MINIMUM COST IMPLEMENTATION OF FULL VCR FUNCTIONALITY IN MPEG VIDEO STREAMING

Chia-Wen Lin[*], *Jian Zhou*[**], *Ming-Ting Sun*[**], and *Hung Hseng Hsu*[+]
[*]Dept. Computer Science & Information Engineering
National Chung Cheng University, Taiwan, R.O.C.
[**]Dept. Electrical Engineering
University of Washington, Washington, U.S.A.
[+]Computer & Communications Research Labs
Industrial Technology Research Institute, Taiwan, R.O.C.

ABSTRACT

With the proliferation of online multimedia content, the popularity of multimedia streaming technology, and the establishment of MPEG video coding standards, it is important to investigate how to efficiently implement an MPEG video streaming system. Digital video cassette recording (VCR) functionality enables quick and user friendly browsing of multimedia content, thus is highly desirable in streaming video applications. The implementation of full VCR functionality, however, presents several technical challenges which have not yet been well resolved. In this paper, we investigate the impacts of the VCR functionality on the video decoder complexity and the network traffic. We also propose a minimum-cost scheme for the efficient implementation of MPEG streaming video system to provide full VCR functionality over a network with minimum requirements on the network bandwidth and the decoder complexity.

1. INTRODUCTION

With the proliferation of online multimedia content, it is highly desirable that multimedia streaming systems support effective and quick browsing. A key technique that enables quick and user friendly browsing of multimedia content is to provide full VCR functionality [1-3]. The set of effective VCR functionality includes forward, backward, step-forward, step-backward, fast-forward, fast-backward, random access, pause, and stop (and return to the beginning). This set of VCR functionality allows the users to have complete controls over the session presentation and is also useful for other applications such as video editing.

However, the implementation of the full VCR functionality with the MPEG [4-6] coded video is not a trivial task. MPEG video compression is based on motion compensated predictive coding with an I-B-P-frame structure. The I-B-P-frame structure allows a straightforward realization of the forward-play, but it imposes several constraints on other trick modes such as random access, backward play, fast-forward play, and fast-backward play. Straightforward implementation of these trick modes requires much higher network bandwidth and decoder complexity compared to those required for the regular forward-play function. With the I-B-P-frame structure, to decode a P-frame, the previously encoded I/P-frames need to be decoded first. To decode a B-frame, both the I/P-frames before and after this B-frame need to be first decoded. To implement a backward-play function, a straightforward implementation at the decoder is to decode the whole group of picture (GOP), store all the decoded frames in a large buffer and play the decoded frames backward. However this will require a huge buffer in the decoder to store the decoded frames. Another possibility 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 the huge buffer but will require the client machine to operate in an extremely high speed which is also not desirable. The extra memory and computational costs soon become unaffordable when the GOP size is large. Besides the problem with backward-play, fast-forward/backward and random-access also present difficulties. When a P/B-frame is requested, all the related previous P/I-frames need to be sent over the network and decoded by the decoder. This requires the network to send all the related frames besides the requested frame at a much higher rate. When many clients request the trick-modes, it may result in much higher network traffics compared to the normal forward-play situation. It also requires high computational complexity in the client decoder to decode all these extra frames.

Some recent works have addressed the implementation of VCR functions for MPEG compressed video in streaming video applications [1,7-9]. Chen *et al.* [1] described a method of transforming an MPEG I-P-B compressed bit-stream into a local I-B form by performing a P-to-I frame conversion to convert all the retrieved P-frames into I-frames at the decoder, thereby breaking the inter-frame dependencies between the P-frames and the I-frames. After the frame syntax conversion and frame reordering, the motion vector swapping approach developed in [7] can be used for backward-play of the new I-B bit-stream. Although this approach does not require the frame memories to store all the decoded frames for the backward-play, it increases the computational complexity and still requires significant storage to store the converted I-frames at the decoder. It also causes drift in the B-pictures since the converted I-pictures are not exactly the same as the original P-pictures. Wee *et al.* [8] presented a method which divides the incoming I-P-B bit-stream into two parts: I-P frames and B-frames. A transcoder is then used to convert the I-P frames into another I-P bit-stream with a reversed frame order. A method of estimating the reverse motion vectors for the new I-P bit-stream based on the forward motion vectors of the original I-P bit-stream as described in [9] is used to reduce the computational complexity of this transcoding process. For B-frames, the motion vector swapping scheme proposed in [7] is used for reverse-play. The transcoding process, however, still requires much computation and will cause drift due to the motion vector approximation [9]. None of the above methods addressed the problem of the extra network traffic and the decoding

0-7803-6685-9/01/$10.00©2001 IEEE

complexity caused by some VCR functions such as fast-forward/backward play and random-access.

