NOTE TO USERS

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DE ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Zhi Jun LEI

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE - DEGREE

School of Information, Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Video Transcoding Techniques for Wireless Video Communications

N. Georganas

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

D. Coll

É. Dubois

L. Guan

J. Zhao

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

UNIVERSITY OF OTTAWA

Video Transcoding Techniques for Wireless Video Communications

by

Zhijun Lei

A dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Ottawa-Carleton Institute of Computer Science

School of Information Technology and Engineering

Your file Votre référence
ISBN: 0-494-01730-9
Our file Notre référence
ISBN: 0-494-01730-9

To my wife, Di, for her constant support and love,

my parents, for their encouragement, guidance, and love.

ACKNOWLEDGEMENTS

I am extremely grateful to Dr. Nicolas D. Georganas, supervisor of my thesis, for his excellent guidance, constant encouragement, and support during the course of this research. He has always been my mentor in my research and beyond the academic scope.

I want to thank most dearly my parents for their love and support. I would also like to thank my wife Di who had been at my side in a supporting role giving me all the love I needed.

The work embodied in this thesis also owes much to the interactions with and encouragement from some professors, colleagues, and friends, most especially Professor Eric Dubois, Professor Jinying Zhao, Professor David Coll, Huiping Guo, Francois Malric, Wenbiao Zhu, Amir Ghavam, and Mojtaba Hosseini. I would like to express my appreciation to them for their valuable suggestion and assistance. I thank the former and current members of the MCRLab and DISCOVER lab. They have all made my life and work at University of Ottawa an enjoyable experience.

ABSTRACT

The transmission of compressed video over channels with different capacities may require a reduction in bit rate if the transmission channel has a lower capacity than the capacity required by the video bit-stream, or when the channel capacity is changing over time. The process of converting a compressed video format into another compressed format is known as transcoding.

This thesis addresses the specific transcoding problem of dynamic bit-rate adaptation for transmission over low bandwidth wireless channels. Transmitting compressed video over lower bandwidth wireless channels require accurate and efficient rate-control schemes. In this thesis, we propose several techniques to improve transcoding performance. Based on our experimental results, we present an approximate linear bit allocation model and macroblock layer rate-control algorithm, which can achieve accurate transcoding bit-rate. By reusing useful statistics information from the incoming compressed video, the bit-rate of the transcoded video can be determined according to the video scene context.

Considering a specific bursty error wireless channel, we propose a solution which combines video transcoding and an ARQ protocol to transmit compressed video over this channel. In order to make sure that the end decoder can decode and play the transcoded video within the required end-to-end delay, we analyze the rate and buffer constraints of the transcoder and derive the conditions that have to be met by the transcoder. In order to test the proposed solution, we use a statistical channel model to simulate the wireless channel and use this model and channel observation to estimate the effective channel bandwidth, which will be fed back to the transcoder for better rate control.

In this thesis, we discuss two applications. For real time video communication over wireless channel, we propose an algorithm that determines the transcoding scaling factor considering end-to-end delay, buffer fullness and effective channel bandwidth. For pre-encoded video distribution over wireless channels, we propose an algorithm which can determine the transcoding bit budget based on end-to-end delay, effective bandwidth, and original video bit profile. The proposed algorithm outperforms H.263 TMN8 in terms of video quality and buffer behavior with the same computational requirements.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. Introduction

1.1 Motivations

In the last decade, both mobile communications and multimedia communications have experienced rapid growth and commercial success. Recent advances in computing and communication technology have stimulated the research interest in digital techniques for encoding and transmitting visual information. Many video services use encoded video for the distribution over channels with different capacities, which may require a reduction in bit-rate, if the transmission medium has a lower capacity than the capacity required by the video bit-stream, or when the channel capacity is changing over time. The process of converting a compressed video format into another compressed format is known as transcoding. With the capability of dynamically changing coding parameters of the compressed video, it is expected that video transcoding will play an important role for Universal Multimedia Access (UMA) by the users with different access links and devices. What differentiates the transcoding techniques from conventional encoding techniques is that the input coded video stream is readily available in the transcoding process. Valuable coding parameters and statistics of the video can be easily obtained from the input video stream, and be reused to facilitate the transcoding process by reducing its computational complexity and improving the coded video quality.

