The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Advanced Digital VCR for Compressed Videos

by

Fu Chang hong

A thesis submitted in partial fulfillment of the requirements for Doctor of Philosophy

December 2007

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

_____ (Name of student)

Abstract

With the establishment of MPEG video coding standards, many video sequences for modern streaming applications are encoded in MPEG formats. However, the MPEG standards employ motion-compensated prediction in which the compressed data is not invariant to changes in frame order. This hinders users from browsing video in a more interactive way. To enrich the user's viewing experience, it is desirable to perform various video cassette recording (VCR) functionality such as backward, fast-forward/backward, random access, etc. in digital video. Therefore, in this thesis, some novel techniques are suggested for the efficient implementation of VCR functionality in a digital video streaming system with minimum requirements on the decoder complexity and the network traffic.

Backward playback is one of the most common VCR functions. One popular approach is to use a reverse transcoder in the server which converts I–P frames into another I–P bitstream in reverse frame order. When a playback device decodes this reverse-encoded bitstream, backward playback can be achieved. To expedite the transcoding process, we propose a fast reverse motion estimation algorithm with smart mode decision for H.264 reverse transcoding. By analyzing the motion vectors and modes decoded from the forward bitstream, the best mode and motion vector for each reverse transcoded macroblock are estimated. A remarkable reduction of computational complexity involved in

reverse motion estimation can be achieved by the proposed algorithm with only negligible impact on the rate-distortion performance.

Afterwards, we propose a compressed-domain approach for an efficient implementation of the MPEG video streaming system to provide backward playback over a network. In the proposed video streaming server, according to the motion information, macroblocks in the requested frame are classified into two categories – backward macroblocks (BMBs) and forward macroblock (FMBs). Two novel macroblock-based techniques are used to manipulate the necessary macroblocks in the compressed domain and the server then sends the processed macroblocks 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 macroblocks to be decoded by the decoder and the number of bits to be sent over the network in the backward-play operation. By identifying the related macroblocks of FMBs in their reference frame, a direct addition technique for discrete cosine transform (DCT) coefficients is designed to further reduce the computational complexity of the decoder. We also contrive a mixed VLC and DCT domain technique for the FMBs to offer better performance of the proposed system. With these compressed-domain techniques, the proposed architecture manipulates macroblocks in the VLC and DCT domain only to achieve a 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.

Although the new scheme exhibits promising results for backward playback, it encounters a problem when backward playback traverses the group-of-pictures (GOP) boundary since the proposed sign inversion technique makes use of the motion relationship between two adjacent frames. But, no inter-frame prediction takes place between the last frame of one GOP and the first frame of the successive GOP. In this thesis, we also provide a novel solution to cope with the GOP discontinuity problem of the video bitstream by re-building the motion linkages across GOP boundaries.

By employing the compressed-domain techniques, the work in this thesis shows significant improvements in terms of the server complexity, the network traffic, and the quality of reconstructed video during backward playback. Undoubtedly, the results of our work will certainly be useful for the future development of digital VCR.

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to my supervisor, Dr. Y.L. Chan, and co-supervisor, Professor W.C. Siu, for their continuous encouragement, guidance and care during the period that I worked on this thesis. They gave me information, suggestions and valuable advice contributing to every success of my research. More importantly, I am deeply impressed with their hard working style and their willingness to devote to the advancement of science and research. This gives me a clear image of a great researcher should be and have inspired me to work hard on the thesis. It is beyond doubt that this will continuously influence my future research and career.

I would again like to express my sincere thank to Dr. Bonnie Law, Mr. Alvin Cheung, Mr. Bill Ip, Miss K.M. Au, Mr. Patrick Chan, Mr. K.C. Hui, Miss K.L. Lau, Miss Karen Wong, Mr. Michael Yiu and Mr. K.T. Fung. The sharing of ideas and experience with them has greatly contributed to make every success of my work. It has been a wonderful time to me in these years to work with them.

Meanwhile, I am glad to express my gratitude to the Department of Electronic and Information Engineering and the Centre of Multimedia Signal Processing for providing me a comfortable working environment, and the Hong Kong

Polytechnic University for their generous financial support to carry to my research work.

Above all, I am deeply grateful to my family for their constant love, encouragement and support. Without their understanding and patience, it is impossible for me to complete this research study.

Table of Contents

List of Figures

List of Tables

Abbreviations

B-frame	Bi-directional Predicted Frame
BMB	Backward Macroblock
BR	Backward Region
DA	Direct Addition
DB	Display Buffer
DCT	Discrete Cosine Transform
DVD	Digital Versatile Disc
FB	Frame Buffer
FBMB	Forward/Backward Macroblock
FLC	Fixed Length Coding
FMB	Forward Macroblock
FPS	Frames per Second
FR	Forward Region
GOP	Group of Pictures
I-frame	Intra-coded Frame
J_{motion}	Joint R-D function for motion estimation
J_{mode}	Joint R-D function for mode decision

JPEG	Joint Picture Experts Group
MB	Macroblock
MC	Motion Compensation
MCMB	Motion-Compensated Macroblock
MCP	Motion-Compensated Prediction
ME	Motion Estimation
MP3	MPEG Layer-3 Audio Coding Standard
MPEG	Moving Picture Experts Group
MV	Motion Vector
P-frame	Forward Predicted Frame
RR	Redundancy Reduction
SI	Sign Inversion
SP-frame	Forward Predicted Switching Frame
TV	Television
VCD	Video Compact Disc
VCR	Video Cassette Recorder
VLC	Variable Length Coding
VLD	Variable Length Decoding
VOD	Video on Demand

List of Publications

International Journal Papers

1. Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu, “Efficient Reverse-Play Algorithms for MPEG Video Streaming with VCR Support,” IEEE Transactions on Circuits and Systems for Video Technology, Volume 16, Issue 1, pp.19-30, January 2006
2. Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu, “Fast Motion Estimation and Mode Decision for H.264 Reverse Transcoding”, Electronics Letters, Volume 42, pp. 1385-1386, November 2006.
3. Chang-Hong Fu, Yui-Lam Chan, Tak-Piu Ip and Wan-Chi Siu, “New Architecture for MPEG Video Streaming System with Backward Playback Support”, IEEE Transactions on Image Processing, Volume 16, Number 9, pp. 2169-2183, September 2007.

International Conference Papers

1. Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu, “Efficient Reverse-Play Algorithm for MPEG Video Streaming with VCR Functionality,” Proceedings of the Fourth International Conference on Information, Communications & Signal Processing and Fourth Pacific-Rim Conference on Multimedia (ICICS-PCM 2003), Volume 3, pp.1356-1360, December. 15-18, 2003, Singapore.

2. Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu, "Macroblock-Based Reverse Play Algorithm For MPEG Video Streaming," Proceedings of the International Symposium on Circuits and Systems (ISCAS2004), Volume 3, pp.753-756, May 23-26, 2004, Vancouver, Canada.
3. Chang-Hong Fu, Yui-Lam Chan, Tak-Piu Ip, and Wan-Chi Siu, "Efficient algorithm for reverse playback in MPEG video streaming," Proceedings of International Symposium on Intelligent Multimedia, Video and Speech Processing (ISIMP 2004), pp.659-662, October 20-22, 2004, Hong Kong.
4. Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu, "Improved macroblock-based reverse play algorithm for MPEG video streaming," Proceedings of the International Conference on Image Processing (ICIP2004), Volume 3, pp.2063-2066, October 24-27, 2004, Singapore
5. Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu, "Reverse-play algorithm for MPEG video streaming," Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, pp. 381-384, May 28-30, 2005, Su Zhou, China.
6. Chang-Hong Fu, Yui-Lam Chan, and Wan-Chi Siu, "Redundancy Reduction in the Reverse-Play Operation of MPEG Streaming System", Proceedings of the twentieth China Symposium on Circuits and Systems, Volume 5, pp.399-402, June 15-17, 2007, Guang Zhou, Shen Zhen, China
7. Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu, "Efficient Motion Estimation in H.264 Reverse Transcoding," Proceedings of the International Conference on Image Processing (ICIP 2007), Volume 5, pp.V317-V320, September. 16-19, 2007, Texas, USA.

Recent Submissions

1. Chang-Hong Fu, Yui-Lam Chan and Wan-Chi Siu, “Establishment of Motion Linkage across GOP boundaries for Reverse Playback on H.264 Video”, revised version submitted to IEEE Transactions on Multimedia.

Chapter 1 Introduction

1.1 Overview

One of the primary mediums for content creation and distribution is video. The last twenty years has witnessed on a digital revolution. Digital data and voice communication have long been around, but the digital format is now used more and more widely along with general usage of computers in life. It is very convenient for people using computer to store, retrieve and exchange digital video. However, digital video requires quite significant storage. For instance, a Digital Versatile Disk (DVD) can only store a few seconds of raw video at television-quality resolution and frame rate, which is not practical. If they are transmitted through a network, considerable bandwidth is needed. Therefore, researchers have endeavored to find methods to compress digital video. It is the emergence of some video compression standards, such as motion JPEG, MPEG-1, MPEG-2, MPEG-4, H.26L, and H.264, which have made video delivery applications such as all-digital TV, video-on-demand and video streaming a reality.

However, these video coding standards are designed mainly for the purposes of efficient video storage and transmission, not browsing. This explains why many current video players [1-4] offer relatively few controls for video browsing. For example, these players have only limited fast-forward/backward play and even they cannot provide backward play. The limitation is due to the use of motion-

compensated prediction in various video coding standards. To enhance the users' viewing experience and expedite the marketability of video delivery services, a key issue is to provide full video cassette recording (VCR) functionality to enable fast and user-friendly browsing of video contents. The set of effective VCR functionality includes forward, backward, stop, pause, step forward, step backward, fast forward, fast backward, and random access. Therefore, in this thesis, we explore ways to implement full VCR functionality efficiently, with minimum requirements made on the network bandwidth and the decoder complexity.

In this chapter, the fundamental principles of digital video and the motives for video compression are introduced. An overview of hybrid motion-compensated predictive coding is then presented. Today, video streaming with set-top devices or software players is becoming very popular and provides a platform for interactive video playback. We therefore discuss the problem of these interactive browsing operations in the current video coding standards. The straightforward implementation of video streaming with VCR support is also discussed. Finally, the motivation, objectives, and the organization of this thesis are presented.

1.2 Hybrid motion-compensated predictive coding

Digital video is video information that is stored and sent in digital form. However, the bit rate of digital video is huge when an analog video signal is converted into digital form at equivalent quality. This bit rate is completely

unacceptable for most networks and processors to handle. The solution is to compress the digital video information, and to store or transmit it in a compressed form. Compression techniques for digital video have been continually improving over the last two decades.

International standards now provide standard techniques for digital video coding. Developing an international standard requires collaboration among many parties from different countries with different infrastructures and commercial interests, and an organization that can support the standardization process as well as enforce the standards. There are two major teams of developing digital video coding standards. They are the ISO/IEC MPEG (Moving Picture Experts Group) and ITU-T VCEG (Video Coding Experts Group). The MPEG is a Working Group of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The goal of MPEG is to develop standards for compression, processing, and representation of moving pictures and audio. It has been responsible for the successful MPEG-1 (ISO/IEC 11172) [5] and MPEG-2 (ISO/IEC 13818) standards [6], which have given rise to widely adopted commercial products and services, such as VCD, DVD, digital television, MP3 players, etc. On the other hand, the VCEG is a working group of the International Telecommunication Union Telecommunication Standardization Sector (ITU-T), whose work has led the development of a series of important standards for video communications over telecommunication networks and computer networks. Starting from the first H.261 videoconferencing standard [7] and following on with the very successful H.263 standard [8], two other standardizations, informally known as H.263+ and

H.263++, have been created to extend the capabilities of H.263. The most powerful and advanced video compression standard at this moment is H.264 or known as MPEG-4 Part 10 [9] which has been developed by the joint efforts of the ITU-T VCEG and the ISO/IEC MPEG.

All the standards make use of the redundancy inherent in digital video information so as to achieve a remarkable reduction in bit rate. A single frame within a video sequence always has a significant amount of spatial redundancy. By eliminating some of this redundancy, it is possible to represent or encode the information in a more compact form. For instance, the image data may be transformed from the spatial domain into another domain in which the information is represented more compactly. Some unimportant components of the transformed information can then be discarded without seriously degrading the visual quality of the decoded image. Apart from the spatial redundancy, scene and objects in the video content changes smoothly and gently over time, so successive frames are often highly correlated. In other words, a moving video sequence contains a large amount of temporal redundancy. It is likely to achieve further compression by removing this temporal redundancy. Motion estimation and compensation is widely used in most of the modern video coding standards. This technique reduces the temporal redundancy between successive frames by forming a predicted frame and subtracting this from the current frame. The predicted frame is created from the past frame, sometimes called the reference frame, by estimating the motion between the current frame and the reference frame. The predicted frame can then be formed by compensating for the motion between the two frames. The output of this

process is a residual frame or predicted difference. The more accurate the prediction process is, the less energy is contained in the residual frame. The residual frame undergoes transformation, quantization, and entropy coding in order to further remove the spatial redundancy before it is stored or sent to the decoder. In the decoder, by using the motion information and the reference frame, the predicted frame is re-created and added with the decoded residual frame to reconstruct the current frame.

By removing spatial and temporal redundancy, three main types of coded pictures are defined in the video coding standards: I-frames, P-frames, and B-frames. I-frames are intraframe encoded without any temporal prediction to other frames. Each frame is treated as a separate picture and encoded independently using transformation, quantization, and entropy coding. Since only spatial redundancy is reduced, the level of compression obtained with I-frames is relatively small. P-frames are interframe encoded using motion compensated prediction from the previous I- or P-frame. Hence, a significantly higher level of compression can be obtained. B-frames are interframe encoded using interpolated motion prediction between the previous I- or P-frame and the next I- or P-frames in the video sequence. The prediction is actually chosen from the previous frame, from the next frame, or by calculating the average of both. This arrangement provides the highest level of compression, but we should note that B-frames never act as a reference for other frames.

The three picture types are grouped together in Group-of-picture (GOP) where GOP consists of one I-frame followed by a number of P- and B-frames. An

example of a GOP structure is shown in Figure 1.1, where each I- or P-frame is followed by three B-frames. Note that either the structure or size of each GOP is not specified in any standard and can be changed to suit different applications.

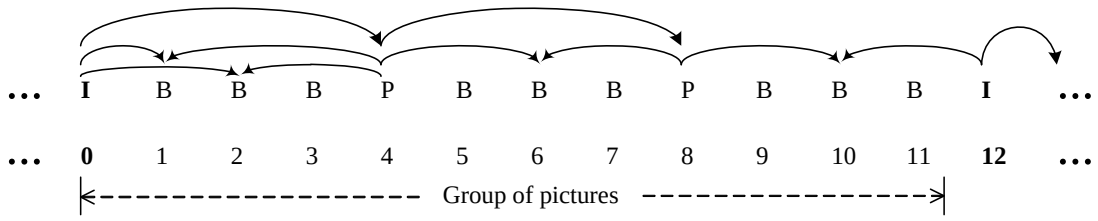


Figure 1.1. Reference relationship in the GOP structure.

1.3 Interactive playout of digital video in online streaming services

Unlike traditional analog video, video servers now store video in a digital form on large hard disks. Streaming digital video over the Internet such as video-on-demand allows users to access and retrieve various videos over networks ubiquitously by using software players or digital set-top box devices. Due to the explosive growth of the Internet and the increasing demand for video contents, real-time video services over the Internet have also received great attention from academia and industry [1-14]. The term real-time multimedia implies timing constraints. For example, video data must be played out continuously. If the data does not arrive in time, the playout process will pause, which is annoying for human eyes. Real-time transport of stored video is the predominant part of real-time multimedia. Transmission of stored video can be done in either the “download mode” or the “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 played out while parts of the content are received and decoded. Due to its 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 high storage-capacity and throughput in the video server and the large bandwidth in the network to deliver the video streams.

Recent advances in computing technology, compression standards [5-9], high-band storage devices, and high-speed networks have made it feasible to provide video streaming applications. The computer software industry has quickly adopted the Internet as a basic platform for digital video delivery. However, the main focus of industry development has been the creation and distribution of video contents, not viewing or browsing. As a consequence, the leading software players such as the Real Networks RealPlayer [1] and Microsoft Windows Media Player [2] only support relatively limited browsing operations. Besides, the consumer electronics industry has begun to incorporate more browsing features in the next generation of hardware video playback devices. Therefore, it is highly desirable that video streaming systems should have the capability of providing fast and effective browsing. This trend has created great opportunities for identifying new interactive video applications, and for developing advanced systems and algorithms to support these applications.

However, current coding standards such as MPEG and H.264 [5-10,15-16] were only very successful for digital video compression. In order to complete the transition to digital video from its current analog state, MPEG/H.264 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 contents is full video cassette recording (VCR) functionality, including forward, backward, step-forward, step-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.

1.4 Motivation and objectives

Digital video compression is an enabling technology for a wide variety of applications, including video delivery over the Internet, digital television broadcasting, and video storage. Although the performance of modern video coding standards such as MPEG and H.264 is efficient in terms of compression, the temporal dependencies of coded frames in MPEG often make VCR trick modes difficult, especially in backward playback. Consequently, fast and efficient algorithms for providing full VCR functionality in MPEG/H.264 video are in great demand.

On the evidence of the recent video players, the implementation of the full VCR functionality with the MPEG/H.264 coded video is not a trivial task. Video players now provide relatively limited controls for video browsing. For instance, only limited fast-forward/backward play is supported by these players and they cannot offer frame-by-frame playback in reverse order. This is due to the fact that MPEG/H.264 video compression is based on motion-compensated prediction with an I-B-P frame structure [16-18]. The I-B-P frame structure allows a straightforward realization of the forward-play function, but imposes several constraints on other trick modes such as backward playback, fast-backward playback, fast-forward playback, and random access. Straightforward implementation of these functions requires much higher network bandwidth and decoder complexity compared to those required for the regular forward-play operation.

For uncompressed video, the solution for backward 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/H.264 video data because the present video compression standards use motion-compensated prediction that is not invariant to changes in frame 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 a simple I-P structure of an MPEG encoded sequence, to decode a P-frame, the previously encoded I-/P-frames need to be transmitted and decoded first. In the simple approach that only decodes I-frames for backward

playback, no P-frames could be displayed and the operation is limited by the GOP size of the bitstream. In practical cases, large GOP size is frequently used and all VCR functions including backward playback only provide access to the limited I-frames, which may miss some important frames. One straightforward approach to implement a backward-play operation of compressed video is to decode all the frames in the whole GOP, store all frames in the large buffer of the decoder, and play the decoded frames in reverse direction. However, this approach requires a significant amount of memory in the client side and this 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. For instance, once a backward operation is requested, it always lasts for a few seconds or minutes. During the period of backward playback, all frames between the current frame and the previously nearest I-frame need to be sent over the network and decoded by the client decoder. This approach 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 undesirable. The problem grows more serious with increases in the GOP size.

In order to enhance the users' viewing experience, the aim of this research is to facilitate the capability of browsing video contents interactively. In this thesis, we propose several cost efficient MPEG video streaming schemes with VCR support. Some novel algorithms are integrated into various schemes in order to provide full VCR functionality over a network with minimum requirements on network bandwidth and decoder complexity.

1.5 Organization of this thesis

This thesis is divided into seven chapters. Prior to our description of the main research in this thesis, a review of a video streaming system with VCR functionality is given in Chapter 2. This review discusses the impact of implementing VCR operations in the video streaming system. We then formulate the traffic requirement of the network as well as the computational requirement of the client machine of different VCR trick modes. Afterwards, a review of the current techniques for the implementation of VCR functionality is presented. Of course, this chapter begins with a brief description of some aspects of the video compression techniques that are relevant to this work.

In Chapter 3, some techniques of performing reverse transcoding on MPEG/H.264 video frames are presented. Techniques including fast motion estimation and mode decision for H.264 reverse transcoding are proposed. By exploring the existing motion information and mode decision in the original H.264 video, a new reverse motion estimation algorithm with smart mode decision is proposed that could expedite the transcoding process.

Chapter 4 mainly investigates a macroblock-based scheme for MPEG video streaming with VCR support. Two novel techniques, called “sign inversion” and “direct addition” are proposed to perform backward playback. All techniques proposed in this chapter manipulate data in the compressed domain in order to reduce the computational complexity of the video server. This scheme could significantly reduce the decoded complexity and network traffic required in the backward-play operation. Chapter 5 then discusses a mixed variable length

coding (VLC) and discrete cosine transform (DCT) domain technique to further improve the performance of the sign inversion and direct addition techniques.

This macroblock-based scheme can significantly reduce the required decoding complexity and network bandwidth during backward playback. However, since there is no inter-frame dependency between the last frame of one GOP and the first frame of its succeeding GOP, the motion relationship disappears and the sign inversion technique cannot be applied in this GOP boundary. In this case, the required complexity of the decoder and the required bandwidth of the network increase remarkably when backward playback traverses GOP boundaries. In Chapter 6, a novel algorithm is proposed to re-build the motion linkages across GOP boundaries to avoid this inherent GOP discontinuity problem.

Chapter 7 is devoted to a summary of the work herein and the conclusions reached as a result. Suggestions are also included for further research in this area.

Chapter 2 Literature Review

2.1 Introduction

Digital video has become extremely popular as a result of the digital revolution in the last two decades. Recent advances in network technology have further simulated the popularity of digital video applications, e.g. video-on-demand, video surveillance, video conferencing, etc. A number of video standards such as MPEG-1/-2/-4, H.26L, and H.264 [5-10] have been developed to provide standard video formats for convenient storage, process, and transmission. Due to the enormous amount of storage required by digital videos, numerous compression techniques such as transform coding and predictive coding are built and employed in the standards in order to reduce the video size while preserving the viewing quality. These techniques exploit all kinds of redundancy in the video data for achieving better compression ratio. At the same time, dependencies are produced among the compressed video data together with compression effects. This is mainly due to the various predictive coding techniques employed in these video standards. These dependencies will not induce any inconvenience in the storage and normal playback of digital video, but cause severe difficulty when more effective browsing of video is desired. Recently, some research work related to digital video browsing has been conducted in different ways and numerous algorithms have been proposed. In this chapter, we introduce the compression techniques employed in the existing video standards that complicate the implementation of video

streaming with VCR functionality. This is followed by the mathematical formulation for the impact of VCR trick modes on video streaming.

This chapter is organized as follows. In the first section, some fundamental concepts about the digital video representation are introduced. Then we illustrate various video compression techniques together with the generic encoder and decoder, followed by the hierarchical structure of a standard video bitstream. Next, a conventional video streaming system with VCR support is presented. This interactive streaming system provides the set of VCR functionality that enables fast and user-friendly video browsing. The backward-play operation is used as an example to demonstrate the working mechanism of this system. The comparisons between forward playback and backward playback in terms of the number of frames to be decoded and the number of bits to be transmitted for displaying each frame are also given. This signifies the impact of the VCR trick modes on the MPEG streaming system. The final section reviews several existing algorithms to implement the VCR operations in the MPEG video streaming system.

2.2 Video Compression Fundamentals

Traditionally, analog video is captured by a number of light sensitive films moving at a fast speed sequentially whilst each film captures the instant time of the scene. During video playback, the recorded films are projected on a screen sequentially with the same speed as recording. To ensure motion appears smooth, about 25-30 frames per second (frame rate) must be used if the scene

is captured by a camera and not generated synthetically. In the modern world, as a result of advances in new digital technology, analog video is converted to a digital form, called digital video, and stored in storage devices such as hard disks, DVD, VCD, etc. Digital video is actually formed by a sequence of digital pictures which are created in the form of a two-dimensional matrix of individual picture elements called pixels. As compared with analog video, digital video includes no fading over time so that quality can be virtually preserved forever. It is more flexible as a storage form and more convenient for transmission through different types of networks. With the help of digital video processing tools, digital video can then be easily modified or edited.

For the above reasons, digital video has been widely used recently. As discussed above, digital video is composed of a sequence of still images or frames. Each natural image is sampled horizontally and vertically to construct its digital form by a light sensitive device (e.g. CCD or CMOS sensor [19]). The number of pixels per line and the number of lines per frame represent the horizontal and vertical resolution of the digital video. The intensity level of each pixel is typically sampled into an integer ranging from 0 to 255 for the grayscale image. When there are more pixels in each frame, the spatial resolution of video is higher and the video quality is better.

A grayscale image uses only one component to represent the brightness or luminance of each spatial sample. Color images, on the other hand, need at least three components per pixel to indicate color accurately. The method chosen to represent brightness and color is described as a color space. The

traditional color space for computer graphics is RGB (red, green, and blue). In the RGB color space, a color image sample is represented with three numbers that indicate the relative proportions of red, green, and blue. Any color can be created by combining red, green, and blue in varying proportions. However, the human visual system is less sensitive to the color components than to the luminance component. The three colors in the RGB color space are equally important and thus they are usually stored at the same resolution. This is likely to represent a color image more efficiently by separating the luminance component from the color information and representing luminance with a higher resolution than color. As a result, the most typical color space used in video coding standards represents luminance and color components separately. An example is the YC_bC_r color space [20-21], which offers a popular way to efficiently represent color images. Its variations come from different sampling formats, such as 4:4:4, 4:2:2, 4:1:1, and 4:2:0. As shown in Figure 2.1, 4:4:4 sampling means that the three components (Y, C_b, and C_r) have same resolution. In 4:2:2 sampling, the color components have the same vertical resolution as the luminance component, but half the horizontal resolution. By contrast, 4:1:1 sampling means that for every four luminance samples in the horizontal direction, there are only one C_b sample and one C_r sample. In 4:2:0 sampling, which is most commonly used in digital video standards, the color samples are centered among four luminance ones, as shown in Figure 2.1(d). By using various sub-sampling, bits allocated for color components are reduced without introducing significant visual quality degradation for the human visual system. We take 4:2:0 sampling as an example. Since each color component

contains one quarter of the number of samples in the Y component, 4:2:0 YC_bC_r video needs exactly half as many samples as 4:4:4 YC_bC_r video.

Representing a moving video sequence is done by taking a rectangular "snapshot" at periodic time intervals. Higher sampling rate (frame rate) provides smoother motion in video with the cost that more bits are required in the same time interval. A typical frame rate is about 30 frames per second (fps) in order to have smooth motion. By contrast, a frame rate of less than 15 fps is sometimes used for very low-bit-rate video communications. A low frame rate leads to what is called flicker, which is caused by the previous image fading from the eye retina before the next one is displayed. Flicker and artificial effects may become visible in fast-motion regions of the video scene.

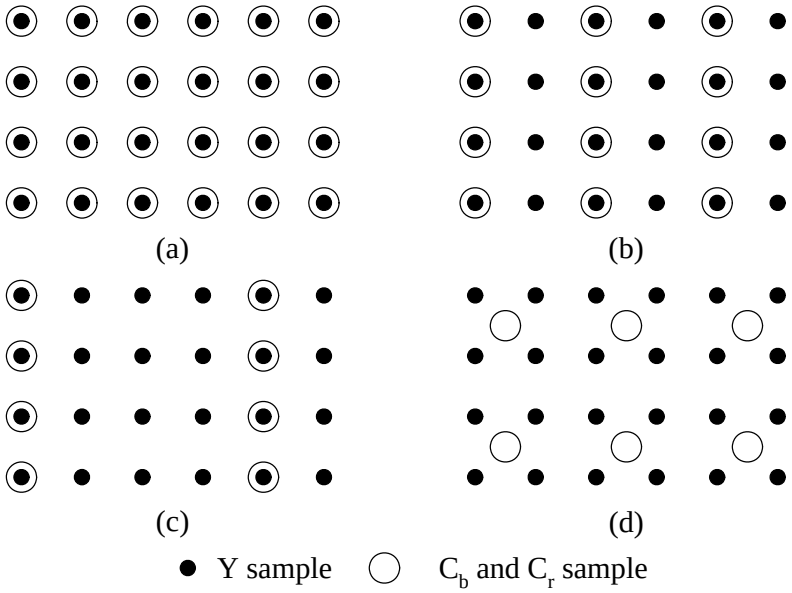


Figure 2.1. YCbCr formats: (a) 4:4:4, (b) 4:2:2, (c) 4:1:1, and (d) 4:2:0.

2.2.1 Video compression techniques

Due to the substantial amount of data in digital video, the compression ratio achieved by sub-sampling of color components only is not sufficient. A number of video coding techniques have been developed over the decades [5-10]. All of the advanced and efficient video coding techniques are built according to the characteristics of natural video and human visual perception. They exploit some of the inherent redundancy in still images and moving sequences in order to provide significant data compression.

Real and natural images are usually smooth spatially. Although each image may consist of many objects and complex background, neighboring pixels in each of these areas have quite similar characteristics and relations. For example, pixels in the dark background area have the values around zero with little variance. This spatial similarity is called spatial redundancy, and can be removed by using transform coding and quantization. Besides, with the normal frame rate of 25-30 fps, neighboring frames in video are quite similar to each other. The similarity existing among neighboring frames is called temporal redundancy, and can be removed by motion-compensated predictive coding. A generic hybrid coding system that exploits all the redundancy mentioned above is illustrated in Figure 2.2. This hybrid motion-compensated coding model is used in nearly all video coding standards.

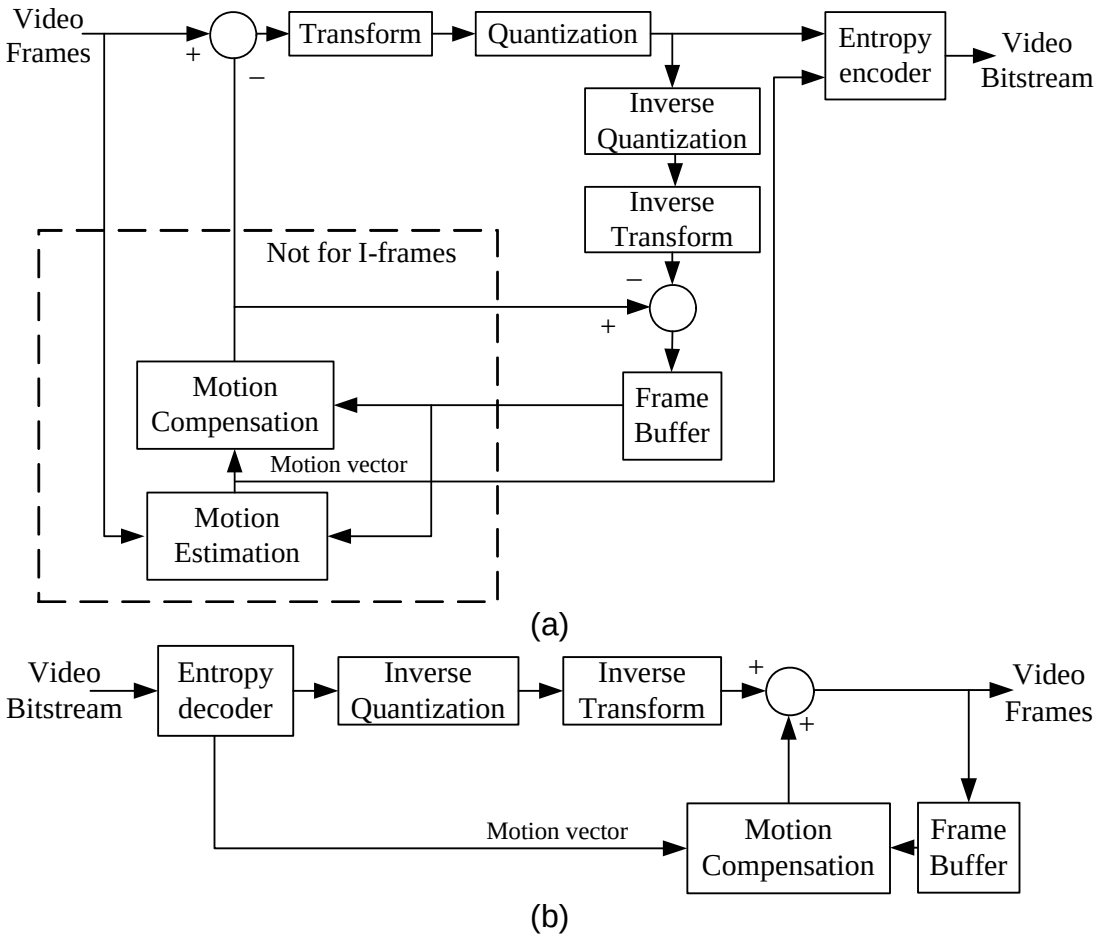


Figure 2.2. Block diagrams of the generic video codec: (a) encoder and (b) decoder.

There are three major frame types adopted in various video standards. They are I-frames, P-frames and B-frames. I-frames, standing for intra-frames, are encoded independently from other frames and do not exploit temporal redundancy. The encoding process of I-frames is included in Figure 2.2(a). During I-frame encoding, the motion estimation and motion compensation processes are deactivated. The whole process is similar to the JPEG image compression standard [22-23], which was developed by the Joint Picture Expert Group. First, the image is divided into 8x8-pixel blocks. Then each block of pixels is transformed into frequency domain to produce a set of transform coefficients. The output coefficients of the transformation are passed to the

quantization process, which represents the sampled data at a finite number of levels. After the quantization, zig-zag scanning is used to arrange the quantized coefficients into a 1-D array for the entropy encoding.

Discrete cosine transform (DCT) [24-31] has been found to be more useful for image and video coding. The transformation packs the energy of the signal into a small number of coefficients, and this process paves the way for efficient compression. In real images, several DCT coefficients could be sufficient to represent the information of the whole block if its pixels vary smoothly without any edges. The DCT coefficients are ordered in an 8x8 matrix by frequency. Each DCT coefficient specifies the contribution of a sinusoidal pattern at a particular frequency to the actual signal. The most upper left corner coefficient is the DC coefficient, and represents the average value of the block. The other coefficients, known as AC coefficients, are arranged from low frequency to high frequency toward the lower right corner. Since the frequency response of the human eyes drops off with increasing spatial frequency, a small variation in intensity is more visible in slowly varying regions than in complex ones. Therefore, the DCT coefficients in the upper left corner, which represent low-frequency components, are more important than the high-frequency coefficients in the lower right corner.

In the quantization process [31-33], a 2-D quantization matrix is provided. The quantization matrix of I-frames in MPEG-2 is shown in Figure 2.3. Each DCT coefficient is divided by the corresponding quantization factor at the same position from the matrix and is then rounded to the nearest integer to obtain the

quantized DCT value. This process aims at reducing the size of the DC and AC coefficients, and discards those unimportant coefficients so that less bandwidth is required for transmission. In practice, the quantization matrix in Figure 2.3 is designed according to the visual sensitivity to different frequency components. As discussed, the low-frequency coefficients in the upper left corner are more important than the high-frequency coefficients in the lower right corner. Therefore, low-frequency coefficients are assigned smaller quantization factors than high-frequency coefficients. In this way, high-frequency coefficients are quantized coarsely to achieve better compression efficiency without greatly affecting visual quality. This also exploits the psycho-visual redundancy phenomenon.

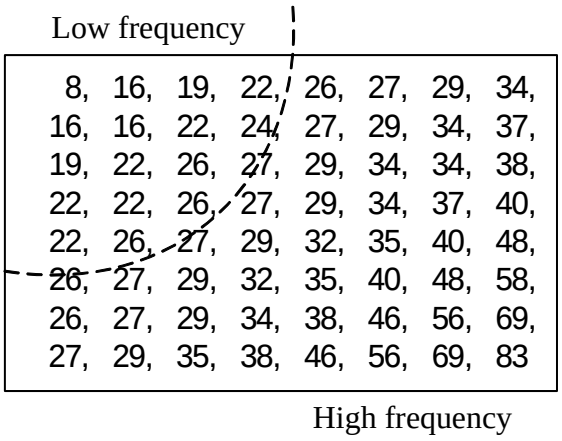


Figure 2.3. Quantization matrix for intra blocks in MPEG-2

For binary encoding, the quantized DCT coefficients are scanned in a predetermined fashion first. One popular strategy is to zig-zag scan the quantized DCT coefficients into a 1-D sequence as depicted in Figure 2.4. This results in DCT coefficients in increasing order of frequency starting with the DC coefficients and ending with the highest frequency AC coefficient. Reordering in this zig-zag way tends to create long “runs” of zero-value coefficients and this is

beneficial to 2-D variable length coding (VLC) [34-36]. To take the data in Figure 2.4 as an example, the coefficients after zig-zag scanning produce the following list: 18, 9, 5, 6, 0, 3, 0, 0, 1, 2, 0, 0, 0, 0, 0, 1, 0, 0, …. It can easily be seen that zig-zag scanning increases the likelihood of grouping all nonzero coefficients together followed by strings of zeros. To exploit this feature, it is necessary to represent them more compactly before VLC such as Huffman coding. This process is generally called run-length coding since the compact form has a series of intermediate symbols (run, level), where run is the number of zeros preceding a nonzero coefficient and level is the magnitude of a nonzero coefficient. The example above would then be encoded as "(0,18), (0,9), (0,5), (0,6), (1,3), (2,1), (0,2), (5,1), EOB", where EOB stands for end of block. EOB signifies that the previous coefficient is the last nonzero value and all remaining coefficients are zero. After run-length coding, the intermediate symbols are mapped to a series of variable length codewords. In the VLC process, the philosophy is that those frequently occurring symbols are represented with short codewords while less common symbols are represented with long codewords. It is noted that the coding method of DC coefficients is different from that of AC values. As discussed, the DC coefficient (e.g. 18 in the example of Figure 2.4) represents the average value of the whole block. Owing to the small physical area covered by each block, the DC coefficient varies only slowly from one block to the next. Hence, it is efficient to use differential encoding in which the DC coefficient of the current block is predicted from the DC value of the previous encoded block. The prediction difference is encoded instead of the DC coefficient itself. Again, in the example of Figure 2.4, if the DC value of the previous block is 16, the corresponding difference value of the

current block is 18-16=2, which is obviously more biased to zero compared to the original DC coefficient. As a result, the biased probability distribution increases the coding efficiency of VLC.

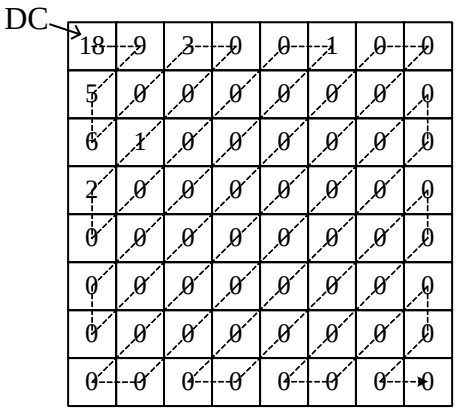


Figure 2.4. An example to illustrate the zig-zag scanning.

