Efficient Reverse-Play Algorithms for MPEG Video With VCR Support

Chang-Hong Fu, Yui-Lam Chan, and Wan-Chi Siu, *Senior Member, IEEE*

***Abstract*—Reverse playback is the most common video cassette recording (VCR) function in digital video players and it involves playing video frames in reverse order. However, the predictive processing techniques employed in MPEG severely complicate the reverse-play operation. For displaying single frame during reverse playback, all frames from the previous I-frame to the requested frame must be sent by the server and decoded by the client machine. It requires much higher bandwidth of the network and complexity of the decoder. In this paper, we propose a compressed-domain approach for an efficient implementation of the MPEG video streaming system to provide reverse playback over a network with the minimal requirements on the network bandwidth and the decoder complexity. In the proposed video streaming server, it classifies macroblocks in the requested frame into two categories—backward macroblocks (BMBs) and forward macroblock (FMBs). Two novel MB-based techniques are used to manipulate the necessary MBs in the compressed domain and the server then sends the processed MBs to the client machine. For BMBs, we propose a sign inversion technique, which is operated in the variable length coding (VLC) domain, to reduce the number of MBs 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. The server also identifies the previous related MBs of FMBs and those related maroblocks coded without motion compensation are then processed by a technique of direction addition of discrete cosine transform (DCT) coefficients to further reduce the computational complexity of the client decoder. With the sign inversion and direct addition of DCT coefficients, the proposed architecture only manipulates MBs 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, 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.**

I. Introduction

DUE to the explosive growth of the Internet and increasing demand for multimedia contents, real-time multimedia services over the Internet has received tremendous attention from academia and industry [1]–[6]. Real-time multimedia implies that it has timing constraints. For example, video data must be played out continuously. If the data do not arrive in time, the playout process will pause, which is annoying to human eyes. Real-time transport of stored video is the predominant part of real-time multimedia. Transmission of stored video can be either "download mode" or "streaming mode." In the download mode, a user downloads the entire video file and then plays back the video file. However, full file transfer in the download mode usually suffers long and perhaps unacceptable transfer time. In contrast, the video data need not be fully downloaded in the streaming mode, but is being played out while parts of the contents are being received and decoded. Due to the real-time nature, video streaming technology plays an important role in media delivery. The realization of a video streaming system has several challenges, such as the high storage-capacity and throughput in the video server and the high bandwidth in the network to deliver video streams. Recent advances in computing technology, compression standards [7]–[10], high-band storage devices, and high speed-networks have made it feasible to provide video streaming applications.

With the rapid growth of online multimedia contents, it is also highly desirable that video streaming systems should have the capability of providing fast and effective browsing. However, current compression standards such as MPEG [8]–[10] were developed primarily for broadcast applications. In order to complete the transition to digital video from its current analog state, MPEG technology needs to encompass not just compression and streaming methodologies but also a video-processing framework. This will allow MPEG to be usable not just for the purposes of efficient storage and transmission of digital video, but also for systems wherein the user needs to interact with the digital video. A key technique that enables fast and user-friendly browsing of video content is to provide full video cassette recording (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 video browsing completely and it is also useful for video editing.

However, MPEG video coding standards were mainly designed for forward-play operations and the predictive processing techniques employed in MPEG [11]–[13] severely complicate reverse-play operations. For uncompressed video, the solution for reverse playback is simple; it just reorders the video frame data in reverse order. The simplicity of this solution relies on two properties: the data for each video frame is self-contained and it is independent of its placement in the data stream. These properties typically do not hold true for MPEG video data because MPEG compression uses predictive processing techniques that are not invariant to changes in frame

Manuscript received July 31, 2003; revised June 26, 2005. This work was supported in part by the Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, Hong Kong Polytechnic University and in part by Grant PolyU 5216/03E from the Research Grants Council of the Hong Kong Special Administrative Region. The work of C.-H. Fu was supported in part by research studentships provided by the University. This paper was recommended by Associate Editor S. Chen.

The authors are with the Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University Kowloon, Hong Kong (e-mail: 02900218r@polyu.edu.hk; enylchan@polyu.edu.hk; enwcsiu@polyu.edu.hk).

Digital Object Identifier 10.1109/TCSVT.2005.856901

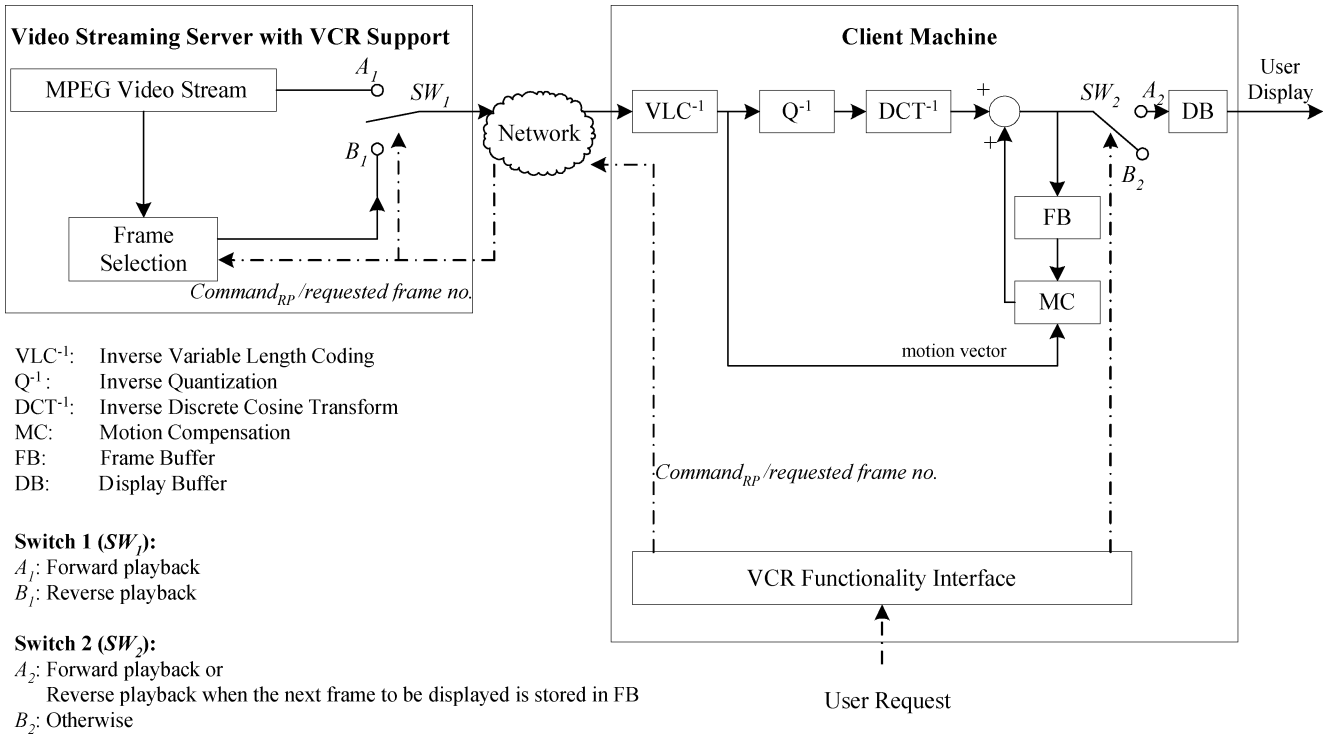


Fig. 1. Architecture of the video streaming system with VCR support.

order. In other words, simply reversing the order of the input frame data will not reverse the order of the decoded video frames. For example, with simple I-P structure of MPEG encoded sequence, to decode a P-frame, the previously encoded I-/P-frames need to be transmitted and decoded first. One straightforward approach to implement a reverse-play operation of MPEG compressed video is to decode all the frames in the whole group of pictures (GOP), store all frames in a large buffer of the decoder, and play the decoded frames reversely. However, this approach will require a significant amount of memory in the client side and it is not desirable. 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 a significant amount of 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.

Recently, some works on the implementation of reverse playback for MPEG compressed video in streaming applications have been introduced [14], [16], [17]. Chen and Kandlur [14] 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. After the frame conversion and frame re-ordering, the motion-vector reversing approach developed in [15] could be used for reverse playback of the new I-B stream. 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. Besides, the method is no longer workable in the new H.264 standard where B-frames are also possible to act as references. Wee and Vasudev [16] described a reverse-play transcoder which is used to convert the I-P-frames into another I-P bitstream with 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 [15] 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 [16]. Lin *et al.* [17] recently proposed to store the forward-encoded bitstream as well as the backward-encoded bitstream in the server to simplify the reverse-play complexity while maintaining the low network bandwidth requirement. However, the storage requirement of the server will be approximately doubled.