This thesis addresses the specific transcoding problem of dynamic bit-rate adaptation of compressed H.263 video bit-streams for transmission over low bit-rate wireless networks. On one hand, pre-compressed videos are usually compressed with high quality or high bit-rate. Furthermore, after compression, the video bit-rate and formats are fixed. On the

other hand, wireless channel bit-rate is low and also changing over time due to interference, signal power, retransmission, etc. In order to transmit pre-compressed video to the end users through wireless channels, the following challenges have to be solved. First, high bit-rate videos have to be converted into low bit-rate ones by transcoding. Second, the transcoded video bit-rates have to be adjusted dynamically according to the changing channel bandwidth. Third, the end-to-end delay has to be controlled within the application requirements. Therefore, in this thesis, we use the concept of bit-rate adaptation to combine these three problems instead of using the concept of bit-rate conversion as in previous research works.

In this chapter, we will first give a brief introduction of the transcoding related works. After that, we will provide the objective of this research and summarize its main contributions and related publications. Finally, the organization of this thesis will be presented.

1.2 Related Works

1.2.1 Video transcoding for bit-rate adaptation

Bit-rate adaptation transcoding has been the most popular research topic among all the different video transcoding research topics. The idea of compressed video bit-rate adaptation is initiated by the applications of transmitting precoded video streams over heterogeneous networks. Due to the variety of different networks comprising the present communication infrastructure, a connection from the video source to the end user may be established through links of different characteristics and capacities. In interactive video applications, the bit-rate of the compressed video was usually adjusted by the source encoder to match the available capacity of the most stringent link used in the connection. The visual quality has to be sacrificed because the encoded video bit-rate has to match

the "weakest link". On the other hand, when pre-encoded video needs to be distributed to users with different connections, such as in Video on Demand (VoD) applications, this is hard to implement because the target transmission channel conditions are generally unknown when the video is originally encoded. Since the encoding process needs to know the channel characteristics, these have to be assumed and given to the encoder while it is running. A great lack of flexibility arises when these bit streams are transmitted through different channels. Alternatively, some service providers encode and store several copies of the same video program, each one encoded with a different target bit-rate assumption. Depending on the number of different channel capacities, the implementation of such a solution may imply keeping a large number of replications of the same material stored in the server. Nevertheless, if the transmission capacity is allowed to change, as it happens in wireless channels, this solution still does not provide the desired flexibility. A similar problem arises in broadcasting and multicasting live encoded programs through channels with diverse characteristics. In this case, the encoder can receive several conflicting requests from distinct switching nodes, according to their own level of congestion or permanent constraints. The encoder has to comply only with the most restrictive node and other users that do not experience any constraints in their transmission paths are unnecessarily penalized [9].

Scalable coding schemes provided in current video coding standards were originally targeted for dynamic bit-rate adaptation. Scalable coding supports a variety of scaled video quality with different peak signal-to-noise ratios (PSNR scalability), frame-rates (temporal scalability), or spatial resolutions (spatial scalability). To achieve different levels of video quality, the video source is first encoded with a low PSNR, low frame-

rate, or low spatial resolution to form a base layer. The residual information between the base layer and the original input is then encoded to form one or more enhancement layers. Successful transmission of the base layer results in a video sequence with basic quality. Additional transmissions of the enhancement layers enhance the quality by adding the residual information. However, this can only provide limited number of enhancement layers, which results in limited levels of video quality. In many networked multimedia applications, a much finer scaling capability is desirable. At the same time, more syntax overhead and complexity is introduced. Recently, fine-granular scalable coding schemes have been proposed to support a fine bit-rate adaptation [55]. However, the schemes still need enhancement layers and require additional complexity and syntax overhead. Furthermore, besides bit-rate adaptation, future video applications require more functions, such as frame size conversion, frame rate conversion, error resilience, coding syntax conversion, etc.

Alternatively, in both of the above cases, a video transcoder can be used at the video source or an intermediate node to convert the video bit-rate. Similar to source encoders, video transcoders can modulate the data they produce by adjusting a number of parameters, including bit-rate, frame rate, and resolution. Using transcoders gives us a second chance to dynamically adjust the video bit-rate according to channel bandwidth. This is particularly useful when there are time variations in the channel characteristics. Compared to scalable coding, converting a previously compressed video bit-stream to another rate bit-stream through transcoding can provide finer and more dynamic adjustments of the bit-rate of the coded video bit-stream [118]. In most bit adaptation cases, a pre-encoded video with high bit-rate and fine visual quality needs to be

converted into low bit-rate video with gracefully degraded visual quality. The ideal result is to achieve visual quality as good as the quality of a video stream that is compressed by a standalone encoder at the lower bit-rate.