In addition to transform coding, temporal predictive coding using motion-compensated prediction [17-18, 37-39] is an important technique for removing temporal redundancy between adjacent frames. In motion-compensated prediction, a pixel is not coded directly; rather, its value is predicted from those of adjacent pixels in a previous frame, which is usually called a reference frame. This is motivated by the fact that pixels in successive frames have similar color values, thus it is wasteful of bits to specify the current value independent of the past values. Both P- and B-frames take advantage of motion-compensated prediction to exploit the temporal redundancy. However, the coding of P-/B-frames is much more complicated than that of I-frames.

Figure 2.2(a) also shows the key steps of motion-compensated prediction. In this case, both motion estimation and compensation are activated. In this encoder, each video frame is divided into blocks of a fixed size and each block

is coded using a combination of motion-compensated prediction and transform coding. For the sake of simplicity, block-based motion estimation is employed in most video coding standards in which a block is predicted from a previously coded reference frame. In practice, the block size for motion estimation may not be the same as that for transform coding. Generally, motion estimation is operated on a larger block known as a macroblock (MB). In most cases, the MB size is 16x16 pixels and the block size is 8x8 pixels. The motion estimation process uses the MB as a basic unit in which pixels of each MB in the current frame are compared on a pixel-by-pixel basis with pixels of the corresponding MB in the preceding I- or P-frame (reference frame). If a close match is located, then the relative displacement between the current MB and the best-matched MB in the reference frame is encoded. This is known as a motion vector. The predicted MB is obtained from the reference frame based on the motion vector using motion compensation. The prediction error of the current MB is then coded by transforming it, quantizing the DCT coefficients, and converting them into variable length code words using entropy coding. In principle, this procedure is similar to that described in encoding of I-frames.

In all video coding standards, neither the search area nor the specific search strategy are specified. Normally, only the contents of luminance are used in the search and a match is said to be found if the sum of the absolute errors in all pixels between the current MB and a possible candidate in the reference frame is the smallest. Instead, the standards only specify how the results of the search (motion vectors) to be encoded. Each motion vector consists of two motion vector components, the horizontal one first, followed by the vertical one,

which are coded independently. Since the physical area of coverage of a MB is small, the motion vectors can have large values. Besides, most moving objects are normally much larger than a single MB. In this situation, adjacent MBs are moved in a similar way as when an object moves. Hence, motion vectors are coded differently with respect to previously decoded motion vectors in order to reduce the number of bits required to represent them. Each component of the differential motion vector is then coded by using a variable length code word. To enhance the efficiency of motion compensation, half-pixel or quarter-pixel motion estimation is adopted in various video coding standards.

The main difference between B- and P- frames is that each B-frame can have more than one reference frame. As shown in Figure 2.5(a), frame *n* (B-frame) has frame *n-1* (I-frame) as the "past" reference frame and the following frame *n+2* (P-frame) as the "future" reference frame. Therefore, the process of motion estimation performs twice for B-frames. Owing to this extra motion estimation, the encoding of B-frames significantly increases the computational burden of the encoder. Moreover, there are two motion vectors for each MB in B-frames – forward and backward motion vectors. The encoder can adaptively select any of the forward, backward or their combined motion-compensated predictions for better prediction accuracy. Note that the use of bidirectional prediction necessitates the coding of frames in an order that is different from the original display order. This out-of-sequence coding is shown in Figure 2.5. Frame *n+2* should be encoded prior to frame *n* (B-frame). Therefore, it incurs extra encoding delay and is typically not recommended for real-time applications such

as video conferencing, though it can provide higher coding efficiency compared to P-frames.

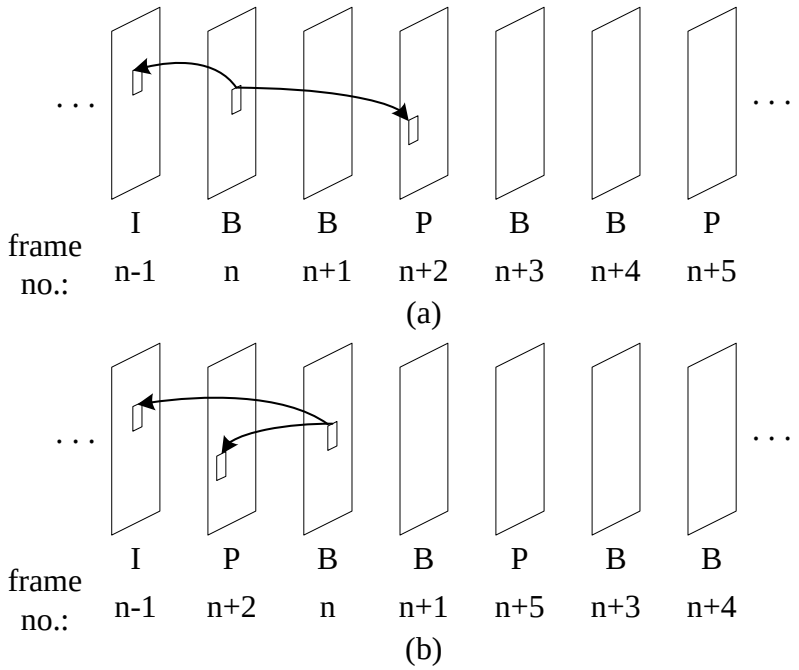


Figure 2.5. Display and encoding order of an I-B-P structure: (a) display order and (b) encoding order.

2.2.2 The hierarchical structure of MPEG bitstream

The video bitstream produced by an MPEG encoder is arranged in a hierarchical structure, as depicted in Figure 2.6. There are totally six layers: sequence layer, group of picture (GOP) layer, picture layer, slice layer, MB layer, and block layer.

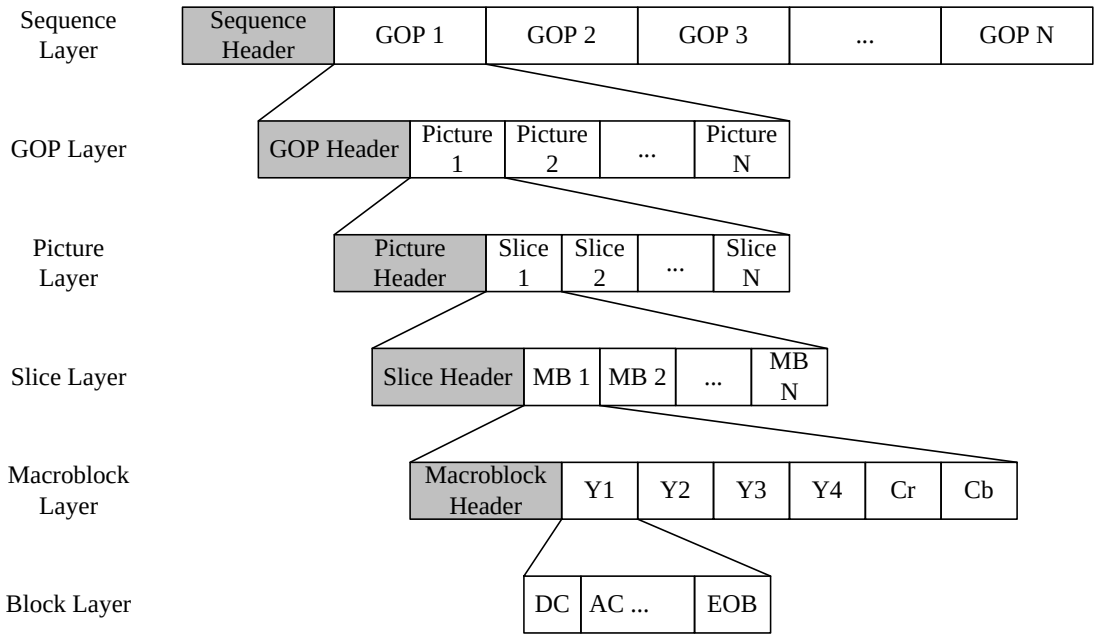


Figure 2.6. MPEG-2 hierarchical structure.

The highest syntactic structure of the coded video bitstream is the video sequence layer. The video sequence layer starts with a sequence header that contains the basic parameters of the whole video such as picture size, frame rate, pixel aspect ratio, chroma format indicating 4:2:0 or 4:2:2, and other global parameters, followed by group of pictures or frames. Each GOP is made up of a GOP header and a series of pictures that are in a continuous display order. The GOP layer is intended to facilitate random access, fast search, and other editing of the sequence. For this purpose, the first picture in a GOP must be an I-frame and it is also the only I-frame in each GOP. The remaining pictures are coded as P-frames or B-frames. Since each GOP is coded independent of previous and next GOPs, it enables a GOP to function as the basic unit for editing and random access. The GOP header contains a time code for synchronization and editing, which gives the hours-minutes-seconds time interval from the beginning of the sequence. This is editing-related information about the picture enclosed in the GOP and a number of picture-related

structures. Each picture consists of a picture header and several slices. The picture layer starts with a picture header indicating the frame type (I, P, or B) and a temporal reference number. This reference number gives its position within the GOP. The next layer is the slice layer. Each slice is a contiguous sequence of macroblocks. The slice layer can help to permit flexibility in signaling changes in some of the coding parameters so as to optimize the quality for a given bitrate or to control the bitrate. For example, a new quantization parameter could be specified in each slice header to overwrite the default one in the sequence header. Therefore, different quantization parameters can be used for different slices to control the bitrate. Another reason for defining a slice is to reset the variable length code to prevent error propagation within a picture. In the slice header, a slice start code with the specified pattern is included for synchronization, which could stop error propagation due to the use of variable length coding. For the first MB of each slice, the DC value is intra encoded rather than predicted from the previously encoded MBs as usual. In this way, the error of one slice can not be propagated to the next slice.

The MB layer is composed of a 16×16 luminance block and its corresponding chrominance block as shown in Figure 2.6. When the common YC_bC_r 4:2:0 color format is used, there are four luminance and two chrominance blocks in each MB. Since motion estimation is performed at the MB level, the motion vector obtained is stored in the MB header. Block, which is the lowest layer of the syntactic structure, consists of 8x8 pixels. DCT coding is applied at this block level.

2.3 Conventional video streaming system with VCR support

Owing to the use of motion-compensated prediction in the current video coding standards, the streaming server finds it difficult to provide VCR trick modes on MPEG encoded video. Figure 2.7 shows the architecture of the conventional video streaming system with VCR support. The video stream is pre-compressed using the MPEG-2 video coding standard [6, 17] and then stored in the storage devices. Upon client request, the streaming server retrieves the 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 both of the video streaming server and the client machine.

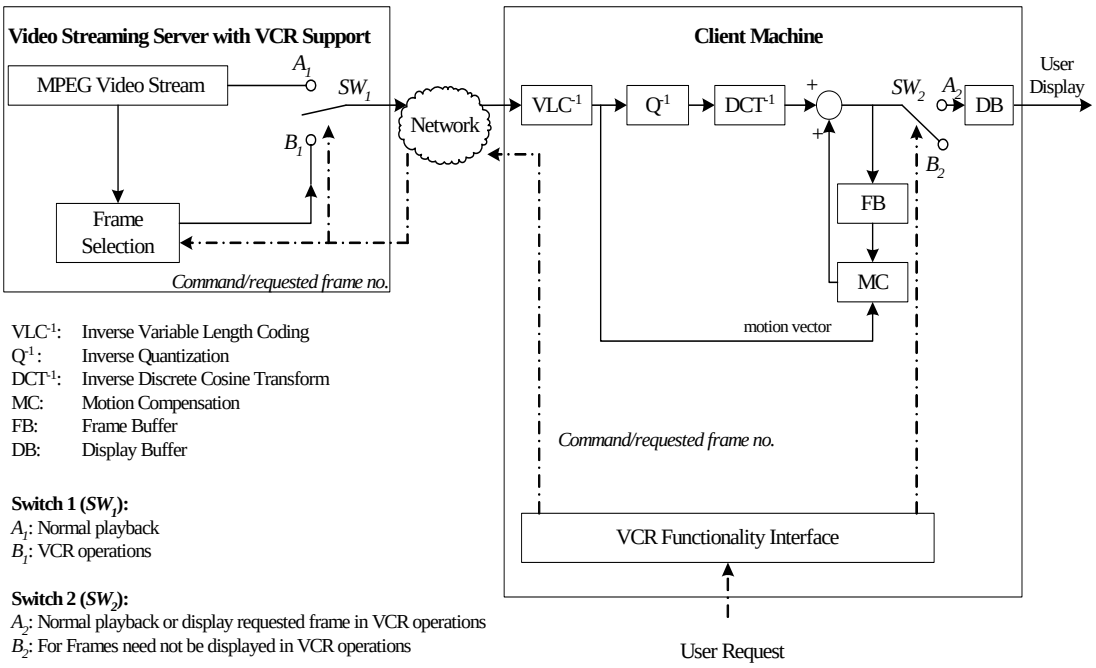


Figure 2.7. The conventional MPEG video streaming system with VCR support.

The client machine consists of an MPEG decoder and an interface for VCR functionality. The MPEG decoder is used to decode the incoming video stream

and deliver it to the appropriate display. The VCR interface is used to translate a user command and then interpret it from the remote control or keyboard to the appropriate signal. The user command includes the specific VCR operation (forward, backward, etc.) and the signal indicating the requested frame number that is sent to the decoder from the server. The interpreted signal then forwards to the server for appropriate actions. At the server side, a control function is built to receive the command and the requested frame number. Necessary frames are selected according to different VCR trick modes and sent to the decoder. The decoder reconstructs these frames and sends them to the display unit.

In the following discussion, backward playback is used as an example to illustrate the operation of this conventional streaming system with VCR support. We also provide experimental results to compare the average number of bits and MBs needed to be sent through the network and decoded at the client side to support the forward-play and backward-play operations, which represent the required network traffic and decoder complexity respectively. Note that B-frames are not used as references for later frames. It means they are not involved in decoding other frames. For simplicity but without loss of generality, we focus our discussions on the case that the MPEG bitstream contains I- and P-frames only.

In Figure 2.7, 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 Figure 2.7. 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 backward-play command at frame *n*, the next frame to be display is frame *n-1*. The client machine generates the backward-play command ($Command_{RP}$) 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 previously nearest I-frame to this requested frame. Since the next frame to be displayed 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 2.8. Suppose frame *n* is the start point of the backward-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 Figure 2.7. 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*. Next, frame *n-1* is decoded and stored in the display buffer so that this frame is displayed on the client screen. At that instant, SW_2 is connected to A_2.

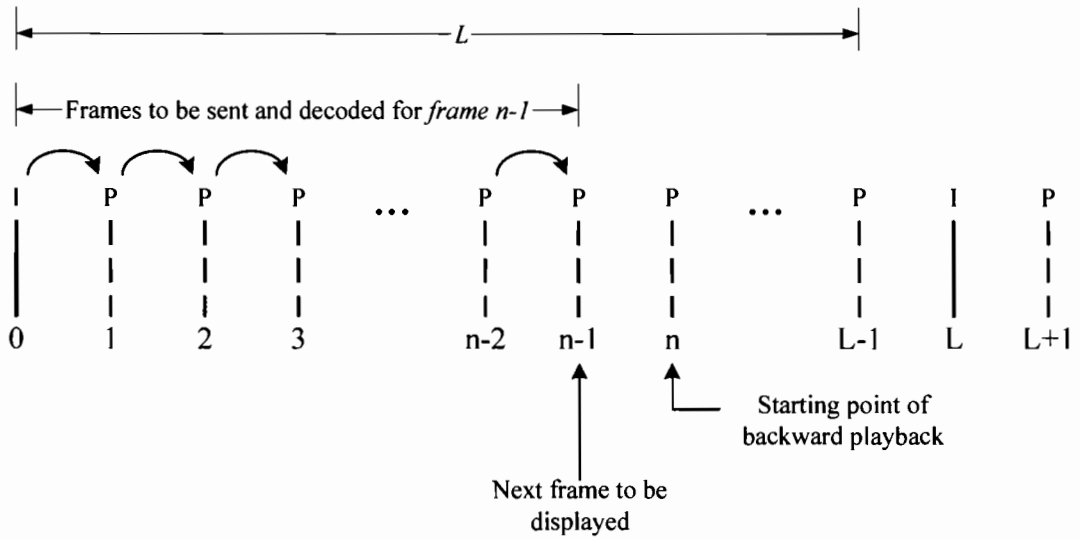
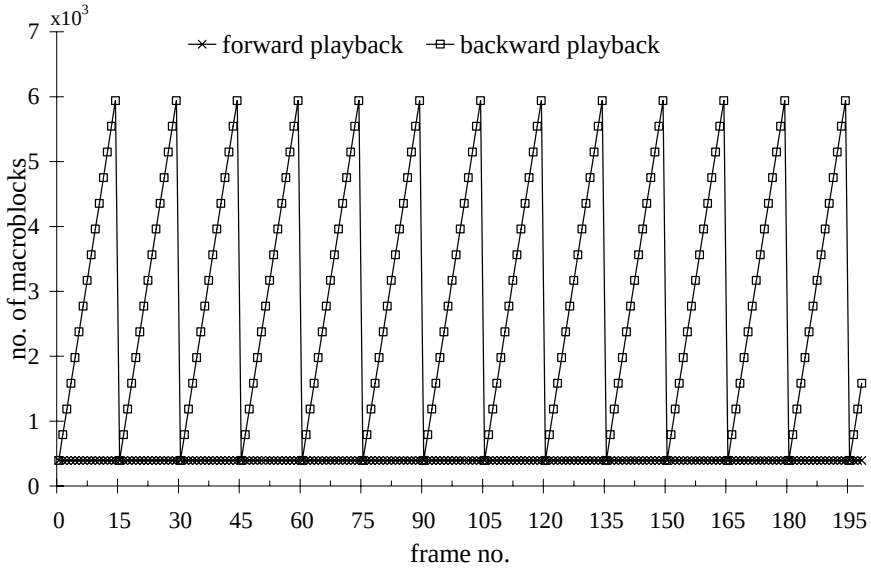


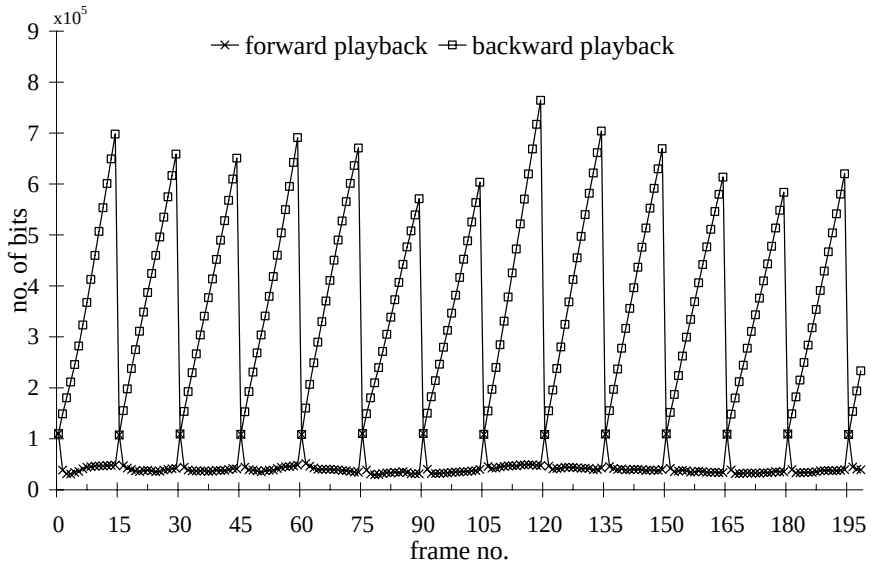
Figure 2.8. Example of backward-play in MPEG streaming system.

The decoder complexity and bandwidth requirement of the backward-play and forward-play operations are depicted in Figure 2.9 where the test video stream used for simulation is a "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 start point of the backward-play operation is at the end of the sequence. This figure shows that the server needs to send an excessively large amount of extra MBs and bits to the decoder to display one frame during backward playback. Thus, straightforward implementation of the backward-play operation requires much higher network bandwidth and decoder complexity than the forward-play operation. Besides, the higher network bandwidth and decoder complexity are distributed in an uneven way. The required MBs and bits will grow almost linearly as frame number increases inside a single GOP. Only one frame is required to display the first I-frame while the whole GOP is required to display the last P-frame of the GOP. This characteristic introduces more delay and requires an additional large buffer to ensure smooth display in streaming conditions. It can be

concluded that the conventional video streaming system with VCR support is not practical and therefore alternative smart schemes are desirable.



(a)



(b)

Figure 2.9. The comparisons of the (a) decoder complexity and (b) bandwidth requirement for sending the “Salesman” sequence over network with respect to forward-play and backward-play operations.

2.4 Problem formulations of digital VCR functions

Before improving the video streaming system with VCR support, it is useful to formulate the impact of VCR functionality in a systematic way. In the following, we provide some mathematical analyses to show the average number of frames to be sent over the network and decoded at the client decoder in order to support different VCR trick modes. For simplicity of the presentation, we again use an example in which the video bitstream 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 following subsections, the average number of frames to be transmitted and decoded for random access, fast forward, backward, and fast backward are denoted by $\overline{N}_{random}$, $\overline{N}_{ff}$, $\overline{N}_{b}$, and $\overline{N}_{fb}$ respectively.

2.4.1 Random access operation

In random access, the requested frame has an arbitrary distance from the current displayed frame. If it is an I-frame, the server only needs to send this frame, and the decoder can decode it immediately. On the other hand, if the requested frame is a P-frame, all the P-frames from the previously nearest I-frame to this requested frame have to be sent. Let us use the example in Figure 2.10 to illustrate this situation. Suppose that the length of all GOPs (N) in the video bitstream is fixed, and frame N_i is the random-access frame. In order to decode frame N_i, frames 0, 1, 2, …, N_{i-1} should also be sent from the server. As the result, the number of frames to be sent is N_i+1. Considering the random-

access points are uniformly distributed, the average number of frames to be sent can be formulated as

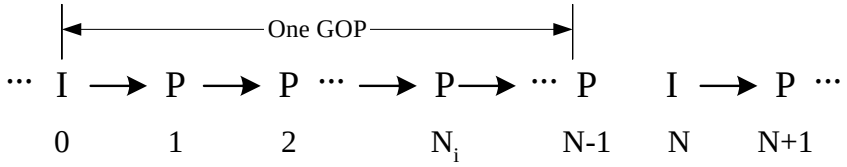


Figure 2.10 . Random Access in N-length GOP.

$$\overline{N}_{random} = \frac{0+1+2+...+(N-2)+(N-1)}{N} = \frac{\frac{N(N+1)}{2}}{N} = \frac{(N+1)}{2} \qquad (2.1)$$

For example, when N is equal to 15, $\overline{N}_{random} = 8$. This means that eight frames on average are transmitted over the network and decoded by the decoder for requesting one frame in the random-access mode.

2.4.2 Fast-forward/backward operation

Fast-forward playback is used to browse video content at a speed faster than at which it would usually play. On the other hand, fast-backward playback is the exact opposite of fast-forward playback, with video frames played backward at a user's desired speed-up factor. But they share a similar property. For instance, in Figure 2.11, the fast-forward and fast-backward operations request the same set of frames if the start point of the fast-forward operation is the end point of the fast-backward operation and their speed-up factor is equal. In this case, the average number of frames to be sent and decoded to support these VCR trick modes is identical. Thus, we will only make an analysis of the fast-forward operation and a similar analysis can be easily extended to the fast-backward operation. Assume that frame N_i is the current displayed frame in the fast-

forward operation, *k* is the speed-up factor, and all GOPs in the video bitstream have the same length *N*. Since the next displayed frame is N_{i+k}, the server needs to send the frames N_{i+1} N_{i+2} N_{i+3} … N_{i+k}. In this case, *k* frames will be received by the client machine. These extra frames are required to be decoded in sequence, though only N_{i+k} will be shown on the display.

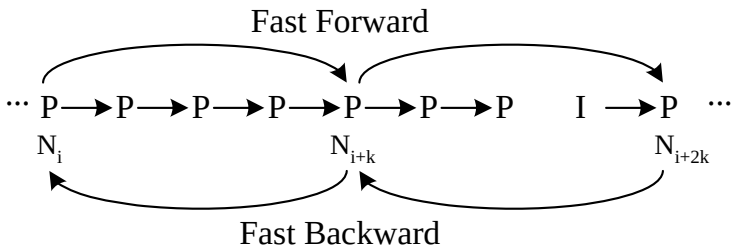


Figure 2.11. Fast forward/backward playback with a speed up factor of k.

In practice, the server may not need to send so many frames due to the existence of I-frames. For example, in Figure 2.11, assume that the current displayed frame is N_{i+k}, and N_{i+2k} is the next frame to be displayed in the fast-forward mode. Obviously, N_{i+k+1} and N_{i+k+2} are not needed because they are not necessary for the decoding of $N_{i+2k.}$ In this example, the server only transmits N_{i+2k-1} and N_{i+2k}. It is difficult to derive a simple formula to show the impact of the fast-forward operation on the decoding complexity and network traffic since the start point of the fast-forward operation can be any frame inside a GOP. However, it is reasonable to consider that the start point of the fast-forward operation is an I-frame because we can always jump to the nearest I-frame first and will not cause annoying effects in viewing the video in most practical applications. By making use of this assumption, after k/L GOPs, where L=GCD(k,N) stands for the greatest common divisor of k and N, the frame to be displayed will be an I-frame again. In other words, the decoding pattern will

repeat every k/L GOPs, including LCM(k,N) frames, where LCM(k,N) is the least common multiple of k and N. According to this periodicity, the analytical closed-form formula can be derived by classifying different combinations of k and N into three classes as follows. Note that mod is the modular operation.

1. k>N, k mod N=0: This is the simplest case. All the P-frames are dropped, and only the non-skipped I-frames are sent and decoded. No extra frames need to be sent for decoding the I-frames. Therefore, $\overline{N}_{ff} = 1$.
2. k>N, k mod N≠0: For every k/L GOPs, the decoding pattern will repeat. Therefore, during each period, N/L frames are requested for display. For the i^{th} requested frame in each period (i=0 to N/L-1), there are (i×k) mod N+1 frames in total are sent and decoded. The average number of frames to be transmitted and decoded for displaying one frame is:

$$\begin{aligned}\overline{N}_{ff}(k,N) &= \frac{L}{N}\sum_{i=0}^{N/L-1}((i\times k)\bmod N+1)\\ &= \frac{L}{N}\sum_{i=0}^{N/L-1}(i\times L+1)\end{aligned} \tag{2.2}$$

3. 2≤k≤N-1, N mod k≠0: In a GOP with an I-P structure, a P-frame needs not be sent only if all the following P-frames inside the same GOP will not be displayed at the client decoder. Therefore, in the first GOP (assume the start point is an I-frame), the number of frames needs not be sent is N mod k. Similarly, the number of the P-frames which need not to be sent in the j^{th} GOP, where 1≤j<k/L, is $(j\times N) \bmod k-1$. Thus, the total number of frames that could be skipped in the k/L GOPs is

$$N_{skip}(k,N) = \sum_{j=0}^{k/L-1}((j\times N) \bmod k - 1) = \sum_{j=1}^{k/L-1}(j\times L - 1) \tag{2.3}$$

When N and k are coprime, i.e., L is equal to 1, (2.3) can be written as

$$N_{skip}(k,N) = \sum_{j=1}^{k-1}(j-1) \tag{2.4}$$

The average number of required frames can be obtained by

$$\overline{N}_{ff}(k,N) = \frac{\frac{k}{L}\times N - N_{skip}(k,N)}{\frac{k}{L}\times\frac{N}{k}} = k - \frac{L}{N}N_{skip}(k,N) \tag{2.5}$$

In summary, the closed-form of the average number of frames to be sent and decoded for a requested frame in fast-forward/backward can be expressed as:

$$\overline{N}_{ff/fb}(k,N) = \begin{cases} 1 & k = 1 \text{ or } k \bmod N = 0 \\ k - \frac{L}{N}\sum_{j=1}^{k/L-1}(j\times L - 1) & 2 \le k \le N-1 \\ \frac{L}{N}\sum_{j=1}^{k/L-1}(j\times L + 1) & k > N, k \bmod N \ne 0. \end{cases} \tag{2.6}$$

where L is the greatest common divisor of k and N.

2.4.3 Backward operation

When a backward operation is requested, it always lasts for a few seconds or minutes. During the period of backward playback, all frames between the current frame and the previously nearest I-frame 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.

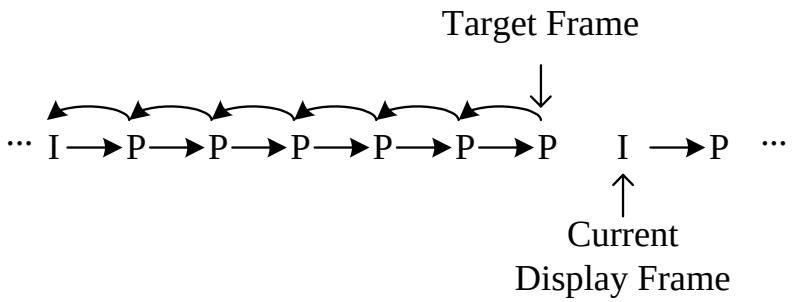


Figure 2.12. Backward operation starting from the end of GOP.

Assume a backward operation is required when the current displayed frame is an I-frame as shown in Figure 2.12. The target frame to be displayed is the P-frame before it, which is the last frame of the previous GOP. In this case, all frames of the GOP are required in order to display the target one. Supposing that all the GOPs in the video stream have the same length N, the server needs to send N frames for the target frame, N-1 frames for the frame before it in the next step and so on. Thus, the total number of frames to be sent for playing one GOP in reverse order (N_b) is

$$N_b = 1 + 2 + 3 + ... + N = \frac{N(N+1)}{2} \tag{2.7}$$

As a result, the average number of frames to be sent and decoded for each frame in the backward-play operation is

$$\overline{N}_b = \frac{N_b}{N} = \frac{N+1}{2} \tag{2.8}$$

Equations (2.1), (2.6), and (2.8) form a theoretical basis for the subsequent analyses of the efficient algorithms on digital VCR.

2.5 Previous algorithms for enabling VCR functionality

In recent years, a number of techniques have been introduced for implementation of VCR support for MPEG compressed video in streaming applications. Generally, they can be classified into three categories: dynamic transmission algorithms, transcoding algorithms, and multiple bitstream algorithms. These algorithms are introduced and discussed in the following subsections.

2.5.1 Dynamic transmission algorithms

In an MPEG video stream, each GOP starts with an I-frame followed by a number of P- and B-frames. This provides a random-access unit for VCR functions. Since an I-frame is encoded independently, storing and transmitting a video stream based on GOPs is efficient. It is practical to support fast scan operations based on GOPs. By considering this, Lee *et al.* [40] proposed a dynamic scheduling scheme to support fast scan operations of a stored video with a certain speed-up playback rate. Whenever a user requests fast scans, the scheme dynamically writes a transmission schedule to a server.

The basic concept of GOP-skipping is illustrated in Figure 2.13. Figure 2.13(a) shows the GOPs selected for normal playback. In normal playback, a server selects all GOPs in a video stream and transmits them to a client over the network. When a user wants to view some frames quickly in the video stream to find a specific scene, the user requests fast scan operations. For instance,

Figure 2.13(b) shows a scenario that a client requests 2×Fast Forward. The server immediately stops the transmission of video data to the client and writes a transmission schedule. The server constructs the set of GOPs selected by a given speed-up factor of 2 and prepares the new transmission schedule. Finally, the server restarts the transmission of video data using the new schedule. Figure 2.13(c) and Figure 2.13(d) shows the situation for jump forward and 3×Fast Backward.

With this “GOP-skipping” concept for VCR operations, dynamic transmission scheduling is used in [40] to support fast scan functions for mxFF or mxFB. The proposed scheme uses a bandwidth smoothing to prepare the transmission schedule of a video stream from a selected frame position. A bandwidth smoothing is applied to the set consisting of multiple GOPs selected by “GOP skipping”. Note that in the fast-backward operation, only the GOPs are played in the reverse direction. In each GOP, the frames are played back in the forward direction as usual. This is different from the desired backward operation we have discussed. Besides, since GOP is used as the fast scan unit in this scheme, the VCR functions can not access all desired frames. When the GOP size is large, this fast scan is quite rough and not adequate for browsing. In fact, the major contribution of this paper is the dynamic transmission scheme for bandwidth smoothing.

Similarly, there are also several other research studies [41-43] on providing VCR functionality in video-on-demand (VOD) systems. Since multicast delivery is the most common solution to reduce the cost in large video-on-demand

systems, these frameworks mainly deal with the difficulties when the multicast delivery is employed in the VOD system. Various methods of buffer allocation, video segmentation and transmission schedule have been proposed to minimize the system, and network resources for supporting full VCR functionality. For example, the scheme proposed in [43] aims to serve a large number of users, and reduce the initial delay and additional channels while supporting VCR services. None of these studies [40-43] has investigated the dependencies among frames. Therefore, the network traffic is still huge when only a single user is considered.

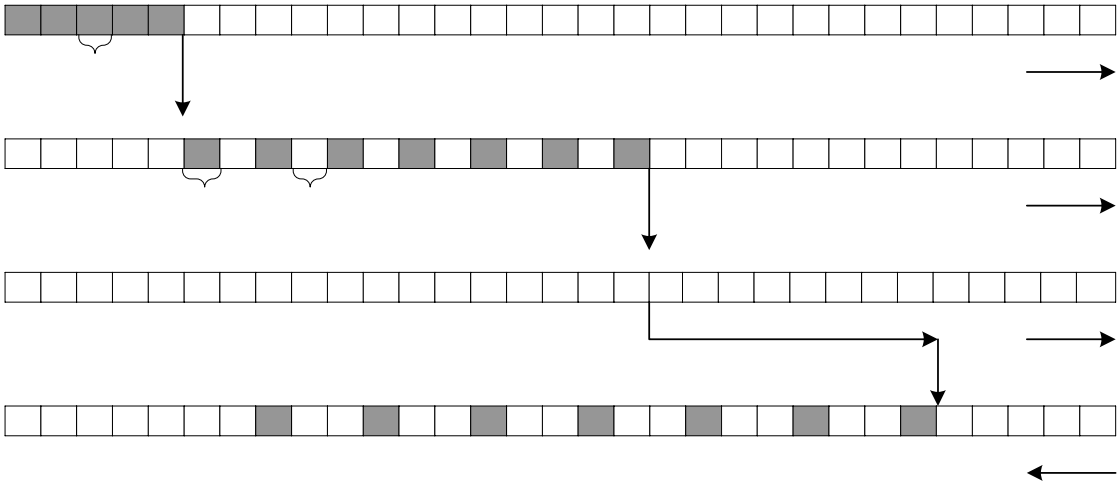


Figure 2.13. Support of VCR functions based on GOP skipping.

2.5.2 Transcoding algorithms

Recently, some transcoding approaches have been proposed to support VCR functionality for MPEG video [5-6, 9-10]. By using the transcoding approach, Chen *et al.* [44] addressed the problem of supporting interactive playout, both forward and backward, of an MPEG encoded video stream at a playback device. A method was proposed to convert the incoming standard MPEG bitstream with

an I-B-P structure into a local bitstream with an I-B structure at the playback device, which then enables the device to support interactive playout even when the buffer space available is constrained. Specifically, a stream conversion scheme is employed to convert all the retrieved P-frames into I-frames after the decompression and playout of each P-frame at the client. An example is shown in Figure 2.14 to illustrate this process. When the bitstream is forward playing initially, after one P-frame, e.g. frame 4, is retrieved and decompressed, it cannot be displayed immediately until the two B-frames, frame 2 and frame 3, are received and decoded, as shown in Figure 2.14(a) and Figure 2.14 (b). At the same time, this decompressed P-frame (frame 4) is encoded as an I-frame and stored in the secondary storage. This P-to-I conversion is performed after decompression, thus there is no additional decoding effort. Besides, there is no extra motion estimation and compensation required for encoding this frame into an I-frame.

This P-to-I frame conversion breaks the inter-frame dependencies between the P-frames and the I-frames. After the frame conversion and reordering, the motion vector swapping technique discussed in [45, 46] is employed to obtain the reverse motion vectors in order to perform the backward-play operation in the new I-B stream. As shown in Figure 2.14(a), when the user stops at frame 14 in the normal playback process and issues a backward-play command, the frames to be displayed are frame 13, frame 12, frame 11, etc. This backward - play order is depicted in Figure 2.14(d). Since frame 14 is a B-frame, the decoded frames 13 and 16 are currently stored in the buffer as the reference frames. As a result, frame 13 can be displayed instantaneously. Then, the next

target is frame 12, which depends on frame 10 and frame 13. With the help of the P-to-I conversion in Figure 2.14(c), frame 10 is encoded as an I-frame and stored in the secondary storage during forward playback. Therefore, frame 12 can also be displayed by retrieving frame 10 from the storage rather than sending and decoding the previous frames up to the I-frame (frame 1) in the conventional approach. Similarly, the subsequent frames can be easily viewed in backward-play operation.

With this new I-B bitstream, VCR operations including random access and fast-forward/backward can be also implemented straightforwardly. Since this approach is based on the download mode, all the VCR operations can only be performed among those frames that have been downloaded and displayed. As explained in section 1.3, full file transfer in the download mode usually suffers long and perhaps unacceptable transfer time. Besides, it requires extra complexity in the decoder to perform the P-to-I conversion and higher storage cost to store the local bitstream in the client.

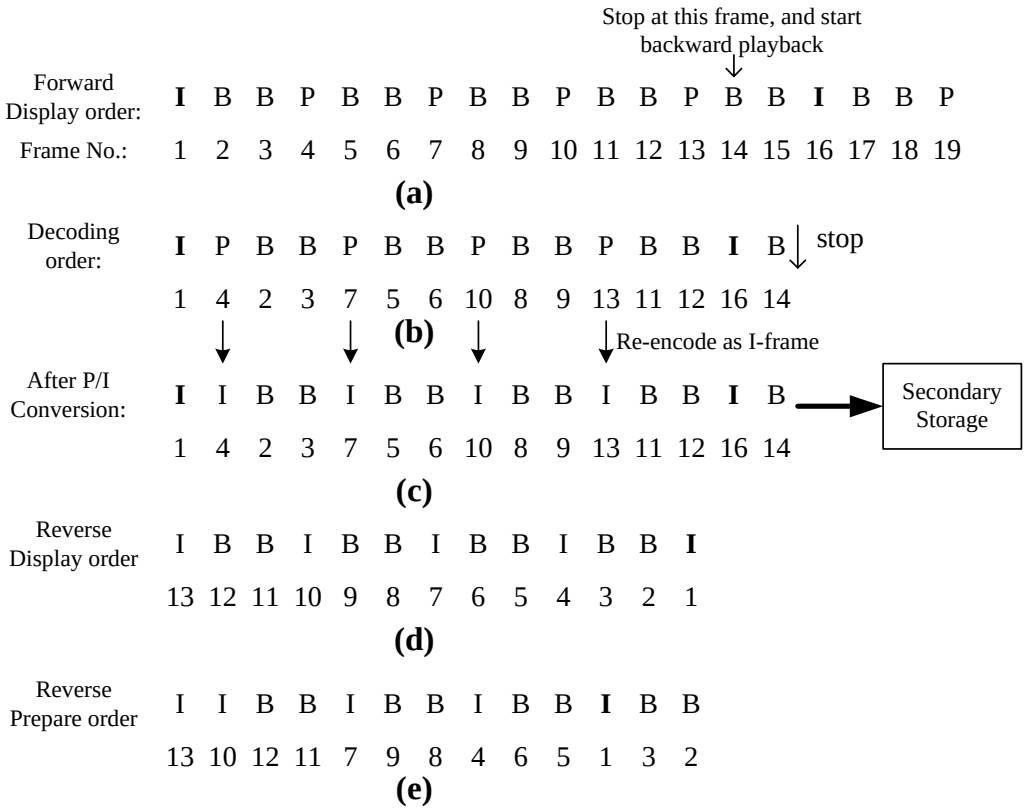


Figure 2.14. P-to-I conversion at the client.