In this paper, we will explore ways for an efficient implementation of the MPEG streaming video system to provide a reverse-play mode of VCR functionality with minimal requirements on the network bandwidth and the decoder complexity. By using the motion information of the compressed bitstream, we propose a novel macroblock (MB) selection scheme at the server to minimize the required network bandwidth and the decoder complexity. The proposed scheme can adaptively select the necessary MBs, manipulate them in the compressed domain and send the processed macraoblocks to the client. Since the proposed scheme mainly operates on the compressed domain, complete decoding and encoding are not required in the server. Thus, an additional processing requirement in the server can be minimized.

The organization of this paper is as follows. Section II of this paper presents an in-depth study of the impacts of reverse playback on decoder complexity and network traffics. The proposed efficient reverse-play algorithms for MPEG video streaming are

then described in Section III. Simulation results are presented in Section IV. Finally, some concluding remarks are provided in Section V.

II. Impacts of Reverse Playback on Decoder Complexity and Network Traffics

Fig. 1 shows a system architecture for video streaming with VCR support. The video stream is pre-compressed using the MPEG-2 video coding standard [10], [18] and then stored in the storage devices. Upon the client request, a streaming server retrieves compressed video data from storage devices and the user can view the video while the video is being streamed over the network. To support VCR services, considerable functionality must be built into the client machine which consists of an MPEG decoder and an interface for VCR functionality. The MPEG decoder is used to decode incoming video stream and deliver it to the appropriate display. The VCR interface then translates user interactions from the remote control or keyboard to appropriate signals for network transfer. It interprets the user commands and forward them to the decoder and server for appropriate actions.

In the following, we provide some discussions and simulation results to show the average number of frames/bits needed to be sent through the network and decoded at the client decoder to support the forward-play and reverse-play operations. 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 discussions on the case that the video stream contains I- and P-frames only. In Fig. 1, there are two switches SW_1 and SW_2 in the server and client machine, respectively. They are used to adapt various VCR operations. In the forward-play operation, switches SW_1 and SW_2 are connected to A_1 and A_2, respectively, as shown in Fig. 1. The server sends the video stream frame-by-frame in MPEG encoding order. The client machine then decodes the incoming video stream and displays the video frames on the client display. If a user issues a reverse-play command at frame n, the next frame to be display is frame $n-1$. The client machine generates the reverse-play command ($\mathrm{Command}_{\mathrm{RP}}$) and the 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 Fig. 2. 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 selects frame 0 to frame $n-1$ from the video stream by switching SW_1 to B_1, as shown in Fig. 1. At the client side, frame 0 to frame $n-2$ do not need to be displayed and only frame $n-1$ should be displayed on the user screen. As a consequence, SW_2 in the client machine is switched to B_2 during the decoding of frame 0 to frame $n-2$. Afterwards, frame $n-1$ is decoded and stored into the display buffer so that this frame is displayed on the client screen. At that instant, SW_2 is connected to A_2.

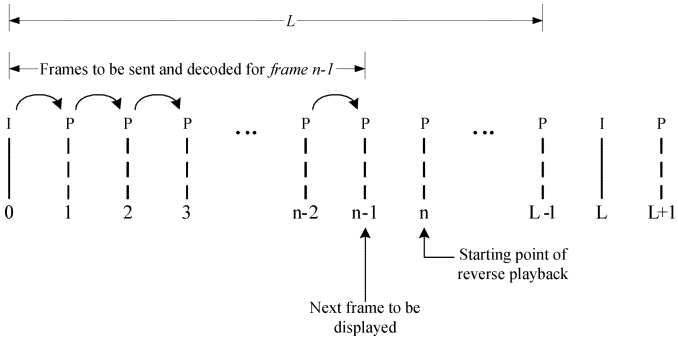


Fig. 2. Example of reverse playback.

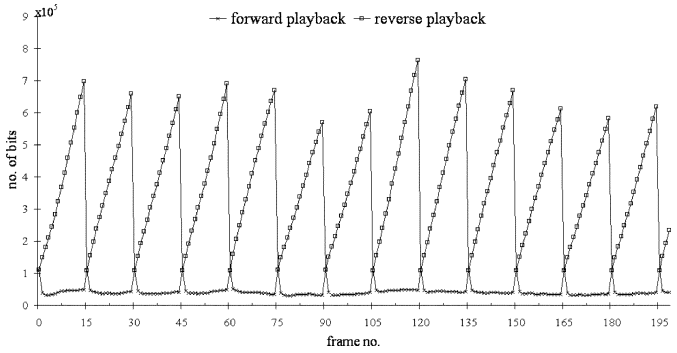


Fig. 3. Bandwidth requirement for sending the "Salesman" sequence over network with respect to forward-play and reverse-play operations.

Now we show the impact of reverse playback on the decoding complexity and network traffics. When a reverse-play operation is requested, it always lasts for few seconds or minutes. During the period of reverse playback, a number of whole GOP need to be sent over the network and decoded by the client decoder. It is useful to derive a formula to compute the number of frames to be sent and decoded for playing the whole GOP in reverse order. Suppose that all the GOPs in the video stream have the same length L, the server needs to send L frames for frame $L-1$, $L-1$ frames for frame $L-2$ and so on. Thus, the total number of frames to be sent for playing one GOP (N_{RP}) reversely is

$$N_{\mathrm{RP}} = \frac{L(L+1)}{2} \tag{1}$$

and the average number of frames to be sent for each frame $\overline{N}_{\mathrm{RP}}$ is

$$\overline{N}_{\mathrm{RP}} = \frac{L+1}{2}. \tag{2}$$

Equation (2) indicates that, $\overline{N}_{\mathrm{RP}}$ will grow almost linearly as L increases. If L is large, it leads to significant increase of the decoding complexity and network traffics. The bandwidth requirement of the reverse-play and forward-play operations are depicted in Fig. 3 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 GOP length is 15 with an I-P structure. The starting point of the reverse-play operation is at the end of the sequence. This figure shows that the server needs to send an excessively large amount of extra bits to the decoder to display

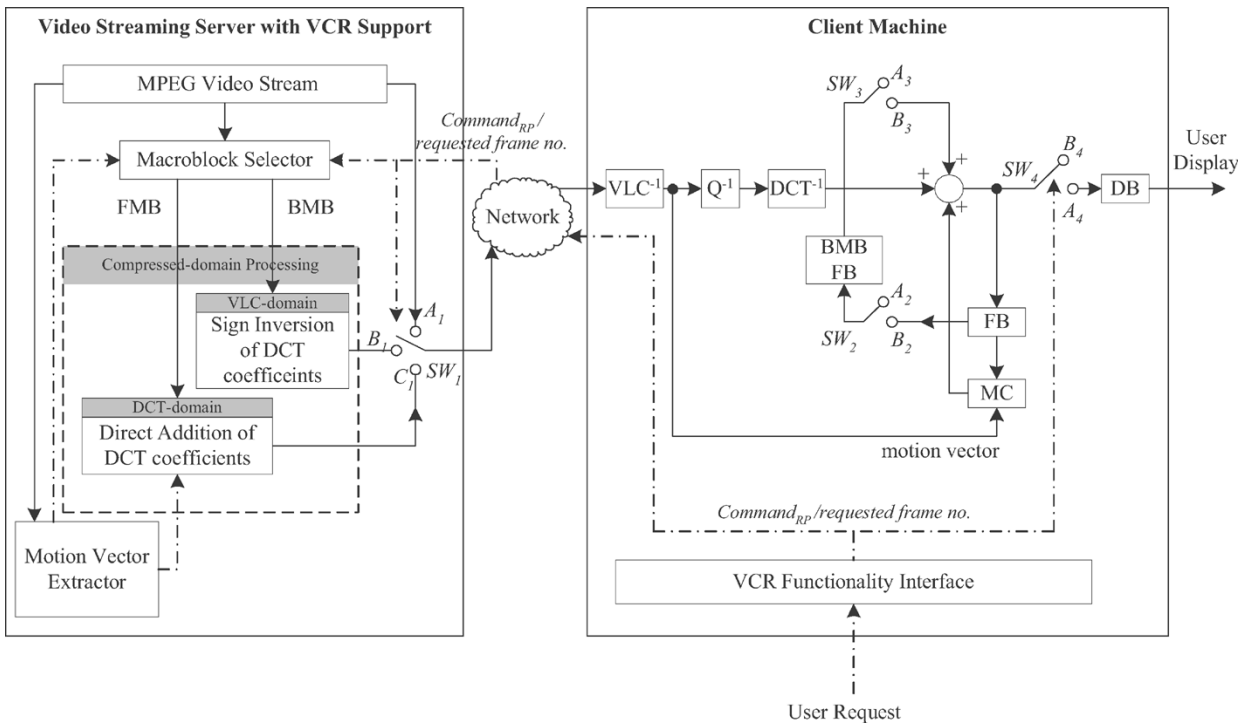


Fig. 4. Proposed architecture for the video streaming system with VCR functionality.