Besides downscaling bit-rate, a video transcoder must have the ability to dynamically control the output bit-rate according to the channel bandwidth. This is the subject of video rate-control. All existing video coding standards assume that the compressed video will be transmitted over a CBR (*Constant Bit-rate*) channel and the rate-control scheme of these standards is based on this assumption. However, this is not the case in reality. For example, if compressed video is transmitted over the Internet, which currently cannot provide a guaranteed constant bit-rate channel for a specific application, when the network is congested, most video packets will be dropped, which results in unacceptable video quality if no other mechanisms are used to protect the video stream. Another case is transmitting video over a wireless channel, which is characterized by high BER *(Bit Error Rate)* and channel fading effects. A lot of research works have been done to solve these problems, from different aspects, while video transcoding could be one of the effective solutions. Rate-control for transcoding differs from standalone rate-control in that the relationship between quantizer step-size and bit-rate for the previously coded video stream is known. This relationship provides extra information for setting the target bit-rate and quantizer step-size on a frame, slice or macroblock basis. Some research works have been done for bit-rate adaptation and bit-rate downscaling using transcoding. In [6-10], transcoding is regarded as a down conversion process, where the bit-rate of a compressed video bit stream is reduced according to a given constraint. Assunção *et al.* proposed a frequency domain transcoder structure and a rate conversion scheme, in

which the problem of optimal transcoding is formulated in an operational rate-distortion context and solved by using a Lagrangian algorithm. New quantizer scales are selected based on classic Rate-Distortion theory for transcoding each MB *(Macroblock)* or group of MB's such that the output rate does not exceed the given constraint while producing a minimum average distortion. In [11], Assunção *et al.* continue their work by considering different frame types of MPEG-2 video and using different compression ratios for each type of picture. In [9], video transcoding is utilized as a mechanism capable of decoupling video encoders from network constraints and providing congestion control of pre-encoded video traffic over ATM networks for video distribution applications. This mechanism provides an effective method of shaping video traffic independent of the initial video encoder's constraints. By using transcoders, video traffic can be controlled at any point along the transmission path and thus the Quality of Service can be maintained without relying on on-line encoders. In [31], Dogan *et al.* addressed the problem of traffic planning for mobile video communications and proposed a video transcoder bank to resolve congestion and/or bandwidth limitation. The proposed architecture presented a layered structure of multiple video rates as required by various networks. The paper also introduces an adaptive method for resolving congestion. The designed system monitors the congestion with a feedback loop within a network and adaptively produces necessary transmission rates while providing the best available service quality. In [110], Xin *et al.* investigate how to use the useful coding statistics from the incoming video stream to perform the picture bit-allocation for transcoding a pre-encoded video bit stream. They proposed a scheme for estimating the complexities of the pictures of the output video using the coding statistics computed from the input video stream. Based on the estimated

picture complexity, a picture bit allocation algorithm is proposed for the rate-control of the transcoding process.

Another problem introduced by bit-rate adaptation transcoding is the transmission buffer requirement and delay. Since video transcoders are introduced in conventional video based applications, the side effects have to be investigated. For compressed video communication, the overall delay has three fundamental components: computational delay, algorithmic structure delay, and buffering delay. As pointed out in [67] and other research works, computational delay and algorithmic structure delay can be overcome with more advanced technology and are not a fundamental barrier to low delay. However, buffering delay is actually caused by the nature of the video. All compression algorithms in which the local compression ratio depends on the characteristics of the input signal inherently generate coded data at a varying rate. The varying rate from the encoder has to be matched to the transmission link by a buffer which smoothes out short term variations. To prevent this buffer from overflowing or under-flowing the rate of coded bits must be controlled. Usually, this is accomplished by monitoring the state of the buffer fullness and deriving a feedback control signal to alter coding parameters such as a quantizer step size. The delay caused by buffering is actually the most significant part of the end-to-end delay of a video system. Since smoothing buffers play a key role in the transmission of CBR coded video, if a transcoder is inserted between an encoder and the corresponding decoder, some modifications are expected to be required in the existing buffering arrangements of a conventional CBR system, which is primarily defined for being used without transcoders. In [12, 13], Assunção *et al.* analyze the buffering implications of inserting a video transcoder within the transmission path. For transcoders with either

fixed or variable compression ratio, the authors show that the encoder buffer size can be maintained as if no transcoder existed while the decoder has to modify its own buffer size. The authors derive the conditions that have to be met by both the encoder and transcoder buffers for preventing the decoder buffer from underflowing or overflowing. Then, based on these conditions, two effective bit-rate-control algorithms for transcoding MPEG-compressed video are proposed.