In [46], Wee *et al.* presented a method that divides the incoming I-B-P bitstream into two parts: I-P frames and B-frames. A transcoder is then used to convert the I-P frames into another I-P bitstream with reverse frame order in a compressed domain. To achieve this, a method of estimating the reverse motion vectors for the new I-P bitstream with reverse frame order is necessary. On the other hand, for B-frames, there are forward and backward motion vectors referring to the “past” and “future” reference frames correspondingly. We only need to Huffman decode them, extract and swap the pair of motion vectors in B-frames, and re-encode them by using Huffman coding. The architecture of reverse transcoding is shown in Figure 2.15.

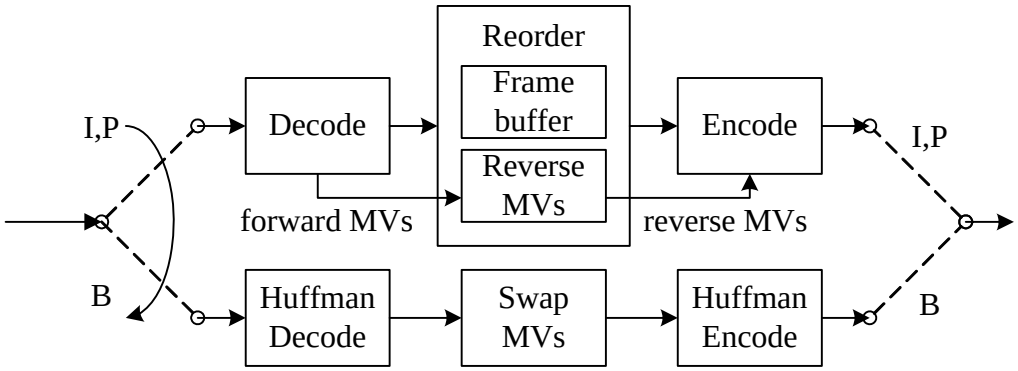


Figure 2.15. The architecture for reverse transcoding.

Motion estimation is known as a rather time consuming process. In order to reduce the computation involved in the motion estimation of the reverse I-P bitstream, several methods of estimating the reverse motion vectors for the reverse bitstream based on the forward motion vectors of the original I-P bitstream were discussed in [46, 47].

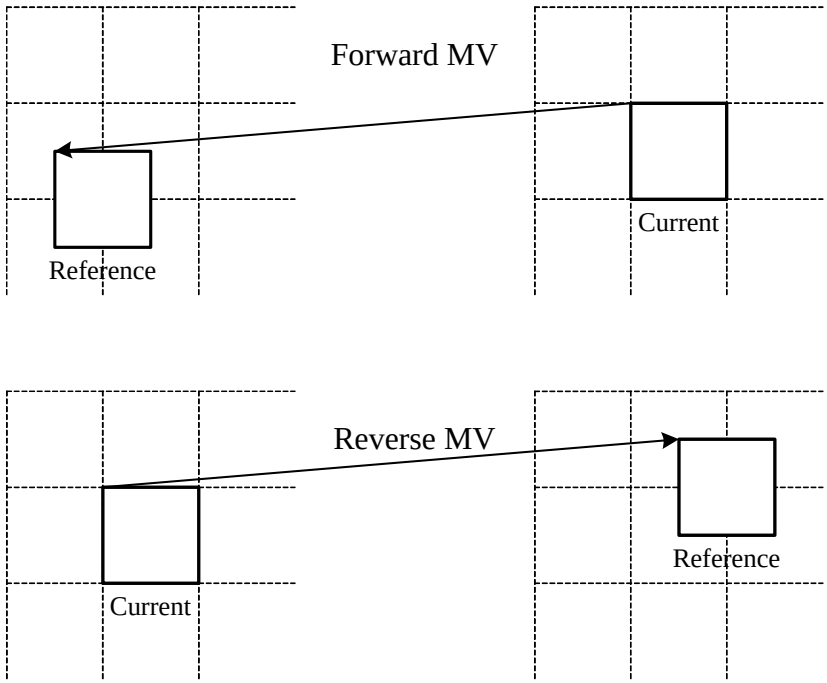


Figure 2.16. Reverse motion vectors by using the in-place reversal algorithm.

In the in-place reverse algorithm, each reverse motion vector is estimated from the forward motion vector in the corresponding spatial location. As shown in

Figure 2.16, the horizontal and vertical components of the corresponding forward motion vector are negated to estimate the horizontal and vertical components of the reverse one. This algorithm is easy to implement, requires little complexity and performs well for the interiors of objects that are stationary or undergo uniform translational motion. However, in the case of large motion areas, it frequently produces inaccurate reverse motion vectors near the object boundaries.

Another category of reverse motion estimation is to exploit the forward motion vectors of the eight neighboring MBs and the one in the corresponding spatial location. It includes maximum-overlap and weighted-overlap algorithms. In these algorithms, each forward motion vector in the neighborhood is assigned a weighted value that represents its relevance to the current MB for which we must estimate the reverse motion vector. As shown in Figure 2.17, the weighted relevance of each forward motion vector is defined as the size of the overlapping area between the current MB and the motion-compensated MB translated by the forward motion vector. In Figure 2.17, the weighted value for motion vectors mv_1, mv_5, and mv_9 are shaded. The maximum-overlap algorithm selects the forward motion vector with the largest weight, e.g. mv_1 in this example, and then negates its horizontal and vertical components. These components are used as the estimate of the reverse motion vector. In the case of very small translations, the maximum-overlap algorithm provides the identical estimate as the in-place reverse algorithm. But, the maximum-overlap algorithm gives better estimates of the reverse motion vectors near the object boundaries in case of larger translations. The weighted-overlap algorithm also

makes use of the corresponding forward motion vector and its neighbors. It constructs the reverse motion vector by summing up all the nine motion vectors with their corresponding weighted values and negating the components of the resulting vector through the following equation.

$$\text{ReverseMV} = -\frac{\sum_{i=1}^{9} \text{weight}_i \times \text{mv}_i}{\sum_{i=1}^{9} \text{weight}_i} \tag{2.9}$$

The weighted-overlap algorithm produces inaccurate motion vectors at the boundaries of objects undergoing translational motion, but it is better suited for areas with rotational motion or camera zooms.

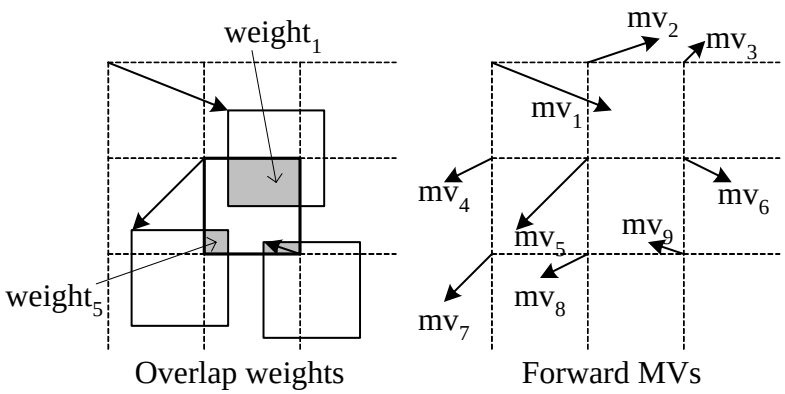


Figure 2.17. Overlap weights with local neighborhood.

To make further improvement, the forward motion vectors as well as the corresponding forward residual error are used to the estimate the reverse motion vector in a number of ways. For example, the residual information can be employed to generate more accurate weights for the forward motion vectors in the overlap algorithms. The energy in the residual error of a particular block can verify the reliability of its forward motion vectors. For instance, a high-energy residual error may signify that the motion vector is not accurate enough

to the actual motion in the sense. By doing such verification, the weight can be adjusted appropriately.

The goal of various transcoding as discussed above is to use a given MPEG data stream to create a new MPEG data stream that, when decoded, displays the video frames in the desired playback direction and speed-up factor. For instance, as compared to the baseline spatial-domain transcoding method, which requires full MPEG decoding and re-encoding to generate the backward-play MPEG bitstream, the use of the aforementioned reverse motion estimation approaches can obviously reduce the computational complexity of the transcoder. However, the transcoding process still requires a great deal of computation for the decoding and re-encoding of residuals. Besides, the reuse of the incoming motion vectors results in non-optimized outgoing motion vectors. This inaccuracy causes quality degradation. In other words, the reconstructed frames in the transcoded bitstream are not exactly the same as the original ones. Moreover, when the video sequence is encoded by the latest MPEG-4 part10/H.264 standard [9], it faces some additional technical challenges that have not yet been satisfactorily resolved. The details will be discussed in the next chapter.

2.5.3 Multiple bitstream algorithms

In [48], a dual-bitstream system was proposed to deal with the problem in different VCR trick modes of the MPEG video streaming system. This approach adds a backward bitstream in the server in addition to the traditional forward

bitstream. The generation of the backward bitstream is simply encoding the video in reverse order. This process is done off-line. As shown in Figure 2.18, the backward bitstream is arranged so that the I-frames in the backward bitstream are interleaved between I-frames in the forward bitstream. Meta files are generated to record the locations of the frames in the compressed bitstream so that the server can access the I-frames directly and easily. These meta files facilitate the switching between the bitstreams during the VCR operations.

When a user issues a backward-play operation during normal playback, it is convenient for the server to stream the backward bitstream. Based on the dual-bitstream structure, the authors in [48] also proposed a frame-selection scheme at the server to minimize the required network bandwidth and the decoder complexity involved in other VCR operations. This scheme determines the frames to be transmitted over a network by switching between the two bitstreams based on a least-cost criterion. 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. It actually measures the distances from the next requested frame to the current displayed frame, the nearest I-frame in the forward bitstream and the nearest I-frame in the backward bitstream. It then picks the frame with the minimum distance as the reference frame to the next requested frame to initiate the decoding. Since this selected reference frame has a shorter distance to the next requested frame, a smaller number of frames need to be sent. Figure 2.18 show an illustrative example. In this example, assume that the previous mode was normal forward playback. The user issues a fast-backward play operation with a speed-up factor of 6 at frame 20, which

needs to display a sequence of frame numbers 14, 8, and 2. Frame 14 will be decoded from the forward bitstream directly since it is an I-frame. For frame 8, since the distance between frame 7 in the backward bitstream (possible reference frame) and the requested frame (frame 8) is less than the distance between the current frame (frame 14, another possible reference frame) and frame 8, frame 7 in the backward bitstream is selected as the reference frame. Frame 7 in the backward bitstream (I-frame) is sent and decoded as the approximation of frame 7 in the forward bitstream (P-frame). Then, frame 8 in the forward bitstream is sent and decoded for display. Similarly, frame 2 is decoded from frame 0 and frame 1 in the forward bitstream.

This approximation will introduce the drift problem since the mismatch of the reference frame exists. Besides, the drift will not only be restricted to the frame at the switching location, but will also propagate to other frames if they are temporally predicted from the drifted frame. Therefore, in [48], two drift-compensated bitstreams D_n^{FR} and D_n^{RF} are added to minimize the drift problem. D_n^{FR} is used for switching from the forward bitstream to the backward bitstream while D_n^{RF} is used for switching in the opposite direction. For example, frame 7 in D_7^{RF} contains the prediction error from frame 7 in the backward bitstream to frame 8 in the forward bitstream. When frame 8 is required in the previous example, after sending frame 7, D_7^{RF} is sent instead of frame 8 in the forward bitstream. The drift compensation process can reduce the drift in bitstream switching. However, the dual-bitstream scheme incurs a huge storage requirement of the server. This problem becomes more serious with the drift-compensated bitstreams.

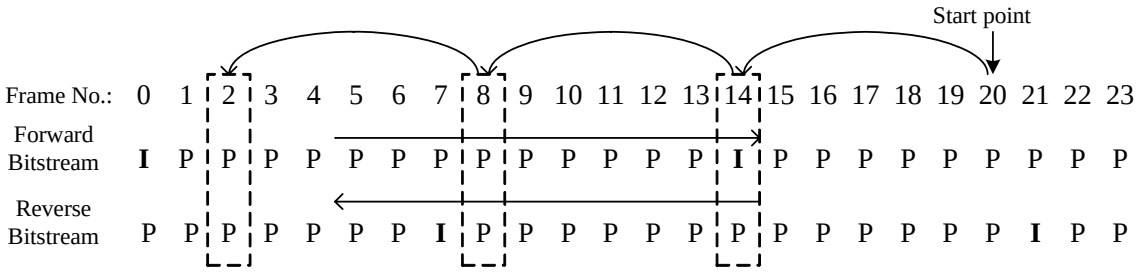


Figure 2.18. The structure of dual-bitstream scheme.

Huang [49] proposed a new dual-bitstream technique for VCR functionality and a transcoding technique for the motion vectors between the dual bitstreams. This technique can solve the drift problem and further reduce the required network bandwidth and decoder complexity. A direct reference bitstream is used to replace the backward bitstream in [48]. As shown in Figure 2.19, the positions of I- and P-frames in the two bitstreams are the same as the original dual-bitstream scheme. In the direct reference bitstream, some I-frames in the forward bitstream are reused. The other P-frames in the direct reference bitstream are directly referenced to their nearest I-frame, either in the forward-encoded bitstream or the direct reference bitstream. In this way, the number of decoding frames to access any I- or P-frame in the VCR operations is restricted to 1 or 2 and no approximation is involved in the process. Besides, the transcoding technique developed in this work [49] was developed to derive the direct reference bitstream with less computation complexity. However, owing to the longer prediction distance in the direct reference bitstream, the correlation between P-frames and their reference I-frames decreases and thus the coding efficiency is reduced. Besides, in terms of the decoder complexity and network traffic, this approach works well in random access, fast forward, and fast

backward operations but performs worse in backward playback compared to the original dual-bitstream scheme.

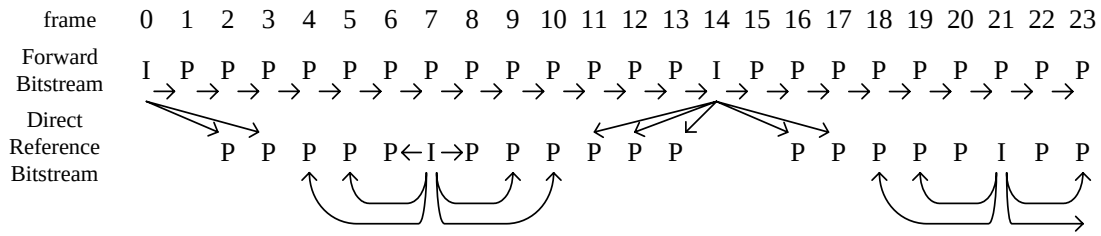


Figure 2.19. The structure of improved dual-bitstream scheme.

Omoigui *et al.* [50] investigated some possible client-server time-compression implementations for fast multimedia browsing. There are two possible time-compression techniques available for video. The first one involves dropping some frames regularly according to the compression rate. For example, when a fast-forward operation with a speed-up factor of 2 requires the compression rate of 50%, and half of the total frames are dropped. The second technique is to change the rate at which video frames are rendered. Thus to get a speed-up factor of 2, the frames are displayed at twice this rate. The main disadvantage of this approach is that it is computationally more expensive for the client, as the decoder has to decode twice as many frames in the same amount of time.

The time compression proposed in [50] could be implemented by storing multiple pre-encoded bitstreams with different temporal resolutions. A bitstream with suitable temporal resolution according to the user's request is then sent. This approach does not introduce excessive network traffic but the speed-up granularity is limited by the number of pre-stored bitstreams, and it forces all users to rely on the author's judgment as to speed-up granularity. Besides,

additional storage is unavoidable at the server since the number of selected pre-encoded bitstreams increases.

2.6 Chapter summary

In order to solve the difficulties of implementing digital video browsing, many recently proposed algorithms have been reviewed in this chapter. We started this chapter by reviewing the compression techniques employed in current video standards. The redundancy exploited by these compression techniques provides good compression effects. The compression techniques also produce dependencies among the frames of the coded bitstream. These dependencies complicate the implementation of the VCR functionality in digital video and create an obstacle for interactive video browsing. The straightforward implementation of a conventional video streaming system with VCR support was also discussed. By introducing the structure of the conventional system, we showed that when users request various VCR modes, it may result in much higher network traffic than the normal-play mode. Next, we systematically analyzed the impact of the VCR functionality on the network traffic and the decoder complexity in the scenario of video streaming. The formulations were devised in terms of the average number of frames to be transmitted to display one frame for different VCR operations. We also reviewed several algorithms for providing interactive video browsing. In dynamic transmission algorithms, they mainly deal with the video stream segmentation and channel manipulation techniques without investigating the characteristics of the MPEG bitstream. These algorithms are only restricted to the multicast VOD systems.

Transcoding algorithms always convert the received bitstream into other formats either in the client or the server for the implementation of VCR functions. However, the transcoding algorithms implemented at the client side require extra storage and complexity in the client machine, and are well suited for a “download mode” rather than a “streaming mode”. On the other hand, the algorithms in the server also incurs huge complexity. Besides, we also mentioned multiple bitstream algorithms which store two or more bitstreams in the server and select the necessary information from the multiple bitstreams to access the desired frame in interactive operations. This type of algorithm demands a huge storage requirement of the server. In some cases, some quality degradation also occurs when bitstream switching is required. The reviews of various algorithms in this chapter indicate that these methods are still primitive, and there is plenty of room for improvement. Therefore, in the following chapters, we examine the possibility of improving the reverse transcoder and the use of compressed-domain processing for providing an efficient and effective video streaming system with VCR support.

Chapter 3 Fast Motion Estimation and Mode Decision for H.264 Reverse Transcoding

3.1 Introduction

In the last chapter, we have shown the impact on one of the most common VCR functions - backward playback in video coding standards. One straightforward approach of performing backward playback on a compressed video bitstream requires decoding the video, storing the uncompressed video frames in the decoder, and displaying them in reverse order. This approach needs a significant amount of memory in playback devices. A more efficient reverse transcoding architecture has been presented in the previous chapter [46-47]. The reverse transcoder in the server is to convert I–P frames into another I–P bitstream with reverse order. When a playback device decodes this reverse-encoded bitstream, backward playback can be achieved. An important problem that arises in the reverse transcoder is to compute reverse motion vectors, which induces heavy computation complexity of the transcoder and may prohibit the use of the reverse transcoder. It is noted that transcoding from the forward bitstream to a new I-P bitstream in reverse direction is also required in the dual-bitstream method [48-49]. To expedite the transcoding process, 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 has been suggested in [47]. The reverse motion estimation in [47] is implemented in the MPEG-2 video

standard. Nowadays, the MPEG-4 part10/H.264 standard [9, 53] (simplified as H.264 later) achieves considerably higher coding efficiency than the previous standards. It is also highly desirable to provide backward playback for this latest standard. However, the H.264 adopts variable block size motion estimation (VBSME) with mode decision in the encoding process and it further complicates the process of reverse transcoding. In this chapter, we thus design a fast motion estimation algorithm with mode decision to speed up the reverse transcoding process.

Part of the contents of this chapter have been published in references [51-52]

3.2 Variable block size motion estimation of H.264

In the work on the new H.264 standard, a number of advanced technical developments have been adopted. For instance, the H.264 standard supports motion estimation using different block sizes such as 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4. The aim is to provide high-precision motion estimation for each macroblock (MB), and hence increase coding efficiency. These different block sizes, that are also called modes, are actually arranged into two-level hierarchy in a MB as depicted in Figure 3.1. The first level (L1) contains block size of 16x16, 16x8, and 8x16 while the second level (L2) is defined as P8x8 mode, of which each 8x8 block can be one of submodes such as 8x8, 8x4, 4x8, or 4x4.

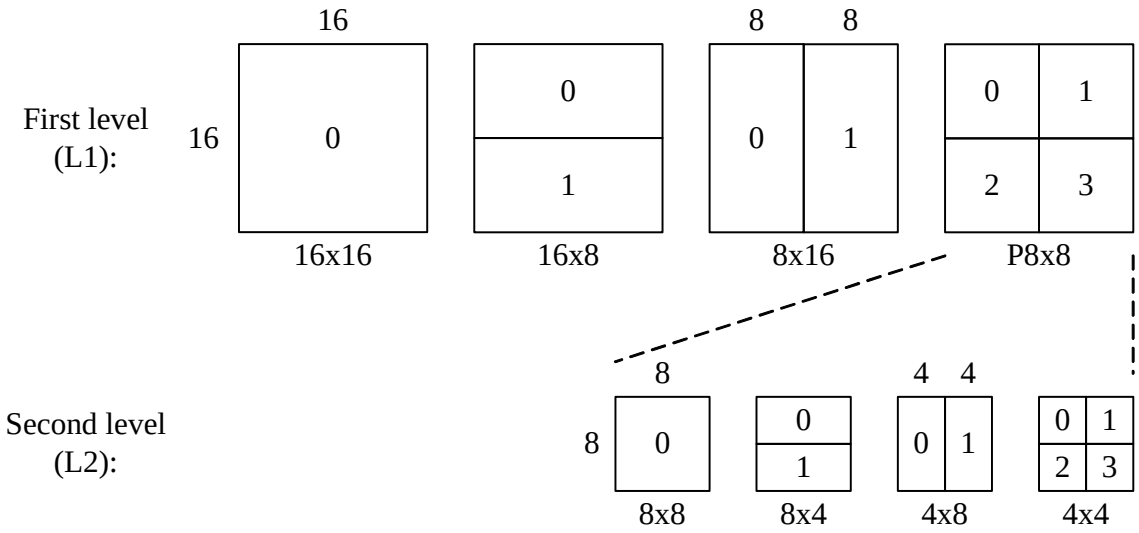


Figure 3.1. Various modes and submodes of VBSME in H.264.

In H.264, the current JVT reference software [53] is based on a Rate-Distortion Optimization (RDO) framework for both motion estimation and mode decision. Motion estimation is separately considered from mode decision and is firstly performed for all modes and submodes independently. For each mode or submode, the motion vector and reference frame are selected by minimizing the cost

$$J_{motion}(RMV, \lambda_{motion}) = SAD(s,r) + \lambda_{motion} \cdot R_{motion}(MV - PMV) \quad (3.1)$$

where *PMV* is the motion vector used for prediction, λ_{motion} is the Lagrangian multiplier for motion estimation, $R_{motion}(MV - PMV)$ is the estimated number of bits for coding the candidate motion vector *MV*, and SAD(*s*,*r*) is sum of absolute differences between the original block *s* and its reference block *r*.

After motion estimation for each mode, a RDO technique is used to select the best mode. This process is called mode decision. The general equation for RDO is given by

$$J_{\text{mode}}(s, c, MODE, \lambda_{\text{mode}}) = SSD(s, c, MODE) + \lambda_{\text{mode}} \cdot R_{\text{mode}}(s, c, MODE)$$

(3.2)

where λ_{mode} is the Lagrangian multiplier for mode decision, *MODE* is one of the following candidate modes: 16x16, 16x8, 8x16, and P8x8, *SSD* is sum of the squared differences between the original block *s* and its reconstructed block *c*, and $R_{\text{mode}}(s, c, MODE)$ represents the number of coding bits associated with the chosen mode. To compute the reconstructed block *c* used in (3.2), the encoding and decoding processes are required. That is to say, for each candidate mode, with the previously determined best motion vector, transform, quantization, inverse quantization, and inverse transform are performed on the original block *s*. Variable length coding is then performed as well to get the number of coding bits $R_{\text{mode}}(s, c, MODE)$. The mode having the minimal cost is decided as the best one.

It is noted that submode decision for P8x8 mode is carried out for each 8x8 block first. Equation (3.2) is evaluated with *MODE* indicating a candidate submode chosen from 8x8, 8x4, 4x8, and 4x4 in L2. *s* and *c* represents the original and reconstructed values of the 8x8 block respectively. The submode with the minimal cost is selected as the best submode for the 8x8 block. After the submode for each 8x8 block is determined, the MB partition for P8x8 mode is fixed with the four selected submodes.

Equation (3.2) is then employed to select the best mode from the candidate modes 16x16, 16x8, 8x16, and P8x8. In this step, *MODE* indicates one of these four candidate modes now. *s* and *c* indicate the original and reconstructed values of the whole MB correspondingly. In the cost evaluation of P8x8 mode, submodes determined before together with the corresponding motion vectors are involved in the reconstruction process of *c*.

Generally, the motion estimation process for each MB is performed seven times in H.264 for computing the best mode. Considering the time consuming of the common motion estimation process, and the encoding and decoding processes for RDO during mode decision, the computation complexity of the variable block size motion estimation in H.264 is extremely high. Numerous algorithms [54-64] were developed to speed up both mode decision and motion estimation for H.264 encoding. All of them exploit the relationship among the motion vectors of different modes to determine the best mode and the motion vector without examining all the candidates. In addition, some transcoding schemes related to H.264 were proposed. For example, [65-66] proposed fast mode decision and motion estimation algorithms for MPEG-2 to H.264 transcoding by reusing motion information. In [67], fast methods for the frame-skipping transcoding in H.264 were also suggested. Obviously, the variable block size motion estimation also increases the complexity of a reverse transcoder and some fast algorithms are desirable.

3.3 Reverse transcoding of H.264 bitstream

In reverse transcoding, consider two frames of a video sequence: frames *n* and *n+1*. In the forward bitstream, frame *n+1* is encoded using frame *n* as the reference. During reverse transcoding, frame *n* is encoded with the new reference, frame *n+1*, and motion estimation is necessary to estimate motion vectors by using frame *n* and frame *n+1* as the current and reference frame respectively. Perhaps the simplest method is to perform brute-force motion estimation between these two frames. The brute-force search can guarantee to obtain the best RD performance, but the calculations of J_{motion} and J_{mode} incur a large amount of computational effort. In MPEG-2/4, reverse motion vectors can be obtained by negating the horizontal and vertical components of the corresponding forward motion vectors in the same spatial location of frame *n+1* as discussed in Section 2.5.2 [46-47]. This idea can be further extended to mode decision in H.264 in which the reverse modes of frame *n* can be directly copied from the modes of frame *n+1* in the corresponding spatial location. This is called “in-place” reverse motion estimation as shown in Figure 3.2. However, a major disadvantage of this method is that it always obtains incorrect modes when the co-located MBs in frames *n* and *n+1* do not have a direct correspondence. In this chapter, we are going to propose a fast mode decision and motion estimation algorithm for the reverse transcoding in H.264.

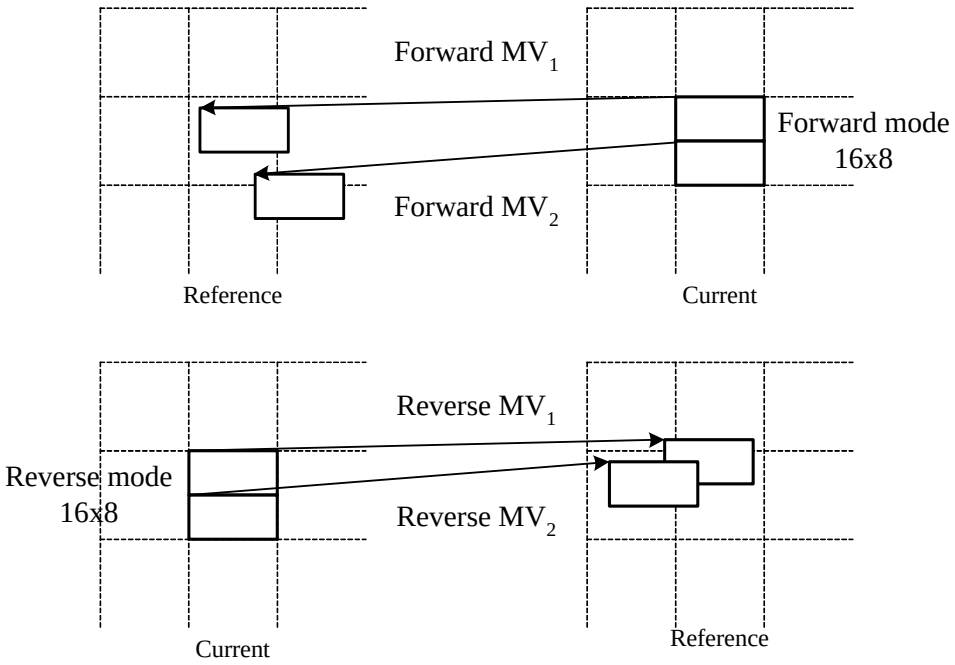


Figure 3.2. The extended "in-place" algorithm for reverse motion estimation in H.264.

3.4 Proposed reverse motion estimation algorithm

In Figure 3.3, we assume that MB_n^k represents the k^{th} MB of frame *n*. In the reverse transcoding process, MB_n^k is encoded with the new reference, frame *n+1*. Let us represent the reverse mode and the set of reverse motion vectors of MB_n^k as $RMODE_n^k$ and RMV_n^k respectively. For variable block size motion estimation, each RMV_n^k contains a number of motion vectors, i.e., $RMV_n^k = \{RMV_n^{k,1}, RMV_n^{k,2} ..., RMV_n^{k,m}\}$, where *m* is the total number of partitions in MB_n^k. Besides, if $RMODE_n^k$ is P8x8 mode, each of the four 8x8 blocks has its submode $RSUBMODE_n^k$. In the forward bitstream, the sets of forward motion vectors of MB_n^k and MB_{n+1}^k are denoted by $FMV_n^k = \{FMV_n^{k,1}, FMV_n^{k,2} ..., FMV_n^{k,m}\}$ and $FMV_{n+1}^k = \{FMV_{n+1}^{k,1}, FMV_{n+1}^{k,2} ..., FMV_{n+1}^{k,m}\}$ respectively. Their corresponding modes (submodes) are $FMODE_n^k$ ($FSUBMODE_n^k$) and

$FMODE_{n+1}^k$ ($FSUBMODE_{n+1}^k$). To expedite mode decision of $RMODE_n^k$, our proposed algorithm exploits the relationship among FMV_n^k , FMV_{n+1}^k, $FMODE_n^k$, and $FMODE_{n+1}^k$.

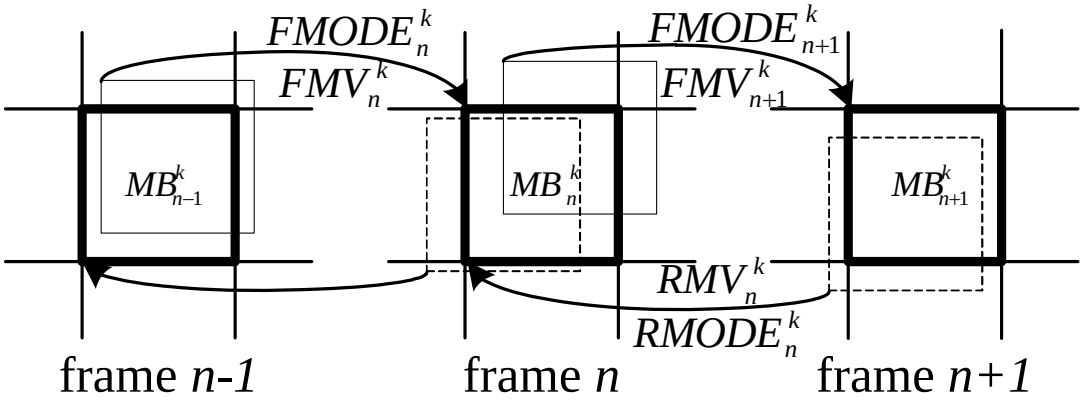


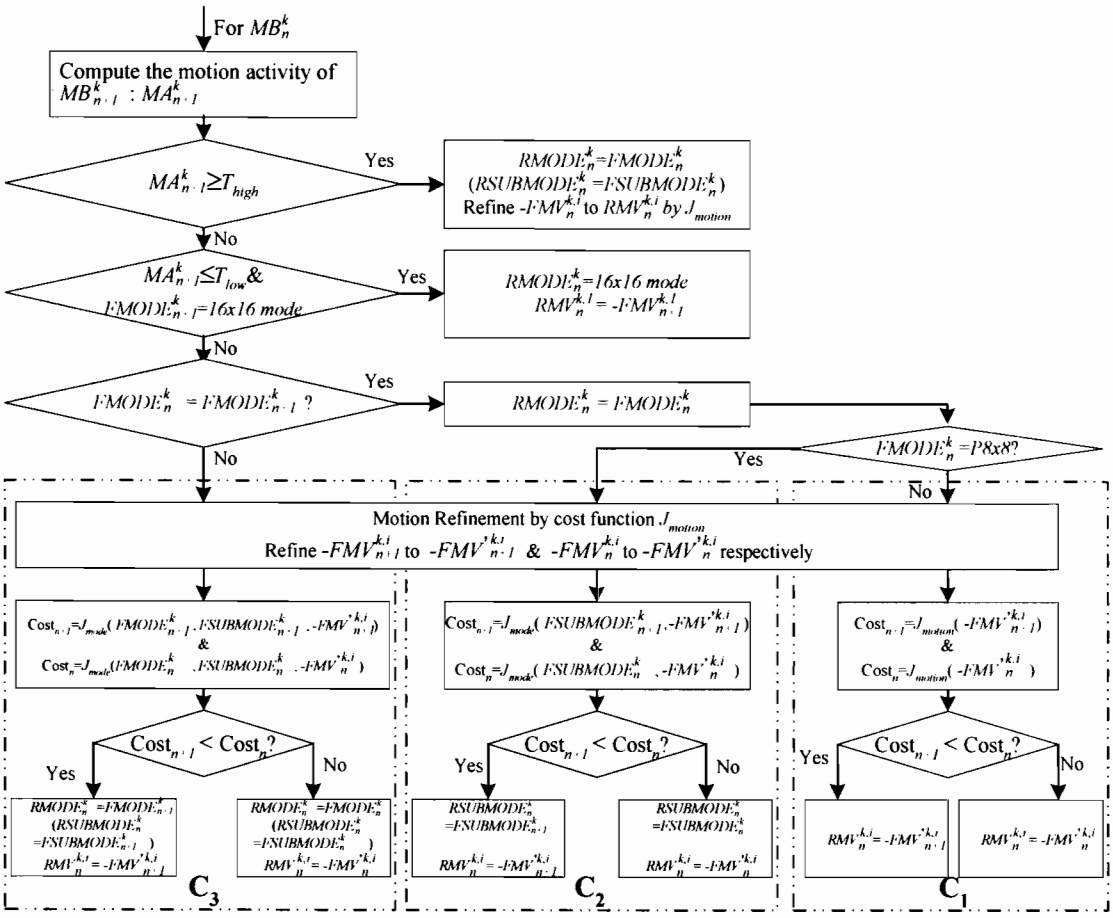
Figure 3.3. Reverse motion estimation of MB_n^k .

In a natural video sequence, the homogeneous areas such as the background are always coded using *16×16* mode. On the contrary, a smaller partition is chosen in the non-homogeneous areas with strong edges. In general, $FMODE_n^k$ provides the spatial partition of MB_n^k and gives a good hint of choosing $RMODE_n^k$. In spite of this, $FMODE_n^k$ only reflects various motions of all objects inside MB_n^k by using frame *n-1* as the reference. But, in choosing the best $RMODE_n^k$, the reference frame is frame *n+1*. Therefore, $FMODE_{n+1}^k$ together with FMV_{n+1}^k of MB_{n+1}^k (the corresponding spatial location of MB_n^k in frame *n+1*) from the forward bitstream is also a good indicator to determine the correlation between modes used in frame *n* and frame *n+1*. Thus an improved way of determining $RMODE_n^k$ can be achieved by adaptively choosing $FMODE_n^k$ and $FMODE_{n+1}^k$.

Generally, the motion activity of MB_{n+1}^{k}, MA_{n+1}^{k}, provides the correlation between $RMODE_{n}^{k}$ and $FMODE_{n+1}^{k}$ and it is defined as

$$MA_{n+1}^{k} = \frac{1}{m}\sum_{i=1}^{m}\left(\left|u_{n+1}^{k,i}\right| + \left|v_{n+1}^{k,i}\right|\right) \tag{3.3}$$

where $u_{n+1}^{k,i}$ and $v_{n+1}^{k,i}$ are the horizontal and vertical components of $FMV_{n+1}^{k,i}$. If $FMODE_{n+1}^{k}$ is the 16x16 mode and its MA_{n+1}^{k} has a small value, it is most likely that $RMODE_{n}^{k}$ is also set to the 16x16 mode due to its temporal stationary. Besides, RMV_{n}^{k} is estimated by $-FMV_{n+1}^{k}$. On the other hand, a large value of MA_{n+1}^{k} signifies that $RMODE_{n}^{k}$ and $FMODE_{n+1}^{k}$ have only small correlation. In this case, only $FMODE_{n}^{k}$ can provide the best estimate of $RMODE_{n}^{k}$. In the actual situation, FMV_{n}^{k} solely represents the motion between frame *n* and frame *n-1*. While RMV_{n}^{k} refers to the motion between frame *n* and frame *n+1*, $-FMV_{n}^{k}$ cannot be directly used, but be taken as a good predictor by considering some kind of temporal smoothness of the motion trajectory. For MA_{n+1}^{k} with medium value, either $FMODE_{n}^{k}$ or $FMODE_{n+1}^{k}$ is possible to be selected. If they are the same and belong to L1 (16×16, 16×8 or 8×16), there is very high chance that this mode can be used in $RMODE_{n}^{k}$. Otherwise, if both $FMODE_{n}^{k}$ and $FMODE_{n+1}^{k}$ belong to L2, the motion activity and spatial structure of MB_{n}^{k} is complex. It is necessary to select $RSUBMODE_{n}^{k}$ of each 8x8 block from $FSUBMODE_{n}^{k}$ and $FSUBMODE_{n+1}^{k}$ by calculating their J_{mode}. The whole algorithm is summarized as follows:



(The superscript ' represents a refined motion vector)

Figure 3.4. Flow chart of the proposed algorithm

Step 1: Compute the motion activity of MB_{n+1}^k, MA_{n+1}^k, using (3.3).