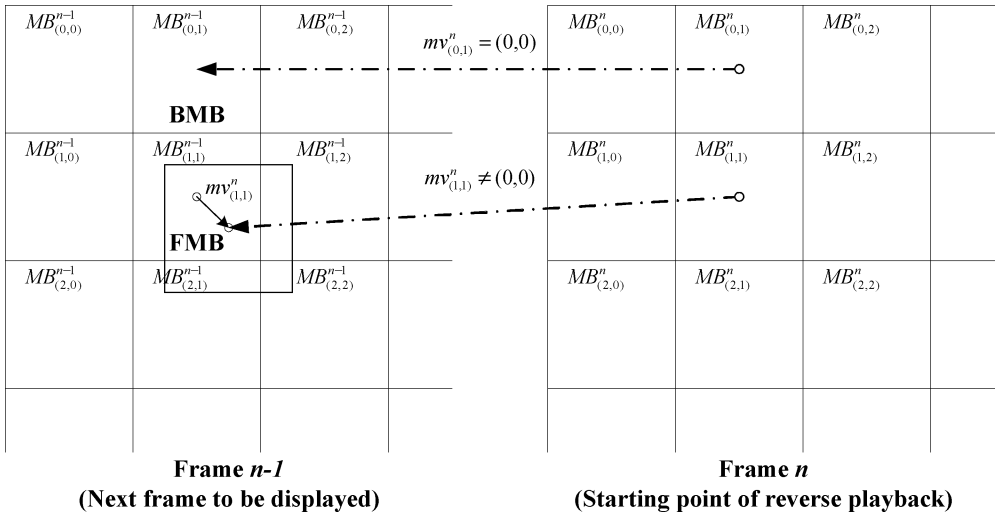


Fig. 5. Definition of the FMB and the BMB.

one frame during reverse playback. 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.

III. MB-Based Reverse-Play Algorithms for Video Streaming System With VCR Functionality

To resolve the above problem, we consider in this paper a MB-based solution for providing reverse-play service of a video streaming system. The architecture of the proposed system is shown in Fig. 4. For simplicity of the presentation, in this paper, we again use an example in which the video is coded in I- and P-frames only. The extension of our discussion to the case with the general I-B-P GOP structure is straightforward. In the forward-play operation, the proposed architecture is the same as the architecture mentioned in Section II in which 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 MB-based scheme is proposed to be used in the reverse-play operation. At the server side of Fig. 4, motion vectors are extracted from the MPEG video stream and these motion vectors are used by a MB selector to identify the types of MBs. Two types of MBs are now defined. For illustration, we use the example in Section II again, assume that a user requests a reverse-play command at frame n, the next frame to be display is frame $n-1$. The situation in MB level is depicted in Fig. 5. We assume that $\mathrm{MB}_{(k,l)}^{n-1}$ represents

TABLE I
PERCENTAGE OF BMB FOR VARIOUS SEQUENCES

Claire	Grandma	Salesman	Carphone	Table Tennis	Foreman	Football
89.75	81.57	61.26	52.37	49.30	43.59	27.51

the MB at the kth row and lth column of frame $n-1$ (the next frame to be displayed). $\mathrm{MB}_{(k,l)}^{n-1}$ is defined as a backward MB (BMB) if the MB in frame n having the same spatial position of $\mathrm{MB}_{(k,l)}^{n-1}$, i.e., $\mathrm{MB}_{(k,l)}^{n}$, is coded without motion compensation (non-MC MB). Otherwise, it is defined as a forward MB (FMB). For example, in Fig. 5, since the motion vector of $\mathrm{MB}_{(0,1)}^{n}$, $mv_{(0,1)}^{n}$, is zero, it means that $\mathrm{MB}_{(0,1)}^{n}$ is a non-MC MB and the MB selector classifies $\mathrm{MB}_{(0,1)}^{n-1}$ as BMB. On the other hand, since $\mathrm{MB}_{(1,1)}^{n}$ is coded with motion compensation (MC-MB), $\mathrm{MB}_{(1,1)}^{n-1}$ is classified as FMB. In this paper, our scheme works at the level of MBs. The server will process each type of MBs in the compressed domain such that a complete decoding and encoding are not required at the server. As it will be described in detail later, for BMB, the server will use only MPEG data from the future frame while FMB needs MPEG data from the past frame. Our contributions lie in the following two areas: 1) a sign inversion of the dicrete cosine transform (DCT) coefficients which can be operated on the variable length coding (VLC) domain for BMB and 2) a direct addition of the DCT coefficients which can be operated in the DCT domain for FMB.

A. VLC-Domain Technique for BMBs

Block MC prediction (MCP) is the type of prediction used in MPEG standards [8]–[10]. This prediction type is what gives the MPEG codecs the advantage over pure still-frame coding methods. In MC prediction, the previously transmitted and decoded frame serves as the prediction for the 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 Fig. 4, each MB in frame n, $\mathrm{MB}_{(k,l)}^{n}$, is reconstructed according to MC prediction and it is given by

$$\mathrm{MB}_{(k,l)}^{n} = \mathrm{MCMB}^{n-1}(mv_{(k,l)}^{n}) + e_{(k,l)}^{n} \tag{3}$$

where $\mathrm{MCMB}^{n-1}(mv_{(k,l)}^{n})$ stands for the MC MB of $\mathrm{MB}_{(k,l)}^{n}$ which is translated by the motion vector $mv_{(k,l)}^{n}$ in the previous reconstructed frame $n-1$ and $e_{(k,l)}^{n}$ is the prediction error between $\mathrm{MB}_{(k,l)}^{n}$ and its MC MB, $\mathrm{MCMB}^{n-1}(mv_{(k,l)}^{n})$. Frame n is then stored in the frame buffer (FB) as it is used 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 $\mathrm{MB}_{(k,l)}^{n-1}$ in frame $n-1$ are requested. To reconstruct each $\mathrm{MB}_{(k,l)}^{n-1}$, all the related previous MBs 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 the GOP size is large. However, if $\mathrm{MB}_{(k,l)}^{n-1}$ is found to be a BMB, its corresponding MB in frame n, $\mathrm{MB}_{(k,l)}^{n}$, is coded without MC. It means that the spatial position of $\mathrm{MB}_{(k,l)}^{n-1}$ is the same as that of $\mathrm{MB}_{(k,l)}^{n}$. Hence, for this specific case, $\mathrm{MCMB}^{n-1}(mv_{(k,l)}^{n})$ is equal to $\mathrm{MB}_{(k,l)}^{n-1}$, and (3) can be rewritten as

$$\mathrm{MB}_{(k,l)}^{n-1} = \mathrm{MB}_{(k,l)}^{n} + \tilde{e}_{(k,l)}^{n} \tag{4}$$

where $\tilde{e}_{(k,l)}^{n} = -e_{(k,l)}^{n}$. 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 $\mathrm{MB}_{(k,l)}^{n}$ are available at the decoder. To reconstruct $\mathrm{MB}_{(k,l)}^{n-1}$ in the reverse-play operation, (4) indicates that, for a BMB, the only data that the server needs to send is the quantized DCT coefficients of $\tilde{e}_{(k,l)}^{n}$. In the following discussions, we will describe how to compute these quantized DCT coefficients of $\tilde{e}_{(k,l)}^{n}$ from the existing MPEG video stream in the server.

