ICICS-PCM 2003
15-18 December 2003
Singapore

Efficient Reverse-Play Algorithm for MPEG Video Streaming with VCR Functionality

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 propose a compressed-domain approach for the efficient implementation of an MPEG video streaming system to provide the reverse-play operation over a network with minimum requirements on the network bandwidth and the decoder complexity. 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. Experimental results show that, as compared to the conventional system, the new streaming system reduces the required network bandwidth and the decoder complexity significantly.

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

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. This set of VCR functionality allows users to control the video browsing completely and it is also useful for video editing.

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 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 reversely. However, this approach will require 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 Kandlur [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-I frame conversion at the client machine. This P-to-I 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-I 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 presents a study of impacts of the reverse-play operation on decoder complexity and network traffics. The proposed reverse-play algorithm is then described in Section 3. Simulation results are presented in Section 4. Finally, some concluding remarks are given in Section 5.

2. Impacts of Reverse-Play on Decoder Complexity and Network Traffics

Now we show the impact of reverse playback on the decoding complexity and network traffics. 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

0-7803-8185-8/03/$17.00 © 2003 IEEE

discussions on the cases that the video stream contains I- and P-frames only. If a user issues a reverse-play operation at frame *n*, the next frame to be display is frame *n-1*. The client machine generates a reverse-play command and a requested frame-number, and sends them to the server. If the requested frame is an I-frame, the server only needs to send this frame, and the decoder can decode it immediately. However, if the requested frame is a P-frame, the server needs to send all the P-frames from the previous nearest I-frame to this requested frame. Since the next frame to be display is frame *n-1*, the client machine does not need to display all the previous frames until frame *n-1*. For example, consider the case as shown in Figure 1. 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 sends frame *0* to frame *n-1* from the video stream. At the client side, frame *0* to frame *n-2* does 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.

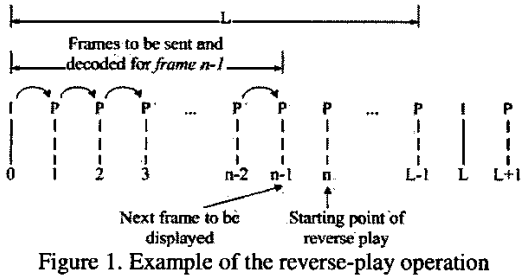


Figure 1. Example of the reverse-play operation

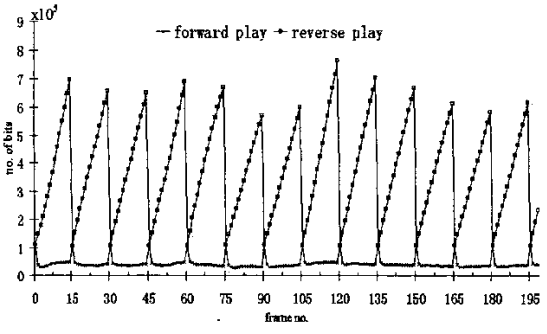
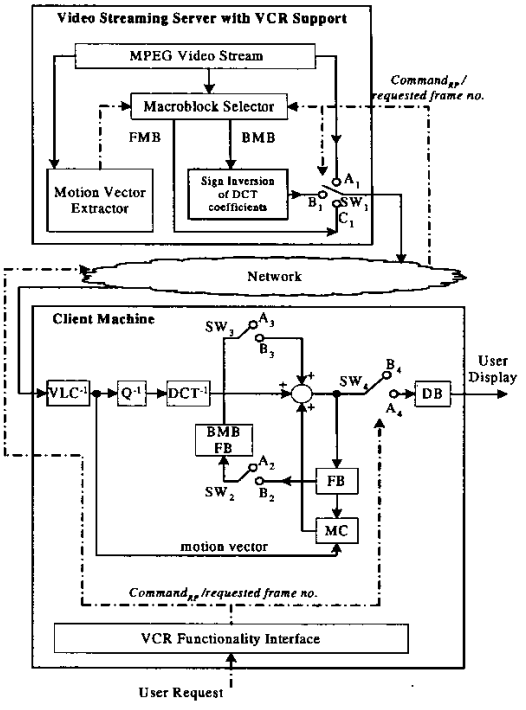


Figure 2. Bandwidth requirement for sending the "Salesman" sequence over network with respect to the forward and reverse-play operations.