Step 2: Compare MA_{n+1}^k with the threshold T_{high}. If $MA_{n+1}^k \geq T_{high}$, choose $FMODE_n^k$ as the final mode of $RMODE_n^k$. Set the initial $RMV_n^{k,i}$ to $-FMV_n^{k,i}$, for i=1, 2,..., m, followed by a refined motion estimation with a significantly reduced search area to obtain the final $RMV_n^{k,i}$.

Step 3: Compare MA_{n+1}^k with the threshold T_{low} ($T_{low} < T_{high}$). If $MA_{n+1}^k \leq T_{low}$ and $FMODE_{n+1}^k$ =16x16 mode, set $RMODE_n^k$ and $RMV_n^{k,1}$ to 16x16 mode and $-FMV_{n+1}^{k,1}$ respectively.

Step 4: Otherwise, choose the best of $RMODE_n^k$ from $FMODE_{n+1}^k$ and $FMODE_n^k$ based on the following three conditions.

C1: If $FMODE_n^k$ = $FMODE_{n+1}^k$ and they both belong to L1, set $RMODE_n^k$ to $FMODE_n^k$ (= $FMODE_{n+1}^k$). Perform motion vector refinement by using both $-FMV_n^{k,i}$ and $-FMV_{n+1}^{k,i}$ as the initial vectors and the one with smallest J_{motion} is selected as $RMV_n^{k,i}$.

C2: If $FMODE_n^k$ = $FMODE_{n+1}^k$ and they both belong to L2, set $RMODE_n^k$ to $FMODE_n^k$ (= $FMODE_{n+1}^k$). Determine $RSUBMODE_n^k$ and RMV_n^k according to the following steps:

- Perform motion vector refinement by using both $-FMV_n^{k,i}$ and $-FMV_{n+1}^{k,i}$ as the initial vectors.
- Choose the best submode of $RSUBMODE_n^k$ from $FSUBMODE_n^k$ and $FSUBMODE_{n+1}^k$ by calculating J_{mode} in (3.2) using their refined motion vectors, and set RMV_n^k to the corresponding refined motion vectors.

C3: If $FMODE_n^k \neq FMODE_{n+1}^k$, determine $RMODE_n^k$ with possible $RSUBMODE_n^k$ and RMV_n^k as follows:

- Refine motion vectors by using both $-FMV_n^{k,i}$ and $-FMV_{n+1}^{k,i}$ as the initial vectors.
- For both $FMODE_n^k$ and $FMODE_{n+1}^k$, calculate J_{mode} by using their refined motion vectors. If one of them is P8x8 mode, reckon its corresponding submode in computing J_{mode}.

- Select the best mode with smaller J_{mode} as $RMODE_n^k$ and set the corresponding refined motion vectors as RMV_n^k. If $RMODE_n^k$ is P8x8 mode, set also $RSUBMODE_n^k$ to the corresponding submodes ($FSUBMODE_n^k$ or $FSUBMODE_{n+1}^k$).

The motion estimation and mode decision process can be summarized in the flowchart as depicted in Figure 3.4. After that, based on the resulted motion vectors of RMV_n^k, a merging process is taken to finish the reverse mode decision. The principle is simple but efficient. If there is a common motion vector among any neighboring partitions, they can be merged to a larger partition with that common motion vector. For example, assume the final mode estimate is P8x8 and all the submodes are 8x8 as shown in Figure 3.5. Since the motion vectors come from $-FMV_n^{k,i}$ and $-FMV_{n+1}^{k,i}$, and refinement is involved, it is possible for them to be equal. If RMV(0,0)=RMV(1,0) and RMV(0,1)=RMV(1,1), the MB partition is reformed to 8x16 as illustrated in Figure 3.5. By using the merging process, less coding bits are needed for coding motion vectors.

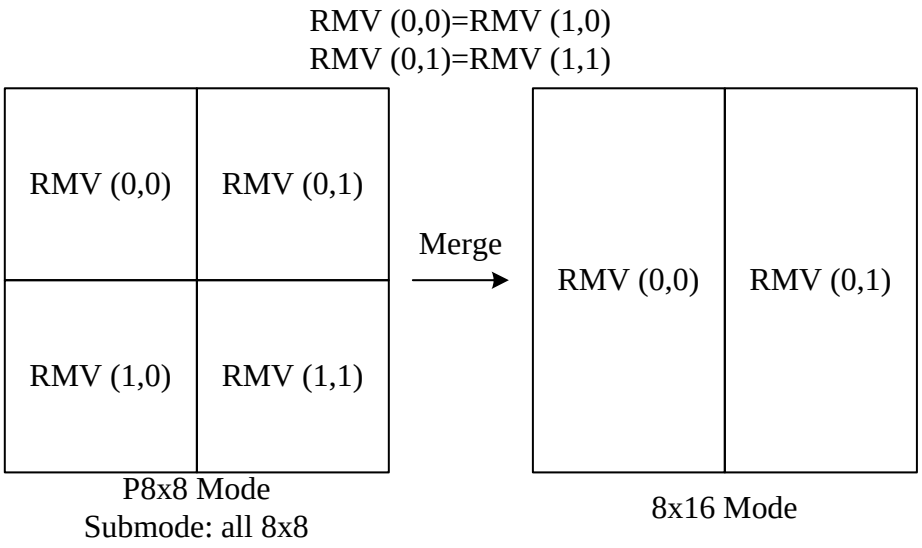


Figure 3.5. Merging process after the mode and motion vector re-estimation.

3.5 Experimental results

The proposed reverse motion estimation algorithm has been implemented based on the JVT JM 9.2 encoder [53] and was tested from five video sequences (Foreman QCIF, Carphone QCIF, Salesman CIF, Tabletennis CIF, and Flower CIF) with 5 quantization parameters, i.e., QP=20, 24, 28, 32, and 36. In our proposed algorithm, T_{high} and T_{low} were set to 256 and 32 respectively, and a search area of ± 1 pixel was used for motion vector refinement. The results are given in Table 3.1 and Table 3.2. In these tables, ΔPSNR, Δbitrate, ΔJ_{motion} and ΔJ_{mode} represent a PSNR change, a bitrate change in percentage, percentage changes in the numbers of J_{motion} and J_{mode} calculations when compared to the "brute-force" algorithm with a search range of ± 8 pixels, respectively. The positive values mean increments whereas negative values mean decrements. From the results in Table 3.2, it is observed that the proposed algorithm can substantially reduce ΔJ_{motion} and ΔJ_{mode} over 99% and 83% respectively. Note that the "in-place" algorithm does not need to compute

J_{motion} and J_{mode} since it directly reuses the modes and motion vectors in the forward bitstream. However, the RD performance of the "in-place" algorithm is greatly deteriorated, as depicted in Figure 3.6, Figure 3.7, and Figure 3.8 where the RD curves of the "Foreman", "Carphone", and "Salesman" sequence are shown respectively. In contrast, the RD performance of the proposed algorithm can achieve similar result to the "brute-force" algorithm. Table 3.1 and Table 3.2 also show that the proposed algorithm has consistent gain in coding efficiency for all video sequences as compared with the "in-place" algorithm. On average, the proposed algorithm can achieve tremendous speed up at a cost of 0.14 dB PSNR drop and 6.44% bitrate increase compared to the "brute-force" algorithm.

Table 3.1. Comparison of the brute-fore, in-place and the proposed algorithms.

	brute-force		in-place		proposed	
	PSNR	Bitrate	PSNR (ΔPSNR)	Bitrate (ΔBitrate)	PSNR (ΔPSNR)	Bitrate (ΔBitrate)
Foreman	33.77	122.92	33.27 (–0.5)	162.13 (+31.90%)	33.59 (–0.18)	131.19 (+6.7%)
Carphone	34.89	107.66	34.22 (–0.67)	151.53 (+40.8%)	34.65 (–0.24)	117.58 (+9.2%)
Salesman	34.26	185.32	34.03 (–0.23)	222.30 (+19.9%)	34.19 (–0.07)	194.72 (+5.1%)
Tabletennis	32.19	661.93	31.82 (–0.37)	845.44 (+27.7%)	32.02 (–0.17)	718.41 (+8.5%)
Flower	31.30	2117.27	31.10 (–0.20)	2382.06 (+12.5%)	31.24 (–0.06)	2174.99 (+2.7%)

Table 3.2. Computation reduction by the proposed algorithm.

	brute-force		proposed	
	J_{motion}	J_{mode}	J_{motion} (ΔJ_{motion})	J_{mode} (ΔJ_{mode})
Foreman	2023	7	10.15 (–99.4%)	0.85 (–87.9%)
Carphone	2023	7	8.43 (–99.5%)	0.70 (–90.1%)
Salesman	2023	7	2.99 (–99.8%)	0.26 (–96.3%)
Tabletennis	2023	7	8.31 (–99.5%)	0.65 (–90.7%)
Flower	2023	7	13.82 (–99.3%)	1.18 (–83.2%)

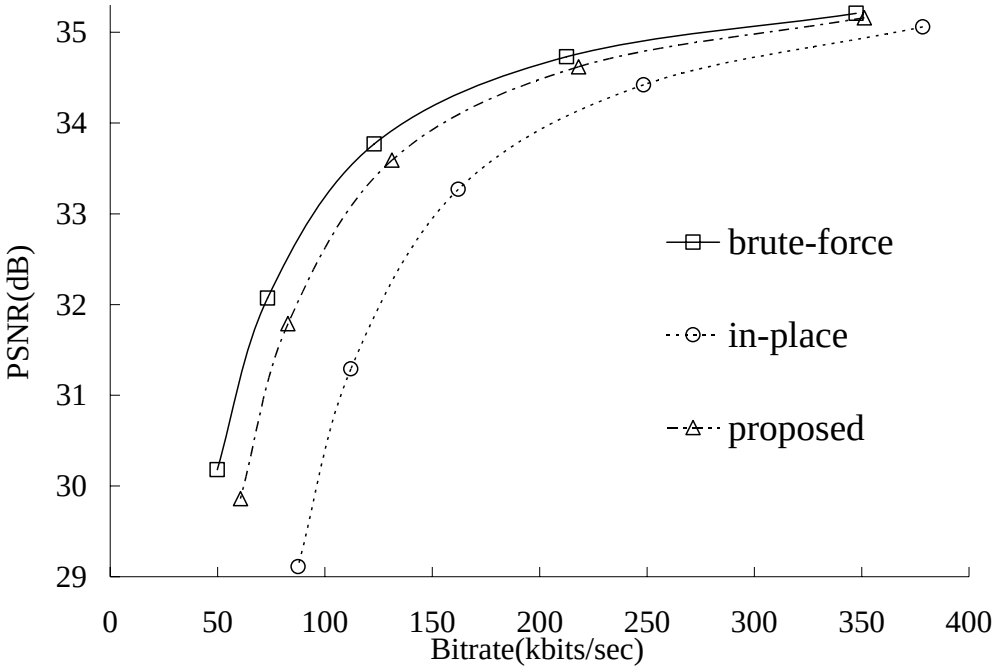


Figure 3.6. Rate-distortion curves comparison for the “Foreman” sequence.

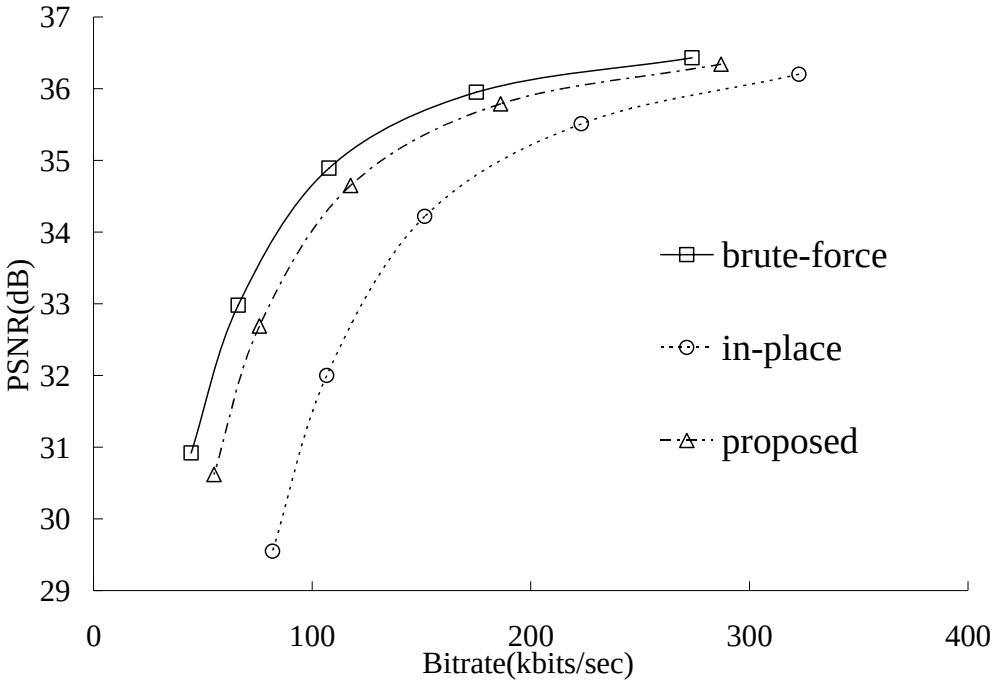


Figure 3.7. Rate-distortion curves comparison for the “Carphone” sequence.

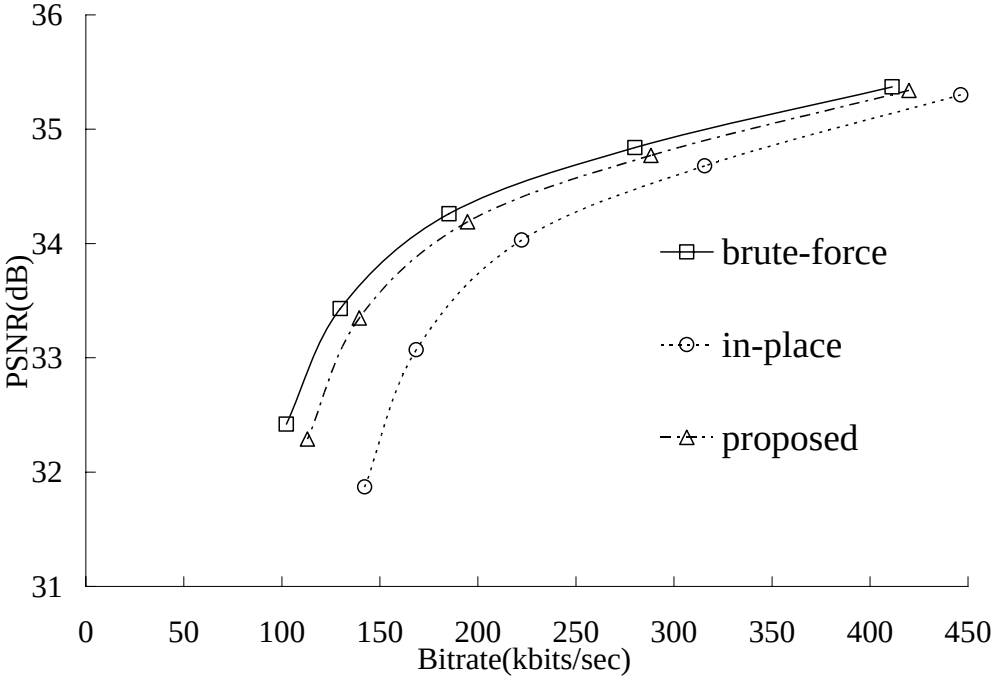


Figure 3.8. Rate-distortion curves comparison for the “Salesman” sequence.

3.6 Conclusion

In this chapter, we have addressed a fast reverse motion estimation and mode decision algorithm for the reverse transcoding in H.264. We mentioned that the straightforward approach might result in much higher complexity of the transcoder; it is not desirable. In our proposed algorithm, the motivation is to exploit the relationship among various modes and motion vectors in the original forward bitstream such that these information can be re-used in reverse transcoding a H.264 bitstream with the support of variable block size motion estimation. Experimental results show that the proposed algorithm can achieve remarkable speed up while maintaining a similar rate-distortion performance as compared to the conventional brute-force algorithm. The rate-distortion performance of the proposed algorithm also outperforms the extended "in-place" algorithm. This reduction in computational complexity helps the real-time VCR implementation of H.264 in reverse transcoding.

Chapter 4 MB-Based Backward-Play Algorithm for MPEG Video Streaming with VCR Support

4.1 Introduction

We have seen from the previous chapter that the use of the proposed reverse motion estimation algorithm could be able to efficiently expedite the reverse transcoding process. Nevertheless, the transcoding process still needs much computation for the decoding and re-encoding of residuals. Besides, this re-encoding process evidently suffers from quality degradation. An alternative approach used in this research work is to avoid the re-computation of residuals in order to maintain the quality of the reconstructed video as good as possible. Therefore, in this chapter, we provide a compressed-domain solution to perform the backward-play operation on the MPEG video streaming system with the minimal requirements on the decoder complexity and network bandwidth. The proposed scheme can adaptively select the necessary macroblocks (MBs), manipulate them in the compressed domain, and send the processed MBs to the client. Since the proposed scheme mainly operates in the compressed domain, complete decoding and re-encoding is not required in the server. Thus, an additional processing requirement in the server can be minimized. Besides, the decoder complexity and network complexity are reduced significantly in the

proposed system compared to the conventional one with only negligible degradation in quality.

Details of the scheme are shown in the following sections, while part of the results of this chapter has been published in references [68-69].

4.2 The proposed video streaming system and some important definitions

The architecture of the proposed video streaming system with VCR support is shown in Figure 4.1. For the sake of simplicity, in the discussion of our proposed solution, we again consider the video bitstream 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 2.3 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 backward-play operation. At the server side of Figure 4.1, 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.

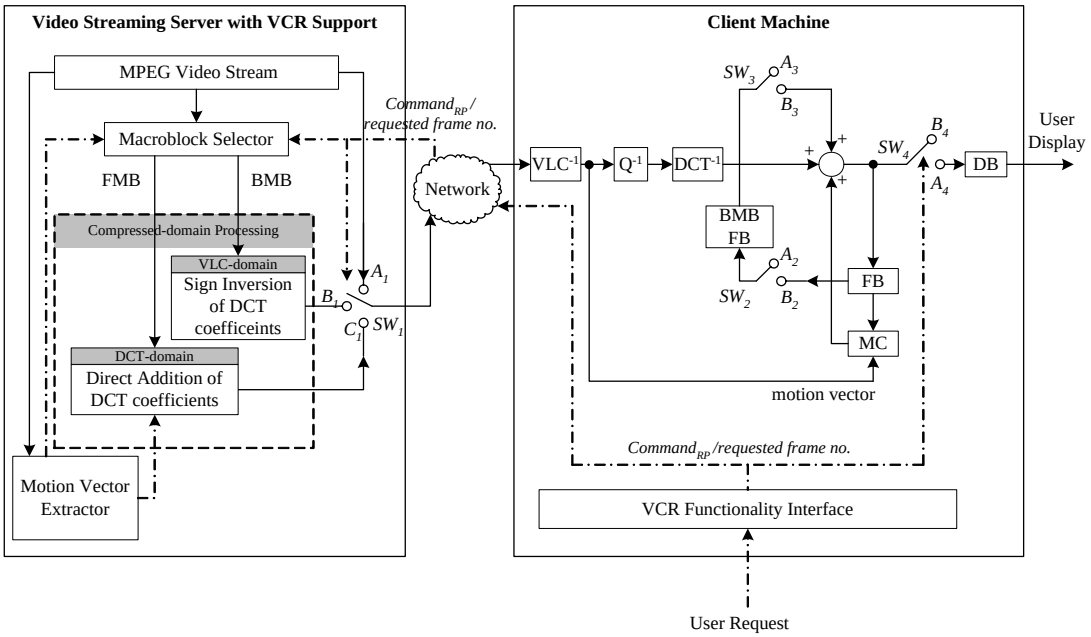


Figure 4.1. The proposed architecture for the video streaming system with VCR functionality.

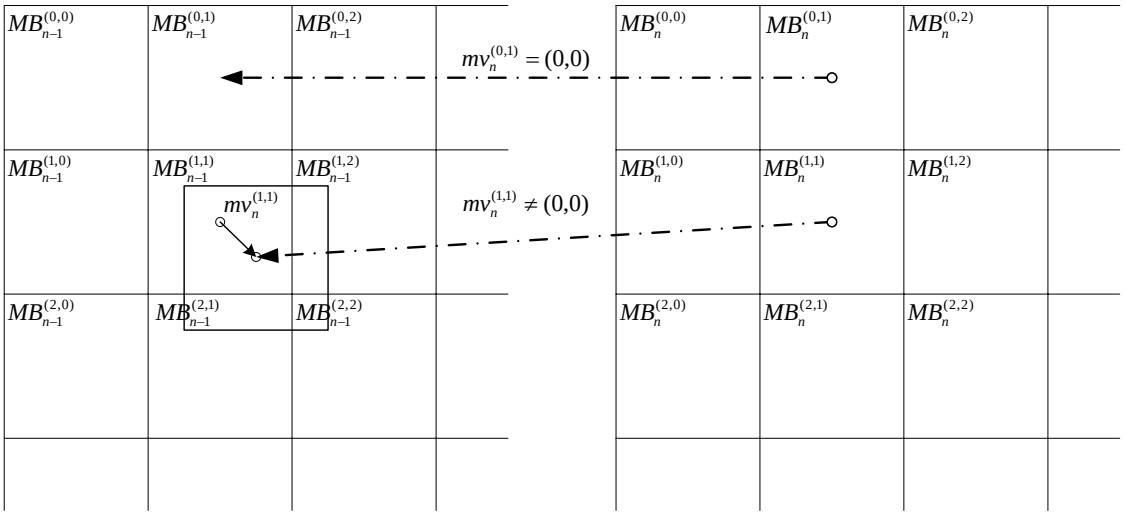


Figure 4.2. Definition of the forward MB (FMB) and the backward MB (BMB).

Two types of MBs are now defined. For illustration, we use the example in Figure 2.8 (Section 2.3) again. Considering a user requests a backward-play command at frame *n*, the next frame to be display is frame *n-1*. The situation in MB level is depicted in Figure 4.2. We assume that $MB_{n-1}^{(k,l)}$ represents the MB at the k^{th} row and l^{th} column of frame *n-1* (the next frame to be displayed).

$MB_{n-1}^{(k,l)}$ is defined as a backward MB (BMB) if the MB in 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 MB). Otherwise, it is defined as a forward MB (FMB). For example, in Figure 4.2, since the motion vector of $MB_{n}^{(0,1)}$, $mv_{n}^{(0,1)}$, is zero, it means that $MB_{n}^{(0,1)}$ is a non-MC MB and the MB selector classifies $MB_{n-1}^{(0,1)}$ as BMB. On the other hand, since $MB_{n}^{(1,1)}$ is coded with motion compensation (MC-MB), $MB_{n-1}^{(1,1)}$ is classified as FMB. The proposed 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 frames. Our contributions lie in the following two areas:

(1) a sign inversion of the DCT coefficients which can be operated in the VLC domain for BMBs, and

(2) a direct addition of the DCT coefficients which can be operated in the DCT domain for FMBs.

4.3 VLC-domain techniques for BMBs

Block motion-compensated prediction (MCP) is the type of prediction used in MPEG standards [5-6, 9-10]. This prediction type is what gives the MPEG codecs the advantage over pure still-frame coding methods. In motion-compensated 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 Figure 4.1, each MB in frame *n*, $MB_n^{(k,l)}$, is reconstructed according to motion-compensated prediction and it is given by

$$MB_n^{(k,l)} = MCMB_{n-1}(mv_n^{(k,l)}) + e_n^{(k,l)} \tag{4.1}$$

where $MCMB_{n-1}(mv_n^{(k,l)})$ stands for the motion-compensated MB 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 MB, $MCMB_{n-1}(mv_n^{(k,l)})$. Frame *n* is then stored in frame buffer (FB) as it is used for decoding subsequent frame *n+1* in the forward play operation.

When a user issues a backward-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 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 $MB_{n-1}^{(k,l)}$ is found to be a BMB, its corresponding MB 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 (4.1) can be rewritten as

$$MB_{n-1}^{(k,l)} = MB_n^{(k,l)} + \tilde{e}_n^{(k,l)} \tag{4.2}$$

where $\tilde{e}_n^{(k,l)} = -e_n^{(k,l)}$. Note that frame *n* is stored in FB at the client machine when a user issues the backward-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 backward-play operation, (4.2) indicates that, for a 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 to $\tilde{e}_n^{(k,l)}$ and considering that the DCT is an odd transform, we can find the DCT of $\tilde{e}_n^{(k,l)}$ in the DCT domain, as indicated below,

$$DCT(\tilde{e}_n^{(k,l)}) = -DCT(e_n^{(k,l)}) \tag{4.3}$$

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

$$QDCT(\tilde{e}_n^{(k,l)}) = -QDCT(e_n^{(k,l)}) \tag{4.4}$$

From (4.4), $Q[DCT(\tilde{e}_n^{(k,l)})]$ can be obtained by inverting the sign of all DCT coefficients in $Q[DCT(e_n^{(k,l)})]$, which can be directly extracted from the video bitstream in the server. $Q[DCT(\tilde{e}_n^{(k,l)})]$ is then transmitted to the client by switching SW_1 to B_1. At the client side, as shown in Figure 4.1, 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 backward 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 [70-75], as demonstrated by the typical examples as shown in Table 4.1 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 BMBs 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 traffic significantly.

Table 4.1. 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

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 $Q[DCT(\tilde{e}_n^{(k,l)})]$ in the VLC domain. For encoding of the quantized DCT coefficients in MPEG, they are arranged into a 1-D array following the zigzag scan order as illustrated in Figure 2.4. This scan order puts the low-frequency coefficients in front of the high-frequency coefficients. Since visually-weighted quantization strongly deemphasizes higher spatial frequencies, only a few

lower-frequency coefficients are nonzero in a typical block. Thus, the zigzag scan order puts the longest runs of zeros at the end of the 1-D array such that the EOB (end-of-block) symbol can efficiently code all of these trailing zero coefficients with a single codeword. Typically, the EOB occurs well before the midpoint of the array. Runs of zero coefficients also occur quite frequently before the EOB. 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.

Table 4.2. 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
:	:	:

Table 4.3. 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

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 4.2 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 $Q[DCT(\tilde{e}_n^{(k,l)})]$ from $Q[DCT(e_n^{(k,l)})]$, the server just parses the MPEG video bitstream and inverts all '*s*' bits of VLCs in a BMB, as shown in Figure 4.3. On the other hand, RUN-LEVEL combinations that are not in the Table 4.3 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 4.3. In this case, the 12-bit FLC for LEVEL is converted into its 2's complement. The bit manipulation of VLCs in a BMB is summarized in Figure 4.3. Since it is not necessary to perform VLC encoding, motion compensation, DCT, quantization, inverse DCT, inverse quantization and VLC decoding in the server, the loading of the server is reduced significantly.

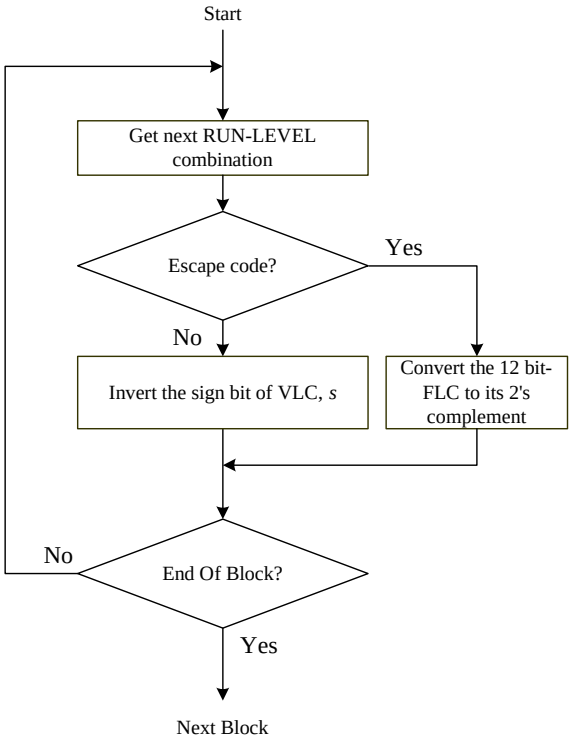


Figure 4.3. Execution flow of the server during bit manipulation of VLCs in a BMB.

4.4 DCT-domain techniques for FMBs

The situation of FMBs is different. The bit manipulation of VLCs mentioned in the previous section cannot be employed since $MCMB_{n-1}(mv_n^{(k,l)})$ is no longer equal to $MB_{n-1}^{(k,l)}$ and then (4.2) does not hold true for FMBs. In other words, $MB_{n-1}^{(k,l)}$ cannot be reconstructed from $MB_n^{(k,l)}$.

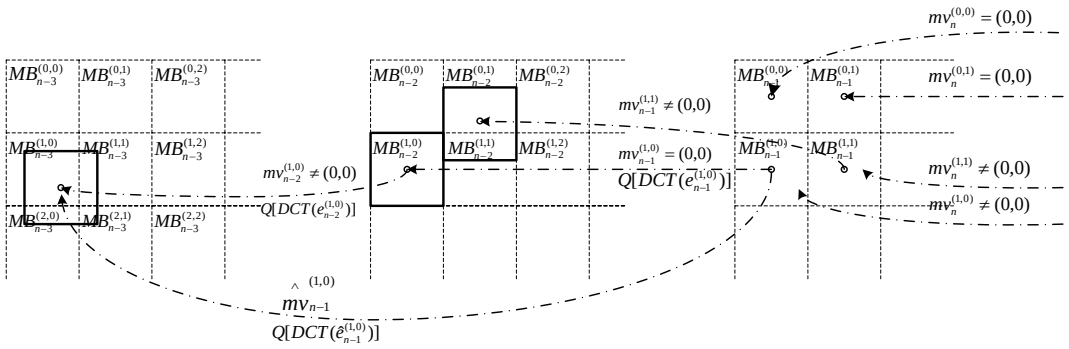


Figure 4.4. A situation in which there are two FMBs in frame *n-1* is illustrated.

In Figure 4.4, a situation in which there are two FMBs in frame *n-1* is illustrated. Three MBs (shaded MBs in Figure 4.4) in frame *n-2* are required to act as references for performing motion compensation of these two FMBs. Those MBs in frame *n-2* further requires their corresponding MBs 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 Figure 4.1. In the client machine, all the switches are open 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 4.4. During decoding FMBs in frame *n-1*, the prediction error between each FMB and its corresponding motion-compensated MB is decoded. Each reconstructed pixel in FMB can be obtained by adding its prediction errors to its motion-compensated pixels of frame *n-2* in FB, as shown in Figure 4.1. 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 backward-play operation. Frame *n-1* is then stored in both 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 backward-play operation.

Table 4.4 . 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
Backward	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 further reduce the decoding complexity and network traffic in processing FMBs, we propose to use a technique involving direct addition of DCT coefficients. This technique is originally designed for frame-skipping video transcoding which is mainly performed in the DCT domain to achieve a transcoder with low complexity [73-75]. Now, we borrow this idea for a FMB when it is coded without motion compensation (non-MC FMB) and further extend it to alleviate the computational burden of the client decoder in the backward-play operation. In Figure 4.4, $MB_{n-1}^{(1,0)}$ 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, $MB_{n-1}^{(1,0)}$ is a BMB instead. In general, a MB in frame *n-1*, $MB_{n-1}^{(k,l)}$, is treated as non-MC FMB when $mv_n^{(k,l)} \neq (0,0)$ and $mv_{n-1}^{(k,l)} = (0,0)$.

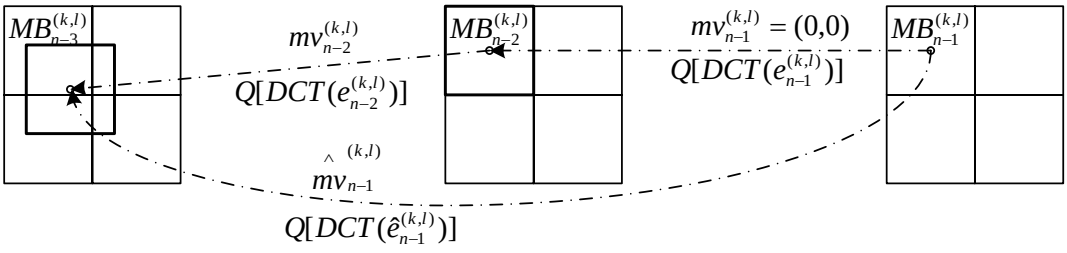


Figure 4.5. Direct addition of DCT coefficients.

In Figure 4.5, a situation in which $MB_{n-1}^{(k,l)}$ 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 $MB_{n-2}^{(k,l)}$ are

used as the reference for $MB_{n-1}^{(k,l)}$ only, it is too wasteful that the client machine to decode $e_{n-2}^{(k,l)}$ for reconstructing $MB_{n-2}^{(k,l)}$. 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 the non-MC FMB. When pixels in $MB_{n-2}^{(k,l)}$ 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, $Q[DCT(e_{n-1}^{(k,l)})]$, 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 $\hat{mv}_{n-1}^{(k,l)}$ and prediction errors in the DCT domain, $Q[DCT(\hat{e}_{n-1}^{(k,l)})]$, by using frame *n-3* as a reference, as shown in Figure 4.5. Since $mv_{n-1}^{(k,l)}$ is zero, we have

$$\hat{mv}_{n-1}^{(k,l)} = mv_{n-2}^{(k,l)} \tag{4.5}$$

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

$$\begin{aligned} Q[DCT(\hat{e}_{n-1}^{(k,l)})] &= Q[DCT(MB_{n-1}^{(k,l)} - MCMB_{n-3}(\hat{mv}_{n-1}^{(k,l)}))] \\ &= Q[DCT(MB_{n-1}^{(k,l)} - MCMB_{n-3}(mv_{n-2}^{(k,l)}))] \end{aligned} \tag{4.6}$$

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 [73]. In the proposed video server, we employ a DCT-domain technique to compute the new $Q[DCT(\hat{e}_{n-1}^{(k,l)})]$. Using (4.1), $MB_{n-1}^{(k,l)}$ can be written as

$$MB_{n-1}^{(k,l)} = MCMB_{n-2}(mv_{n-1}^{(k,l)}) + e_{n-1}^{(k,l)} \tag{4.7}$$

If $MB_{n-1}^{(k,l)}$ is coded without motion compensation, $mv_{n-1}^{(k,l)}$ is zero and $MCMB_{n-2}(mv_{n-1}^{(k,l)})$ is equal to $MB_{n-2}^{(k,l)}$. Equation (4.7) can be simplified to (4.8)

$$MB_{n-1}^{(k,l)} = MB_{n-2}^{(k,l)} + e_{n-1}^{(k,l)} \tag{4.8}$$

Similarly,

$$MB_{n-2}^{(k,l)} = MCMB_{n-3}(mv_{n-2}^{(k,l)}) + e_{n-2}^{(k,l)} \tag{4.9}$$

Substituting (4.9) into (4.8), we obtain

$$MB_{n-1}^{(k,l)} - MCMB_{n-3}(mv_{n-2}^{(k,l)}) = e_{n-1}^{(k,l)} + e_{n-2}^{(k,l)} \tag{4.10}$$

Using (4.6) and (4.10), the newly quantized DCT coefficients of the prediction error between the current non-MC FMB and its corresponding reference MB in frame *n-3*, $Q[DCT(\hat{e}_{n-1}^{(k,l)})]$, can be written as

$$Q[DCT(\hat{e}_{n-1}^{(k,l)})] = Q[DCT(e_{n-1}^{(k,l)} + e_{n-2}^{(k,l)})] \tag{4.11}$$

Taking into account the linearity of DCT, (4.11) becomes

$$Q[DCT(\hat{e}_{n-1}^{(k,l)})] = Q[DCT(e_{n-1}^{(k,l)}) + DCT(e_{n-2}^{(k,l)})] \tag{4.12}$$

Note that, in general, quantization is not a linear operation because of the integer truncation. However, $DCT(e_{n-1}^{(k,l)})$ and $DCT(e_{n-2}^{(k,l)})$ are divisible by the

quantizer step-size. Thus, we obtain the final expression of $Q[DCT(\hat{e}_{n-1}^{(k,l)})]$ by using frame *n-3* as a reference,

$$Q[DCT(\hat{e}_{n-1}^{(k,l)})] = Q[DCT(e_{n-1}^{(k,l)})] + Q[DCT(e_{n-2}^{(k,l)})] \quad (4.13)$$

Equation (4.13) implies that the newly quantized DCT coefficient $Q[DCT(\hat{e}_{n-1}^{(k,l)})]$ can be computed in the DCT domain by adding $Q[DCT(e_{n-1}^{(k,l)})]$ and $Q[DCT(e_{n-2}^{(k,l)})]$. Both of them can be directly extracted from the MPEG video bitstream 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 Figure 4.4 again. Instead of transmitting and decoding $Q[DCT(e_{n-1}^{(1,0)})]$ and $Q[DCT(e_{n-2}^{(1,0)})]$ for $MB_{n-1}^{(1,0)}$, by using the direct addition of DCT coefficients, the server only needs to send $Q[DCT(\hat{e}_{n-1}^{(1,0)})]$ 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 $MB_{n-3}^{(k,l)}$ are used as the reference for $MB_{n-1}^{(k,l)}$ only, as depicted in Figure 4.6.

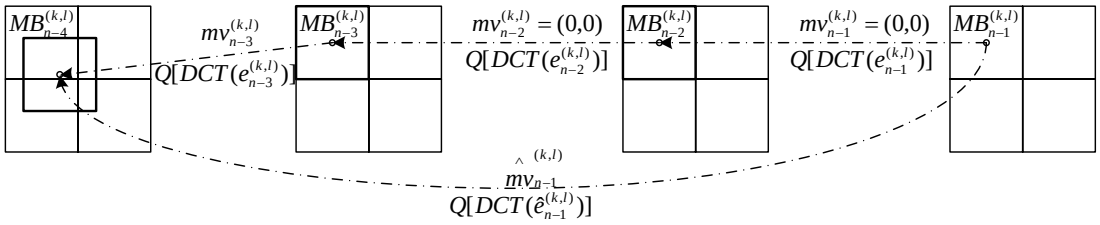


Figure 4.6. Using direct addition of DCT coefficients iteratively.

4.5 Experimental results for the proposed MB-based system

In this section, we present some experimental results to evaluate the performances of the proposed techniques including the sign inversion for BMBs and the direct addition for FMBs when applied to the video streaming system with VCR support. An MPEG-2 encoder [16] was employed to encode various video sequences with different spatial resolutions and motion characteristics. The video sequences used in the experiments are tabulated in Table 4.5. All the video 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 start point of the backward-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 4.5. Spatial resolutions and motion characteristics of the testing sequences.