Since network-bandwidth and the decoder complexity are major concerns in most streaming video applications, in this paper, we investigate effective techniques to implement the full VCR functionality in an MPEG video streaming system with minimum network bandwidth requirement and decoder complexity. We propose to use dual bit-streams at the server to resolve the problem of reverse-play. Based on the dual-bit-stream structure, we propose a novel frame-selection scheme at the server to minimize the required network bandwidth and the decoder complexity. This scheme determines the frames to stream over the networks by switching between the two bit-streams based on a least-cost criterion. We also discuss a drift compensation scheme to prevent the drift caused by the bit-stream switching.

2. SUPPORTING FULL VCR FUNCTIONALITY WITH LEAST COST

To solve the problem in the backward-play operation, we propose to add a reverse-encoded bit-stream in the server. To generate the reverse-encoded bit-stream, in the encoding process, after we finish the encoding and reach the last frame of the video sequence, we encode the video frames in the reverse order. The proposed dual-bit-stream structure is shown in Fig. 1, where "*F*" and "*R*" stand for the forward and reverse encoded bit-streams respectively. For clarity of the presentation but without loss of generality, in this paper, we use an example in which the video is coded in I/P-frames with a GOP size of 14 frames as shown below. The extension of our discussion to the case with the general I-B-P GOP structure is straightforward.

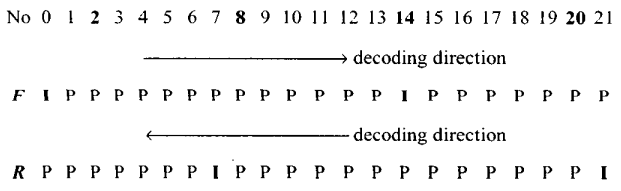


Fig. 1. Proposed dual-bit-stream structure

As shown above, we arrange the encoding so that the I-frames in the reverse bit-stream are interleaved between the I-frames in the forward bit-stream. In this way, the required number of frames sent by the server and decoded by the decoder can be further reduced in other trick modes as will be explained later. Two metadata files recording the location of the frames in each compressed bit-stream are also generated so that the server can switch from the forward-encoded bit-stream to the reverse-encoded bit-stream and vice versa easily. With the reverse-encoded bit-stream, when the client requests the backward-play mode, the server will stream the bits from the reverse-encoded bit-stream. Using this scheme, the complexity of the client machine and the required network bandwidth for the backward-play mode can be minimized. The storage requirement of the server will be about doubled. However, this is usually much more desirable than to require a large network bandwidth and to increase the complexity of the client machines since the network bandwidth is more precious and there may be a large number of client machines in the streaming video applications. Since the encoding of the video is done off-line, the extra time needed in producing the reverse bit-stream is not an important concern.

To reduce the decoder complexity and the network traffic in the fast forward/backward and the random-access modes, we propose a frame-selection scheme which minimizes a predefined "cost" using bit-stream switching. The cost can be the decoding effort at the client decoder or the traffic over the networks, or a combination of both. This is further explained as follows.

Let c_{R_C} stands for the cost of decoding the next requested P-frame from the current displayed frame, c_{R_FI} for the cost of decoding the next requested P-frame from the closest I-frame in the forward bit-stream, and c_{R_RI} for the cost of decoding the next requested frame from the closest I-frame of the reverse-encoded bit-stream. To minimize the number of frames sent to the decoder, the costs can be the distances from the possible reference frames to the next requested frame. To minimize the network traffic, the costs can be the total number of bits from the possible reference frames required for decoding the next requested frame. It is also possible to use different weights to combine the two costs according to the channel condition and the client capability. Based on the current play-direction, the requested mode, and the costs c_{R_C}, c_{R_FI}, and c_{R_RI}, the reference frame to the next requested frame with the least cost will be chosen to initiate the decoding. This will also determine the selection of the next bit-stream and the decoding direction. This least-cost criterion will only be activated in the fast forward/backward and the random access modes to avoid frequent bit-stream switching in the normal operations.