1.2.2 Video transcoding for spatial resolution downscaling

Video spatial resolution downscaling is important since most current handheld devices are characterized by limited screen sizes. For downscaling the video into lower spatial resolution, motion vectors from the incoming video cannot be reused directly, but have to be re-sampled and downscaled. Based on the new motion vectors, predictive residues are recalculated and compressed. For transcoding a compressed video stream with $M \times N$ spatial resolution into a stream with smaller spatial resolution, such as $M/2 \times N/2$ or $M/4 \times N/4$, there are two problems that have to be addressed:

- How to downscale four 8×8 DCT blocks into one 8×8 DCT block (using downscaling by 2 as an example)?
- Given the motion vectors for a group of four 16×16 macroblocks of the original video, how to estimate the motion vectors for the 16×16 macroblocks in the downscaled video? This problem is very important because the accuracy of the new motion vectors directly affects visual quality of the reconstructed images.

For the second problem, there are several methods that have been proposed in the literature. In [82], Shen *et al.* proposed an Adaptive Motion Vector Re-sampling (AMVR) scheme to approximate the optimal motion vectors using the original motion

information from the incoming bit stream of the $M \times N$ video sequence. The DCT block activity is calculated for adaptive motion vector estimation. In [114], Yin *et al.* proposed an algorithm to recalculate the motion vectors for the outputting video from the incoming video, considering the motion activity of the four macroblocks and the eight neighboring macroblocks. In [104], Wong *et al.* proposed a motion vector re-sampling algorithm also considering the original motion vector. In [80], Shanableh *et al.* compare several motion vector re-sampling methods for spatial resolution down sampling, such as average, median, AMVR, etc. These methods only have good effect on videos with little motion. However, for video streams with high motion and complicated camera operations, such as scene cut, zooming and panning, the re-sampled motion vectors are not accurate enough, which results in unacceptable video quality. To solve this problem, re-sampled motion vectors have to be refined to accurately describe the motion of new video. Refinement is carried out around the re-sampled motion vectors to yield more accurate values of predicted motion. Thus, transcoding with the motion vector refinement scheme does not involve complex full-scale motion re-estimation operations. Different refinement algorithms have been proposed in the literature. In refinement algorithms, the refinement window has conventionally been chosen to be small to avoid the high complexity operations of the motion re-estimation process in the video transcoder. Indeed, small window sizes give reasonable quality improvement and achieve a good trade-off between complexity and transcoding quality. If the refinement window size is increased, only a small degree of quality improvement can be achieved. In [116, 117], Youn *et al.* proposed a fast-search motion vector refinement scheme that is capable of providing video quality comparable to which can be achieved by performing a new full-

scale motion estimation but with much less computation. In their work, motion vectors from the incoming video are refined to minimize the sum of the quantization error.

Motion vector refinement can only be accomplished in the pixel domain with the use of a locally reconstructed video frame, and therefore DCT-domain transcoding algorithms fail to provide an acceptable motion data refinement [79]. There are also some other research works investigating other aspects related with spatial resolution downscaling. In [100], Vetro *et al.* presented a detailed complexity-quality analysis of various MPEG-2 to MPEG-4 transcoding architectures that perform spatial resolution reduction. In [2], Alshaykh *et al.* proposed a technique for decoding a full-resolution video bit-stream at low memory cost and displaying the signal at a lower resolution. It minimizes the accumulation of drift errors by tracking the decoded pixels. In [115], Yin *et al.* analyze the drift error to identify the source of quality degradation when transcoding to lower resolution. Two types of drift error are considered: a reference picture error, and error due to the non-commutative property of motion compensation and down-sampling. In [90, 91], Vetro *et al.* investigate the possible means of down converting an incoming HDTV signal into one which is suitable for display onto an NTSC monitor. The authors proposed a novel frequency synthesis algorithm which constructs a set of 8×8 DCT coefficients from four original 8×8 DCT blocks. A down-conversion decoder employing a unique adaptive motion compensation algorithm is presented. In [101], Vetro *et al.* proposed a new technique to compensate the drift error based on the principle of intra-block refresh, which actually converts some percentage of inter-coded blocks to intra-coded blocks to control drift propagation. Some other related research work can be found in [16, 17, 43, 44, 89, 98, 103, 113, 121]. For this topic, most existing work focuses on

the case of downscaling by 2 or 3. How to downscale the incoming video into any factor still needs to be addressed. In the pixel domain, this can be easily implemented by down-sampling or pixel interpolation. However, this requires fully decoding the incoming video. Spatial resolution downscaling in the DCT domain is still a challenge.