Sequences	Resolutions	Motion characteristics
Salesman	352x288	Low
Football	352x240	High
Table Tennis	352x240	High
Foreman	176x144	High
Carphone	176x144	Low
Claire	176x144	Low
Grandma	176x144	Low

Table 4.6. 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 (MBs)			Average no. of bits to be sent over the network (bits)		
		Conventional System	SI	SI+DA	Conventional System	SI	SI+DA
Salesman (*352×288*)	1.5M	3109	1912	1575	373996	224223	191773
	3M	3109	1912	1575	796185	474678	392541
Football (*352×240*)	1.5M	2591	2159	2001	368697	321932	306286
	3M	2591	2159	2001	815826	701160	660250
Table Tennis (*352×240*)	1.5M	2591	1719	1441	368660	287232	263341
	3M	2591	1719	1441	747080	573522	519223
Foreman (*176×144*)	64K	777	625	530	9610	7399	7198
	128K	777	625	530	26175	20357	18686
Carphone (*176×144*)	64K	777	586	452	8466	6176	5980
	128K	777	586	452	25192	19082	16966
Claire (*176×144*)	64K	777	225	125	10582	1931	1643
	128K	777	225	125	31481	8252	5985
Grandma (*176×144*)	64K	777	342	240	12554	3278	2758
	128K	777	342	240	30608	8452	6185

To verify the performances of the proposed techniques, extensive experiments were carried out. First, we have simulated the situation of the I-P structure with *L=15*. Results of the experiments are used to compare the performance of the conventional video streaming system mentioned in Section 2.3. Two different versions of our proposed streaming systems have been realized, 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 4.6. The average number of MBs sent for decoding is directly proportional to the decoder complexity. In Table 4.6, 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 Table 4.7 and Table 4.8. The savings in both the bits to be sent over the network and MBs to be decoded by the decoder are *39-82*%. for these sequences. In other words, the decoder complexity for playing those

sequences in reverse order is reduced by *39-82*%. 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 is still a saving of *16-34*%.

Table 4.7. Performance improvement of the proposed systems, SI and SI+DA, over the conventional system in terms of the number of MBs to be decoded by the decoder.

Sequences	Bitrate	Saving of macroblocks to be decoded by the decoder	
		SI	SI+DA
Salesman (*352×288*)	1.5M	38.49%	49.32%
	3M	38.49%	49.32%
Football (*352×240*)	1.5M	16.64%	22.76%
	3M	16.64%	22.76%
Table Tennis (*352×240*)	1.5M	33.65%	44.39%
	3M	33.65%	44.39%
Foreman (*176×144*)	64K	19.58%	31.84%
	128K	19.58%	31.84%
Carphone (*176×144*)	64K	24.64%	41.81%
	128K	24.64%	41.81%
Claire (*176×144*)	64K	71.06%	83.86%
	128K	71.06%	83.86%
Grandma (*176×144*)	64K	56.05%	69.06%
	128K	56.05%	69.06%

Table 4.8. 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 (*352×288*)	1.5M	40.05%	48.72%
	3M	40.38%	50.70%
Football (*352×240*)	1.5M	12.68%	16.93%
	3M	14.06%	19.07%
Table Tennis (*352×240*)	1.5M	22.09%	28.57%
	3M	23.23%	30.50%
Foreman (*176×144*)	64K	23.00%	25.10%
	128K	22.23%	28.61%
Carphone (*176×144*)	64K	27.05%	29.36%
	128K	24.25%	32.65%
Claire (*176×144*)	64K	81.75%	84.47%
	128K	73.79%	80.99%
Grandma (*176×144*)	64K	73.89%	78.03%
	128K	72.39%	79.79%

To further reduce the number of MBs 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 motion-compensated MB into one for FMBs. Table 4.6, Table 4.7, and Table 4.8 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 4.7. 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 backward 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 interesting 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 4.8. 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 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.

Figure 4.7, Figure 4.8, and Figure 4.9 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” , “Carphone”, and “Claire” sequences in the backward-play operation respectively. It is obvious that the proposed methods can achieve significant performance improvement in terms of decoder complexity and network traffic. Note that, as shown in Figure 4.7, Figure 4.8, and Figure 4.9, 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. However, SI+DA can get some savings of the last frame of the GOP in both the bits to be sent and the MBs to be decoded due to the contribution of the DCT-domain technique. For this frame, there is no BMB, all the MBs are treated as FMBs. When some non-MC MBs exist for reconstructing one FMB, the technique of direct addition of DCT coefficients can combine several non-MC MBs into one. The savings can then be achieved. This is another improvement of SI+DA over SI as shown in Figure 4.7, Figure 4.8, and Figure 4.9.

Theoretically, both VLC-domain and DCT-domain techniques for BMBs and FMBs will not introduce any quality degradation during backward 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 8x8 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 backward 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 4.9 shows the PSNR degradation of the proposed systems. 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 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 nearly maintained as shown in Table 4.9.

Table 4.9. PSNR performances of SI and SI+DA.

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

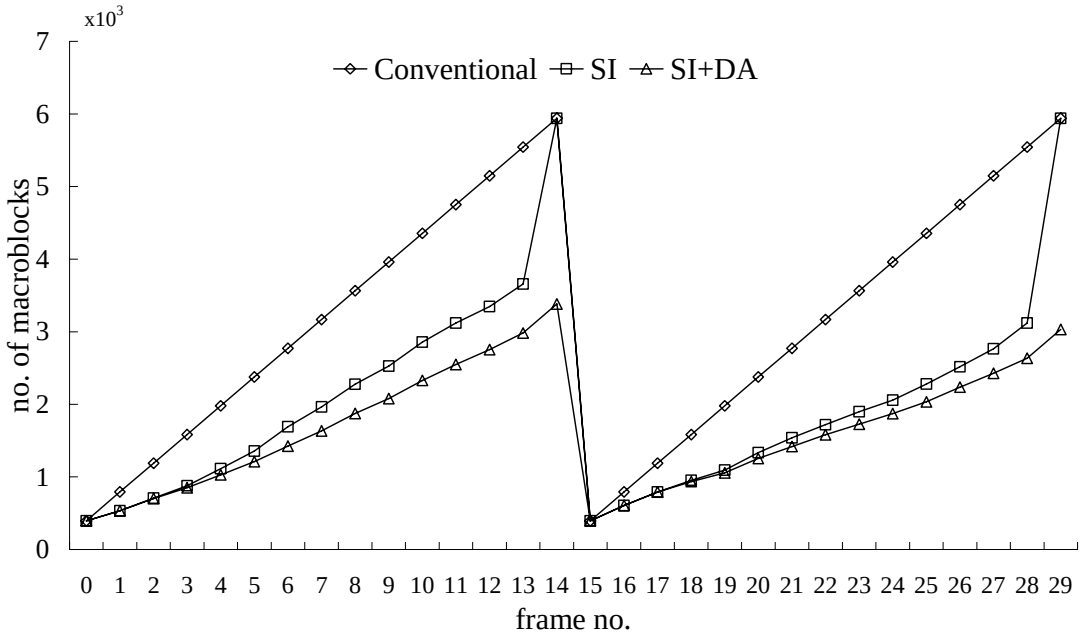
Second, let us demonstrate the performance of the proposed SI+DA with different *L*. Table 4.10 and Table 4.11 depict the performance improvement of the average number of MBs to be decoded and the average number of bits to be sent with respect to different *L* respectively. Equation (2) indicates that, if *L* is large, the average number of frames need to be transmitted and decoded for a requested frame in the backward-play operation is increased, thereby leading to a significant increase of decoding complexity and network traffic. From Table 4.10 and Table 4.11, we show that SI+DA has a better improvement in both decoder complexity and network traffic for large *L*. This further demonstrates the effect of the proposed techniques when applied to the MPEG video system with VCR functionality.

Table 4.10. The saving of the average number of MBs that need to be decoded of SI+DA as compared with the conventional system for different *L*.

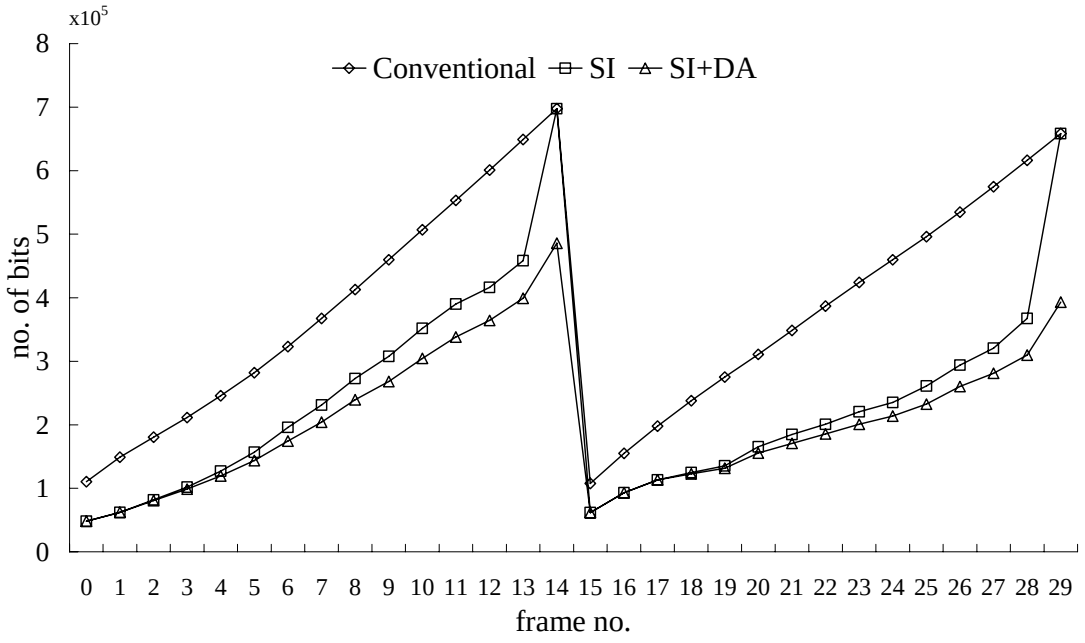
Sequences	Bitrate	Saving of macroblocks to be sent over the network		
		L=7	*L=15*	*L=30*
Salesman (*352×288*)	1.5M	39.98%	49.32%	53.26%
	3M	39.98%	49.32%	53.26%
Football (*352×240*)	1.5M	18.01%	22.76%	25.53%
	3M	18.01%	22.76%	25.53%
Table Tennis (*352×240*)	1.5M	35.03	44.39%	49.74%
	3M	35.03	44.39%	49.74%
Foreman (*176×144*)	64K	25.32	31.84%	32.75%
	128K	25.32	31.84%	32.75%
Carphone (*176×144*)	64K	34.01	41.81%	47.33%
	128K	34.01	41.81%	47.33%
Claire (*176×144*)	64K	68.45	83.86%	90.79%
	128K	68.45	83.86%	90.79%
Grandma (*176×144*)	64K	57.36	69.06%	74.51%
	128K	57.36	69.06%	74.51%

Table 4.11. The 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 (*352×288*)	1.5M	44.43%	48.72%	50.56%
	3M	43.30%	50.70%	54.12%
Football (*352×240*)	1.5M	16.22%	16.93%	16.65%
	3M	16.85%	19.07%	19.87%
Table Tennis (*352×240*)	1.5M	29.96%	28.57%	27.69%
	3M	29.39%	30.50%	31.73%
Foreman (*176×144*)	64K	29.53%	25.10%	20.99%
	128K	28.18%	28.61%	27.34%
Carphone (*176×144*)	64K	33.11%	29.36%	26.49%
	128K	32.33%	32.65%	35.37%
Claire (*176×144*)	64K	78.54%	84.47%	86.54%
	128K	73.51%	80.99%	84.90%
Grandma (*176×144*)	64K	73.79%	78.03%	80.32%
	128K	73.22%	79.79%	83.25%

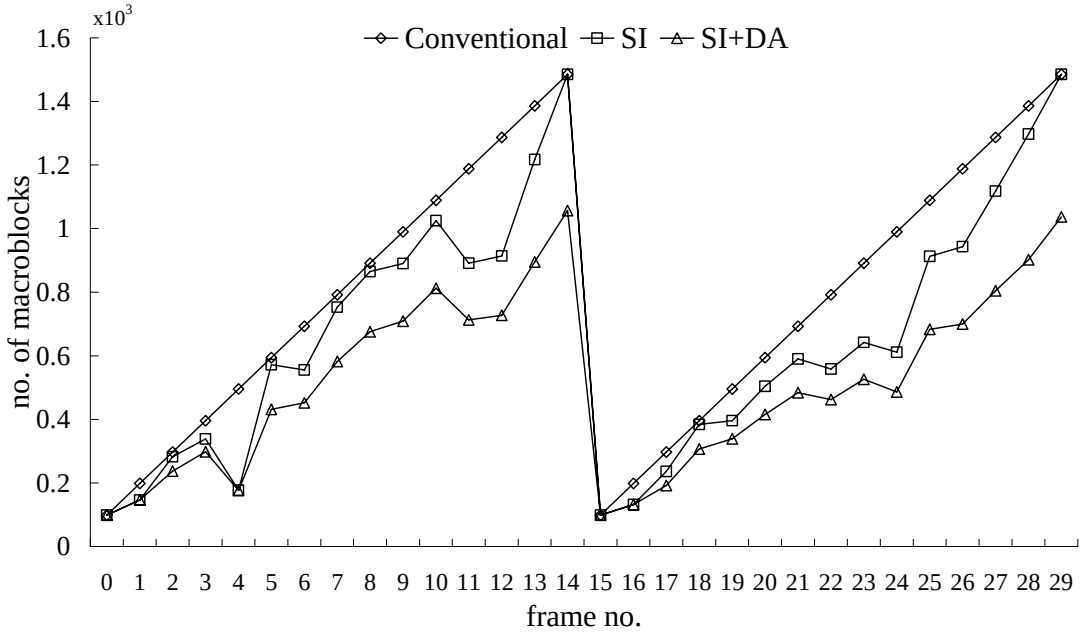


(a)

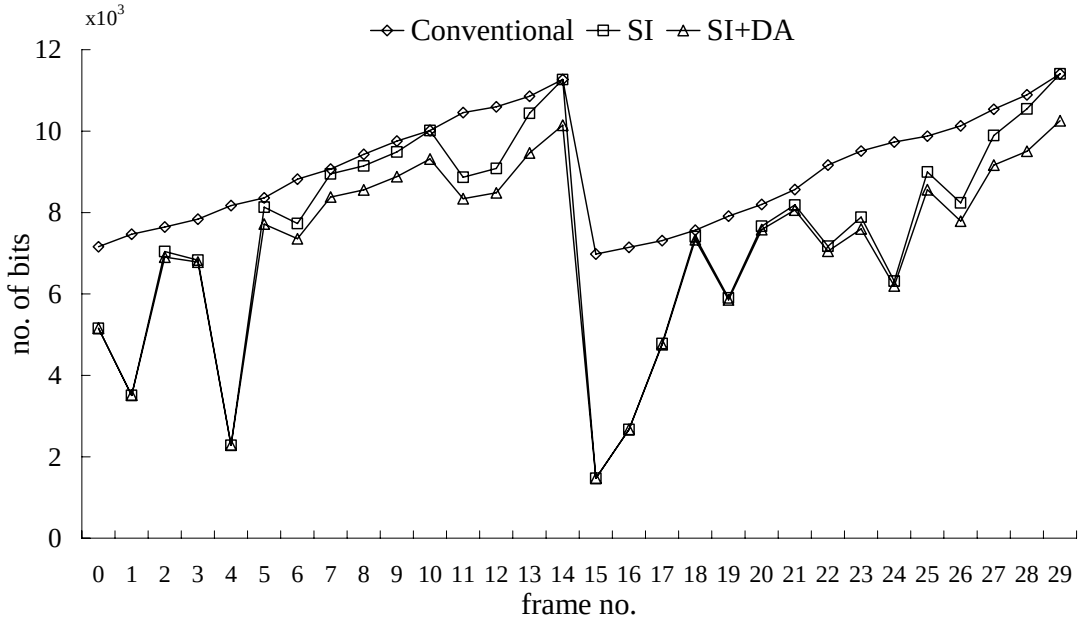


(b)

Figure 4.7. Performance of the conventional system and the proposed systems: SI and SI+DA, for the “Salesman” sequence encoded at 1.5Mb/s in the backward-play operation. (a) Number of MBs to be decoded by the decoder, and (b) number of bits to be sent over

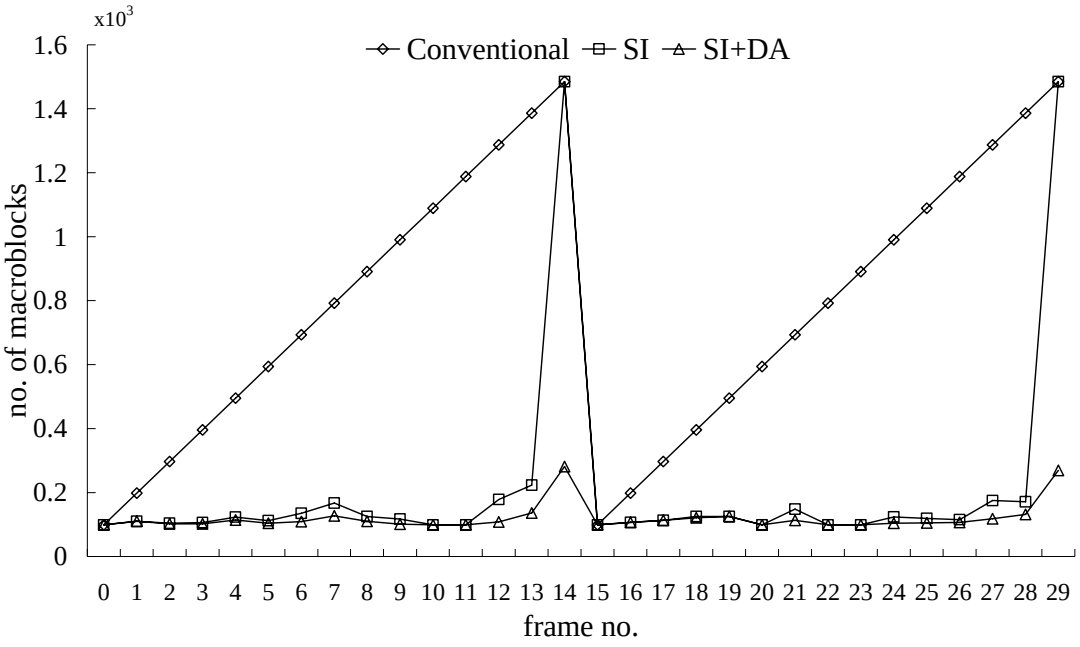


(a)

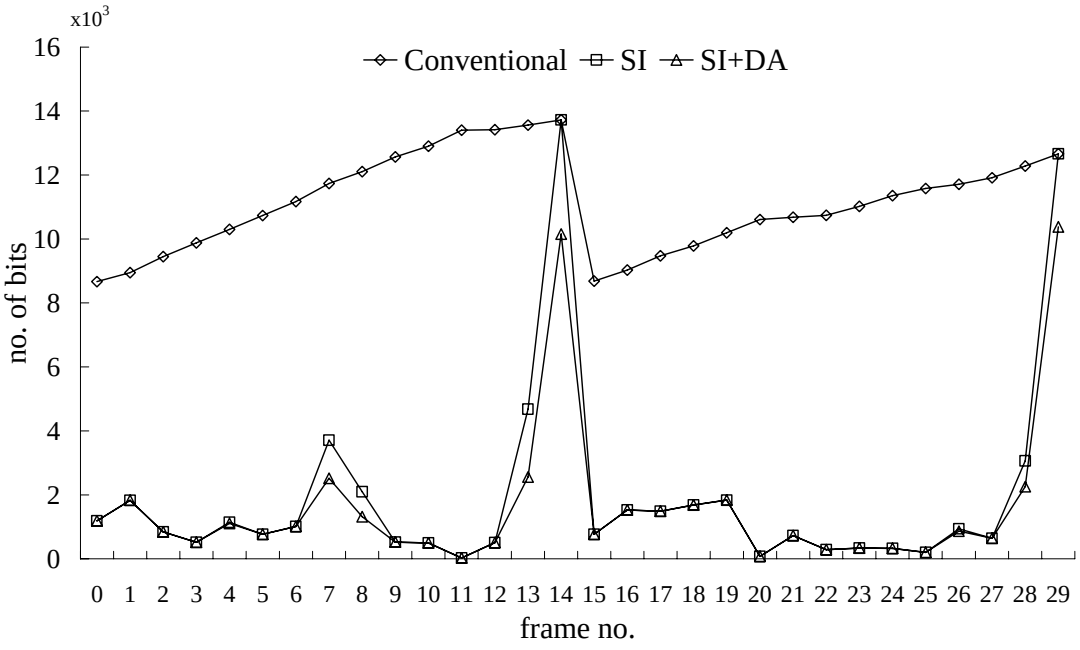


(b)

Figure 4.8. Performance of the conventional system and the proposed systems: SI and SI+DA, for the “Carphone” sequence encoded at 64 Kb/s in the backward-play operation. (a) Number of MBs to be decoded by the decoder, and (b) number of bits to be sent over



(a)



(b)

Figure 4.9. Performance of the conventional system and the proposed systems: SI and SI+DA, for the “Claire” sequence encoded at 64 Kb/s in the backward-play operation. (a) Number of MBs to be decoded by the decoder, and (b) number of bits to be sent over

4.6 Chapter Summary

In this chapter, we have proposed two efficient backward-play techniques for implementing an MPEG video streaming system with VCR functionality. The proposed techniques are motivated by the center-biased motion vector distribution of real-world video sequences. With the motion information, the video streaming server organizes the MBs in the requested frame into two categories – backward MBs (BMBs) and forward MBs (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 motion compensation, 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 backward-play operations. Since BMBs and FMBs are manipulated in 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 backward playback will only be degraded negligibly. Experimental results show that, with our proposed techniques, the MPEG video system with backward-play functionality can minimize the required network bandwidth and decoder complexity significantly. Therefore, the proposed system is a promising solution for MPEG video streaming with VCR functionality.

Chapter 5 Mixed VLC/DCT-domain Technique for FMBs in the MB-based Video Streaming System

5.1 Introduction

The key to high performances in the MB-based system proposed in Chapter 4 lies in the center-biased motion vector distribution of real-world video sequences. For instance, the sign inversion technique manipulating BMBs could be quite helpful in the video sequences containing gentle, smooth and slow motion. When the amount of motion activity in a video sequence increases, the performance of the proposed scheme is dropped since the sequence contains more FMBs in which the sign inversion technique cannot be employed. Though the direct addition technique applied on FMBs can help to reduce the computational burden of the client decoder during backward playback, the reduction is not significant enough as compared with the use of the sign inversion technique. It is due to the fact that only one MB is needed for each BMB. To further enhance our proposed scheme, in this chapter, we investigate whether the sign inversion technique can be applied to FMBs as well.

Some results in this chapter have been reported in references [76].

5.2 Modified architecture using a mixed VLC/DCT-domain technique

In Chapter 4, we have proposed the MB-based architecture for providing backward-play service of a video streaming system with VCR support to reduce the requirements on the decoder complexity and network traffic. Two techniques involving sign inversion and direct addition have been adopted in the proposed architecture to handle BMBs and FMBs in the requested frame, respectively. For the sign inversion technique, it is operated in the VLC domain. The main benefit for this technique is that only one MB from the future frame is needed for the required BMB. On the other hand, the direct addition technique is operated in the DCT domain. Normally, more than one MB (all the related MBs from the previous nearest I-frame to the requested frame) are necessary to be transmitted and decoded. In the following, we try to manipulate FMBs as much as possible in the VLC domain in order to get the benefit of the sign inversion technique.

In this chapter, we modify the video streaming system in Figure 4.1 with the help of the mixed VLC/DCT-domain technique. The architecture of the proposed system is shown in Figure 5.1. The redundant operations involved in backward playback are further reduced by dividing each FMB into two regions. These two regions can be handled either in the DCT domain or VLC domain. The major modification is the functional block highlighted in Figure 5.1.

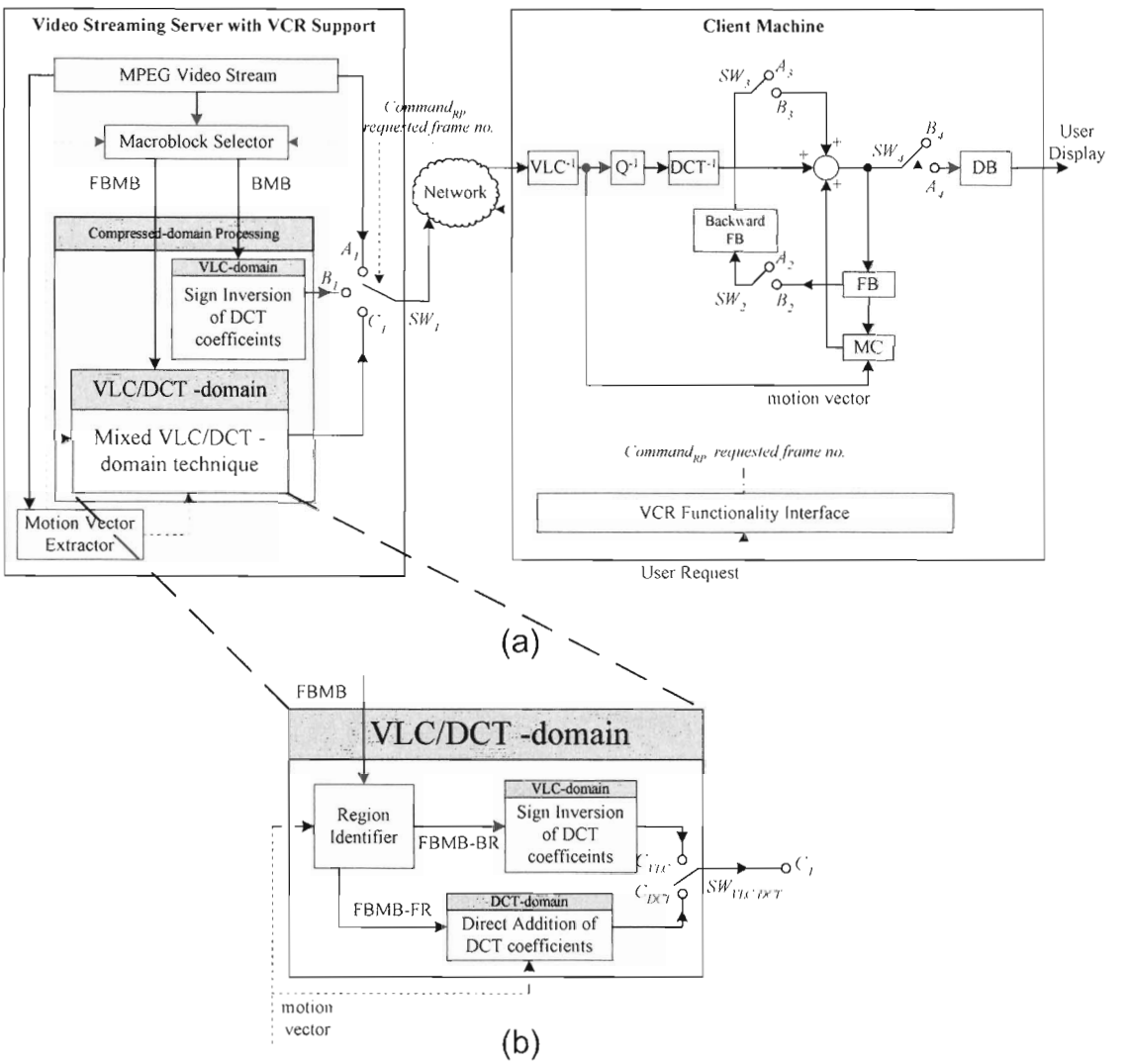


Figure 5.1. The proposed video streaming system with VCR functionality: (a) the overall architecture, and (b) the block diagram of the mixed VLC/DCT-domain technique for FBMBs.

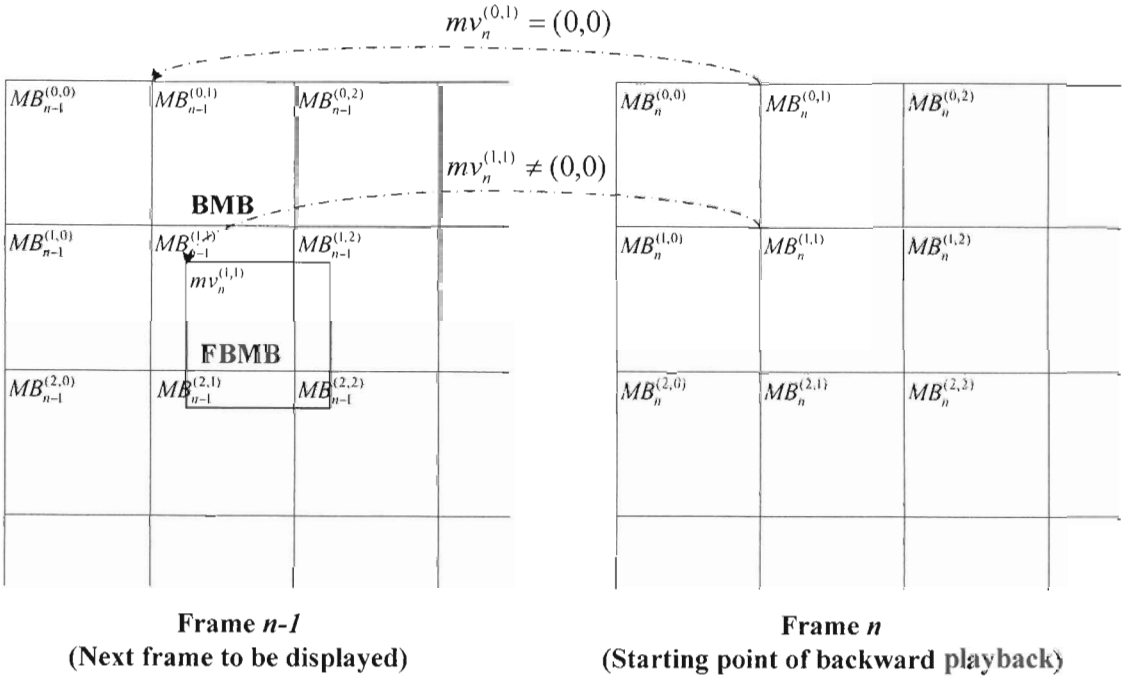


Figure 5.2. Definition of the backward MB (BMB) and the forward/backward MB (FBMB).

For the sake of consistency, we use the example in Figure 2.8 to illustrate the new idea. In this example, a backward-play command is requested at frame *n*. Therefore, frame *n-1* is displayed next. For convenience of our discussion, the MB viewpoint of the proposed scheme in Figure 4.2 is redrawn with some modifications, as shown in Figure 5.2. Again, $MB_{n-1}^{(k,l)}$ stands for the MB at the k^{th} row and l^{th} column of frame *n-1* (the next displayed frame). As compared with Figure 4.2, a FMB is now replaced by a FBMB (forward/backward MB). This symbol is more suitable to discuss the complexity reduction of handling a FMB. As discussed later, BMBs use only the MPEG data of the future frame while FBMBs need the MPEG data from both the past and future frames. By using this concept, our idea in this chapter is to let the DCT-domain technique or the VLC-domain technique to be used in different regions of a FBMB adaptively, and we refer this to as the mixed VLC/DCT-domain technique.

5.3 Region definition in FBMB

As explained in Section 4.4, the VLC-domain technique of sign inversion in Section 4.3 cannot be directly applied to FBMBs since (4.2) is no longer valid for FBMBs. That means $MB_{n-1}^{(k,l)}$ cannot be reconstructed from $MB_{n}^{(k,l)}$. To reconstruct FBMBs of frame *n-1* directly, the server examines the motion vectors in the MPEG video stream and all the related MBs from the previous nearest I-frame to frame *n-2* should be transmitted and decoded. In Figure 5.3, a situation in which $MB_{n-1}^{(1,1)}$ is a FBMB with the motion vector $mv_{n-1}^{(1,1)}$ is illustrated. Two MBs (the shaded MBs) in frame *n-2* are used as references for performing motion compensation of $MB_{n-1}^{(1,1)}$. These two MBs in frame *n-2* again depend on their corresponding MBs in frame *n-3*. The related MBs are continuously identified until the previous nearest I-frame. In this example, the server needs to send seven MBs for $MB_{n-1}^{(1,1)}$ from frame *n-1* to frame *n-3*.

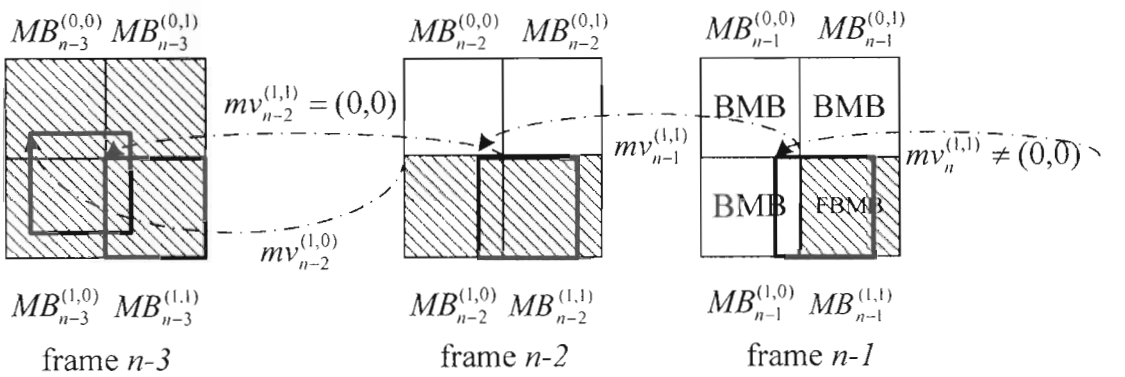


Figure 5.3. A situation in which there is one FBMB in frame *n-1*.

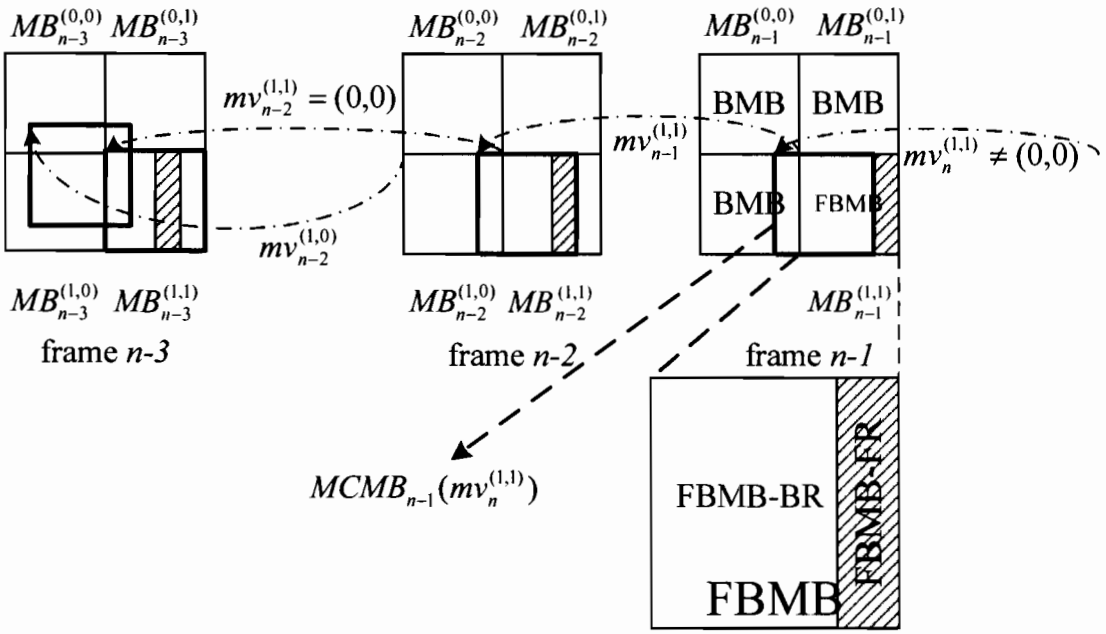


Figure 5.4. Illustration of the forward/backward MB – a backward region (FBMB-BR) and a forward region (FBMB-FR).

To mitigate the decoding complexity and network traffic of processing FBMBs, we suggest exploring the redundancy among the MBs that are required for the reconstruction of FBMBs. Let us use Figure 5.4 to give a clearer account of our idea for reconstructing FBMBs. In Figure 5.4, $MB_{n-1}^{(1,1)}$ is a FBMB whose pixels are divided into two regions. The MB enclosed by the thick line in frame *n-1* is the motion-compensated MB of $MB_n^{(1,1)}, MCMB_{n-1}(mv_n^{(1,1)})$. Pixels in $MB_{n-1}^{(1,1)}$ which overlap with $MCMB_{n-1}(mv_n^{(1,1)})$ belong to a backward region (FBMB-BR). Otherwise they are considered as a forward region (FBMB-FR). In the following, we show that the VLC-domain technique can still be used for FBMB-BRs so that all pixels of a FBMB-BR would be backward reconstructed from the future frame in order to reduce the decoder and channel burdens. On the other hand, the pixels of the corresponding FBMB-FR are forward reconstructed from the past frames. Section 5.4 describes how the VLC-domain technique can be applied to FBMB-BRs, while Section 5.5 presents the DCT-domain technique for the

case of FBMB-FRs. By doing so, we can further improve the performance of the proposed video streaming system with VCR supported as mentioned in Chapter 4.

5.4 VLC-domain technique for backward region in FBMB

The VLC-domain technique mentioned in Section 4.3 of the last chapter helps to reduce both network traffic and decoder complexity. Besides, this technique can also minimize the computational complexity required for the server since it only processes MBs in the VLC-domain. In order to keep the benefits of the VLC-domain technique, we propose to process the MPEG bitstream as much as possible in the VLC domain. Specifically, we propose to reconstruct all pixels in each FBMB-BR in the VLC-domain according to the following formulation. For block motion-compensated prediction, each MB in frame *n*, $MB_n^{(k,l)}$, is reconstructed by motion-compensated prediction and it is given by (4.1), which can be rewritten as

$$MCMB_{n-1}(mv_n^{(k,l)}) = MB_n^{(k,l)} + \tilde{e}_n^{(k,l)} \tag{5.1}$$

where, $\tilde{e}_n^{(k,l)} = -e_n^{(k,l)}$. Note that pixel values of $MB_n^{(k,l)}$ can be obtained in the frame buffer of the decoder. (5.1) implies that pixels in the FBMB-BR can be reconstructed by only sending the sign inversion of the quantized DCT coefficients of $e_n^{(k,l)}$. Similar to the technique used in BMBs, all these sign-inverted coefficients can be generated in the VLC-domain. In other words, the pixels in the requested FBMB of frame *n-1* overlapping with $MCMB_{n-1}(mv_n^{(1,1)})$ (FBMB-BR) are computed from frame *n*, which is already stored

in the client machine. This signifies that the server only needs to send the prediction errors of one MB to the client side for decoding the FBMB-BR.