By applying the DCT to $\tilde{e}_{(k,l)}^{n}$ and considering that the DCT is an odd transform, we can find the DCT of $\tilde{e}_{(k,l)}^{n}$ in the DCT domain, as indicated below

$$\mathrm{DCT}(\tilde{e}_{(k,l)}^{n}) = -\mathrm{DCT}(e_{(k,l)}^{n}). \tag{5}$$

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

$$Q[\mathrm{DCT}(\tilde{e}_{(k,l)}^{n})] = -Q[\mathrm{DCT}(e_{(k,l)}^{n})]. \tag{6}$$

From (6), $Q[\mathrm{DCT}(\tilde{e}_{(k,l)}^{n})]$ can be obtained by inverting the sign of all DCT coefficients in $Q[\mathrm{DCT}(e_{(k,l)}^{n})]$, which can be directly extracted from the video stream in the server. $Q[\mathrm{DCT}(\tilde{e}_{(k,l)}^{n})]$ is then transmitted to the client by switching SW_1 to B_1. At the client side, as shown in Fig. 4, 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 reverse playback. The 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 MB 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 [19]–[21], as demonstrated by the typical examples as shown in Table I which shows the distribution of BMB for various sequences including "Claire," "Grandma," "Salesman," "Carphone," "Foreman," "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 MBs 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 significantly.

In the server, the sign inversion of DCT coefficients requires additional variable length decoding and re-encoding. To reduce the computational load of the server, we propose to compute the newly quantized DCT coefficients $Q[\mathrm{DCT}(\tilde{e}_{(k,l)}^{n})]$ in the

TABLE II
VLC TABLE FOR RUN-LEVEL COMBINATIONS. THE SIGN BIT "S" IS "0" FOR POSITIVE AND "1" FOR NEGATIVE

Variable length codes	Run	Level
10	End of Block	
11 s	0	1
011 s	1	1
0100 s	0	2
0101 s	2	1
0010 1s	0	3
:	:	:
:	:	:
0000 101 s	9	1
0000 01	Escape	
0010 0110 s	0	5
:	:	:

VLC domain. In MPEG video encoding, DCT coefficients representing high spatial frequencies are almost always zero, whereas low-frequency coefficients are often nonzero. To exploit this behavior, the DCT coefficients are arranged qualitatively from low to high spatial frequency following zigzag scan order. This zigzag scan approximately orders the coefficients according to their probability of being zero. Even with zigzag ordering, many DCT coefficients are zero in a typical 8×8 block. In this case, better coding efficiency is obtained when codewords are defined by combining the length of the zero coefficient run with the amplitude of the nonzero coefficient terminating the run.

Each nonzero sequence of DCT coefficients is then coded in the RUN-LEVEL symbol structure with different variable length codes (VLCs). RUN refers to the number of zero coefficients before the next nonzero coefficient; LEVEL refers to the amplitude of the nonzero coefficient. Table II illustrates this. The trailing bit of each VLC is the "s" bit that codes the sign of the nonzero coefficient. If "s" is 0, the coefficient is positive; otherwise, it is negative. To convert $Q[\mathrm{DCT}(\tilde{e}^n_{(k,l)})]$ from $Q[\mathrm{DCT}(e^n_{(k,l)})]$, the server just parses the MPEG video bitstream and inverts all 's' bits of VLCs in BMB, as shown in Fig. 6. On the other hand, RUN-LEVEL combinations that are not in the Table II 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. The FLCs for RUN and LEVEL are shown in Table III. 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 Fig. 6. Since it is not necessary to perform VLC encoding, MC, DCT, quantization, inverse DCT, inverse quantization, and VLC decoding in the server, the loading of the server is reduced significantly.

B. DCT-Domain Technique for FMBs

The situation of FMB is different. The bit manipulation of VLCs mentioned in Section III.A cannot be employed since $\mathrm{MCMB}^{n-1}(mv^n_{(k,l)})$ is no longer equal to $\mathrm{MB}^{n-1}_{(k,l)}$ and then (4) does not hold true for FMB. In other words, $\mathrm{MB}^{n-1}_{(k,l)}$ cannot be reconstructed from $\mathrm{MB}^n_{(k,l)}$. In Fig. 7, a situation in which there are two FMBs in frame $n-1$ is illustrated. Three MBs (shaded MBs in Fig. 7) in frame $n-2$ are required to act as references for performing MC of these two FMBs. Those macrobocks in frame $n-2$ further requires their corresponding MBs

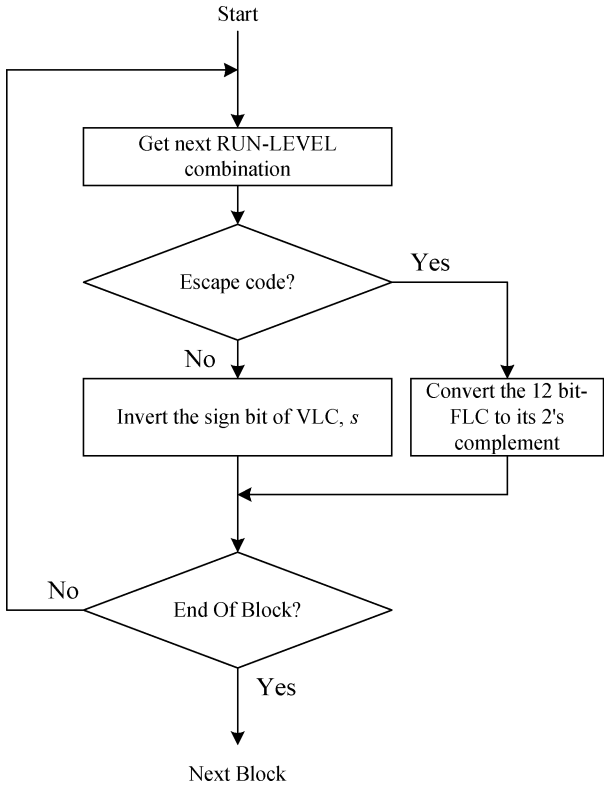


Fig. 6. Execution flow of the server during bit manipulation of VLCs in BMB.

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 MB selector extracts all the related MBs from the previous nearest I-frame to frame $n-2$ and sends them to the client machine, as shown in Fig. 4. In the client machine, all the switches are opened and the decoder decodes the necessary MBs in forward order from the previous nearest I-frame to frame $n-2$. All the decoded MBs 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 IV. During decoding FMBs in frame $n-1$, the prediction error between each FMB and its corresponding MC MB is decoded. Each reconstructed pixel in FMB can be obtained by adding its prediction errors to its MC pixels of frame $n-2$ in FB, as shown in Fig. 4. 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. Frame $n-1$ is then stored in both the display buffer (DB) and FB. The frame in DB is used for display 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-play operation.

To further reduce the decoding complexity and network traffics in processing FMBs, we propose to use a technique involving direct addition of DCT coefficients. This technique is originally designed for the frame-skipping video transcoder which is mainly performed on the DCT domain to achieve a transcoder with low complexity [22]–[24]. Now, we borrow this idea for FMB when it is coded without MC (non-MC FMB)

TABLE III
FLC TABLE FOR RUNS AND LEVELS. IT IS USED FOLLOWING THE ESCAPE CODE OF A VLC

Fixed Length codes	Run
0000 00	0
0000 01	1
0000 10	2
:	:
:	:
:	:
:	:
:	:
:	:
:	:
1111 11	63

Fixed Length codes	Signed_level
1000 0000 0001	-2047
1000 0000 0010	-2046
:	:
1111 1111 1111	-1
0000 0000 0000	Forbidden
0000 0000 0001	+1
: :	: :
0111 1111 1111	+2047

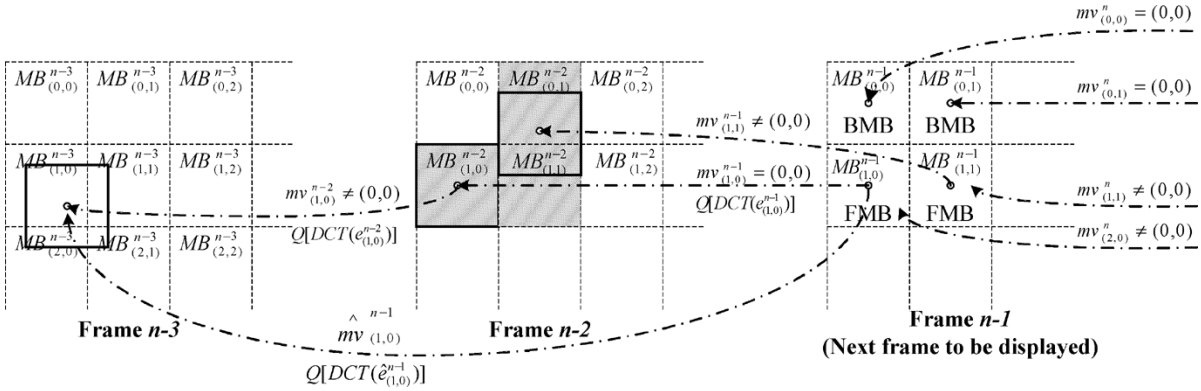


Fig. 7. Situation in which there are two FMBs in frame $n-1$ is illustrated.

and further extend it to alleviate the computational burden of the client decoder in the reverse-play operation. In Fig. 7, $\mathrm{MB}^{n-1}_{(1,0)}$ is a non-MC FMB since its motion vector, $mv^{n-1}_{(1,0)}$, is equal to zero. Note that if $mv^{n}_{(1,0)}$ is also equal to zero, $\mathrm{MB}^{n-1}_{(1,0)}$ is a BMB instead. In general, a MB in frame $n-1$, $\mathrm{MB}^{n-1}_{(k,l)}$, is treated as non-MC FMB when $mv^{n}_{(k,l)} \neq (0,0)$ and $mv^{n-1}_{(k,l)} = (0,0)$.