1.2.3 Video transcoding for temporal resolution conversion

To transcode an incoming compressed video bit-stream for a low bandwidth outgoing channel, such as a wireless network, a high transcoding ratio is often required. However, high transcoding ratios may result in unacceptable video quality when the incoming bit-stream is transcoded with the full frame rate as the incoming bit-stream. For example, in a narrow-band wireless network, which may have a less than 20 kbps bandwidth, the quality degradation due to the low bit-rate is significant at 25 or 30 fps. Frame-rate conversion or frame-skipping is often used as an efficient scheme to allocate more bits to the remaining frames, so that acceptable quality can be maintained for each frame. In addition, frame-rate conversion is also needed when an end system supports only a lower frame-rate. For example, some handheld devices can only decode and display at most 15 video frames per second. Frame rate conversion can be simply accomplished by arbitrary frame dropping. For instance, dropping every other frame in a sequential order leads to a half rate reduction in the transcoded sequence. As discussed in the previous sections, usually the motion vectors decoded from the bit-stream are reused in transcoders to speed up the re-encoding processing. However, when frames are skipped, the motion vectors cannot be directly reused because the motion vectors from the incoming video frame are pointed to the immediately previous frame. If the previous frame is dropped in the transcoder, the link between two frames is broken and the end decoder will not be able to reconstruct the picture by these motion vectors. One solution for this problem is to

perform a full-scale motion re-estimation based on the previous non-skipped frame. In this case, the incoming frame has to be completely decoded and reconstructed into the pixel domain, and the full-scale motion estimation also requires numerous computations. To overcome this motion vector reuse problem, a linear interpolation method can be used to estimate the motion vectors from the current frame to its previous non-skipped frame if some frames between them are skipped [47]. The basic idea is that the motion vectors of the currently transcoded frame are re-estimated by interpolating their values using motion vectors in previous frames till the previous non-skipped frames.

In [119], Youn *et al.* proposed a Forward Dominant Vector Selection (FDVS) scheme to overcome some drawbacks in the bilinear interpolation method. The proposed method selects one dominant motion vector from the four neighboring macroblocks. A dominant motion vector is defined as the motion vector carried by a dominant macroblock, which is a macroblock that has the largest overlapping segment with the block pointed by the incoming motion vector. However, both the bilinear interpolation and FDVS have to be combined with a refinement scheme to produce accurate motion vectors. In [21], Chen *et al.* proposed an algorithm called Activity Dominant Vector Selection (ADVS), which utilizes the quantized discrete cosine transform coefficients of residual blocks for composing a MV from the incoming ones. In addition, a motion-vector refinement algorithm called variable step-size search is presented. In [35], Fung *et al.* proposed a frame-skipping transcoder in which direct summation of the DCT coefficients for macroblocks coded without motion compensation reduces most complex modules of the transcoder. The authors proposed a criterion which employs incoming motion vectors and re-encoding error for dynamically adjusting the frame rate. In [99], Vetro *et al.* consider

the rate-distortion models to adjust the transcoding frame rate and optimize the output visual quality. Since the major challenge in temporal resolution transcoding is producing new motion vectors, which is similar as the challenge in spatial resolution transcoding, some algorithms in spatial resolution transcoding can also be used for temporal resolution transcoding. However, when the MPEG standard is used, in which the bi-directional predicted frame (B frame) uses the previous I and successive P frame as reference, temporal resolution transcoding is still a challenge. At the same time, a reordering operation has to be carried out in the transcoder when some frames are skipped.