We are now going to explain the significance of using the VLC-domain technique in the FBMB-BR. Let us refer to the example in Figure 5.3 again. We can see that seven MBs are required for the reconstruction of $MB_{n-1}^{(1,1)}$. The situation gets worse when the requested FBMB is far away from the previous nearest I-frame. However, by applying the VLC-domain technique, we find that only FBMB-FR of the requested FBMB needs to be reconstructed from the MBs in the previous frames. This is illustrated in the example as shown in Figure 5.4. In this example, only the shaded region of $MB_{n-1}^{(1,1)}$ (FBMB-FR) is reconstructed from the previous frames. In frame *n*-2, only $MB_{n-2}^{(1,1)}$ is actually required. Although another MB $MB_{n-2}^{(1,0)}$ is also covered by the motion-compensated MB of $MB_{n-1}^{(1,1)}$, it is no longer required. The reason is that the FBMB-BR of $MB_{n-1}^{(1,1)}$ can be reconstructed from frame *n*, which is available at the decoder. Similarly, in frame *n*-3, only $MB_{n-3}^{(0,1)}$ needs to be sent over the network and decoded by the decoder. Altogether, instead of sending seven MBs as depicted in Figure 5.3, the server needs to transmit only four MBs, including three MBs from the previous frames and one MB from frame *n* for the FBMB-FR and FBMB-BR respectively. The necessary MBs used to reconstruct $MB_{n-1}^{(1,1)}$ can be reduced considerably. If the length of GOP is longer, the savings of the proposed technique could be even larger.

5.5 DCT-domain technique for forward region in FBMB

After obtaining the required pixels of the FBMB-BR, we are now interested in reconstructing pixels of each FBMB-FR from the previous frames. To keep the decoding effort as little as possible, the technique of direct addition of DCT coefficients for FMBs proposed in Section 4.4 can be employed for the FBMB-FRs in the similar way.

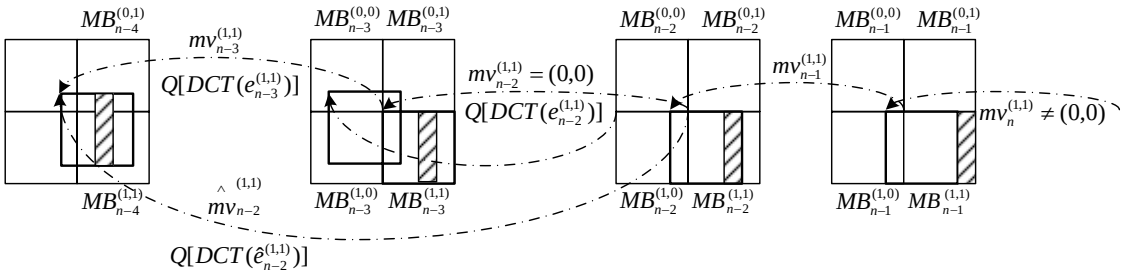


Figure 5.5. Direct addition of DCT coefficients for a FBMB-FR.

Figure 5.5 further extends the process of motion compensation to frame *n-4* of the example as shown in Figure 5.4. Among those referenced MBs of $MB_{n-1}^{(1,1)}$, $MB_{n-2}^{(1,1)}$ is a non-MC MB since its motion vector, $mv_{n-2}^{(0,1)}$, is equal to zero. In the reconstruction process of $MB_{n-1}^{(1,1)}$, $MB_{n-2}^{(1,1)}$ is needed. Therefore, the decoder requires decoding the prediction errors $e_{n-2}^{(1,1)}$ in frame *n-2*, $e_{n-3}^{(1,1)}$ in frame *n-3*, and so on. If pixels in $MB_{n-3}^{(1,1)}$ are used as the reference for $MB_{n-2}^{(1,1)}$ only, it is wasteful to decode $e_{n-3}^{(1,1)}$ in the decoder. In this situation, we can apply the DCT-domain technique in combining $e_{n-2}^{(1,1)}$ and $e_{n-3}^{(1,1)}$ in one single MB for the non-MC MB. This strategy can reduce the required number of MBs that are decoded by the decoder.

The use of direct addition of DCT coefficient to combine $e_{n-2}^{(1,1)}$ and $e_{n-3}^{(1,1)}$ in the server for FBMB-FR of $MB_{n-2}^{(1,1)}$ is similar as the process described in Section 4.4. For example, pixels in $MB_{n-3}^{(1,1)}$ are not necessary to be decoded directly in the client machine. Thus, the incoming quantized DCT coefficients of the prediction error from the original video stream, $Q[DCT(e_{n-2}^{(1,1)})]$, are no longer valid because they refer to the pixels, which are not stored in FB. Since $mv_{n-2}^{(0,1)}$ is equal to zero in this example, the direct addition technique can then be used. From (4.5) and (4.13), the server can adopt the direction addition technique to compute the new motion vector, $\hat{mv}_{n-2}^{(1,1)}$, and prediction errors in the DCT-domain, $Q[DCT(\hat{e}_{n-2}^{(1,1)})]$, by using frame *n-4* as the reference (see Figure 5.5):

$$\begin{aligned}\hat{mv}_{n-2}^{(1,1)} &= mv_{n-2}^{(1,1)} + mv_{n-3}^{(1,1)} \\ &= mv_{n-3}^{(1,1)}\end{aligned} \tag{5.2}$$

and

$$Q[DCT(\hat{e}_{n-2}^{(1,1)})] = Q[DCT(e_{n-2}^{(1,1)})] + Q[DCT(e_{n-3}^{(1,1)})] \tag{5.3}$$

This equation signifies the fundamental idea of direct addition of DCT coefficients. In (5.3), $Q[DCT(\hat{e}_{n-2}^{(1,1)})]$ can be computed in the DCT-domain by adding $Q[DCT(e_{n-2}^{(1,1)})]$ and $Q[DCT(e_{n-3}^{(1,1)})]$, which can be directly retrieved from the MPEG video stream in the server. By doing so, no decoding and re-encoding is necessary, and therefore the computational complexity required for the server is very insignificant. In the client machine, there is another advantage. Instead of decoding $Q[DCT(e_{n-2}^{(1,1)})]$ and $Q[DCT(e_{n-3}^{(1,1)})]$ for $MB_{n-2}^{(1,1)}$, the client machine only needs to decode one MB ($Q[DCT(\hat{e}_{n-2}^{(1,1)})]$) when the

direct addition of DCT coefficients is adopted, which can further minimize the computational complexity of the decoder. It is noted that this DCT-domain technique can also be applied iteratively if $mv_{n-3}^{(1,1)}$ is also equal to zero and pixels in $MB_{n-4}^{(1,1)}$ are used as the reference for $MB_{n-3}^{(1,1)}$ only, as illustrated in Figure 5.6.

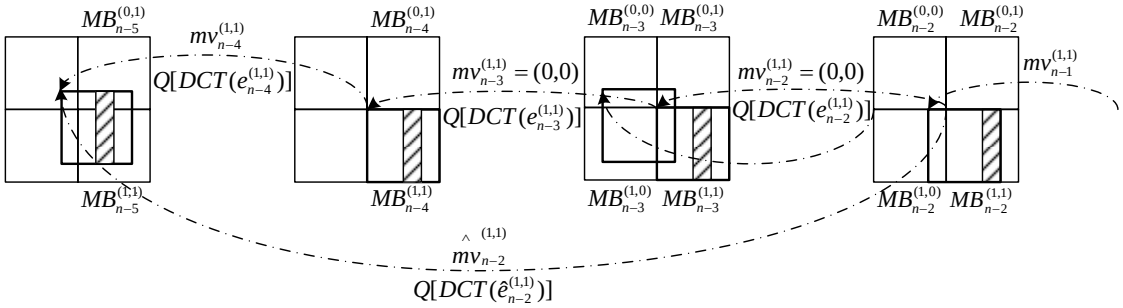


Figure 5.6. Using direct addition iteratively for FBMB-FR.

Subsequently, we will examine the switch positions of the improved video streaming system when FBMBs are employed for the backward-play operation. The internal switch $SW_{VLC/DCT}$ is employed to facilitate the use of the mixed VLC/DCT technique for FBMBs, as depicted in Figure 5.1(b). Owing to the adoption of the VLC-domain technique, for processing FBMB-FRs, switch $SW_{VLC/DCT}$ is connected to C_{VLC}, and the switch positions of SW_2, SW_3 and SW_4 are the same as that of BMBs, as shown in Table 4.4, so that all reconstructed pixels of FBMB-BRs are stored in the backward-FB. To reconstruct the FBMB-FR in frame *n-1,* switches SW_1 and $SW_{VLC/DCT}$ in the server is connected to C_1 and C_{DCT}, respectively, when the DCT-domain technique is adopted. The MB selector then extracts all the related MBs from the previous nearest I-frame to frame *n-2*. These MBs may be manipulated by the DCT-domain technique if non-MC MBs exist and the combined MBs will be sent to the client machine. In

the client machine, all the switches are open and the decoder decodes the necessary MBs from the previous nearest I-frame to frame *n-2*. The decoded pixels in frame *n-2*, which are referred by FBMB-FRs in frame *n-1*, are then put into FB. After that, SW_3 and SW_4 are switched to B_3 and A_4 respectively. Table 5.1 summaries the switch positions of the proposed video streaming system. When FBMB-FRs in frame *n-1* is being decoded, the prediction error between each FBMB and its corresponding motion-compensated MB is required. Reconstructed pixels in the FBMB-FR can be constructed by adding its prediction error to its motion-compensated pixels of frame *n-2* in FB, as depicted in Figure 5.1. Meanwhile, pixels of BMBs and FBMB-BRs stored in backward-FB are then composed with the reconstructed pixels of FBMB-FRs to form frame *n-1*. Note that frame *n-1* is the target frame that should be displayed next in the backward-play operation. Therefore, frame *n-1* is stored in both DB and FB. In addition, frame *n-1* stored in FB can then be used for reconstructing BMBs and FBMB-BRs of the consequent frame, frame *n-2*, during backward playback.

Table 5.1. Switch positions of the proposed video streaming system with the support of the mixed VLC/DCT technique.

Playback modes	MB type	SW_1	SW_2	SW_3	SW_4	$SW_{VLC/DCT}$
Forward	—	A_1	A_2	A_3	A_4	—
Backward	BMB	B_1	B_2	A_3	B_4	—
	FBMB-BR	C_1	B_2	A_3	B_4	C_{DCT}
	FBMB-FR (nearest I-frame to frame *n-2*)	C_1	A_2	A_3	B_4	C_{DCT}
	FBMB-FR (frame *n-1*)	C_1	A_2	B_3	A_4	C_{VLC}

5.6 Experimental results of using the mixed VLC/DCT-domain technique

Extensive experiments have been conducted to evaluate the performances of the mixed VLC/DCT-domain techniques. An MPEG-2 encoder [16] was used to encode the same set of the video sequences in Table 4.5 of Chapter 4. These sequences cover various spatial resolutions and motion characteristics. Again, the start point of the backward-play operation is at the end of the sequence. For all video sequences, the frame-rate of the video stream was *30* frames/s and the GOP length was set to 15.

In the experiments of this section, different techniques have been applied to our improved streaming system: SI+DA and SI+RR (stands for Redundancy Reduction). SI+DA uses the sign inversion technique of DCT coefficients (operated in the VLC-domain) for BMBs and the direct addition technique of DCT coefficients for FBMBs. For SI+RR, we identify two regions in each FBMB and use the sign inversion technique for the FBMB-BR in order to further achieve redundancy reduction for the FBMB-FR. Besides, SI+RR also adopts the DCT-domain technique to manipulate the FBMB-FR. Note that, both SI+DA and SI+RR retain almost the same reconstruction quality as that of the conventional system since re-encoding is not necessary in the proposed techniques.

The performance comparisons in terms of the average number of MBs to be decoded and bits to be sent are given in Table 5.2. As discussed in section 4.5,

SI+DA can obtain significant savings as compared with the conventional system in all sequences, as also shown in Table 5.2. In this table, we also show that SI+RR can provide further reductions on the decoder complexity and network traffic. Table 4.7 and Table 4.8 clearly figure out that the average number of MBs to be decoded and bits to be sent can be reduced by up to about 10% in the sequences “Salesman”, “Foreman”, and “Carphone” for SI+RR. Since the motion activities of the FBMBs are limited, the mixed DCT/VLC-domain technique can make good contributions in these sequences. For the rather low motion sequences such as “Claire” and “Grandma”, in which BMBs are dominant, the benefit of using the mixed VLC/DCT-domain technique is not so significant although motion activities of FBMBs are low as well. It is because the SI+DA technique has already saved about 80% in terms of MBs to be decoded and bits to be sent, the potential for further improvement is very restricted.

Table 5.2. Detailed comparisons among SI+DA, SI+RR, and the conventional system.

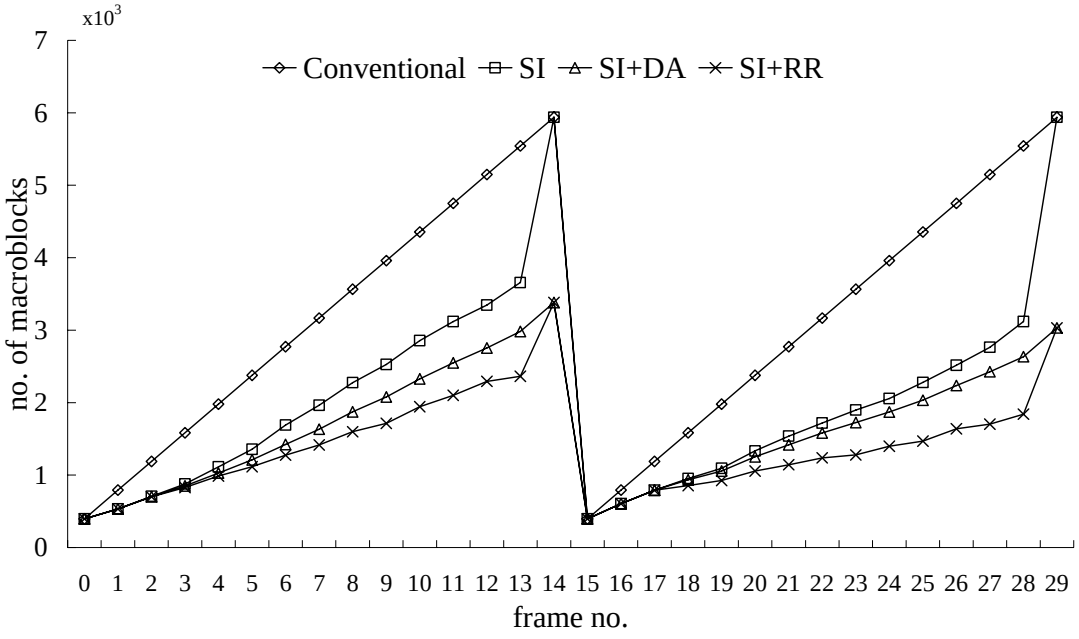
Sequences	Bitrate	Average no. of MBs to be decoded by the decoder (MBs)			Average no. of bits to be sent over the network (bits)		
		Conventional System	SI+DA	SI+RR	Conventional System	SI+DA	SI+RR
Salesman (*352×288*)	1.5M	3109	1575	1310	373996	191773	159827
	3M	3109	1575	1310	796185	392541	326949
Football (*352×240*)	1.5M	2591	2001	1929	368697	306286	296807
	3M	2591	2001	1929	815826	660250	638655
Table Tennis (*352×240*)	1.5M	2591	1441	1404	368660	263341	257556
	3M	2591	1441	1404	747080	519223	507933
Foreman (*176×144*)	64K	777	530	474	9610	7198	6102
	128K	777	530	474	26175	18686	16263
Carphone (*176×144*)	64K	777	452	383	8466	5980	4857
	128K	777	452	383	25192	16966	14072
Claire (*176×144*)	64K	777	125	119	10582	1643	1467
	128K	777	125	119	31481	5985	5406
Grandma (*176×144*)	64K	777	240	205	12554	2758	2114
	128K	777	240	205	30608	6185	4809

Table 5.3. Performance improvements of SI+DA and SI+RR, over the conventional system in terms of the number of MBs to be decoded by the decoder.

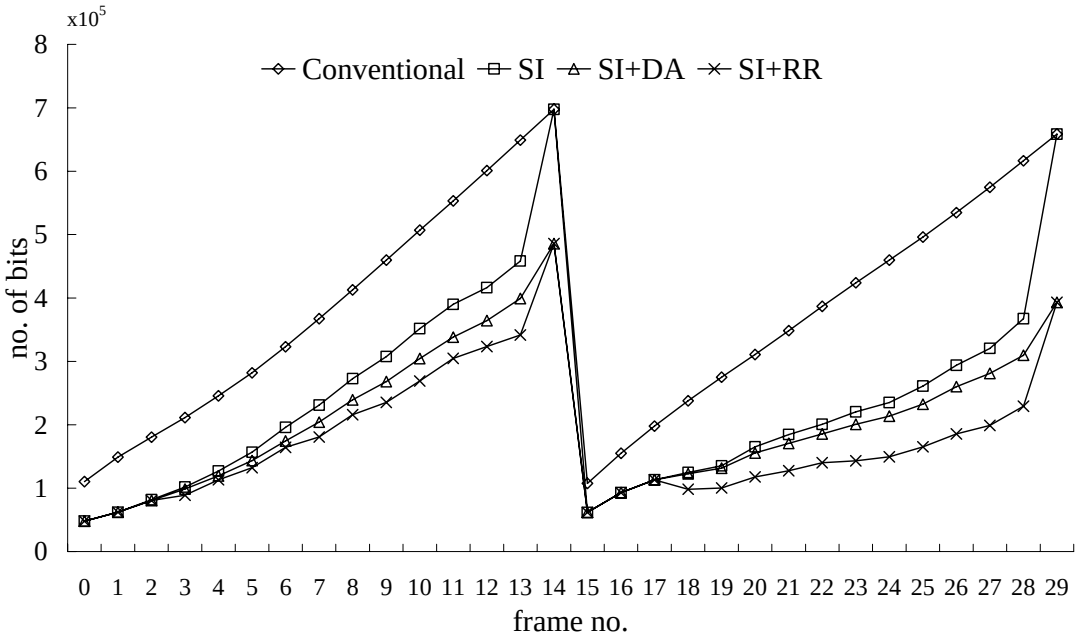
Sequences	Bitrate	Saving of macroblocks to be decoded by the decoder	
		SI+DA	SI+RR
Salesman (*352×288*)	1.5M	49.32%	57.85%
	3M	49.32%	57.85%
Football (*352×240*)	1.5M	22.76%	25.52%
	3M	22.76%	25.52%
Table Tennis (*352×240*)	1.5M	44.39%	45.80%
	3M	44.39%	45.80%
Foreman (*176×144*)	64K	31.84%	39.02%
	128K	31.84%	39.02%
Carphone (*176×144*)	64K	41.81%	50.75%
	128K	41.81%	50.75%
Claire (*176×144*)	64K	83.86%	84.75%
	128K	83.86%	84.75%
Grandma (*176×144*)	64K	69.06%	73.59%
	128K	69.06%	73.59%

Table 5.4. Performance improvements of SI+DA and SI+RR, 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+DA	SI+RR
Salesman (*352×288*)	1.5M	48.72%	57.27%
	3M	50.70%	58.94%
Football (*352×240*)	1.5M	16.93%	19.50%
	3M	19.07%	21.72%
Table Tennis (*352×240*)	1.5M	28.57%	30.14%
	3M	30.50%	32.01%
Foreman (*176×144*)	64K	25.10%	36.51%
	128K	28.61%	37.87%
Carphone (*176×144*)	64K	29.36%	42.63%
	128K	32.65%	44.14%
Claire (*176×144*)	64K	84.47%	86.14%
	128K	80.99%	82.83%
Grandma (*176×144*)	64K	78.03%	83.16%
	128K	79.79%	84.29%

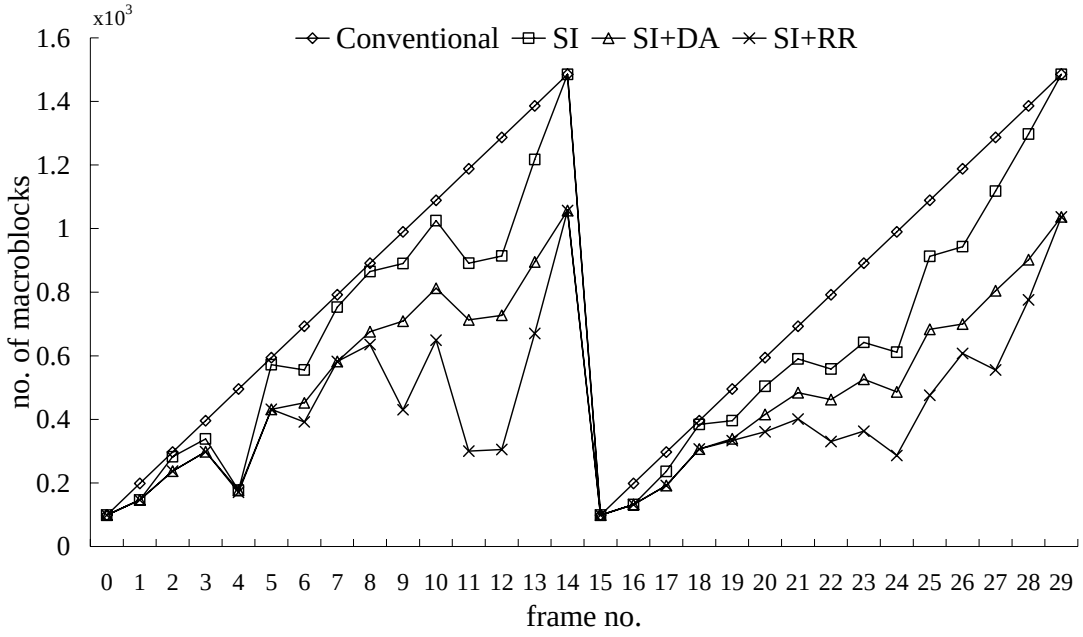


(a)

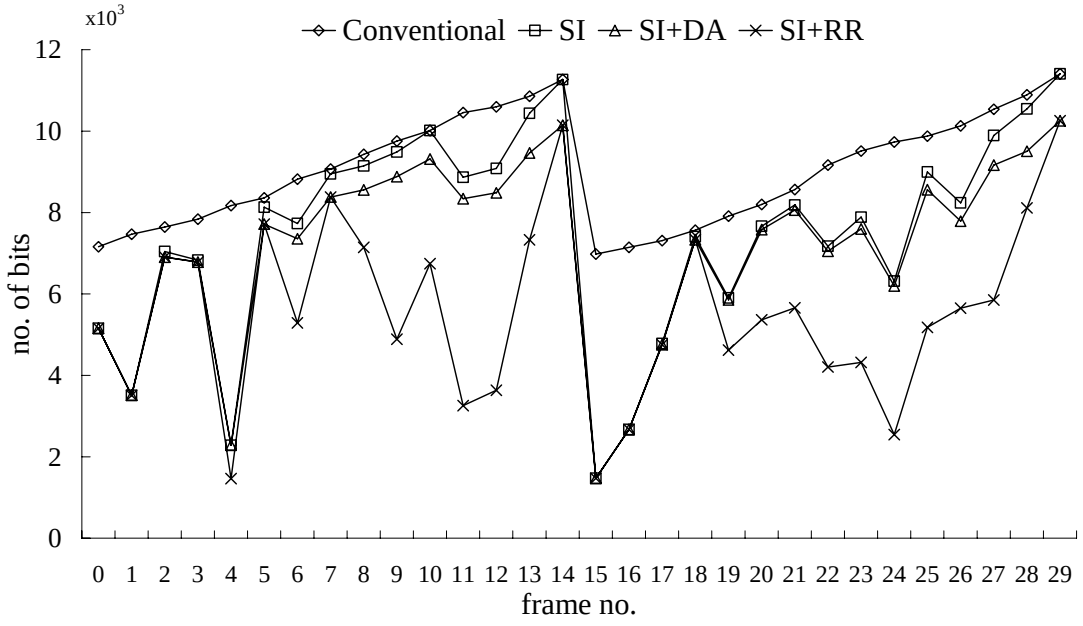


(b)

Figure 5.7. Performance of the conventional system and the proposed systems: SI, SI+DA and SI+RR, for the “Salesman” sequence encoded at 1.5 Mb/s in the backward-play operation. (a) Number of MBs to be decoded by the decoder, and (b) number of bits to be sent over the network.

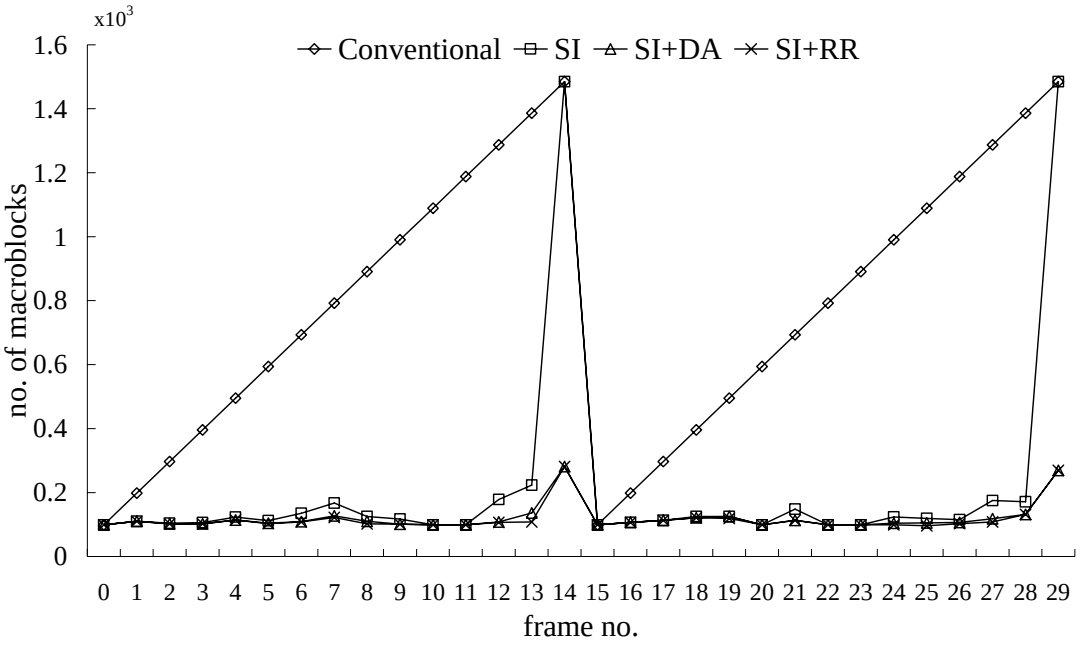


(a)

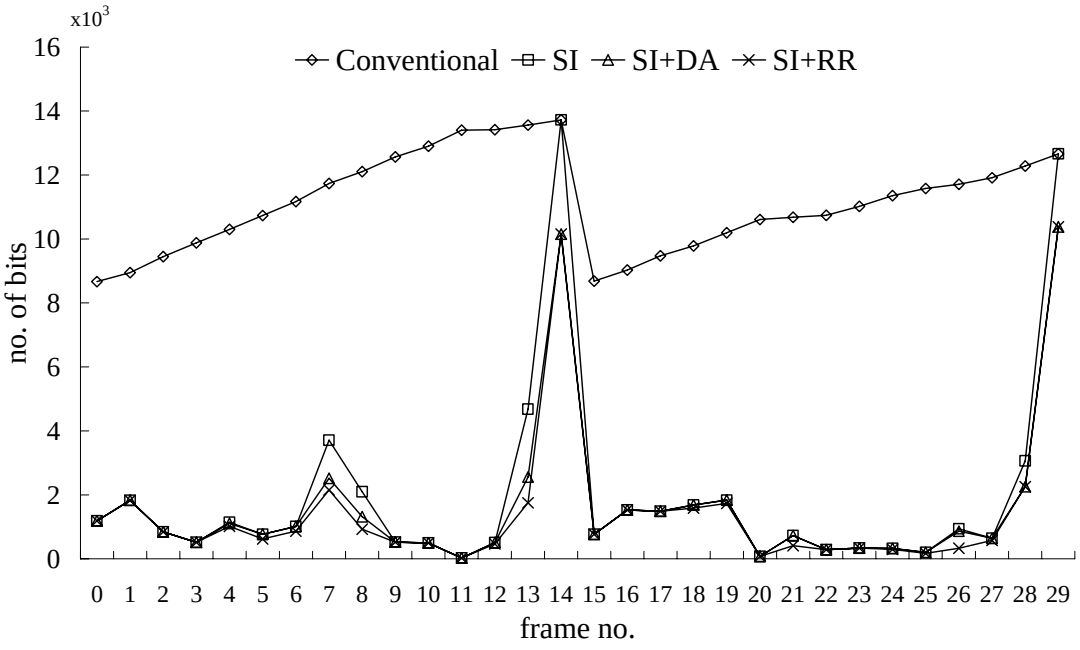


(b)

Figure 5.8. Performance of the conventional system and the proposed systems: SI, SI+DA and SI+RR, for the “Carphone” sequence encoded at 64kb/s in the backward-play operation. (a) Number of MBs to be decoded by the decoder, and (b) number of bits to be sent over the network.



(a)



(b)

Figure 5.9. Performance of the conventional system and the proposed systems: SI, SI+DA and SI+RR, for the “Claire” sequence encoded at 64kb/s in the backward-play operation. (a) Number of MBs to be decoded by the decoder, and (b) number of bits to be sent over the network.

The frame-by-frame comparisons of the required number of MBs to be decoded and the required number of bits to be transmitted of the conventional approach, SI, SI+DA and SI+RR are shown in Figure 5.7, Figure 5.8, and Figure 5.9 for

the “Salesman”, “Carphone”, and “Claire” sequences in the backward-play operation respectively. These figures show that SI+RR has the best performances in terms of the decoder complexity and network traffic among all techniques. The performances of the newly proposed SI+RR have more significant when the length of the GOP, *L*, increases, as shown in Table 5.5 and Table 5.6. For *L* equal to 30, the saving of SI+RR over SI+DA even increases up to 50%. The effective of the new mixed VLC/DCT-domain technique comes from the use of the sign inversion technique on FBMB-BR such that more pixels are handled in the VLC domain for large *L*.

Table 5.5. The saving of the average number of MBs that need to be decoded of SI+RR as compared with SI+DA for different *L*.

Sequences	Bitrate	Saving of macroblocks to be sent over the network		
		L=7	*L=15*	*L=30*
Salesman (*352×288*)	1.5M	3.87%	16.82%	30.75%
	3M	3.87%	16.82%	30.75%
Football (*352×240*)	1.5M	0.18%	3.58%	16.45%
	3M	0.18%	3.58%	16.45%
Table Tennis (*352×240*)	1.5M	0.46%	2.54%	8.97%
	3M	0.46%	2.54%	8.97%
Foreman (*176×144*)	64K	1.05%	10.53%	24.11%
	128K	1.05%	10.53%	24.11%
Carphone (*176×144*)	64K	2.94%	15.73%	32.34%
	128K	2.94%	15.73%	32.34%
Claire (*176×144*)	64K	0.62%	5.52%	21.28%
	128K	0.62%	5.52%	21.28%
Grandma (*176×144*)	64K	2.51%	14.64%	31.88%
	128K	2.51%	14.64%	31.88%

Table 5.6. The saving of the average number of bits that need to be sent of SI+RR as compared with SI+DA for different *L*.

Sequences	Bitrate	Saving of bits to be sent over the network		
		L=7	*L=15*	*L=30*
Salesman (*352×288*)	1.5M	5.09%	16.66%	30.26%
	3M	4.25%	16.71%	31.07%
Football (*352×240*)	1.5M	0.23%	3.09%	15.38%
	3M	0.20%	3.27%	15.84%
Table Tennis (*352×240*)	1.5M	0.57%	2.20%	7.27%
	3M	0.48%	2.17%	7.55%
Foreman (*176×144*)	64K	3.81%	15.23%	25.83%
	128K	3.05%	12.97%	24.49%
Carphone (*176×144*)	64K	7.97%	18.78%	28.98%
	128K	6.56%	17.06%	28.92%
Claire (*176×144*)	64K	2.67%	10.73%	29.06%
	128K	1.91%	9.67%	29.75%
Grandma (*176×144*)	64K	5.53%	23.33%	44.77%
	128K	3.98%	22.25%	51.18%

5.7 Chapter Summary

In this chapter, we have designed a mixed VLC/DCT technique for FBMBs to further mitigate the decoder complexity and network bandwidth requirements when backward playback is requested. For BMBs, the sign inversion technique mentioned in Chapter 4, which is operated in the VLC-domain, is used. The VLC-domain manipulation is very effective since only one MB from the future frame is needed for each BMB. To maximize the benefit of the sign inversion technique, the mixed VLC/DCT-domain technique divides the FBMB into two regions. We have proven that some partitions of FBMB can still be handled in the VLC domain while the remaining parts can be manipulated efficiently by using direction addition of DCT coefficients. With the help of the proposed VLC/DCT-domain technique, FBMBs can be manipulated as much as possible in the VLC domain.

All the points related to our proposed scheme have been verified experimentally. Experimental results show that our MPEG video streaming system provides remarkable backward-play quality and is able to further minimize the required network bandwidth and decoder complexity significantly. As a concluding remark, the new VLC/DCT-domain technique in this chapter can strength the practicality of the proposed video streaming system with VCR support.

Chapter 6 Establishment of Linkage across GOP boundaries for Backward Playback

6.1 Introduction

By exploring the motion relationship between two adjacent frames, we have developed a macroblock-based (MB-based) scheme in Chapter 4 and made further enhancement in Chapter 5 for efficiently implementing backward playback on compressed video bitstreams. This scheme can significantly reduce the required decoding complexity and network bandwidth during backward playback without introducing additional storage in the server. However, it suffers from one drawback. Since there is no inter-frame prediction between the last frame of one GOP and the first frame of the successive GOP, the motion relationship disappears. As a result, the sign inversion technique becomes useless for traversing GOP boundaries in the reverse frame order. It leads to drastic increases in the number of MBs to be decoded and the number of bits to be sent at that moment. As shown in Figure 5.7 to Figure 5.9, although the peak bitrate caused by the last frame of each GOP could be alleviated to some extent by using the direct addtion technique, it still exists. These high bitrate peaks in streaming video might result in dropped packets, which then causes jerky video during reverse playback. In this chapter, by making use of a new SP picture type supported in the H.264, we solve the GOP discontinuity problem by building linkages across GOP boundaries.

6.2 One-dimensional View of the MB-based Scheme

For the convenience of illustration, the same example used previously for this MB-based scheme is depicted in Figure 6.1 in its one-dimensional view with a GOP length of *L*. In this figure, each frame has three MBs. Again, we assume that MB_n^i represents the i^{th} MB in frame *n*. Since the motion vectors of both forward and backward directions are used in this chapter to establish linkages across GOP boundaries, the corresponding motion vector of MB_n^i is represented by $mv_{n-1\to n}^i$, where frame *n* and frame *n-1* is the current frame and the reference frame respectively. Suppose a backward-play command is issued at frame *n+1*, the next frame to be displayed is frame *n*, that is to say, our target is to reconstruct all MBs in frame *n*. As discussed in Chapter 4, our MB-based scheme exerts the center-biased motion vector distribution of real-world video sequences on backward playback. With the motion vectors, the video server classifies MB_n^i into two categories – backward MBs (BMBs) and forward MBs (FMBs). MB_n^i is classified as a BMB if its co-located MB in frame *n+1*, i.e. MB_{n+1}^i, is coded with zero motion vector. Otherwise, it is classified as a FMB. In the example shown in Figure 6.1, MB_n^i and MB_n^{i+2} in frame *n* are classified as BMBs since $mv_{n\to n+1}^i$ and $mv_{n\to n+1}^{i+2}$ are equal to zero. On the other hand, MB_n^{i+1} is a FMB due to non-zero motion vector $mv_{n\to n+1}^{i+1}$. During backward playback with our proposed sign inversion technique, BMBs use only MBs from the future frame while FMBs need MBs from the past frames. Taking MB_n^i as an example, since $mv_{n\to n+1}^i$ is equal to zero and MB_n^i is found to be a BMB, the equation (4.1) in Chapter 4 can be written as

$$MB_n^i = MB_{n+1}^i + (-e_{n+1}^i) \tag{6.1}$$

where e_{n+1}^i is the prediction error between MB_{n+1}^i and MB_n^i. It is noted that frame *n+1* is stored in the frame buffer at the decoder when the backward-play operation is requested at frame *n+1*. In other words, pixels of MB_{n+1}^i have been already available at the decoder. As demonstrated in Chapter 4, in the server, the quantized transform coefficients of e_{n+1}^i, $Q\left[T\left(e_{n+1}^i\right)\right]$, can be easily extracted from the video bitstream. By performing the sign inversion technique on these coefficients in the VLC domain, we can obtain and transmit $Q\left[T\left(-e_{n+1}^i\right)\right]$, which are decoded to $-e_{n+1}^i$ in the decoder side. The situation of FMB MB_n^{i+1} is different since (6.1) does not hold true for it. In Chapter 4, we also proposed a direct addition technique to deal with the construction of FMBs. A further enhancement was made in Chapter 5 to maximize the benefit of the sign inversion technique in the VLC domain. Since this enhancement will not affect the linkage establishment across GOP boundaries, for the sake of simplicity, we ignore this enhancement in the following discussion.

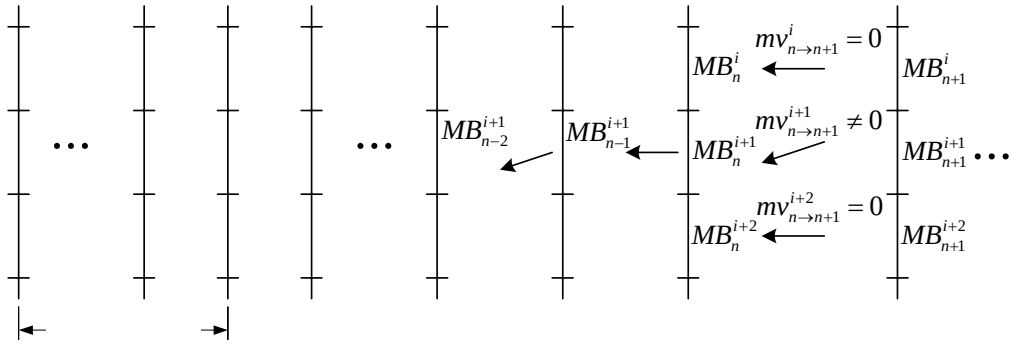


Figure 6.1. One-dimensional view of the MB-based scheme for backward playback.

By adopting the sign inversion technique into the MB-based scheme, huge reductions on both of the decoder complexity and network bandwidth are

achievable, as shown in Figure 5.7 to Figure 5.9. It shows that lower network bandwidth and decoder complexity are required by applying the sign inversion technique into the MB-based scheme. However, the situation gets worse at the GOP boundaries. As we mentioned before, it is due to the reason that inter-frame dependency between the last frame of one GOP and the first frame of the succeeding GOP does not exist, for instance, frame *L-1* (P_{L-1}) and frame *L* (I_L) in Figure 6.1.