In Fig. 8, a situation in which $\mathrm{MB}^{n-1}_{(k,l)}$ is a non-MC FMB is illustrated. To reconstruct this non-MC FMB, the decoder needs to decode the prediction errors $e^{n-1}_{(k,l)}$ in frame $n-1$, $e^{n-2}_{(k,l)}$ in frame $n-2$, and so on. If pixels in $\mathrm{MB}^{n-2}_{(k,l)}$ are used as reference for $\mathrm{MB}^{n-1}_{(k,l)}$ only, it is too wasteful that the client machine to decode $e^{n-2}_{(k,l)}$ for reconstructing $\mathrm{MB}^{n-2}_{(k,l)}$. 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 MB for the non-MC FMB. The required number of MBs, 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 $\mathrm{MB}^{n-2}_{(k,l)}$ 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, $Q[\mathrm{DCT}(e^{n-1}_{(k,l)})]$, 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 $\widehat{mv}^{n-1}_{(k,l)}$ and prediction errors in the DCT domain, $Q[\mathrm{DCT}(\hat{e}^{n-1}_{(k,l)})]$, by using frame $n-3$ as a reference, as shown in Fig. 8. Since $mv^{n-1}_{(k,l)}$ is zero, we have

$$\widehat{mv}^{n-1}_{(k,l)} = mv^{n-2}_{(k,l)}. \tag{7}$$

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

$$\begin{aligned} &Q[\mathrm{DCT}(\hat{e}^{n-1}_{(k,l)})] \\ &= Q[\mathrm{DCT}(\mathrm{MB}^{n-1}_{(k,l)} - \mathrm{MCMB}^{n-3}(\widehat{mv}^{n-1}_{(k,l)}))] \\ &= Q[\mathrm{DCT}(\mathrm{MB}^{n-1}_{(k,l)} - \mathrm{MCMB}^{n-3}(mv^{n-2}_{(k,l)}))]. \end{aligned} \tag{8}$$

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 [22]. In the proposed video server, we employ a DCT-

TABLE IV
SWITCH POSITIONS OF THE PROPOSED VIDEO STREAMING 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

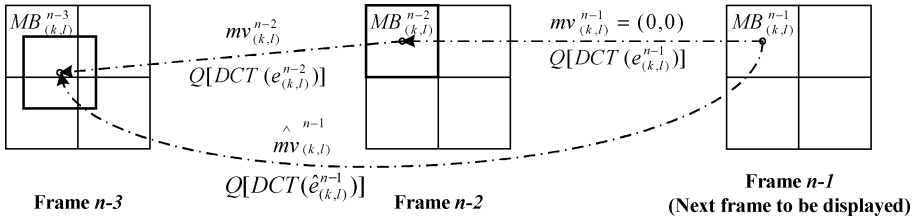


Fig. 8. Direct addition of DCT coefficients.

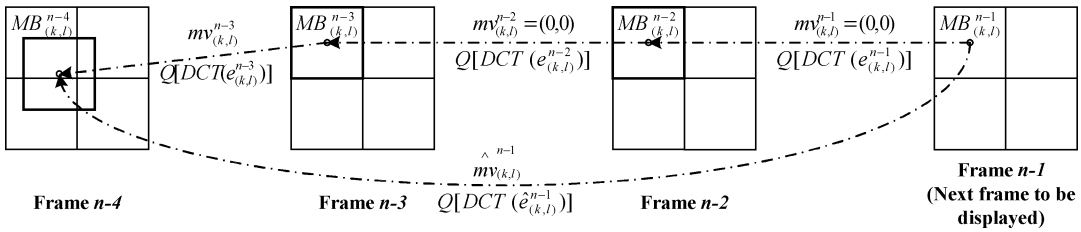


Fig. 9. Using direct addition of DCT coefficients iteratively.

domain technique to compute the new $Q[\mathrm{DCT}(\hat{e}^{n-1}_{(k,l)})]$. Using (3), $\mathrm{MB}^{n-1}_{(k,l)}$ can be written as

$$\mathrm{MB}^{n-1}_{(k,l)} = \mathrm{MCMB}^{n-2}(mv^{n-1}_{(k,l)}) + e^{n-1}_{(k,l)}. \quad (9)$$

If $\mathrm{MB}^{n-1}_{(k,l)}$ is coded without MC, $mv^{n-1}_{(k,l)}$ is zero and $\mathrm{MCMB}^{n-2}(mv^{n-1}_{(k,l)})$ is equal to $\mathrm{MB}^{n-2}_{(k,l)}$. Equation (9) can be simplified to (10)

$$\mathrm{MB}^{n-1}_{(k,l)} = \mathrm{MB}^{n-2}_{(k,l)} + e^{n-1}_{(k,l)}. \quad (10)$$

Similarly

$$\mathrm{MB}^{n-2}_{(k,l)} = \mathrm{MCMB}^{n-3}(mv^{n-2}_{(k,l)}) + e^{n-2}_{(k,l)}. \quad (11)$$

Substituting (11) into (10), we obtain

$$\mathrm{MB}^{n-1}_{(k,l)} - \mathrm{MCMB}^{n-3}(mv^{n-2}_{(k,l)}) = e^{n-1}_{(k,l)} + e^{n-2}_{(k,l)}. \quad (12)$$

Using (8) and (12), the newly quantized DCT coefficients of the prediction error between the current non-MC FMB and its corresponding reference MB in frame $n-3$, $Q[\mathrm{DCT}(\hat{e}^{n-1}_{(k,l)})]$, can be written as

$$Q[\mathrm{DCT}(\hat{e}^{n-1}_{(k,l)})] = Q[\mathrm{DCT}(e^{n-1}_{(k,l)} + e^{n-2}_{(k,l)})]. \quad (13)$$

Taking into account the linearity of DCT, (13) becomes

$$Q[\mathrm{DCT}(\hat{e}^{n-1}_{(k,l)})] = Q[\mathrm{DCT}(e^{n-1}_{(k,l)}) + \mathrm{DCT}(e^{n-2}_{(k,l)})]. \quad (14)$$

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

$$Q[\mathrm{DCT}(\hat{e}^{n-1}_{(k,l)})] = Q[\mathrm{DCT}(e^{n-1}_{(k,l)})] + Q[\mathrm{DCT}(e^{n-2}_{(k,l)})]. \quad (15)$$

Equation (15) implies that the newly quantized DCT coefficient $Q[\mathrm{DCT}(\hat{e}^{n-1}_{(k,l)})]$ can be computed in the DCT domain by adding $Q[\mathrm{DCT}(e^{n-1}_{(k,l)})]$ and $Q[\mathrm{DCT}(e^{n-2}_{(k,l)})]$. 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 the scheme, we use the example in Fig. 7 again. Instead of transmitting and decoding $Q[\mathrm{DCT}(e^{n-1}_{(1,0)})]$ and $Q[\mathrm{DCT}(e^{n-2}_{(1,0)})]$ for $\mathrm{MB}^{n-1}_{(1,0)}$, by using the direct addition of DCT coefficients, the server only needs to send $Q[\mathrm{DCT}(\hat{e}^{n-1}_{(1,0)})]$ and only one MB requires to be decoded in the client machine. The DCT-domain technique can thus minimize the computational complexity of the decoder. Furthermore, the direct addition of DCT coefficients can be computed iteratively if $mv^{n-2}_{(k,l)}$ is also equal to zero and pixels in $\mathrm{MB}^{n-3}_{(k,l)}$ are used as reference for $\mathrm{MB}^{n-1}_{(k,l)}$ only, as depicted in Fig. 9.