To illustrate the scheme, we use the above example again, assuming that the previous mode was backward-play and the requested mode is fast-backward with a speed-up factor 6 which needs to display a sequence of frames with frame-numbers 20, 14, 8, 2, ... For simplicity, in the following examples we use the minimum decoding distance criterion (which roughly corresponds to minimum decoder complexity) to illustrate the selection of the next reference frame and the effectiveness of the proposed method.

Our method will operate as follows:

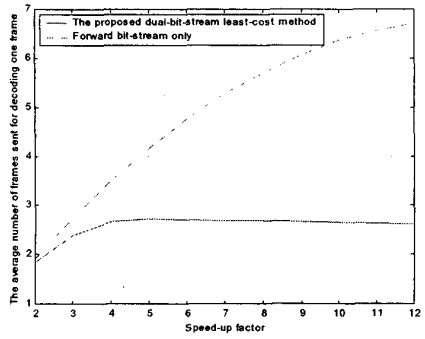
1. The current position is frame 20 which was decoded using the reverse bit-stream (*R*).
2. Frame 14 will be decoded from the forward bit-stream (*F*) directly since it is an I-frame.
3. Frame 8 will be decoded from frame 7 of the backward bit-stream, since the distance between frame 7 of the reverse bit-stream (an I-frame) and the requested frame (frame 8) is less than the distances between the requested frame and the current decoded frame (frame 14 of the reverse bit-stream), and the closest I-frame of the forward bit-stream (it's also frame 14). Note that, in this case, we use frame 7 of the reverse bit-stream (an I-frame) as an approximation of frame 7 of the forward bit-stream (a P-frame) to predict frame 8 of the forward bit-stream. This will cause some drift. However, in the fast-forward/backward modes, the drift is relatively insensitive to human eyes due to the fast change of the content displayed. When it resumes normal forward/reverse play, the I-frames will terminate the drift. The drift problem will be further investigated in the next section.
4. Frame 2 will be decoded from frames 0 and 1, using the forward bit-stream, since the decoding effort from frame 0 of the forward bit-stream (an I-frame) is the minimum.

The bit-stream sent from the server will have the following form:

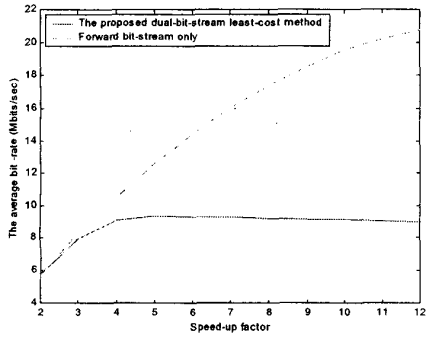
P	**I**	I	**P**	I	P	**P**	...	frame type
20	**14**	7	**8**	0	1	**2**	...	frame number
R	***F***	*R*	***F***	*F*	*F*	***F***	...	selected bit-stream

In this way, we only need to send and decode 6 frames. Without the least-cost decoding scheme, we will need to send and decode 13 frames from the reverse bit-stream.

If the minimum decoding distance criterion is used (i.e., to minimize the number of frames sent to the decoder), the proposed scheme will guarantee that the maximum amount of decoding to access any frame in the sequence is less than GOP/4 frames if the I-frames in the forward and the reverse bit-streams are interleaved. In addition, no huge temporary buffer is required in the decoder. If the I-frames in the forward and the reverse bit-streams are aligned, the maximum amount of decoding to access any frame will be less then GOP/2 frames.



(a)



(b)

Fig. 2. (a) The average number of frames; (b) the average bit-rate to send the "Mobil and Calendar" sequence over network using the proposed method with respect to different speed-up factors.