The bandwidth requirement of the reverse-play operation is depicted in Figure 2 where the test video stream used for simulation is "Salesman" sequence with a length of 200 frames. The "Salesman" sequence is encoded at 1.5 Mb/s with a frame-rate of 30 fps and the length of GOP (L) is 15 with an I-P structure. The starting point of the reverse playback is at the end of the sequence. This figure shows that the server needs to send extra bits to the decoder to display one frame. Thus, straightforward implementation of the reverse-play operation requires much higher network bandwidth and decoder complexity compared to those required for the forward play operation.



VLC^{-1}: Inverse Variable Length Coding
Q^{-1}: Inverse Quantization
DCT^{-1}: Inverse Discrete Cosine Transform
MC: Motion Compensation
FB: Frame Buffer
DB: Display Buffer
BMB FB: Frame Buffer for BMB

Figure 3. The proposed architecture for the video streaming system with VCR functionality.

3. The Proposed Video Streaming System with VCR Functionality

The architecture of the proposed system is shown in Figure 3. In the forward-play operation, the switches SW_1, SW_2, SW_3 and SW_4 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 3, 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. For illustration, we use the example in Section 2 again, assuming that a user requests a reverse-play command at frame *n*, the next frame to be display is frame *n-1*, which is depicted in Figure 4. We assume that $MB_{(k,l)}^{n-1}$ represents the macroblock at the k^{th} row and l^{th} column of frame *n-1* (the next frame to be displayed). $MB_{(k,l)}^{n-1}$ is defined as a backward macroblock

(BMB) if the macroblock of frame n having the same spatial position of $MB^{n-1}_{(k,l)}$, i.e. $MB^{n}_{(k,l)}$, is coded without motion compensation (non-MC macroblock). Otherwise, it is defined as a forward macroblock (FMB). For example, in Figure 4, since the motion vector in $MB^{n}_{(0,1)}$, $mv^{n}_{(0,1)}$, is zero, it means that $MB^{n}_{(0,1)}$ is a non-MC macroblock and the macroblock selector classifies $MB^{n-1}_{(0,1)}$ as BMB. On the other hand, since $MB^{n}_{(1,1)}$ is coded with motion compensation (MC-macroblock), $MB^{n-1}_{(1,1)}$ 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.

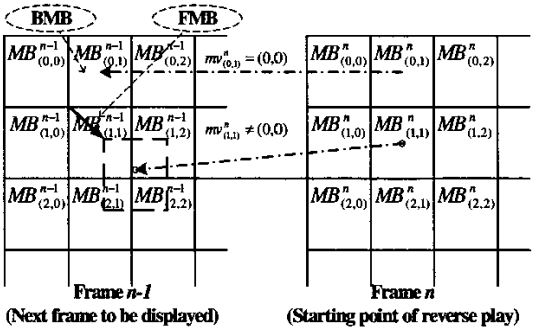


Figure 4. Definition of the forward macroblock (FMB) and the backward macroblock (BMB).

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 for current frame. 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 3, each macroblock in frame n, $MB^{n}_{(k,l)}$, is reconstructed according to motion-compensated prediction and it is given by

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

where $MCMB^{n-1}(mv^{n}_{(k,l)})$ stands for the motion-compensated macroblock of $MB^{n}_{(k,l)}$ which is translated by the motion vector $mv^{n}_{(k,l)}$ in the previous reconstructed frame $n-1$ and $e^{n}_{(k,l)}$ is the prediction error between $MB^{n}_{(k,l)}$ and its motion-compensated macroblock, $MCMB^{n-1}(mv^{n}_{(k,l)})$. 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 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 over the network and decoded by the decoder in the conventional video streaming system. It becomes impractical when GOP is large. However, if $MB^{n-1}_{(k,l)}$ 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 $MB^{n}_{(k,l)}$. Hence, for this specific case, $MCMB^{n-1}(mv^{n}_{(k,l)})$ is equal to $MB^{n-1}_{(k,l)}$, and equation (1) can be rewritten as