IV. Simulation Results

In this section, we present some simulation results. Computer simulations were conducted to evaluate the performances of the proposed techniques including the sign inversion for backward MBs and the direct addition for forward MBs when applied to the video streaming system with VCR support. MPEG-2 encoder [18] was employed to encode various video sequences with different spatial resolutions and motion characteristics. All the test sequences have a length of 200 frames. "Claire," "Grandma," and "Carphone" are typical videophone sequences in QCIF (176×144 pixels) format. "Salesman," "Table Tennis," and "Football" are in either CIF (352×288 pixels) format or SIF (352×240 pixels) format. All these sequences were encoded at two different bitrates. 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.

To verify the performances of the proposed techniques, extensive simulations were carried out. First, we have simulated the situation of the I-P structure with $L = 15$. Results of the simulations are used to compare the performance of the conventional video streaming system mentioned in Section II. Two different versions of our proposed streaming systems have been realized,

TABLE V
DETAILED COMPARISONS BETWEEN THE PROPOSED SYSTEMS, *SI* AND *SI+DA*, 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	*SI*	*SI+DA*	Conventional System	*SI*	*SI+DA*
Salesman	0.75M	3109	1866	1532	194384	114909	101366
(*352×288*)	1.5M	3109	1866	1532	373996	220677	188546
Football	1.5M	2591	2024	1864	368697	300759	284381
(*352×240*)	3M	2591	2024	1864	815827	656238	613946
Table Tennis	1.5M	2591	1644	1371	368660	276529	252640
(*352×240*)	3M	2591	1644	1371	747080	550476	496382
Foreman	64K	777	581	481	10288	7292	7017
(*176×144*)	128K	777	581	481	27015	19383	17491
Carphone	64K	777	554	421	9620	6763	6478
(*176×144*)	128K	777	554	421	26122	19048	16782
Claire	64K	777	223	124	12035	2823	2435
(*176×144*)	128K	777	223	124	31481	9034	6774
Grandma	64K	777	314	212	10455	2888	2562
(*176×144*)	128K	777	314	212	30608	8646	6488

TABLE VI
PERFORMANCE IMPROVEMENT OF THE PROPOSED SYSTEMS, *SI* AND *SI+DA*, OVER THE CONVENTIONAL SYSTEM IN TERMS OF THE NUMBER OF MACROBLOCKS TO BE DECODED BY THE DECODER

Sequences	Bitrate	Saving of macroblocks to be decoded by the decoder	
		SI	*SI+DA*
Salesman	0.75M	39.98%	50.73%
(*352×288*)	1.5M	39.98%	50.73%
Football	1.5M	21.88%	28.03%
(*352×240*)	3M	21.88%	28.03%
Table Tennis	1.5M	36.53%	47.06%
(*352×240*)	3M	36.53%	47.06%
Foreman	64K	25.20%	38.12%
(*176×144*)	128K	25.20%	38.12%
Carphone	64K	28.74%	45.80%
(*176×144*)	128K	28.74%	45.80%
Claire	64K	71.30%	84.01%
(*176×144*)	128K	71.30%	84.01%
Grandma	64K	59.59%	72.79%
(*176×144*)	128K	59.59%	72.79%

TABLE VII
PERFORMANCE IMPROVEMENT OF THE PROPOSED SYSTEMS, *SI* AND *SI+DA*, OVER THE CONVENTIONAL SYSTEM IN TERMS OF THE NUMBER OF BITS TO BE SENT OVER THE NETWORK

Sequences	Bitrate	Saving of bits to be sent over the network	
		SI	*SI+DA*
Salesman	0.75M	40.89%	47.85%
(*352×288*)	1.5M	40.99%	49.59%
Football	1.5M	18.43%	22.87%
(*352×240*)	3M	19.56%	24.75%
Table Tennis	1.5M	24.99%	31.47%
(*352×240*)	3M	26.32%	33.56%
Foreman	64K	29.12%	31.79%
(*176×144*)	128K	28.25%	35.25%
Carphone	64K	29.69%	32.66%
(*176×144*)	128K	27.08%	35.75%
Claire	64K	76.54%	79.77%
(*176×144*)	128K	71.30%	78.48%
Grandma	64K	72.37%	75.49%
(*176×144*)	128K	71.75%	78.80%

and let us call them *SI* and *SI+DA*. *SI* uses the technique of sign inversion of DCT coefficients for BMBs while *SI+DA* employs the technique of direction addition of DCT coefficients for FMBs as well. For our implementation, the operation of the DCT-domain technique used in *SI+DA* is restricted to 16-bits. The detailed comparisons of the average number of MBs to be decoded and bits to be sent are tabulated in Table V. The average number of MBs sent for decoding is directly proportional to the decoder complexity. In Table V, we show that *SI* outperforms the conventional approach in all sequences. The results are more significant for the sequences "Claire," "Grandma," and "Salesman" as shown in Tables VI and VII. The savings in both the bits to be sent over the network and MBs to be decoded by the decoder are from 40%–75% for these sequences. In other words, the decoder complexity for playing those sequences in reverse order is reduced by 40%–75%. It is due to the reason that these sequences contain more BMBs in which the technique of sign inversion can be employed. For sequences containing high motion activities such as "Football," "Table Tennis," "Foreman," and "Carphone", there still has a saving of 20%–40%.

To further reduce the number of macrblocks to be decoded and bits to be sent, *SI* can work with the technique of direct addition of DCT coefficients, *SI+DA*, which combines several non-MC MBs into one for FMB. Tables V–VII show that by using the technique of direction addition of DCT coefficients, *SI+DA* produces further savings in terms of both number of MBs to be decoded and bits to be sent as compared with that of *SI*. Note that the number of MBs requested by the decoder is kept constant at different bitrates, as shown in Table VI. It is because the number of MBs to be decoded only depends on the numbers of BMBs and non-MC FMBs in the encoded sequence during reverse playback. In fact, the types of MBs are computed based on the distribution of motion vectors in the encoded sequence, which is not varied at different encoded bitrates. On the other hand, it is significant to note that the number of bits to be sent over the network can be reduced more significantly for sequences encoded at high bitrate, as shown in Table VII. The reason behind is that, at low bitrate, a considerable percentage of DCT blocks have a significant amount of zero elements. For direct addition of the DCT coefficients, all newly quantized DCT coefficients are computed in the DCT domain by adding two

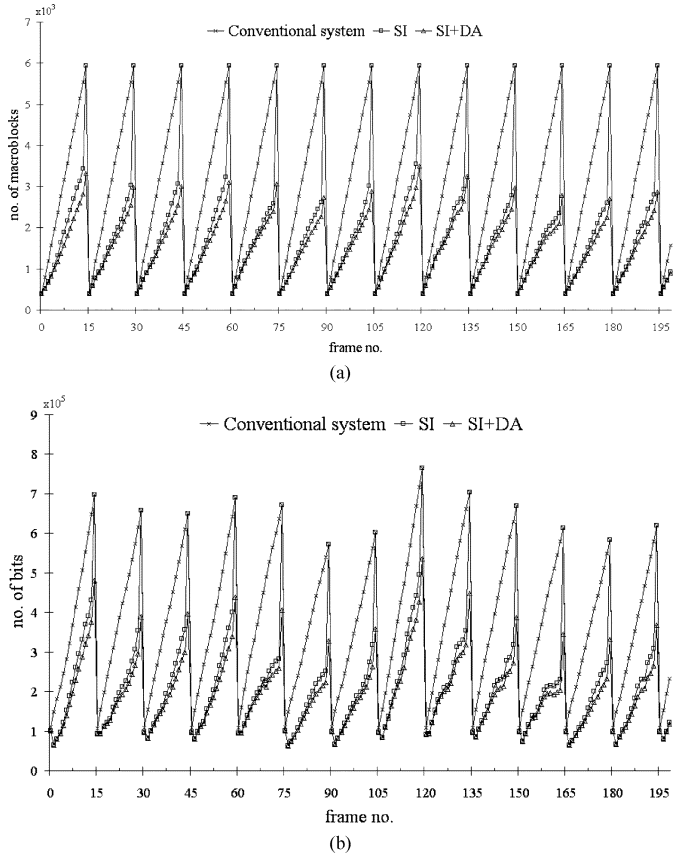


Fig. 10. Performance of the conventional system and the proposed systems, *SI* and *SI+DA*, for the "Salesman" sequence encoded at 1.5 Mb/s in the reverse-play operation. (a) Number of MBs to be decoded by the decoder. (b) Number of bits to be sent over the network.