Fig. 2 compares the average numbers of frames sent to the decoder for decoding a frame and the average network traffics with and without the proposed dual-bit-stream least-cost method with respect to different speed-up factors in the fast-forward operation. Two bit-streams generated by forward and reverse encoding a 280-frame (20 GOPs in our example) "Mobile and Calendar" test sequence at 3 Mbps with a frame-rate of 30 fps are used for simulation. From the above analyses, in the fast-forward/backward and random-access operations, the sever needs to send several extra frames to the decoder to display one frame, thereby resulting in a heavy burden on the networks (especially when the number of users is large) and increasing the decoder complexity. Note that, with the proposed method, when the speed-up factor reaches GOP/4 (e.g., 3.5 in our example), the decoding complexity and the network traffic will not continue to grow even when the speed-up factor gets higher. This is because, when the speed-up factor k is larger than GOP/4, the server always can find an I-frame in one of the two bit-streams which has shorter distance to the next displayed frame from the current displayed P-frame, since the distance for the nearest I-frame is guaranteed to be less than GOP/4. In this case the number of frames to be sent for displaying a requested frame will have a range of [1, GOP/4+1]. If the distribution is uniform, the average number of frames can be approximated by GOP/8+1, which is 2.75 when GOP=14, very close to the simulation result. From Fig. 2, it is obvious that the proposed method can achieve significant performance improvement in terms of the decoder complexity and the network traffic load. When the speed-up factor $k \geq$ GOP/4, the proposed method guarantees a nearly constant decoding and network traffic cost.

3. DRIFT COMPENSATION

As mentioned above, in the proposed scheme, I- or P-frames of one bit-stream may be used to approximate P-frames of the other bit-stream. This approximation, however, will lead to frame mismatch and thus cause drift when the approximated frames are used as the reference frames to predict the following P/B-frames as illustrated in Fig. 3. In Fig. 3, the "Mobile &. Calendar" sequence is encoded at 3 Mbps. The GOP size is 14 and the speed-up factor is 6. When the server performs an I-to-P or a P-to-P approximation by using the bit-stream switching, there is a PSNR drop. The drift caused by the bit-stream switching can be as large as 2.5 dB and will last until the next I-frame as shown. However, the subjective degradation observed is not significant, since the fast display speed in the fast forward/backward modes will mask most of the spatial distortions.

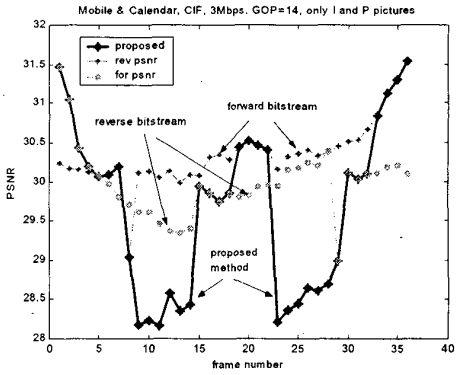
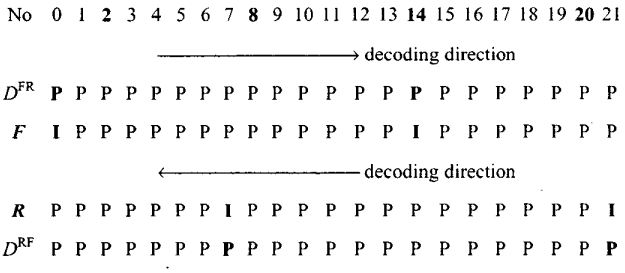


Fig. 3. PSNR comparison of the forward bit-stream, the reverse bit-stream, and the bit-stream generated using the proposed method.

In the random access mode, the drift will only last a few frames within a GOP, thus will not cause serious degradation. In the fast forward/backward mode, the drift is relatively insensitive to human eyes due to the fast changes of the content displayed.

However, in some applications, it may still be desirable to prevent the drift. The drift problem can be resolved by adding two bit-streams consist of all P-frames for the drift-compensated bit-stream switching as explained using the following example:



where D^{FR} is a bit-stream used for switching from the I- or P-frames of the forward bit-stream to the P-frames of the reverse bit-stream, while D^{RF} is used for switching from the I- or P-frames of the reverse bit-stream to the P-frames of the forward bit-stream. The bit-stream D^{FR} is obtained as follow.