$$MB^{n-1}_{(k,l)} = MB^{n}_{(k,l)} + \tilde{e}^{n}_{(k,l)} \quad (2)$$

where $\tilde{e}^{n}_{(k,l)} = -e^{n}_{(k,l)}$. It is noted 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 (2) indicates that, for BMB, the only data that the server needs to send is the quantized DCT coefficients of $\tilde{e}^{n}_{(k,l)}$. In the following discussions, we will describe how to compute these quantized DCT coefficients of $\tilde{e}^{n}_{(k,l)}$ from the existing MPEG video stream in the server.

By applying the DCT for $\tilde{e}^{n}_{(k,l)}$ and considering that DCT is an odd transform, we obtain the expression of $\tilde{e}^{n}_{(k,l)}$ in the DCT-domain.

$$DCT(\tilde{e}^{n}_{(k,l)}) = -DCT(e^{n}_{(k,l)}) \quad (3)$$

Then the quantized DCT coefficients of $\tilde{e}^{n}_{(k,l)}$ are given by

$$Q[DCT(\tilde{e}^{n}_{(k,l)})] = -Q[DCT(e^{n}_{(k,l)})] \quad (4)$$

From equation (4), $Q[DCT(\tilde{e}^{n}_{(k,l)})]$ can be obtained by inverting the sign of all DCT coefficients in $Q[DCT(e^{n}_{(k,l)})]$, which can be extracted from the video stream in the server directly. $Q[DCT(\tilde{e}^{n}_{(k,l)})]$ is then transmitted to the client by switching SW_1 to B_1. At the client side, as shown in Figure 3, switch SW_2 is connected to B_2 so that all reconstructed BMBs are stored in BMB-FB which is an additional frame buffer in the client machine for BMB. Those reconstructed BMBs stored in BMB-FB are used for further composition of FMBs. From the above derivation, we can conclude that the server and the client only need to send and decode one macroblock for each BMB respectively.

For a real world image sequence, the block motion field is usually gentle, smooth, and varies slowly. As a consequence, the distribution of motion vector is center-biased [8-9], as demonstrated by the typical examples as shown in Table 1 which shows the distribution of BMB for various sequences including "Claire", "Grandma", "Salesman", "Carphone", "Table Tennis" and "Football".

These sequences have been selected to emphasize different amount of motion activities. It is clear that over 90% and 27% of the macroblocks are BMB for sequences containing low and high amount of motion activities respectively. By inverting the sign of all DCT coefficients in the server, the sequence containing more BMBs can alleviate the decoder complexity and network traffics more significantly.

Table 1. Percentage of BMB for various sequences.

Claire	Grandma	Salesman	Carphone	Table Tennis	Football
89.75	81.57	61.26	52.37	49.30	27.51

The situation of FMB is different. The sign inversion of DCT coefficients cannot be employed since $MCMB^{n-1}(mv^{n}_{(k,l)})$ is no longer equal to $MB^{n-1}_{(k,l)}$ and then equation (2) is not valid for FMB. In other words, $MB^{n-1}_{(k,l)}$ cannot be reconstructed from $MB^{n}_{(k,l)}$. In Figure 4, 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*. The required macroblocks in frame *n-2* further requires their related motion-compensated macroblocks in frame *n-3*. This process continues until the previous nearest I-frame. To reconstruct FMBs in frame *n-1*, switch SW_1 in the server is connected to C_1 and the macroblock selector extracts all the related macroblocks from the previous nearest I-frame to frame *n-2* and sends them to the client machine, as shown in Figure 3. In the client machine, all the switches are opened and the decoder decodes the necessary macroblocks in the forward order from the previous nearest I-frame to frame *n-2*. All the decoded macroblocks in frame *n-2*, which are referred by FMBs in frame *n-1*, are stored in FB. Afterwards, switches SW_3 and SW_4 are connected to B_3 and A_4 respectively. The switch positions of the proposed video streaming system are summarized in Table 2. During decoding FMBs in frame *n-1*, the prediction error between each FMB and its corresponding motion-compensated macroblock is decoded. Each reconstructed pixels in FMB can be obtained by adding its prediction errors to its motion-compensated pixels of frame *n-2* stored in FB. At the same time, the BMBs stored in BMB-FB are then composited with the reconstructed FMBs to form frame *n-1*, which is the desired frame to be displayed in the reverse-play operation. The frame *n-1* is then stored in both DB and FB. The frame in DB is used for displaying purpose. On the other hand, frame *n-1* stored in FB can be further used for reconstructing BMBs of the consequent frame, frame *n-2*, in the reverse playback.