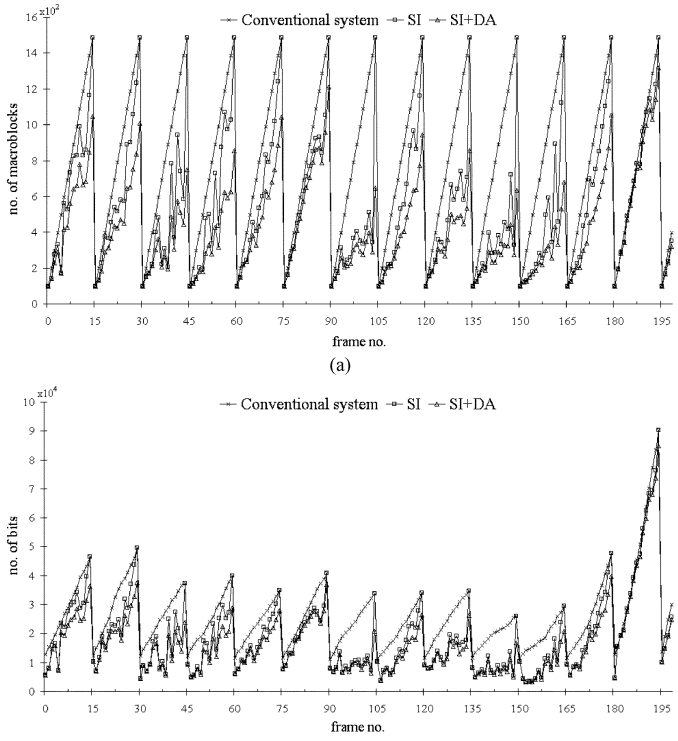


Fig. 11. Performance of the conventional system and the proposed systems, *SI* and *SI+DA*, for the "Carphone" sequence encoded at 128 Kb/s in the reverse-play operation. (a) Number of MBs to be decoded by the decode. (b) Number of bits to be sent over the network.

DCT coefficients, which are directly extracted from the MPEG video stream in the server. If either one of the DCT coefficients is zero, it does not save the bits required to encoded the combined DCT coefficients. Although the direction addition of the DCT coefficients can combine several MBs into one and save the number of MBs to be sent, it cannot achieve as much saving of bits as sending over the network at low bitrate.

Figs. 10 and 11 show the frame-by-frame comparisons of the required number of MBs decoded by the decoder and the required number of bits transmitted over the network of the conventional system, *SI* and *SI+DA* for the "Salesman" and "Carphone" sequences in the reverse-play operation, respectively. It is obvious that the proposed methods can achieve significant performance improvement in terms of decoder complexity and network traffics. Note that, as shown in Figs. 10 and 11, the required number of MBs and bits at each last frame of GOPs of the conventional system and *SI* 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. In this 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.

Theoretically, both VLC-domain and DCT-domain techniques for BMBs and FMBs will not introduce any quality degradation during reverse playback. However, such techniques will cause quality drift due to the mismatch control and clipping operations of the MPEG-2 video algorithm. To alleviate such quality drift, we suggest implementing a suitable compensation at the decoder. For instance, to minimize the error accumulation due to the IDCT mismatch at the MPEG-2 decoder, the mismatch control is performed before the IDCT. This involves clipping the decoded coefficients to the range $[-2048, 2047]$ and summing all coefficients in the block. If the sum is even, the mismatch control is applied to the last coefficient of the block. If the last coefficient is odd, it is decremented by 1, and if even incremented by 1. This process will cause the problem of decoding the 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 8×8 blocks is "3." In the forward playback, the mismatch control at the decoder adjusts the last coefficient to "2." In contrast, this coefficient is sign-inversed in the VLC 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 the mismatch control is activated for BMBs at the decoder, the last coefficient of this block after the mismatch control is examined. If this coefficient after the mismatch control is even, it is further increased by 2, otherwise it is decreased by 2. In the above example, the last coefficient after the 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 the mismatch control in the BMBs can be fully compensated. Table VIII shows the PSNR degradation of the proposed algorithms. Compared with the conventional system, only a negligibly small PSNR degradation of the proposed *SI* is shown. The small degradation of PSNR is mainly caused by the clipping operations. For *SI+DA*, the PSNR degradation

TABLE VIII
PSNR PERFORMANCES OF *SI* AND *SI+DA*

Sequences	Bitrate	PSNR degradation during reverse playback	
		SI	*SI+DA*
Salesman	0.75M	0.0005	0.0052
(*352×288*)	1.5M	0.0014	0.0215
Football	1.5M	0.0055	0.0073
(*352×240*)	3M	0.0064	0.0293
Table Tennis	1.5M	0.0217	0.0381
(*352×240*)	3M	0.0189	0.0624
Foreman	64K	0.0000	0.0001
(*176×144*)	128K	0.0012	0.0000
Carphone	64K	0.0002	0.0000
(*176×144*)	128K	0.0000	0.0000
Claire	64K	0.0285	0.0340
(*176×144*)	128K	0.0792	0.0808
Grandma	64K	0.0001	0.0029
(*176×144*)	128K	0.0030	0.0107

TABLE IX
SAVING OF THE AVERAGE NUMBER OF MACROBLOCKS THAT NEED TO BE DECODED OF *SI+DA* AS COMPARED WITH THE CONVENTIONAL SYSTEM FOR DIFFERENT L

Sequences	Bitrate	Saving of macroblocks to be sent over the network		
		L=7	*L=15*	*L=30*
Salesman	0.75M	40.93%	50.73%	55.02%
(*352×288*)	1.5M	40.93%	50.73%	55.02%
Football	1.5M	21.20%	28.03%	32.19%
(*352×240*)	3M	21.20%	28.03%	32.19%
Table Tennis	1.5M	36.51%	47.06%	53.06%
(*352×240*)	3M	36.51%	47.06%	53.06%
Foreman	64K	29.01%	38.14%	39.84%
(*176×144*)	128K	29.01%	38.14%	39.84%
Carphone	64K	36.71%	45.80%	51.42%
(*176×144*)	128K	36.71%	45.80%	51.42%
Claire	64K	65.87%	79.77%	90.95%
(*176×144*)	128K	65.87%	79.77%	90.95%
Grandma	64K	59.85%	72.79%	78.56%
(*176×144*)	128K	59.85%	72.79%	78.56%

TABLE X
SAVING OF THE AVERAGE NUMBER OF BITS THAT NEED TO BE SENT OF *SI+DA* AS COMPARED WITH CONVENTIONAL SYSTEM FOR DIFFERENT L

Sequences	Bitrate	Saving of bits to be sent over the network		
		L=7	*L=15*	*L=30*
Salesman	0.75M	46.33%	47.85%	47.42%
(*352×288*)	1.5M	42.58%	49.59%	56.13%
Football	1.5M	19.20%	22.87%	24.71%
(*352×240*)	3M	19.42%	24.75%	27.52%
Table Tennis	1.5M	29.14%	31.47%	32.73%
(*352×240*)	3M	29.00%	33.60%	36.56%
Foreman	64K	33.30%	31.79%	28.93%
(*176×144*)	128K	32.08%	35.15%	34.66%
Carphone	64K	35.53%	32.66%	70.69%
(*176×144*)	128K	34.03%	35.75%	38.39%
Claire	64K	67.14%	84.01%	84.70%
(*176×144*)	128K	65.89%	78.48%	84.39%
Grandma	64K	64.79%	75.49%	81.02%
(*176×144*)	128K	66.35%	78.80%	84.00%

increases slightly, but is still insignificant. It is due to the effect of mismatch control and the clipping operations when the technique of direct addition is adopted. With all of these factors, the visual quality is maintained as shown in Table VIII.

Second, let us demonstrate the performance of the proposed *SI+DA* with different L. Tables IX and X depict the performance improvement of the average number of MBs to be decoded and the average number of bits to be sent with respect to different L, respectively. Equation (2) indicates that, if L is large, the average number of frames need to be transmitted and decoded for a requested frame in the reverse-play operation is increased, thereby leading to a significant increase of decoding complexity and network traffics. From Tables IX and X, we show that *SI+DA* has a better improvement in both decoder complexity and network traffics for large L. This further demonstrates the effect of the proposed techniques when applied to the MPEG video system with VCR functionality.