$$D_n^{FR} = \text{Pred}(F_n, R_{n-1}) \quad (1)$$

and

$$D_n^{RF} = \text{Pred}(R_n, F_{n+1}) \quad (2)$$

where Pred(A,B) represents an inter-frame prediction process that frame B is predicted from the reference frame A. When performing the bit-stream switching, the correctly predicted frame is used for switching between the forward and the reverse bit-streams. For example, if the bit-stream is switched from F_n (an I- or P-frame) to R_{n-1} (a P-frame), then the server will send the frames as ... F_n, D_n^{FR}, R_{n-2} ..., instead of sending ...F_n, R_{n-1}, R_{n-2}, ... With the two drift correction bit-streams, for the above fast-backward example, the proposed method will generate a bit-stream for the above example as follows:

P	**I**	I	**P**	I	P	**P**	...	frame type
20	**14**	7	**8**	0	1	**2**	...	frame number
R	***F***	*R*	***D^{RF}***	*F*	*F*	***F***	...	selected bit-stream

Since D^{RF} is encoded based on the decoded frames from the forward and the reverse bit-streams, the drift can be compensated. If the prediction errors of the drift-compensated predictive frames in D^{RF} and D^{FR} are losslessly encoded, there will be no drift. Otherwise, there will be small drift. The drift will depend on the quantization step-sizes used in the encoding. A finer quantizer will lead to lower drift, while increasing the storage for the drift compensation bit-streams. Since the encoding process to obtain all the bit-streams is done off-line in streaming video applications, the encoding complexity is not a major concern.

It should be noted that if the I-frames of the two bit-streams are interleaved, and the speed-up factor is high enough (e.g., the frame skipping distance ≥ GOP/4), in the proposed method, only replacing P-frames with I-frames will be sufficient because we always can find an I-frame in one of the two bit-streams which has shorter distance to the next requested frame than the current decoded P-frame. In this case, we only need to store the drift compensation frames for all the I-frames of both bit-streams. In the fast-forward/backward operations with small speed-up factors (e.g., 2 or 3), however, the proposed least-cost scheme has limited gain on the decoding complexity and the network traffics as shown in Fig. 2. Thus, a possible low-complexity solution for the fast-forward/reverse play is:

If k < GOP/4

Use dual bit-streams without performing bit-stream switching.

else

Use bit-stream switching with I -> P drift-compensation only.

Using this modified scheme, only the drift-compensation frames for the I -> P approximations need to be created, thus the storage cost for the drift-compensation frames can be reduced drastically without significant performance sacrifice in typical applications.

4. CONCLUSIONS

In this paper, we discussed issues in implementing an MPEG video streaming system with full VCR functionality. We showed that when the users request reverse-play, fast-forward/reverse-play, or random access, it may result in much higher network traffics than the normal-play mode. These trick-modes may also require high client machine complexity. We proposed to use a reverse-encoded bit-stream to simplify the client terminal complexity while maintaining the low network bandwidth requirement. We also proposed a minimum-cost frame-selection scheme which can minimize the number of frames needed to be sent over the network and to be decoded. We also discussed a drift compensation scheme to prevent the drift caused by the bit-stream switching. We showed that with our proposed scheme, an MPEG-4 video streaming system with full VCR functionality can be efficiently implemented to minimize the required network bandwidth and decoder complexity.

5. REFERENCES

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

[2] T. D.C. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, vol. 13, pp. 14-24, Aug. 1994.

[3] F. C. Li *et al.*, "Browsing digital video," Technical Report: MSR-TR-99-67, Microsoft Research, Sep. 1999, ftp://ftp.research.microsoft.com/pub/tr/tr-99-67.pdf

[4] ISO/IEC 11172, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s," (MPEG-1), Oct. 1993.

[5] ISO/IEC 13818-2 "Generic coding of moving pictures and associated audio". (MPEG-2), Nov. 1993.

[6] ISO/IEC JTC1/SC29/WG11 "Coding of moving pictures and associated audio MPEG98/W2194," (MPEG-4), Mar. 1998.

[7] S. Chen, "Reverse playback of MPEG video," U.S. Patent 5,739,862.

[8] S. J. Wee and B. Vasudev, "Compressed-domain reverse play of MPEG video streams," *Proc. SPIE Conf. Multimedia Syst. and Appl.*, pp. 237-248, Nov. 1998.

[9] S. J. Wee, "reversing motion vector fields," *Proc. IEEE Int. Conf. Image Proc.*, Oct. 1998.