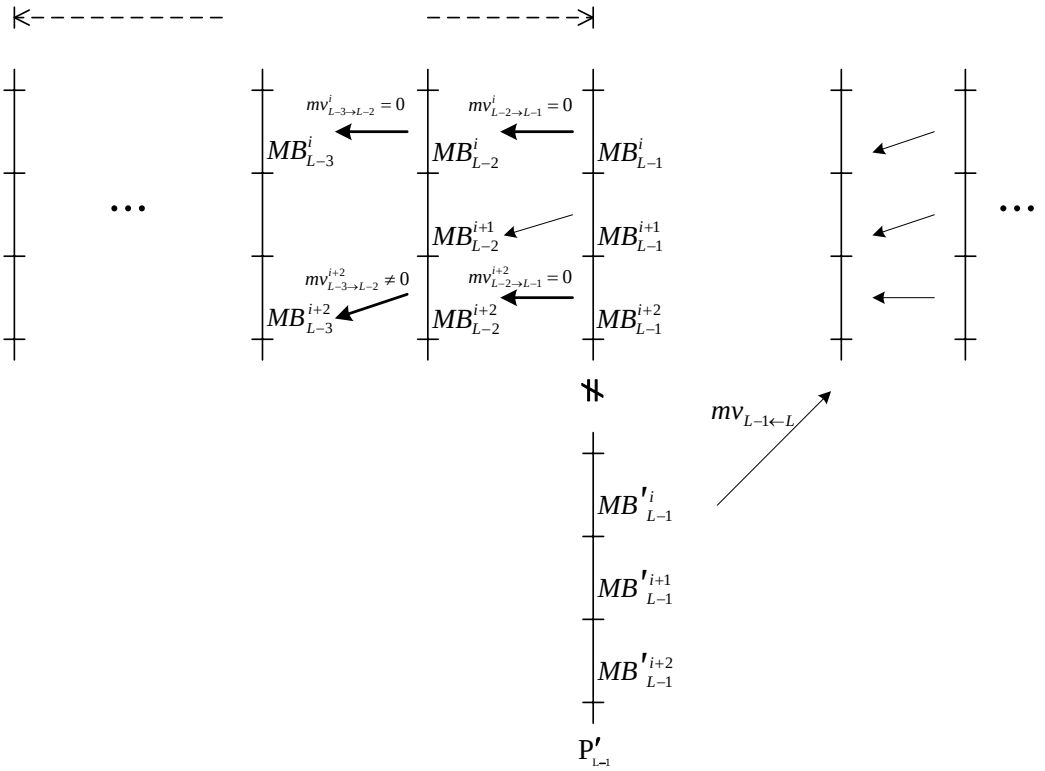
6.3 Establishment of Linkages across GOP Boundaries in the MB-based Backward-play Scheme

In order to cope with the GOP discontinuity of the video bitstream, an extra frame is added to establish the linkage between adjacent GOPs. One possible way for building the relationship between frame *L-1* (P_{L-1}) and frame *L* (I_L) is to re-encode frame *L-1* as P'_{L-1} which uses I_L as the reference, as depicted in Figure 6.2(a). It is noted that the reference frame used in P_{L-1} in the original bitstream is P_{L-2} while the reference frame used in the re-encoded P'_{L-1} is I_L. It means that the reconstructed values of P_{L-1} and P'_{L-1} are not identical because they are predicted from different reference frames. During backward playback across the GOP boundary as shown in Figure 6.2(a), (6.1) can be rewritten as

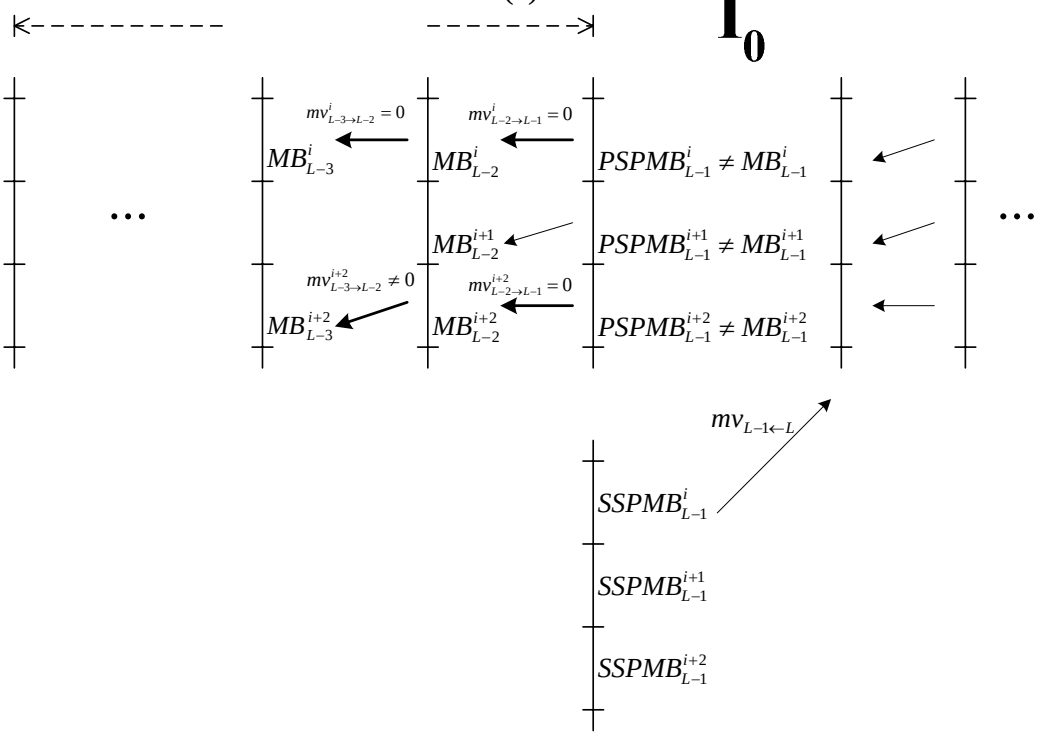
$$MB^i_{L-2} = MB^i_{L-1} + (-e^i_{L-1}) \tag{6.2}$$

In this case, only the decoded MB from P'_{L-1}, MB'^i_{L-1}, is available which is not exactly identical to MB^i_{L-1}. Therefore, from (6.2), the playback quality of MB^i_{L-2} is degraded. The mismatch between P_{L-1} and P'_{L-1} would not only be

confined to a single frame but would further propagate to frame *L-2* due to the use of the sign inversion technique for backward playback.



(a)



(b)

Figure 6.2. Possible ways to establish the linkage across the GOP boundary by using (a) a P-frame, and (b) an SP-frame.

6.3.1 The use of SP-frames across GOP boundaries

An SP-frame is a new picture type supported by H.264 for drift-free switching between compressed video bitstreams of different bit rates to accommodate the bandwidth variation [77-80]. The advantage of SP-frames is to allow identical reconstruction of the frames even when different reference frames are used for prediction. This property motivates us to adopt SP-frames across GOP boundaries for the linkage establishment between adjacent GOPs, as shown in Figure 6.2(b). In principle, SP-frames are encoded in pairs – a primary SP-frame and a secondary SP-frame. In Figure 6.2(b), PSP_{L-1} is a primary SP-frame at frame *L-1* and it is encoded by using frame *L-2* as the reference. On the other hand, SSP_{L-1} is its corresponding secondary-SP frame predicted from I_L and this secondary SP-frame is decoded when backward playback traverses the GOP boundary. The coding of PSP_{L-1} and SSP_{L-1} ensures that same reconstructed values of frame *L-1* can be obtained for playing in both forward and backward directions.

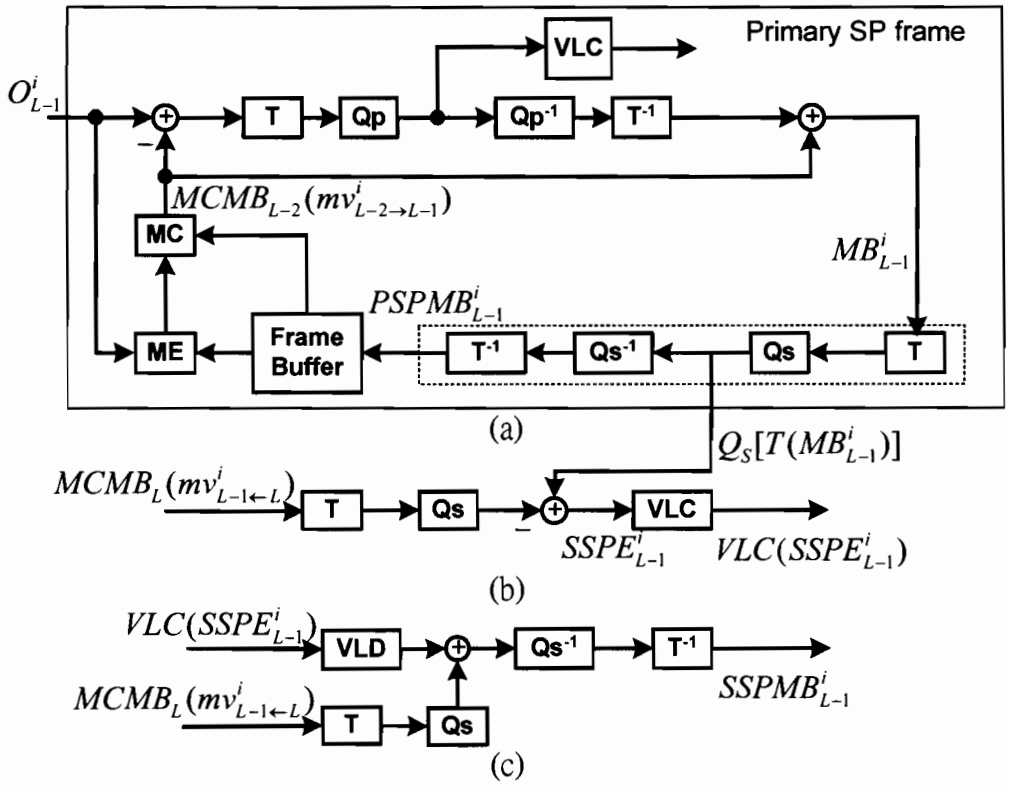


Figure 6.3. The block diagrams of (a) primary SP-frame encoding, (b) secondary SP-frame encoding, and (c) secondary SP-frame decoding.

The way to encode the i^{th} MB of PSP_{L-1}, $PSPMB^i_{L-1}$, is similar to that of a P-frame as shown in Figure 6.3(a). First, the original MB^i_{L-1}, O^i_{L-1}, is coded in the same way as a normal P-macroblock. The pixel values of MB^i_{L-1} can be represented by

$$MB^i_{L-1} = MCMB_{L-2}(mv^i_{L-2\to L-1}) + T^{-1}\{Q_P{}^{-1}\{Q_P[T(O^i_{L-1} - MCMB_{L-2}(mv^i_{L-2\to L-1}))]\}\} \quad (6.3)$$

where $MCMB_{L-2}(mv^i_{L-2\to L-1})$ represents the motion-compensated MB of MB^i_{L-1} which is translated by the motion vector $mv^i_{L-2\to L-1}$ in frame *L-2*. For primary SP-frame encoding, additional transform/quantization and dequantization/inverse transform steps with the quantization level Q_S (enclosed within the dotted line in Figure 6.3(a)) are performed on MB^i_{L-1}. These additional steps are required to avoid mismatch when different reference frames

are used for reconstructing frame *L-1*, and a more detailed treatment of the additional quantization process can be found in [77-80]. The i^{th} MB of PSP_{L-1}, $PSPMB^i_{L-1}$, is then stored in the frame buffer, and can be written as

$$PSPMB^i_{L-1} = T^{-1}\{Q_S^{-1}\{Q_S[T(MB^i_{L-1})]\}\} \tag{6.4}$$

To encode the corresponding MB in SSP_{L-1}, $SSPMB^i_{L-1}$, $Q_S[T(MB^i_{L-1})]$ is taken from the primary SP encoder as the input data of the secondary SP encoder, as depicted in Figure 6.3(b). Its motion-compensated MB from frame I_L, $MCMB_L(mv^i_{L-1\leftarrow L})$, is also transformed and quantized using Q_S before generating the prediction error with $Q_S[T(MB^i_{L-1})]$. It is noted that $mv^i_{L-1\leftarrow L}$ is the motion vector of $SSPMB^i_{L-1}$ by using I_L as the reference. The prediction error in the quantized transform domain ($SSPE^i_{L-1}$) can then be computed as

$$SSPE^i_{L-1} = Q_S[T(MB^i_{L-1})] - Q_S\{T[MCMB_L(mv^i_{L-1\leftarrow L})]\} \tag{6.5}$$

which is then entropy encoded as its binary representation $(VLC(SSPE^i_{L-1}))$ as shown in Figure 6.3(b). Both $Q_S[T(MB^i_{L-1})]$ and $Q_S\{T[MCMB_L(mv^i_{L-1\leftarrow L})]\}$ are thus synchronized to Q_S and there is no further quantization from this point. It means that $SSPE^i_{L-1}$ is also synchronized to Q_S. On traversing the GOP boundary reversely, the decoder receives $VLC(SSPE^i_{L-1})$ with $mv^i_{L-1\leftarrow L}$. Besides, I_L is already in the decoder buffer, $Q_S\{T[MCMB_L(mv^i_{L-1\leftarrow L})]\}$ is then generated in the same way as the encoder and summed up with $SSPE^i_{L-1}$, as shown in Figure 6.3(c). After dequantization and inverse transformation, $SSPMB^i_{L-1}$ can be obtained as

$$SSPMB_{L-1}^{i} = T^{-1}\{Q_S^{-1}\{Q_S\{T[MCMB_L(mv_{L-1\leftarrow L}^{i})]\} + SSPE_{L-1}^{i}\}\} \quad (6.6)$$

Substituting (6.5) into (6.6), we obtain

$$SSPMB_{L-1}^{i} = T^{-1}\{Q_S^{-1}\{Q_S[T(MB_{L-1}^{i})]\}\} \quad (6.7)$$

From (6.4) and (6.7), we can conclude that

$$SSPMB_{L-1}^{i} = PSPMB_{L-1}^{i} \quad (6.8)$$

In this way, when the backward playback traverses the GOP boundary as shown in Figure 6.2(b), after displaying frame *L*, $VLC(SSPE_{L-1}^{i})$ and $mv_{L-1\leftarrow L}^{i}$ are going to be transmitted and decoded to reconstructed pixel values of SSP_{L-1}, which is exactly equal to that of PSP_{L-1}. This means that the mismatch between the forward and backward playback at frame *L-1* can be eliminated due to the use of SP-frames at the GOP boundary.

Although this coding arrangement can achieve identical reconstruction of frame *L-1* for forward and backward playback, it will lead to quality degradation of each MB in backward playback as compared with the MB which is P-frame encoded. For example, consider the i^{th} MB of frame *L-1* as shown in Figure 6.2(b). If frame *L-1* is encoded as an SP-frame, during backward playback across the GOP boundary, $SSPMB_{L-1}^{i}$ is reconstructed as $T^{-1}\{Q_S^{-1}\{Q_S[T(MB_{L-1}^{i})]\}\}$. This signifies that $SSPMB_{L-1}^{i}$ is equal to $PSPMB_{L-1}^{i}$ instead of MB_{L-1}^{i}. In other words, the pixel values stored in the decoder buffer, $SSPMB_{L-1}^{i}$, is no longer equal to MB_{L-1}^{i}. On reconstructing frame *L-2* by adopting the sign inversion technique technique, (6.2) indicates exact MB_{L-1}^{i} is required in the decoder buffer. Otherwise, the decoder encounters the

mismatch problem between MB^{i}_{L-1} and $PSPMB^{i}_{L-1}$. This mismatch again introduces distortion of MB^{i}_{L-2} and will influence the quality of the subsequent P-frames within the same GOP for backward playback. The extent of mismatch caused by the SP frame will be illustrated later in the experimental results.

6.3.2 The strategy of allocating PSPMBs

For backward playback, the above discussion notifies that the sign inversion technique cannot be used for PSPMBs/SSPMBs due to the extra quantization. This means that the primary SP-frames and their corresponding secondary SP-frames placed before I-frames are not sufficient to achieve mismatch-free video reconstruction between forward and backward playback after traversing GOP boundaries. Instead of arranging the primary SP-frame before the I-frame, we propose a novel strategy to allocate different PSPMBs within the GOP to cope with the mismatch problem.

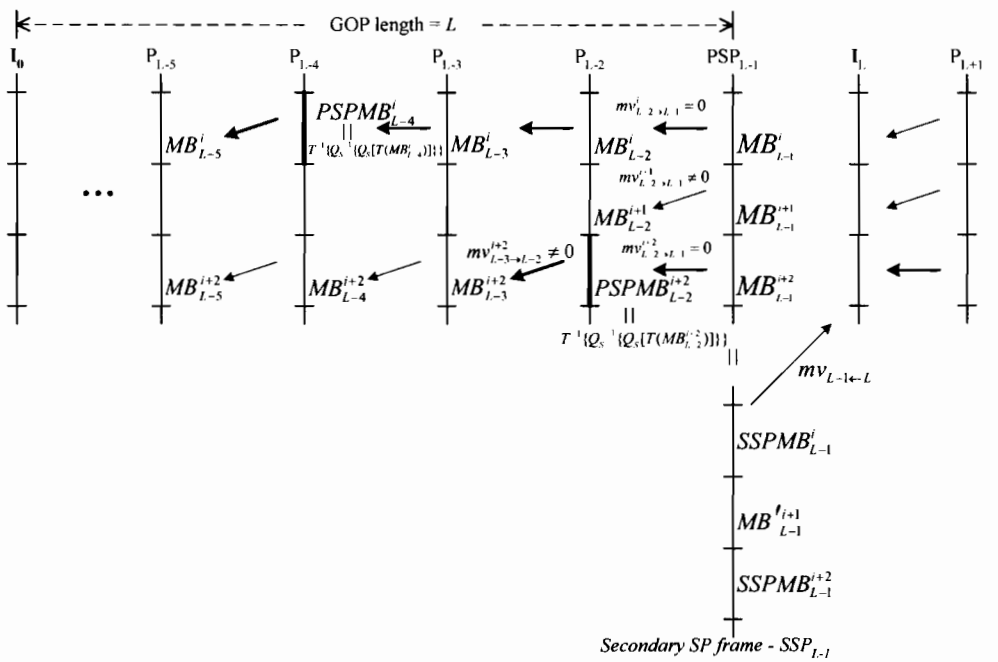


Figure 6.4. The strategy of allocating PSPMBs in the proposed scheme.

From (6.4) and (6.7), it points out that (6.2) does not hold true for PSPMBs/SSPMBs. To avoid the mismatch due to use of the sign inversion technique in PSPMBs/SSPMBs, we make a special arrangement for SP coding. In the new arrangement, only the last BMB (in the backward direction) of the BMB chain is PSPMB-encoded. For illustration, Figure 6.4 shows the BMB chain at the *i*th position, in which it consists of three BMBs consecutively - MB_{L-2}^{i}, MB_{L-3}^{i}, and MB_{L-4}^{i}. Only the last BMB, MB_{L-4}^{i}, is allowed to be PSPMB-encoded ($PSPMB_{L-4}^{i}$). Since it is at the end of the BMB chain, MB_{L-5}^{i} in this example is a FMB and no sign inversion technique is applied to MB_{L-5}^{i} by using $PSPMB_{L-4}^{i}$. In other words, $PSPMB_{L-4}^{i}$ would not be used to reconstruct MB_{L-5}^{i} and the mismatch does not happen during backward playback. Similarly, in Figure 6.4, MB_{L-2}^{i+2} is also PSPMB-encoded as $PSPMB_{L-2}^{i+2}$. On the other hand, no MB is encoded as PSPMB at the *i+1*th position since MB_{L-2}^{i+1} is already a FMB. This strategy for distributing PSPMBs in different frames within the same GOP can guarantee no mismatch to be appeared during backward playback and the PSPMB-encoded MBs will have no influence on the use of the sign inversion technique.

In the following, we again use the MB at the *i*th position as an example to provide a detailed formulation of how this strategy can provide mismatch-free solution. In this BMB chain, only MB_{L-4}^{i} is PSPMB-encoded as $PSPMB_{L-4}^{i}$ since it is the last BMB (in the backward direction). According to the encoding

structure in Figure 6.3(a), MB^i_{L-4} needs to pass through extra quantization/dequantization and $PSPMB^i_{L-4}$ can be computed as

$$PSPMB^i_{L-4} = T^{-1}\{Q_S{}^{-1}\{Q_S[T(MB^i_{L-4})]\}\} \tag{6.9}$$

It is interesting to note that its subsequent co-located MBs in the forward direction (MB^i_{L-3}, MB^i_{L-2}, and MB^i_{L-1}) are not necessary to be PSPMB-encoded. For instance, MB^i_{L-3} is encoded by using the normal P-frame encoding procedure. The prediction of MB^i_{L-3} is $PSPMB^i_{L-4}$ since the motion vector of MB^i_{L-3} is zero. MB^i_{L-3} can then be reconstructed as

$$MB^i_{L-3} = PSPMB^i_{L-4} + T^{-1}\{Q_P{}^{-1}\{Q_P[T(e^i_{L-3})]\}\} \tag{6.10}$$

where e^i_{L-3} is the prediction error between MB^i_{L-3} and $PSPMB^i_{L-4}$, and its prediction error coefficients are quantized and dequantized using the quantization level Q_P. These quantization/dequantization are the normal processes defined in the P-frame encoding. Similarly, MB^i_{L-2} and MB^i_{L-1} can be represented by

$$MB^i_{L-2} = MB^i_{L-3} + T^{-1}\{Q_P{}^{-1}\{Q_P[T(e^i_{L-2})]\}\} \tag{6.11}$$

and

$$MB^i_{L-1} = MB^i_{L-2} + T^{-1}\{Q_P{}^{-1}\{Q_P[T(e^i_{L-1})]\}\} \tag{6.12}$$

respectively. Putting (6.10) – (6.11) into (6.12), it becomes

$$MB^i_{L-1} = PSPMB^i_{L-4} + \sum_{k=1}^{3} T^{-1}\{Q_P{}^{-1}\{Q_P[T(e^i_{L-k})]\}\} \tag{6.13}$$

Performing the transformation on both sides and taking into account the linearity of transformation, (6.13) can be re-written as

$$T(MB_{L-1}^{i}) = T(PSPMB_{L-4}^{i}) + \sum_{k=1}^{3} Q_P{}^{-1}\{Q_P[T(e_{L-k}^{i})]\} \quad (6.14)$$

Referring to (6.9), we see that the transform coefficients of the first term in the right-hand side of (6.14), i.e. $T(PSPMB_{L-4}^{i})$, is divisible by Q_S. On the other hand, $\sum_{k=1}^{3} Q_P{}^{-1}\{Q_P[T(e_{L-k}^{i})]\}$ in (6.14) is only divisible by Q_P. If we set $Q_S = Q_P$ during SP-frame encoding, the terms inside the summation in (6.14) are also divisible by Q_S. Therefore, $T(MB_{L-1}^{i})$ is now divisible by Q_S if Q_S is set to Q_P. The divisibility of Q_S in $T(MB_{L-1}^{i})$ provides the fundamental for solving the mismatch problem for backward playback across GOP boundaries. It is noted that, as comparing to the SP-frame arrangement mentioned in Section 6.3.1, $T(MB_{L-1}^{i})$ in (6.3) is impossible to be divisible by Q_S since the motion-compensated MB in the transform domain $T[MCMB_{L-2}(mv_{L-2\rightarrow L-1}^{i})]$ in (6.3) is not divisible by Q_S.

Figure 6.4 also shows SSP_{L-1} which is necessary to build the linkage between two successive GOPs. To reconstruct $SSPMB_{L-1}^{i}$ during backward playback, $SSPE_{L-1}^{i}$ is required. In the proposed primary and secondary SP-frame coding arrangement, the PSPMBs and SSPMBs are no longer at the same frame. For instance, at the *i*th MB in Figure 6.4, $PSPMB_{L-4}^{i}$ is at frame *L-4* while $SSPMB_{L-1}^{i}$ is at frame *L-1*. From (6.5), $SSPE_{L-1}^{i}$ can be generated by computing the difference between $Q_S[T(MB_{L-1}^{i})]$ and $Q_S\{T[MCMB_L(mv_{L-1\leftarrow L}^{i})]\}$. Again, $SSPE_{L-1}^{i}$ is synchronized to Q_S. According to (6.7), during traversing across the GOP

boundary in reverse frame order, the decoder with I_L, $SSPE^i_{L-1}$, and $mv^i_{L-1\leftarrow L}$ can perfectly reconstruct $SSPMB^i_{L-1}$ as $T^{-1}\{Q_S^{-1}\{Q_S[T(MB^i_{L-1})]\}\}$. From (6.14), since all transform coefficients in $T(MB^i_{L-1})$ is quantized and dequantized without loss at Q_S given $Q_S = Q_P$, the term $T^{-1}\{Q_S^{-1}\{Q_S[T(MB^i_{L-1})]\}\}$ is equal to MB^i_{L-1}. Therefore, we obtain

$$SSPMB^i_{L-1} = MB^i_{L-1} \tag{6.15}$$

As discussed in Section 6.3.1, only $SSPMB^i_{L-1}$ is available in the decoder buffer after the backward play traverses the GOP boundary. But now, $SSPMB^i_{L-1}$ in the buffer is exactly the same as MB^i_{L-1}. That means the sign inversion technique can then be applied to reconstruct MB^i_{L-2} from $SSPMB^i_{L-1}$ without introducing any mismatch. During backward playback, the sign inversion technique continues to be used until the last BMB of this chain. To reconstruct MB^i_{L-5}, the sign inversion technique is no longer applied since it is a FMB. Therefore, the mismatch does not happen although $PSPMB^i_{L-4}$ is not equal to MB^i_{L-4}, as formulated in (6.14).

By encoding the last BMB of the BMB chain to be PSPMB-encoded, ($PSPMB^i_{L-4}$ and $PSPMB^i_{L-2}$ in Figure 6.4), the linkage across the GOP boundary between frame *L* and frame *L-1* is established without mismatch. As shown in Figure 6.4, when backward playback passes across the GOP boundary, $SSPMB^i_{L-1}$, MB^{i+1}_{L-1}, and $SSPMB^{i+2}_{L-1}$ are transmitted and decoded as frame *L-1*. For $SSPMB^i_{L-1}$ and $SSPMB^{i+2}_{L-1}$, their decoded pixel values are the same as compared with MB^i_{L-1} and MB^{i+2}_{L-1} for forward play, as formulated in (6.15), which ensures the sign

inversion technique can be applied on these MBs without any mismatch to further decode frame *L-2* during backward play. In contrast, MB'^{i+1}_{L-1} is encoded as a normal P-macroblock. The decoded pixel values of MB'^{i+1}_{L-1} are thus not equal to MB^{i+1}_{L-1}. This will not introduce mismatch for backward playback since there is no BMB at this position, and the sign inversion technique will not be used for MB^{i+1}_{L-2}. Therefore, the mismatch between MB^{i+1}_{L-1} and MB'^{i+1}_{L-1} has no influence on backward playback. Although MB^{i+1}_{L-1} and MB'^{i+1}_{L-1} can be encoded as $PSPMB^{i+1}_{L-1}$ and $SSPMB^{i+1}_{L-1}$, respectively, to avoid mismatch, it is not necessary for the sake of better performance for forward playback. It is due to the fact that the addition quantization step with Q_S introduces quality degradation of MB^{i+1}_{L-1} during forward playback.

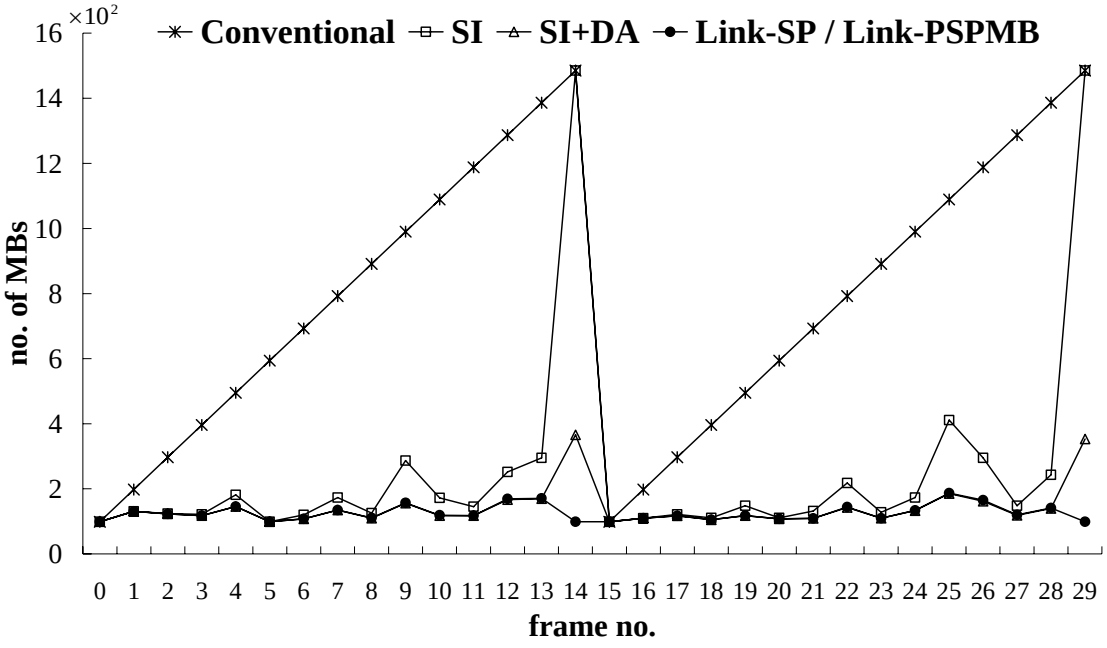
6.4 Experimental results

The section presents experimental results and evaluates the performances of various methods for linkage establishment across GOP boundaries. JVT JM 10.1 encoder [53] was employed to encode same set of the video sequences in Table 4.5 of Chapter 4. Consistently, for all testing sequences, the frame-rate of the video stream was *30* frames/s and the GOP size was 15.

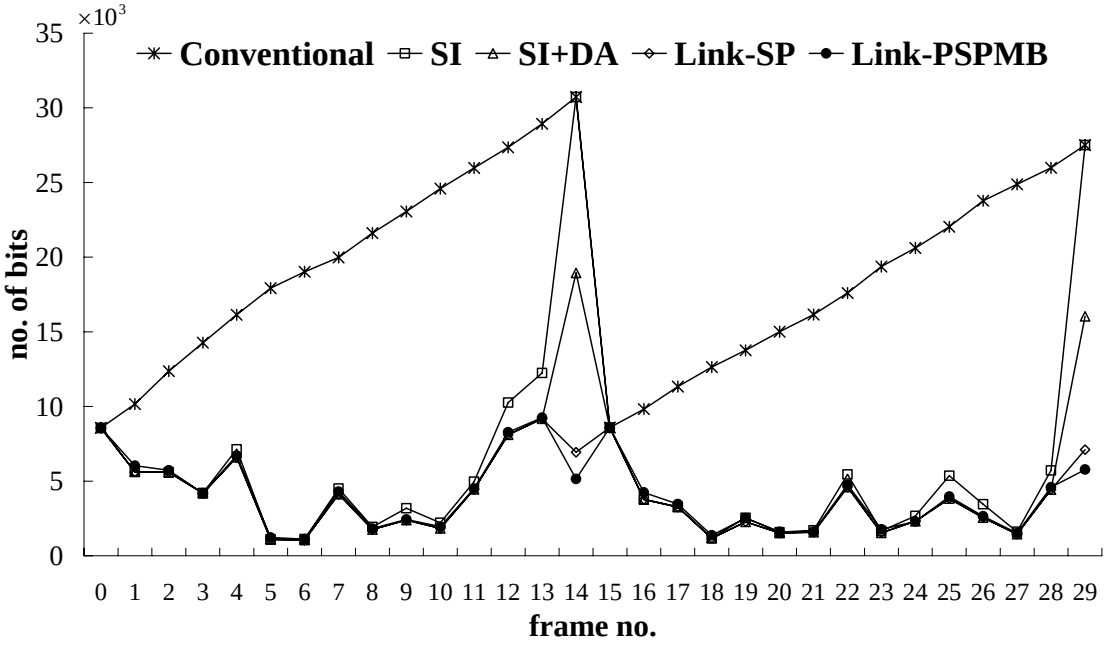
Our expectation is to give a fair and comprehensive comparison among the algorithms. Both schemes with and without establishing linkages across GOP boundaries were tested for comparison on various video sequences. They include the conventional scheme, the SI scheme [68] and SI+DA scheme [69]

mentioned in Chapter 4, and the schemes with linkage establishment across GOP boundaries. Two varieties for building linkages across GOP boundaries have been applied to the SI+DA scheme. Let us call them Link-SP and Link-PSPMB. Link-SP uses SP-frames for building linkages across GOP boundaries. For Link-PSPMB, the proposed PSPMB allocation strategy suggested in Section 6.3.2 is adopted to avoid the mismatch problem. We have simulated the same situation that the start point of the backward-play operation is at the end of the sequence. The comparison of the average number of MBs to be decoded in the last frames of GOPs (MB_{last}) is given in Table 6.1. The average number of MBs sent for decoding is directly proportional to the decoder complexity. In Table 6.1, we show that SI cannot help too much because there is no inter-frame prediction between the last frame of one GOP and the first frame of its succeeding GOP. The little saving is due to the fact that, in the MB-based scheme, those MBs that are not used as the reference for others are not necessary to be sent and decoded. Therefore, the sign inversion technique is not so useful in GOP boundaries. In Table 6.1, SI+DA can reduce MB_{last} to a certain extent by combining several MBs into one prior to transmission [69]. However, the reduction is only significant for sequences with low motion activities such as the “Claire” sequence. It is also clear from Table 6.1 that, by using the Link-SP and Link-PSPMB, MB_{last} can be remarkably reduced by 93.3% (330 MBs and 99 MBs are decoded for SIF and QCIF sequences, respectively) as compared to the conventional scheme. This result is expected since only MBs in secondary SP-frames are transmitted and decoded. Table 6.2 then shows the average number of bits to be sent in the last frames of GOPs (Bit_{last}) for different schemes, and it indicates the peak bitrates due to the

use of SI. In Table 6.2, Link-SP can reduce Bit_{last} significantly, about 73%-86%, as compared with that of the conventional scheme. Link-PSPMB produces further reduction on Bit_{last} since some MBs in secondary SP-frames are encoded as normal P-macroblocks, as shown in Figure 6.4, which provides better coding efficiency as compared with MBs encoded as secondary SP-macroblocks [77-80]. Figure 6.5 shows the frame-by-frame comparisons of the required number of MBs decoded by the decoder and bits transmitted over the networks of both schemes. Note that Link-SP and Link-PSPMB require the same number of MBs to be decoded, for simplicity, we use the same curve to show their performances. It is obvious that the peaks caused by the last frame of GOP in SI and SI+DA can be smoothed away by using the proposed Link-SP and Link-PSPMB. Building the linkages across GOP boundaries is thus efficient in reducing burstiness of the network traffic and decoder complexity during backward playback across GOP boundaries. The drastic reduction on MB_{last} and Bit_{last} are also beneficial to the average number of MBs to be decoded (MB_{all}) and bits to be sent (Bit_{all}) for the whole sequence. Table 6.3 and Table 6.4 show the Link-SP and Link-PSPMB can provide further savings for MB_{all} and Bit_{all}.



(a)



(b)

Figure 6.5. Performance of the conventional, SI, SI+DA, Link-SP, and Link-PSPMB schemes for the "Claire" sequence in the backward playback. (a) Number of MBs to be decoded by the decoder, and (b) number of bits to be sent over the network.

Table 6.1 Detailed comparisons of the conventional, SI, SI+DA, Link-SP, and Link-PSPMB schemes in terms of the average number of MBs to be decoded for the last frame of GOPs (MB_{last}). The numbers in brackets represent the saving of MB_{last} for various schemes as compared with the conventional scheme.

	Conventional	SI	SI+DA	Link-SP	Link-PSPMB
Salesman	5940	5934 (-0.10%)	1843 (-68.9%)	396 (-93.3%)	396 (-93.3%)
Football	4950	4928 (-0.44%)	2980 (-39.8%)	330 (-93.3%)	330 (-93.3%)
Tabletennis	4950	4911 (-0.77%)	3576 (-27.8%)	330 (-93.3%)	330 (-93.3%)
Foreman	1485	1485 (0.00%)	1286 (-13.4%)	99 (-93.3%)	99 (-93.3%)
Carphone	1485	1484.5 (-0.03%)	1123 (-24.4%)	99 (-93.3%)	99 (-93.3%)
Claire	1485	1484.8 (-0.01%)	379 (-74.5%)	99 (-93.3%)	99 (-93.3%)
Grandma	1485	1485 (0.00%)	347 (-76.6%)	99 (-93.3%)	99 (-93.3%)

Table 6.2. Detailed comparisons of the conventional, SI, SI+DA, Link-SP and Link-PSPMB schemes in terms of the average number of bits to be sent for the last frame of GOPs (Bit_{last}). The numbers in brackets represent the saving of Bit_{last} for various schemes as compared with the conventional scheme.

	Conventional	SI	SI+DA	Link-SP	Link-PSPMB
Salesman	194203	194145 (-0.03%)	155030 (-20.2%)	51833 (-73.31%)	39153 (-79.8%)
Football	751053	749419 (-0.22%)	661448 (-11.9%)	103086 (-86.27%)	69077 (-90.8%)
Tabletennis	607667	603765 (-0.64%)	558047 (-8.2%)	89395 (-85.29%)	36192 (-94.0%)
Foreman	110703	110703 (0.00%)	104008 (-6.1%)	21389 (-80.6%)	10080 (-90.9%)
Carphone	84450	84422 (-0.03%)	75492 (-10.6%)	15695 (-81.42%)	7498 (-91.1%)
Claire	30528	30519 (-0.03%)	19185 (-37.1%)	7224 (-76.3%)	5265 (-82.7%)
Grandma	40020	40020 (0.00%)	28103 (-29.8%)	10557 (-73.6%)	8746 (-78.1%)

Table 6.3. Detailed comparisons of the conventional, SI, SI+DA, Link-SP, and Link-PSPMB in terms of the average number of MBs to be decoded for the whole GOPs (MB_{all}). The numbers in brackets represent the saving of MB_{all} for various schemes as compared with the conventional scheme.

