table 3. The average number of macroblocks sent for decoding is directly proportional to the decoder complexity. In Table 3, we show that the proposed system outperforms the conventional one in all sequences. The results are more significant for the sequences “Claire”, “Grandma”, and “Salesman”. The saving in both the bits to be sent over the network and macroblocks to be decoded by the decoder is from 40-75% for those sequences. In other words, the decoder complexity for playing those sequences in the reverse order is reduced by 40-75%. 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. Figure 3 shows the 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 the proposed method can achieve significant performance improvement in terms of the decoder complexity and the network traffic load. Note that, as shown in Figure 3, the required number of macroblocks and bits at each last frame of GOP of the proposed system 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

3. Mismatch Error Compensation

Theoretically, the proposed DCT-domain technique for BMBs will not introduce any quality degradation during reverse playback. However, the technique will cause quality drift due to mismatch control and the clipping operations of the MPEG-2 video algorithm. To alleviate such quality drift, we suggest implementing suitable compensation at the decoder. For instance, to minimize the error accumulation due to IDCT mismatch at the MPEG-2 decoder, mismatch control is performed before IDCT.
BMB exists in the last frame of the current GOP. Thus, the technique of sign inversion cannot be applied in the last frame of each GOP.

To verify the effect of the proposed algorithm on the visual quality during reverse playback, the display frames of the proposed system and the conventional system in reverse playback are compared. Table 4 shows the PSNR degradation of the proposed algorithm. Compared with the conventional system, only a negligibly small PSNR degradation of the proposed algorithm is shown. The reason behind is that the quality drift introduced by mismatch control in BMBs has been fully compensated and the small degradation of PSNR is mainly caused by the clipping operations in the MPEG decoder.

5. Conclusion

In this paper, we have proposed an efficient reverse-play algorithm for an MPEG video streaming system. The proposed algorithm is motivated by the center-biased motion vector distribution characteristics of real-world video sequences. With the motion information, the video streaming server organizes the macroblocks in the requested frame into two categories. Then it selects the necessary macroblocks adaptively, processes them in the compressed-domain and sends the processed macroblocks to the client machine. The technique of sign inversion of DCT coefficients is proposed to simplify the decoder complexity while maintaining the low network bandwidth requirement. We also propose a mismatch error compensation method to maintain the PSNR quality in a negligible level. Simulation results show that, with our proposed scheme, an MPEG video streaming system with reverse-play functionality can minimize the required network bandwidth and decoder complexity significantly with almost the same visual quality compared to the conventional one.

Acknowledgments

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 scholarships provided by the University.

References


Table 3. Performance improvement of the proposed system over the conventional system.

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Bitrate</th>
<th>Saving of macroblocks to be decoded by the decoder</th>
<th>Saving of bits to be sent over the network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesman</td>
<td>1.5M</td>
<td>40.0%</td>
<td>41.0%</td>
</tr>
<tr>
<td>Football</td>
<td>1.5M</td>
<td>21.9%</td>
<td>18.4%</td>
</tr>
<tr>
<td>Table Tennis</td>
<td>1.5M</td>
<td>36.5%</td>
<td>25.0%</td>
</tr>
<tr>
<td>Carphone</td>
<td>64K</td>
<td>28.7%</td>
<td>29.7%</td>
</tr>
<tr>
<td>Claire</td>
<td>64K</td>
<td>71.3%</td>
<td>76.5%</td>
</tr>
<tr>
<td>Grandma</td>
<td>64K</td>
<td>59.60%</td>
<td>72.4%</td>
</tr>
</tbody>
</table>

Table 4. PSNR performances of the proposed system.

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Bitrate</th>
<th>PSNR degradation during reverse playback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesman</td>
<td>1.5M</td>
<td>0.0014</td>
</tr>
<tr>
<td>Football</td>
<td>1.5M</td>
<td>0.0055</td>
</tr>
<tr>
<td>Table Tennis</td>
<td>1.5M</td>
<td>0.0217</td>
</tr>
<tr>
<td>Foreman</td>
<td>64K</td>
<td>0.0000</td>
</tr>
<tr>
<td>Carphone</td>
<td>64K</td>
<td>0.0002</td>
</tr>
<tr>
<td>Claire</td>
<td>64K</td>
<td>0.0285</td>
</tr>
<tr>
<td>Grandma</td>
<td>64K</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Figure 3. Performance of the proposed system and the conventional system for the “Salesman” sequence in the reverse-play operation.