4. Simulation Results

A series of computer simulations were conducted to evaluate the performances 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 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 frames/s.

Table 2. Switch positions for different VCR modes of the proposed system.

Playback modes	Macroblock type	SW_1	SW_2	SW_3	SW_4
Forward	—	A_1	A_2	A_3	A_4
Reverse	BMB	B_1	B_2	A_3	B_4
	FMB (nearest I-frame to frame *n-2*)	C_1	A_2	A_3	B_4
	FMB (frame *n-1*)	C_1	A_2	B_3	A_4

To verify the performances of the proposed system, extensive simulations were carried out. Results of the simulations are used to compare the performance of the conventional video streaming system mentioned in Section 2. The detailed comparisons of the 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" as shown in Table 4. 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 5 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 5, 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 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.

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. We also proposed to use a technique of sign inversion of DCT coefficients to simplify the decoder complexity while maintaining the low network bandwidth requirement. 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.

6. 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 studentships provided by the University.

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

[8] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," IEEE Trans. Circuits Syst. for Video Technol., vol. 8, pp. 369-377, Aug. 1998.

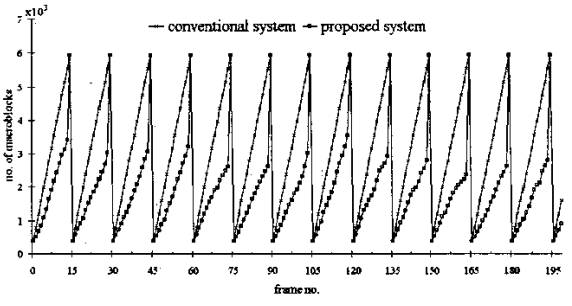
[9] K.T. Fung, Y.L Chan and W.C. Siu, "New architecture for dynamic frame-skipping transcoder," IEEE Trans. on Image Processing, vol. 11, no. 8, pp. 886-900, Aug. 2002.

Table 3. Detailed comparisons between the proposed system and the conventional system.

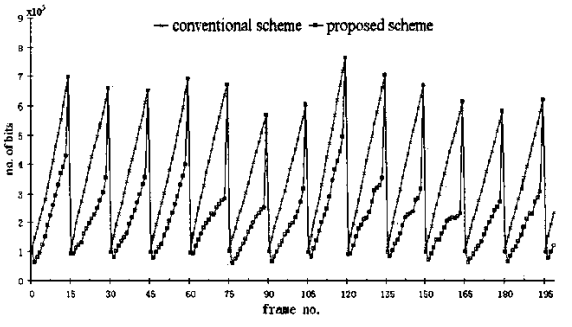
Sequences	Bitrate	Average no. of macroblocks to be decoded by the decoder		Average no. of bits to be sent over the network	
		Conventional System	Proposed System	Conventional System	Proposed System
Salesman	1.5M	3109	1866	373996	220677
Football	1.5M	2591	2024	368697	300760
Table Tennis	1.5M	2591	1644	368660	276529
Carphone	64K	777	554	9620	6763
Claire	64K	777	223	12035	2823
Grandma	64K	777	314	10455	2888

Table 4. 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
Salesman	1.5M	40.0 %	41.0 %
Football	1.5M	21.9 %	18.4 %
Table Tennis	1.5M	36.5 %	25.0 %
Carphone	64K	28.7 %	29.7 %
Claire	64K	71.3 %	76.5 %
Grandma	64K	59.60 %	72.4 %



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



(b) Number of bits transmitted over the network.

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