V. CONCLUSION

In this paper, we addressed issues in implementing an MPEG video system with VCR functionality. We have showed that when a user requests a reverse-play operation, it may result in much higher network traffics than a forward-play operation. This reverse-play operation also requires high decoder complexity. Efficient reverse-play techniques for the MPEG video system have been proposed in this paper. The proposed techniques are motivated by the center-biased motion vector distribution characteristics of real-world video sequences. With the motion information, the video streaming server organizes the MBs in the requested frame into two categories—BMBs and FMBs. Then it selects the necessary MBs adaptively, processes them in the compressed domain and sends the processed MBs to the client machine. For BMBs, we have proposed a technique of sign inversion of DCT coefficients to simplify the decoder complexity while maintaining low network bandwidth requirement. For FMBs coded without MC, we have also shown that a technique of using direction addition of DCT coefficients when applied to the video streaming system with VCR support can further alleviate the computational burden of the client decoder in reverse-play operations. Since BMB and FMB are manipulated on the VLC domain and DCT domain, respectively, it is not necessary to perform decoding and re-encoding of the video streams in the server. There is little additional computational complexity required for the server. Furthermore, since the process of re-encoding is not required, the visual quality during reverse playback will only be degraded negligibly. Simulation results show that, with our proposed scheme, the MPEG video system with reverse-play functionality can minimize the required network bandwidth and decoder complexity significantly. Basically, all techniques proposed here can be easily extended to the new video coding standard—H.264. Besides, in H.264, the integer transform supersedes the DCT such that all operations can be carried out using integer arithmetic without loss of decoding accuracy. Mismatch control is then not necessary to be used at the decoder. As a result, the visual quality degradation during reverse playback due to the technique of direct addition could be further reduced. However, many problems still remain to be investigated in H.264. For example, one of our future works could focus on techniques to implement the proposed algorithms with the variable block size motion estimation which is supported in H.264. As a concluding remark, it is believed that the results of the present work will certainly be useful for the future development of H.264 codecs in digital VCR applications.

REFERENCES

[1] H. J. Stuttgen, "Network evolution and multimedia communication," *IEEE Multimedia*, vol. 2, no. 3, pp. 42–59, Fall 1995.

[2] T. D. C. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, vol. 13, no. 3, pp. 14–24, Fall 1994.

[3] L. Press, "The Internet and interactive television," *Commun. ACM*, vol. 36, no. 12, pp. 19–23, Dec. 1993.

[4] D. Wu, Y. T. Thomas, and Y.-Q. Zhang, "Transporting real-time video over the Internet: Challenges and approaches," *Proc. IEEE*, vol. 88, no. 12, pp. 1855–1877, Dec. 2000.

[5] Real Networks RealPlayer [Online]. Available: http://www.real.com/

[6] Microsoft Window Media. Microsoft Corporation Inc. [Online]. Available: http://www.microsoft.com/windows/windowsmedia/

[7] "Video coding for low bitrate communication," ITU-T Recommendation H.263, 1997.

[8] L. Chiariglione, "The development of an integrated audiovisual coding standard: MPEG," *Proc. IEEE*, vol. 83, no. 2, pp. 151–157, Feb. 1995.

[9] "Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s—Part 2: Video," ISO/IEC 11 172–2, 1993.

[10] "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video,", ISO/IEC 13 818–2, 1996.

[11] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. 33, no. 9, pp. 1011–1015, Sep. 1985.

[12] Y.-L. Chan and W.-C. Siu, "New adaptive pixel decimation for block motion vector estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 1, pp. 113–118, Feb. 1996.

[13] ——, "An efficient search strategy for block motion estimation using image features," *IEEE Trans. Image Process.*, vol. 10, no. 8, pp. 1223–1238, Aug. 2001.

[14] 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 Syst.*, 1995, pp. 73–80.

[15] S. J. Wee, "Reversing motion vector fields," in *Proc. IEEE Int. Conf. Image Process. 1998 (ICIP98)*, Oct. 1998, pp. 209–212.

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

[17] C. W. Lin, J. Zhou, J. Youn, and M. T. Sun, "MPEG video streaming with VCR functionality," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 415–425, Mar. 2001.

[18] "Video Codec Test Model, TMN8," ITU-T/SG15, 97.

[19] 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. Video Technol.*, vol. 8, no. 5, pp. 369–377, Aug. 1998.

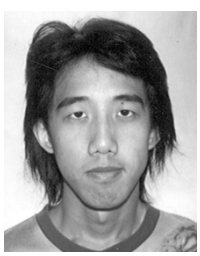
[20] L.-M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[21] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438–442, Aug. 1994.

[22] K.-T. Fung, Y.-L. Chan, and W.-C. Siu, "Low-complexity and high quality frame-skipping transcoder," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sydney, Australia, May 2001, pp. 29–32.

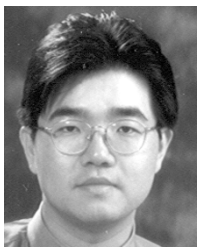
[23] ——, "New architecture for dynamic frame-skipping transcoder," *IEEE Trans. Image Process.*, vol. 11, no. 8, pp. 886–900, Aug. 2002.

[24] ——, "Low-complexity and high-quality frame-skipping transcoder for continuous presence multipoint video conferencing," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 31–46, Feb. 2004.



Chang-Hong Fu received the B.Eng. degree (first class hons.) from the Hong Kong Polytechnic University, Hong Kong, in 2002, where he is currently working toward the Ph.D. degree.

His research interests include multimedia technologies, signal processing, and video coding.



Yui-Lam Chan received the B.Eng. (first class hons.) and the Ph.D degree from the Hong Kong Polytechnic University, Hong Kong, in 1993 and 1997, respectively.

He joined the Hong Kong Polytechnic University in 1997, and is now an Assistant Professor in the Department of Electronic and Information Engineering. He has published over 40 research papers in various international journals and conferences. His research and technical interests include multimedia technologies, signal processing, image and video compression, video streaming, video transcoding and video conferencing, digital TV, and digital VCR.

Dr. Chan was the recipient of more than ten famous prizes, scholarships, and fellowships for his outstanding academic achievement, such as being the Champion in Varsity Competition in Electronic Design, the Sir Edward Youde Memorial Fellowship, and the Croucher Foundation Scholarships.

Wan-Chi Siu (M'77-SM'90) received the Associateship from The Hong Kong Polytechnic University, the M.Phil. degree from The Chinese University of Hong Kong, and the Ph.D. degree from the Imperial College of Science, Technology, and Medicine, London, U.K., in 1975, 1977, and 1984, respectively.

He was with The Chinese University of Hong Kong as a Tutor, initially, and later as an Engineer, from 1975 and 1980. He then joined The Hong Kong Polytechnic University as a Lecturer in 1980 and was subsequently supported by the University in October 1982 for further studies in the U.K. He was promoted to Senior Lecturer, Principle Lecturer, and Reader in 1985, 1987, and 1990, respectively, and has been Chair Professor in the Department of Electronic and Information Engineering since 1992. He was Head of the Electronic and Information Engineering Department and subsequently Dean of Engineering Faculty from 1994 and 2002. He is currently the Director of the Centre for Multimedia Signal Processing. He has published over 250 research papers, over 120 of which appeared in international journals, including IEEE publications and *IEE Proceedings*. He is also an Editor of the recent book *Multimedia Information Retrieval and Management* (Berlin, Germany: Springer, 2003) and a member of the editorial board of the *Journal of VLSI Signal Processing Systems for Signal, Image, Video Technology*, the EURASIP *Journal on Applied Signal Processing*, and a few other journals. His research interests include digital signal processing, fast computational algorithms, transforms, wavelets, image and video coding, and computational aspects of pattern recognition and neural networks.

Prof. Siu was Guest Editor and Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: ANALOG AND DIGITAL SIGNAL PROCESSING from 1995 to 1997. He received many awards, such as the Distinguished Presenter Award of the University Management Workshop of the PolyU (1997), the IEEE Third Millennium Medal for Outstanding Achievements and Contributions (2000), and, most recently, the Best Teacher Award, EIE, and the Outstanding Award in Research and Scholarly Activities, Faculty of Engineering, PolyU (2003). He has been a keynote speaker at a number of international conferences, including, most recently, the 2002 Third IEEE Pacific Rim Conference on Multimedia (PCM2002), Taiwan, R.O.C., and the 2003 IEEE International Conference on Neural Networks and Signal Processing, Nanjing, China. He has held the position of General Chair or Technical Program Chair of many international conferences. In particular, he was a Technical Program Chair of the IEEE International Symposium on Circuits and Systems (ISCAS 1997), and he is the General Chair of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003). From 1991 and 1995, he was a member of the Physical Sciences and Engineering Panel of the Research Grants Council (RGC), Hong Kong Government, and, in 1994, he chaired the first Engineering and Information Technology Panel to assess the research quality of 19 cost centers (departments) from all universities in Hong Kong.