	Conventional	SI	SI+DA	Link-SP	Link-PSPMB
Salesman	3060	1278 (-58.23%)	872 (-71.5%)	790 (-74.2%)	790 (-74.2%)
Football	2550	1831 (-28.2%)	1535 (-39.8%)	1379 (-45.9%)	1379 (-45.9%)
Tabletennis	2550	2189 (-14.2%)	2008 (-21.3%)	1780 (-30.2%)	1780 (-30.2%)
Foreman	765	713 (-6.8%)	658 (-13.9%)	586 (-23.3%)	586 (-23.3%)
Carphone	765	671 (-12.3%)	549 (-28.3%)	487 (-36.3%)	487 (-36.3%)
Claire	765	276 (-63.9%)	160 (-79.1%)	144 (-81.2%)	144 (-81.2%)
Grandma	765	248 (-67.4%)	151 (-80.2%)	137 (-82.0%)	137 (-82.0%)

Table 6.4. Detailed comparisons of the conventional, SI, SI+DA, Link-SP, and Link-PSPMB schemes in terms of the average number of bits to be sent for the whole GOPs (Bit_{all}). The numbers in brackets represent the saving of Bit_{all} for various schemes as compared with the conventional scheme.

	Conventional	SI	SI+DA	Link-SP	Link-PSPMB
Salesman	125912	63359 (-49.7%)	58083 (-53.9%)	51829 (-58.8%)	52484 (-58.32%)
Football	413459	347024 (-16.1%)	329539 (-20.3%)	295699 (-28.5%)	294714 (-28.7%)
Tabletennis	304636	265317 (-12.9%)	255244 (-16.2%)	226841 (-25.54%)	223951 (-26.5%)
Foreman	62327	57985 (-6.97%)	56368 (-9.6%)	51361 (-17.59%)	50807 (-18.5%)
Carphone	48163	43336 (-10.0%)	40107 (-16.7%)	36483 (-24.25%)	36111 (-25.0%)
Claire	19156	7640 (-60.1%)	6323 (-66.9%)	5598 (-70.78%)	5594 (-70.8%)
Grandma	26150	9680 (-62.9%)	8369 (-67.9%)	7305 (-72.1%)	7435 (-71.6%)

Table 6.5. Mismatch (in terms of ΔPSNR) between forward and backward playback for different schemes.

Sequences	SI/SI+DA (dB)	Link-SP (dB)	Link- PSPMB (dB)
Salesman	0.018	0.667	0.017
Football	0.025	0.260	0.031
Tabletennis	0.019	0.059	0.021
Foreman	0.005	0.090	0.004
Carphone	0.008	0.109	0.007
Claire	0.059	0.816	0.048
Grandma	0.019	0.827	0.020

Table 6.6. Rate-distortion performance for Link-PSPMB during forward playback.

Sequences	ΔPSNR (dB)	ΔBitrate (%)
Salesman	-0.29	+13.5%
Football	-0.065	+1.33
Tabletennis	-0.038	+1.34
Foreman	-0.040	+2.2
Carphone	-0.048	+1.99
Claire	-0.38	+6.28
Grandma	-0.44	+15.8%

We have also demonstrated mismatch between forward and backward playback for different schemes. For the conventional scheme, the backward playback retains the same reconstruction quality as that of the forward playback. It is interesting to note that, theoretically, both SI and SI+DA will not introduce any PSNR degradation during backward playback. However, such techniques will cause quality degradation due to the clipping operation of the H.264 video algorithm, as discussed in Chapter 4. The mismatch becomes significant for Link-SP as shown in Figure 6.6. This figure shows that except the frames before I-frame, all frames cause serious quality degradation for backward playback. It is not unexpected since $SSPMB_{L-1}^{i}$ is equal to $PSPMB_{L-1}^{i}$ instead of MB_{L-1}^{i}, as illustrated in Figure 6.2(b). In this case, the sign inversion technique introduce mismatch for reconstructing frame *L-2*. In Figure 6.6, we show that

the mismatch problem can be alleviated by using Link-PSPMB because $SSPMB_{L-1}^{i}$ is now equal to MB_{L-1}^{i} when the PSPMB allocation strategy is adopted. On the other hand, the mismatch in Link-PSPMB is also only come from the clipping operation of adopting the sign inversion technique, and is almost the same as SI and SI+DA. Table 6.5 also gives the mismatch between forward and backward playback for various sequences. It shows that the proposed Link-PSPMB is comparable to SI and SI+DA. Therefore, the proposed Link-PSPMB can avoid the quality degradation due to the adoption of PSPMB and SSPMB while smoothing out the burstiness of decoder complexity and network traffic.

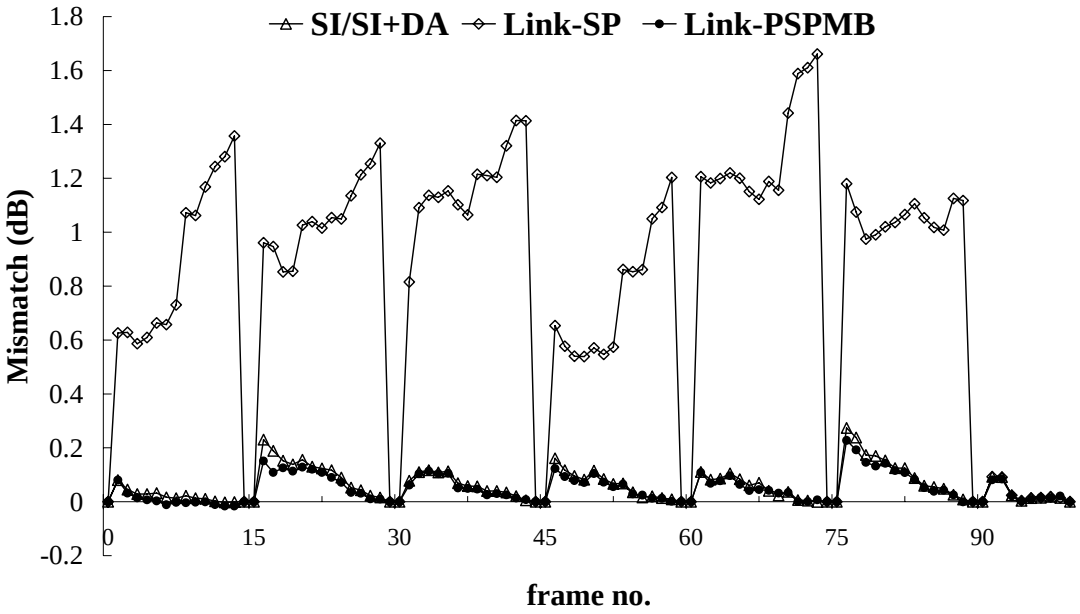


Figure 6.6. Mismatch between forward and backward playback in terms of ΔPSNR for the “Claire” sequence.

However, in Figure 6.4, the additional quantization step in SP-frame encoding has little influence on the rate-distortion performance of PSPMBs, and these PSPMBs further affects their related MBs in future frames. Consequently, it degrades the reconstruction quality of forward playback. This is illustrated in

Table 6.6 in which ΔPSNR and Δbitrate represent a PSNR change and a bitrate change in percentage, respectively, of the video bitstream encoded by using PSPMBs when compared to the bitstream without PSPMBs. The positive values mean increments whereas negative values mean decrements. From the results in Table 6.6, it is observed that the coding efficiency of Link-PSPMB slightly decreases, but it is negligible in most sequences. But, for the "Claire", "Salesman", and "Grandma" sequences, the loss in rate-distortion performance increases. It is due to the fact that for a sequence that contains low motion activities, the BMB chain in Figure 6.4 becomes longer, and more MBs in the subsequent frames are affected. However, it can be noticed that due to the characteristic of these sequences, the scheme SI+DA has achieved a relatively large saving at the GOP boundaries, as shown in Table 6.3 and Table 6.4. As a result, the requirement of establishing the linkages across GOP boundaries for backward playback on these sequences is not as desired as other sequences.

6.5 Chapter Summary

In this chapter, we have addressed issues on a substantial improvement of the MB-based backward-play scheme for compressed video bitstreams. We have shown that when backward playback traverses GOP boundaries, the required number of MBs to be decoded and the required number of bits to be sent increase significantly, which hinder its use in video streaming applications. The reason behind is that the inter-frame dependency between the last frame of one GOP and the first frame of the successive GOP disappears. At that moment, the sign inversion technique, which makes use of the motion information

between two frames during backward playback, for the MB-based scheme cannot be applied. In order to cope with the GOP discontinuity of the video bitstream, SP-frames have been proposed to establish the missing linkage to a previous GOP thereby ensuring continuity of backward playback across GOP boundaries. With the linkage establishment using SP-frames at GOP boundaries, the peak bitrate and decoding complexity caused by the last frames of GOPs can be alleviated. However, a direct implementation of using the SP coding concept arouses serious mismatch between forward and backward playback. Through a novel arrangement of the PSPMBs inside the GOP, a new allocation strategy has been proposed to encode the last BMB of the BMB chain as PSPMB. This arrangement is able to ensure that the backward playback retains nearly the same reconstruction quality as that of the forward playback, and the only difference comes from the mismatch due to the use of the sign inversion technique in the MB-based backward-play scheme. Experimental results confirm that the proposed algorithm can smooth out the burstiness of decoder complexity and network traffic while avoiding the mismatch due to the adoption of SP-frame coding for backward playback. This significantly strengthens the proposed MB-based backward-play scheme.

Chapter 7 Conclusion & Suggestions for Future Research

7.1 Conclusion to the present work

In this thesis, we have investigated several compressed-domain algorithms for the backward-play operation in the MPEG/H.264 video streaming system with VCR support. These compressed-domain algorithms have been shown to be conducive to the implementation of VCR functions in the compressed video bitstream, with minimum requirements on the server/decoder complexity and the network bandwidth.

The solution for VCR functions is simple for analog video on tapes, since the data for each video frame is self-contained and independent of each other. However, when the video is coded with any of the well-known video coding standards such as MPEG-1, -2, -4, and H.264, temporal dependencies among frames are produced along with their competent compression effects. The conventional MPEG video streaming system with VCR support was presented in Chapter 2, and detailed formulations in terms of the average number of frames to be sent for various VCR operations have been made. These formulations showed that the decoder complexity and network bandwidth involved in the VCR functions are significantly heavier than those in normal playback. These create obstacles for interactive browsing operations on digital

video, which are desirable features in video-on-demand and high-definition TV. Although some techniques in the literature have been proposed to solve these problems, results of our study indicated that these techniques are still primitive and introduce high processing complexity of the server/client and quality degradation of the browsing video. A further need may arise for mitigating the required server/client complexity and network bandwidth with high-quality guarantee for VCR functions in the digital video system.

Therefore, in this research, two different approaches were employed to provide efficient solutions for backward playback. The approach in Chapter 3 used a transcoding technique in which a fast motion estimation and mode decision algorithm for reverse transcoding was suggested. The proposed fast algorithm reduced the inherent complexity of the transcoder. In Chapters 4 to 6, we designed various compressed-domain approaches to efficiently manipulate video data for backward playback. These approaches give a new direction for the video streaming system with VCR support at MB level. In previous studies, the investigation of video browsing has focused on the frame-level arrangement. In fact, different MBs in the requested frame have different properties in the compressed domain. A MB-based scheme has thus been proven in this thesis to provide good performance for backward playback.

The aim of reverse transcoding is to produce a reverse-encoded bitstream that, when decoded, displays the video frames in reverse order from the original bitstream. This reverse transcoding process is difficult since reversing the order of frames in the original bitstream does not result in a reversed motion vectors

and modes in the H.264 standard. In Chapter 3, we found that much of the motion vector and mode information contained in the original H.264 video bitstream can be used to save a significant number of computations. We thus proposed a fast motion estimation algorithm with mode decision in H.264 reverse transcoding by utilizing both of the mode and motion vector information from the original H.264 video bitstream. This provides a better estimation of the reverse motion vectors and modes. Motion activities of MBs are considered to measure the correlation between neighboring frames. According to the motion activities, the relationship between the mode and spatial information, and the temporal smoothness of the motion trajectory, the best motion vector and mode predictors are selected to speed up the reverse transcoding process of H.264 video bitstream. Experimental results confirmed that the proposed algorithm achieved better rate-distortion performance than the well-known “in-place” algorithm, and maintained low computational complexity in the reverse transcoding process.

However, the decoding and re-encoding of residuals in reverse transcoding still needs a great deal of computation, even though our fast motion estimation and mode decision algorithm can speed up the transcoding process. Quality degradation of browsing video incurred by the re-encoding process is unavoidable. Therefore, the discussion in Chapter 4 proposed a novel MB-based solution in the server for the realization of backward playback. In our study, it found that different MBs in one particular frame have different properties in the compressed domain for backward playback. Therefore, by using the motion information, a MB-selection scheme in the server was

designed to classify MBs into two types: BMBs and FMBs. Following the classification of MBs with our proposed server, BMBs are manipulated by the sign inversion (SI) technique to reverse the motion compensation process from using the previously decoded MBs as the reference in order to obtain the requested MB. The derivation discussed in Chapter 4 has also shown that only a limited amount of data needs to be transmitted to the client. In addition, this sign inversion technique performs all computations in the VLC domain, and it achieves noticeable savings in decoder complexity and network traffic. For FMBs, we also designed a direct addition (DA) technique, which operates in the DCT domain, to further alleviate the decoder complexity while maintaining low network bandwidth requirement. Experimental results showed that, by using the SI and DA techniques, the required network bandwidth and decoder complexity during backward playback can be reduced remarkably. At the server side, the additional computation requirement is very limited since the SI and DA techniques are operated in the VLC and DCT domains respectively. Moreover, no obvious quality degradation for backward playback is observed since no re-encoding is involved in the server.

In Chapter 5, in order to maximize the benefits of the sign inversion technique as much as possible, a mixed VLC/DCT technique was contrived for FMBs to further reduce the decoder complexity and the network bandwidth. To do so, FMBs (renamed as FBMBs in Chapter 5) can be divided into two partitions - backward region and forward region. We found that the sign inversion technique can be employed in the backward regions of FBMBs again. On the other hand, the remaining forward regions can still be manipulated using the DA

technique. By using this arrangement, it is possible to maximize the handling of FBMBs in the VLC domain. Experimental results showed that further savings of up to 10% in terms of the number of MBs to be decoded and bits to be sent over the network in the backward-playback can be achieved by using the new VLC/DCT-domain technique.

The solutions suggested in Chapter 4 and Chapter 5 could successfully alleviate decoder complexity the network traffic during backward playback. Nevertheless, the required decoder complexity and network bandwidth surge when backward playback traverses across GOP boundaries since there is no inter-frame prediction between the last frame of one GOP and the first frame of the successive GOP. In this situation, the sign inversion technique cannot be used because it relies on the inter-frame motion dependency. In Chapter 6, we considered establishing the missing linkages at GOP boundaries with the help of SP-frames. This ensures the continuity of backward playback across the GOP boundaries, and the peak bitrate and decoding complexity incurred by the GOP boundaries is then reduced significantly. However, the straightforward implementation of using SP-frames results in a serious mismatch between forward and backward playback. Therefore, in Chapter 6, a novel arrangement of PSPMBs inside the GOP was made through the new allocation strategy. This strategy only encodes the last BMB of the BMB chain as PSPMB. The derivation in Chapter 6 has proven that the backward playback retains almost the same reconstruction quality as that of the forward playback. Experimental results verified that the burstiness of the decoder complexity and network traffic at GOP boundaries could be removed by using the proposed technique without

introducing mismatch between forward and backward playback. This further strengthens our proposed MB-based backward-play scheme.

In conclusion, we expect the amount of video content available to grow in line with the widespread adoption of Internet video streaming and the rapid development of playback devices. The demand for interactive browsing of video is likely to increase enormously in the near future. In our present work, a number of techniques have been investigated that can enhance the capability of video browsing in various aspects. We believe that the results obtained in this work contribute significantly to the efficient realization of modern video streaming system with VCR support.

7.2 Future Directions

In this thesis, we have proposed several techniques to resolve the problems of backward playback in the digital video streaming system. Video compression is an active research topic. Different video compression techniques are designed to encode the digital video in compact form with fewer bits. The design philosophy of current video coding standards sets out to ensure that only minimum processing resources are needed if a compressed video is decoded for normal playback in a pre-determined order. For freely browsing digital video using VCR functions, the challenge is how to deal with the correlation exploited in these compression standards appropriately, and decode the video sequence in any order efficiently. Based on the successful techniques described in this

thesis and proven by a wide range of experimental work, we propose here some directions for future research.

7.2.1 MB-based techniques in the latest standard H.264

Excepting the reverse transcoding methods in Chapter 3, the MB-based techniques in Chapter 4 and 5 were implemented based on the MPEG-2 system at the initial stage of our study. Generally, all of these techniques can be easily extended to the latest video coding standard - H.264. The emerging H.264 standard achieves much higher coding efficiency than the previous video coding standards. The new standard supports a collection of new encoding techniques including integer transform, SP-frames, variable block size, multiple reference frames for motion prediction, etc. These new features supported in H.264 have both desirable and less desirable effects on VCR operations. In Chapter 6, we tried to extend the MB-based schemes to the H.264 standard by using the SP-frame coding concept to solve the motion discontinuity at the GOP boundaries. In the experiments, the H.264 codec (JVT JM 10.1 [53]) was employed, and new features such as integer transform, SP-frame option, and adaptive context-based variable length coding were activated. The results proved that the MB-based scheme could work well in H.264 bitstreams. However, many problems still remain to be investigated in H.264. For example, future work could focus on whether MBs or smaller partitions such as 4x4 blocks are appropriate for our proposed SI and DA techniques. Besides, in H.264, the use of motion estimation and compensation with multiple reference frames severely complicates VCR operations. This feature generates a very

complicated frame relationship in the video bitstream where more than one frame is used as the reference for the current frame. Generally, our proposed scheme still works when multiple reference frames are used, but some modifications on definitions may be necessary. For example, when the reference of $MB_n^{(k,l)}$ is not referred to the previous frame n-1 but frame n-2, the corresponding MB in frame n-1, $MB_{n-1}^{(k,l)}$,will be defined as FBMB no matter the motion vector of $MB_n^{(k,l)}$ is zero or not. This reduces the percentage of BMBs and causes some impacts on our achieved improvements. Fortunately, according to the statistics and analysis of works on multiple reference frames, a large percentage (about 80%) of macroblocks still refer to their previous nearest frame for most sequences. As a result, our results will not be affected too much. Besides, when multiple reference frames are enabled, more than one reference frame buffers are employed in the decoder. Manipulation of these frame buffers may facilitate the VCR functions implementation. This provides a great challenge for video browsing in the H.264 bitstream. It is a crucial issue that should be investigated before they can be put into practical use for browsing the video bitstream compressed by the newest H.264 standard.

7.2.2 Extension of the proposed techniques in other VCR functions

It is noted that not all of our proposed techniques in Chapters 4 and 5 are restricted to backward playback, but can also be beneficial to other VCR operations, especially when the GOP size is large. For instance, if the next requested frame in random-access or fast backward is in the same GOP as the

displayed frame shown in Figure 7.1, the SI and DA techniques may be applied in both directions in order to save network traffic and decoder complexity. This further shows that the results of our work will be useful for the future development of digital VCR.

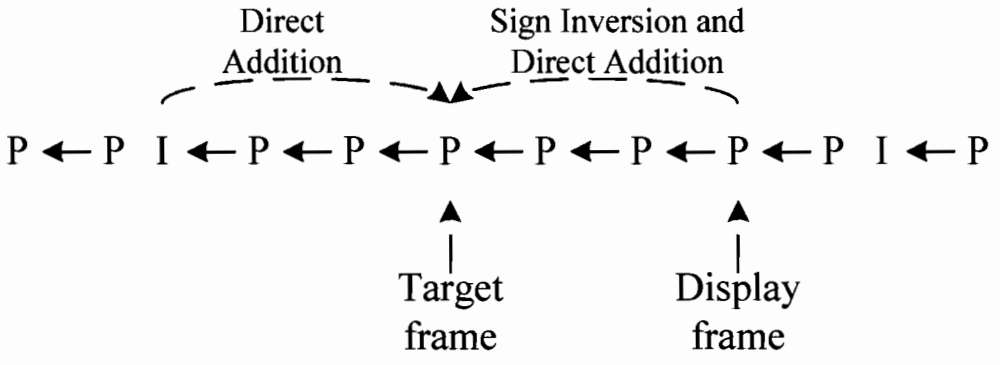


Figure 7.1. Possible directions for random access or fast-backward playback.

On the other hand, if the next requested frame in random-access or fast-forward/backward is outside the same GOP, it is possible that the server first finds an I-frame which has a shorter distance to the requested frame and then jumps to the requested P-frame. This is quite similar to the case of development of the frame-skipping techniques [73-75]. Several pixel-domain frame-skipping techniques [81-82] for bitrate reduction of compressed video have been devised in recent years. In [83], a frame control scheme was proposed to dynamically adjust the number of skipped frames. We believe that techniques being used in frame-skipping are also useful for fast-forward/backward and random-access operations in digital VCR. Future research could explore structures similar to the frame-skipping techniques for VCR functionality in the compressed domain.

7.2.3 Issues on the use of SP-frames in other VCR functions

In Chapter 6, SP-frames were shown to solve the motion discontinuity across GOP boundaries in backward playback. We believe that it would also be useful in other VCR functions. For example, if the frame at the middle of GOP is encoded as a pair of SP-frames: primary SP (PSP) and secondary SP (SSP) as depicted in Figure 7.2, the reconstruction value of these two frames are exactly the same. The pair of SP-frames provides one more access point inside the GOP in the random access process without mismatch. Although the size of SSP-frames is relatively large, they are only utilized in VCR operations such as random access or fast forward rather than forward playback. In the normal playback, only PSP is used.

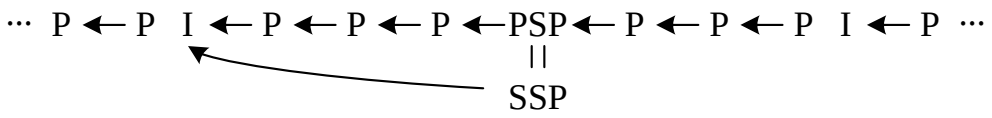


Figure 7.2. Using SP-frames for other VCR functions.

With this powerful “no mismatch” feature, SP-frames have great potential in providing VCR functions. However, the coding process of the SP-frame has not yet been optimized, and improving the coding efficiency of the SP-frame is another important topic in this field. One possible improvement may come from the different characteristics of quantized transform coefficients in SP-frames. The current entropy coding algorithms in H.264 adopts context adaptive variable length coding or context adaptive arithmetic coding for quantized transform coefficients. Both are designed for the normal distribution of coefficients in I-/P-frames. For I-/P-frames, in each quantized transform

coefficients block, non-zero coefficients concentrates on the low-frequency part and consecutive zeros in the high-frequency part. In the SSP-frames, the distribution is different. Since the subtraction of the current frame from the reference frame is performed after the transformation and quantization, more non-zero coefficients exist in high frequency region compared to I-/P-frames as shown in Figure 7.3. An entropy coding algorithm is specifically designed for SP-frames. This would provide better coding efficiency for SP-frames and make them more practical in implementing VCR functions.

18	9	1	0
5	0	0	0
2	0	0	0
0	0	0	0

(a)

18	9	1	0
5	0	-2	0
2	1	1	0
2	0	-1	0

(b)

Figure 7.3. Comparison of non-zero coefficients distribution between (a) P-frame and (b) SP-frame.

References

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

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

[3] SonicBlue Inc. [Online]. Available: http://www.replay.com/

[4] TiVo Inc. [Online]. Available: http://www.tivo.com/

[5] 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

[6] ISO/IEC 13818-2, “Information Technology -- Generic Coding of Moving Pictures and Associated Audio Information: Video,” 1996.

[7] Video Codec for Audiovisual Services at p×64 kbit/s, ITU-T Recommendation H.261., 1993

[8] Video Coding for Low Bitrate Communication, ITU-T Recommendation H.263, May 1997.

[9] ISO/IEC 14496-10 and ITU-T Rec. H.264, Advanced Video Coding, 2003.

[10] ISO/IEC 14496-2, “Information Technology – Coding of audio-visual Objects – Part 2: Video,” 2004.

[11] H.J. Stuttgen, “Network evolution and multimedia communication,” IEEE Multimedia, vol. 2, pp. 42-59, Fall 1995.

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

[13] L. Press, "The Internet and interactive television," Comm. ACM, vol. 36, no. 12, pp. 19-23, Dec. 1993.

[14] D. Wu, Y. T. Thomas, and Y.-Q. Zhang, "Transporting real-time Video over the internet: challenges and Approaches," Proceedings of the IEEE, vol. 88, no. 12, December 2000.

[15] L. Chiariglione, "The development of an integrated audiovisual coding standard: MPEG," Proceedings of IEEE, vol. 83, pp. 151-157, Feb. 1995.

[16] ITU-T/SG15, "Video codec test model, TMN8," June 97.

[17] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," IEEE Transactions on Communications, vol. 33, no. 9, pp. 1011-1015, September 1985.

[18] Y. L. Chan and W. C. Siu, "New adaptive pixel decimation for block motion vector estimation," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 1, pp.113 -118, February 1996, U.S.A.

[19] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Second Edition, Prentice Hall, 2001.

[20] ITU-R Recommendation BT.601, Encoding parameters of digital television for studios, 1982.

[21] K. R. Rao and J.J. Hwang "Techniques and Standards for Image, Video, and Audio Coding" Prentice Hall, 1996.

[22] ISO/IEC 10918-1, "Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines", 1994.

[23] Y. M. Lei and M. Ouhyoung, "Software-based motion JPEG with progressive refinement for computer animation," IEEE Trans. Consumer Electronics, vol. 40, pp. 557-562, Aug. 1994.

[24] N. Ahmed, T. Natrajan, and K. R. Rao, “Discrete cosine transform”, IEEE trans. Computers,. January 1974.

[25] F. Arguello and E. L. Zapata, “Fast cosine transform based on successive doubling methods,” Electronic Letters, vol. 26, pp. 1616-1618, Sept. 1990.

[26] N. I. Cho and S. U. Lee, “Fast algorithm and implementation of 3D discrete cosine transform,” IEEE Trans. Circuits and Systems, vol. 38, pp.297-305, Mar. 1991.

[27] T. Eude et al., “On the distribution of the DCT coefficients,” ICASSP’94, vol. 5, pp. 365-368, Adelaide, Australia, Apr. 1994.

[28] A. W. Johnson, M. H. Chan, and J. Princess, “ Frequency scalable video coding using the MDCT,” ICACCP’94, vol. 5, pp. 477-480, Adelaide, Australia, Apr. 1994.

[29] J. H. Kim and G. M. Park, “Two layered DCT based coding scheme for a new digital HD-VCR,” ICCE’94, pp. 24-25, Chicago, IL, June 1994.

[30] R. J. Chen and B. C. Chieu, “A full adaptive DCT based color image sequence coder,” Signal Processing: Image Communication, vol. 6, pp.289-301, Aug. 1994.

[31] P. Pirsch, “Design of DPCM quantizers for video signals using subjective tests,” IEEE Trans. Commum. Vol. COM-29, pp. 990-1000, July 1981.

[32] Heng-Yao Lin, Yi-Chih Chao, Che-Hong Chen, Bin-Da Liu, and Jar-Ferr Yang, “Combined 2-D Transform and Quantization Architectures for H.264 Video Coders,” International Symposium on Circuits and Systems, Vol. 2, pp. 1802-1805, May 2005.

[13] L. Press, “The Internet and interactive television,” Comm. ACM, vol. 36, no. 12, pp. 19-23, Dec. 1993.

[14] D. Wu, Y. T. Thomas, and Y.-Q. Zhang, “Transporting real-time Video over the internet: challenges and Approaches,” Proceedings of the IEEE, vol. 88, no. 12, December 2000.

[15] L. Chiariglione, “The development of an integrated audiovisual coding standard: MPEG,” Proceedings of IEEE, vol. 83, pp. 151-157, Feb. 1995.

[16] ITU-T/SG15, “Video codec test model, TMN8,” June 97.

[17] R. Srinivasan and K. R. Rao, “Predictive coding based on efficient motion estimation,” IEEE Transactions on Communications, vol. 33, no. 9, pp. 1011-1015, September 1985.

[18] Y. L. Chan and W. C. Siu, “New adaptive pixel decimation for block motion vector estimation,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 1, pp.113 -118, February 1996, U.S.A.

[19] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Second Edition, Prentice Hall, 2001.

[20] ITU-R Recommendation BT.601, Encoding parameters of digital television for studios, 1982.

[21] K. R. Rao and J.J. Hwang “Techniques and Standards for Image, Video, and Audio Coding” Prentice Hall, 1996.

[22] ISO/IEC 10918-1, “Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines”, 1994.

[23] Y. M. Lei and M. Ouhyoung, “Software-based motion JPEG with progressive refinement for computer animation,” IEEE Trans. Consumer Electronics, vol. 40, pp. 557-562, Aug. 1994.

[33] Wedi T. and Wittmann S., “Quantization offsets for video coding,” International Symposium on Circuits and Systems, Vol. 1, pp. 324-327, May 2005.

[34] D. A. Huffman, “A method for the construction of minimum redundancy codes,” Proceedings of The IRE , vol. 40, pp. 1098-1101, September 1952.

[35] Langdon G., and Rissanen J., “Compression of black-white images with arithmetic coding,” IEEE Transactions on Communications, vol. COM-29, no. 6, pp. 858-867, June 1981.

[36] Dihong Tian, W.H. Chen, Pi Sheng Chang, Al Regib, G., and Mersereau, R.M., “Hybrid Variable Length Coding for Image and Video Compression,” IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 1, pp. I-1133 – I-1136, April 2007.

[37] Moorhead R. II, and Rajala S., “Motion-compensated interframe coding,” IEEE International Conference on Acoustics, Speech and Signal Processing, vol.10, pp347-350, Apr. 1985.

[38] Girod B., “The Efficiency of Motion-Compensating Prediction for Hvbrid Coding; of Video Sequences,” IEEE Journal on Selected Areas in Communications, vol.5, issue 7, pp1140-1154, Aug. 1987.

[39] Ichino K., Yoshida T., Nishihara I., and Sakai Y., “2D/3D hybrid video coding based on motion compensation,” IEEE International Conference on Image Processing, vol. 2, pp.644-648, Oct 1999..

[40] J. H. Lee and S. S. Lee, “A GoP-skipping-based dynamic transmission scheme for supporting fast scan functions of a stored video,” TENCON 99. Proceedings of the IEEE Region 10 Conference, Volume 2, 15-17 Sept. 1999 Page(s): 919 – 922.

[41] E. L. Abram-Profeta and K. G. Shin, “Providing unrestricted VCR functions in multicast video-on-demand servers,” in Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS’98), 1998.

[42] Wilson W. F. Poon, and K. T. Lo, “Design of multicast delivery for providing VCR functionality in interactive video-on-demand systems,” Broadcasting, IEEE Transactions on, Volume 45, Issue 1, March 1999 Page(s): 141 – 148.

[43] J. M. Choi, S. W. Lee, and K. D. Chung, “A multicast delivery scheme for VCR operations in a large VOD system,” Parallel and Distributed Systems, 2001. ICPADS 2001. Proceedings. Eighth International Conference on 26-29 June 2001 Page(s): 555 – 561.

[44] 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.

[45] S. Cen, “Reverse playback of MPEG video,” U.S. Patent 5739862, Apr. 14, 1998.

[46] S. J. Wee and B. Vasudev, “Compressed-domain reverse play of MPEG video streams,” in Proc. SPIE Conf. Multimedia Systems and Applications, pp. 237-248, November 1998.

[47] S. J. Wee, “Reversing motion vector fields,” in Proc. IEEE Int. Conf. Image Processing 1998 (ICIP98), pp. 209-212, October 1998.

[48] C. W. Lin, J. Zhou, J. Youn, and M. T. Sun, “MPEG video streaming with VCR functionality,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, pp. 415-425, March 2001.

[49] S. Y. Huang, “Improved techniques for dual-bitstream MPEG video streaming with VCR Functionalities,”

[50] N. Omoigui, L. He, A.Gupta, J. Grudin, and E. Sanocki, “Time-compression: system concerns, usage, and benefit,” in Proc. ACM SIGHI Conf., pp. 136-143, May 1999.

[51] C.H. Fu, Y.L. Chan, and W.C. Siu, “Fast motion estimation and mode decision for H.264 reverse transcoding”, pp.1385-1386, Vol.42, No.24, 23 November, 2006, Electronic Letters.

[52] C.H. Fu, Y.L. Chan, and W.C. Siu, “Efficient motion estimation in H.264 reverse transcoding”, in Proc. IEEE International Conference on Image Processing 2007, pp.V317-320, September 16-19, 2007.

[53] JVT Reference Software from http://iphome.hhi.de/suehring/tml/download/

[54] A. Chang, O.C. Au, and Y.M. Yeung, “A novel approach to fast multi-block motion estimation for H.264 video coding,” International Conference on Multimedia and Expo, Volume 1, 6-9 July 2003.

[55] Y. Peng, H. Y. C. Tourapis, A. M. Tourapis, and J. Boyce, “Fast mode decision and motion estimation for JVT/H.264,” International Conference on Image Processing 2003, Volume 3, pp.853-856, September 2003

[56] B. Jeon, “Fast mode decision for H.264,” Input document to JVT meeting, 10th Meeting, Waikoloa, Hawaii, USA, 8-12 December 2003.

[57] B. Jeon, and J. Lee, “Fast mode decision for H.264,” International Conference on Multimedia and Expo, Volume 1, pp.1131-1134, 27-30 June 2004.

[58] X. Jing, and L.-P Chau, “Fast approach for H.264 inter mode decision,” Electronics Letters, Volume 40, Issue 17, pp. 1050 – 1052, 19 August. 2004.

[59] Y. H. Kim, J. W. Yoo, S. W. Lee, J. Shin, J. Paik, and H. K. Jung, “Adaptive mode decision for H.264 encoder,” Electronics Letters, Volume 40, Issue 19, pp. 1172 – 1173, 16 September 2004.

[60] Q. H. Dai, D. D. Zhu, and R. Ding, “Fast mode decision for inter prediction in H.264,” International Conference on Image Processing 2004, Volume 1, pp.119-122, 24-27 Oct. 2004.

[61] Y. F. Shen, D. M. Zhang, C. Huang, and J. T. Li, “Fast mode selection based on texture analysis and local motion activity,” International Conference on Communications, Circuits and Systems, 2004, Volume 1, pp. 539 – 542, 27-29 June 2004.

[62] Z. Zhou, M. T. Sun, and Y. F. Hsu, “Fast variable block-size motion estimation algorithm based on merge and slit procedures for H.264,” International Symposium on Circuits and Systems, 2004, Volume 3, pp.725-728, 23-26 May 2004.

[63] H. Zhu, C. K. Wu, Y. L. Wang, and Y. Fang, “Fast Mode Decision for H.264/AVC Based on Macroblock Correlation,” 19th International Conference on Advanced Information Networking and Applications, Volume 1, pp. 775-780, 28-30 March 2005.

[64] C. F. Lin and J. J. Leou, “An adaptive fast full search motion estimation algorithm for H.264,” IEEE International Symposium on Circuits and Systems 2005, Volume 2, pp. 1493-1496, 23-26 May 2005.

[65] X. A. Lu, Tourapis A. M., Yin Peng, and J. Boyce, “Fast mode decision and motion estimation for H.264 with a focus on MPEG-2/H.264 transcoding,”

IEEE International Symposium on Circuits and Systems, volume 2, pp. 1246-1249, 23-26 May 2005.

[66] Z. Zhou, S. J. Sun, S. M. Lei, and M. T. Sun, “Motion information and coding mode reuse for MPEG-2 to H.264 transcoding,” IEEE International Symposium on Circuits and Systems, 2005, pp. 1230-1233,23-26 May 2005

[67] I. H. Shin, Y. L. Lee, and H. W. Park, “Motion estimation for frame-rate reduction in H.264 transcoding,” Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2004, pp. 63-67, 11-12 May 2004.

[68] C. H. Fu, Y. L. Chan, and W. C. Siu, “Efficient reverse play algorithms for MPEG video streaming with VCR functionality” in Proc. International Conf. on Information, Communications and Signal Processing, and the fourth IEEE Pacific-Rim Conf. on Multimedia, 1356-1359, Dec., 2003.

[69] C. H. Fu, Y. L. Chan, and W.C. Siu, “Efficient reverse-play algorithms for MPEG video with VCR support”, IEEE Trans. Circuits Syst. Video Technol., vol.16, issue 1, pp.19-30, Jan. 2006.

[70] 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, pp. 369-377, Aug. 1998.

[71] 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, pp. 313-317, June 1996.

[72] 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, pp.438-442, Aug. 1994.

[73] K. T. Fung, Y. L. Chan, and W. C. Siu, “Low-complexity and high quality frame-skipping transcoder,” in Proc. IEEE Int. Symposium on Circuits and Syst.'2001, Sydney, Australia, pp. 29-32, May 6-9, 2001.

[74] K. T. Fung, Y. L. Chan, and W. C. Siu, “New architecture for dynamic frame-skipping transcoder,” IEEE Trans. Image Processing, Vol. 11, No. 8, pp. 886-900, August 2002, U.S.A.

[75] K. T. Fung, Y. L. Chan, and W. C. Siu, “Low-complexity and high-quality frame-skipping transcoder for continuous presence multipoint video conferencing,” IEEE Trans. Multimedia, Vol. 6, No. 1, pp. 31 – 46, Feb. 2004.

[76] C.H. Fu, Y.L. Chan, and W.C. Siu, ‘New Architecture for MPEG Video Streaming System with Backward Playback Support’, IEEE Trans. on Image Processing, vol.16, no.9, pp.2169-2183, Sep. 2007.

[77] R. Kurceren and M. Karczewisz, “A Proposal for SP-frames,” ITU-T Q.6/SG 16, VCERG-L27, Germany, Jan 2001.

[78] X. Sun, F. Wu, S. Li, W. Gao, and Y.-Q. Zhang, “Improved SP coding technique,” JVT-B097, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Geneva, Jan. 2002.

[79] X. Sun, F. Wu, S. Li, and R. Kurceren, “Efficient and flexible drift-free video bitstream switching at predictive frames,” in Proc. ICME’2003, pp.541-544, Sept. 2003.

[80] E. Setton and B. Girod, “Rate-distortion analysis and streaming of SP and SI frames,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 16, No. 6, pp. 733-743, June 2006.

[81] J. Youn, M. T. Sun, and C. W. Lin, “Motion Vector Refinement for High-performance Transcoding,” IEEE Transactions on Multimedia, Vol. 1, pp. 30-40, March 1999, U.S.A.

[82] J. Youn, M. T. Sun, and C. W. Lin, “Motion Estimation for High Performance Transcoding,” IEEE Transactions on Consumer Electronics, Vol. 44, pp. 649-658, Aug. 1998, U.S.A.

[83] J. N. Hwang, T. D. Wu, and C. W. Lin, “Dynamic Frame-skipping in Video Transcoding,” 1998 IEEE Second Workshop on Multimedia Signal Processing, pp. 616 –621, 1998.