1.2.4 Video transcoding for error resilience

An area of increasing importance in personal communications is error resilience in compressed video as video emerges as a new medium in the error prone network environment, e.g. mobile communication networks. In a video bit stream, improved compression performance is often achieved using variable length code words to represent both transform coefficients and motion vectors. Clearly protection of the bit stream from errors can be added to the bit stream but this is done by re-introducing some of the redundancy that has been removed during the compression process. Furthermore, conventional error detection or error correction codes added to the bit stream are often simply insufficient to detect or correct bit errors due to their capability limitation. Video transcoding can be used as an alternative solution to protect the video against the channel errors and network congestion by dynamically introducing error resilient features in the video stream. For error resilience purposes, the video transcoder can apply data partitioning and insertion of re-synchronization markers into the incoming bit stream. Furthermore, the transcoder can apply unequal error protection to various segments of video data. Thus, the header data are assigned the highest protection, whereas the DCT

coefficients are transcoded with a negligible level of protection. In [30], Dogan *et al.* proposed a video transcoding algorithm, called Adaptive Intra Refresh (AIR), which adaptively convert some INTER blocks in the video frame into INTRA mode ones to provide temporal resilience between transcoded video frames and prevent the error from propagating to the following frames. In [86], Swann *et al.* proposed a transcoder to transcode the MPEG-II video stream into a more resilient stream without increasing the bit-rate by using the Error-Resilient Entropy Coding (EREC) approach. In [74, 75], Reyes *et al.* use a transcoder to modify the resilience of the encoded bit stream by using source coding techniques to provide appropriate level of resilience.

1.2.5 Video transcoding for multipoint video bridging

An important application of video transcoders is the Multipoint Control Unit (MCU) in multipoint video conferencing applications. Multipoint video conferencing is a natural extension of point-to-point video conferencing. For a multipoint videoconference over a wide-area network, the conference participants are connected to an MCU in a central office. A video combiner in the MCU combines the multiple coded digital video streams from the conference participants into a coded video stream that conforms to the syntax of the video coding standard and sends it back to the conference participants for decoding and presentation.

One approach used in the video combiner to combine the multiple coded digital video streams is the coded domain combining approach. In this approach, the video combiner modifies the headers of the individual coded video bit-streams from the conference participants, multiplexes the bit-streams, and generates new headers to produce a legal coded and combined video bit-stream conforming to the video coding standard. Despite its simplicity, the coded domain combining approach offers limited flexibility for users to

manipulate the video bit-streams. Since the coded domain combiner simply concatenates bit-streams, the combined bit-rate will be about the sum of the bit-rates of all the individual bit-streams. Thus, for each conference participant, the upstream and downstream video bit-rates are highly asymmetrical. Also, with the coded domain combining approach, it is difficult to utilize the video characteristics in a multipoint videoconference, where only one or two persons are active at one time while other persons are just listening and have little motion. It is difficult to dynamically allocate extra bandwidth to active persons using the coded domain approach. When using the transcoding approach for video combining, the video combiner decodes each coded video stream, combines the decoded video streams, and re-encodes the combined video at the transmission channel rate. The transcoding approach allows each conference participant to encode the video using the full bandwidth of the transmission channel. Since each user can use the full bandwidth, and in the re-encoding process the rate-control strategies in video coding algorithms will dynamically allocate more bits to the active frames or areas, the overall quality can be more uniform compared to the coded domain combining approach where the quality of the active person may be much worse than the inactive persons. In [60-63], Lin *et al.* proposed a dynamic rate-control method that operates in the video transcoder to enhance the visual quality and allow Region of Interest (ROI) coding in multipoint video conferencing. The proposed method first identified the active participants from the multiple incoming video streams by calculating the temporal and the spatial activities of the participant sub-windows. The sub-windows of inactive participants are dropped and the saved bits are allocated to the active sub-windows by using a rate-distortion optimized bit allocation approach. In [108], Xin *et al.* proposed an

approach to estimate the picture-complexity of the transcoded output video streams from the input streams. Based on the picture-complexity estimation, a bit-allocation strategy was presented for joint transcoding of multiple pre-encoded MPEG video streams. Other related discussions can be found in [83, 85, 107, 109].

1.2.6 Video transcoding for syntax conversion

Video transcoding for syntax conversion is usually called heterogeneous transcoding, which transcodes the video stream compressed by one codec standard into a video stream that can be decoded by another codec standard, such as MPEG-to-H.261/H.263. Because most current codec standards, such as H.261, H.263, MPEG-1, 2, 4, are all based on DCT, the syntax conversion can be performed without changing the DCT coefficients. The conversion algorithm for these standards mainly consists of the following steps:

- Video frame header adjustment;
- Video data translation from one syntax to another;
- Necessary bit stream stuffing for different synchronization requirements of different standards.

Video data translation is the major process of the entire transcoding process. This process consists of enhanced mapping operations that involve transforming video parameters from one type of syntax to another. However, this mapping processing is still a much less complex and hence less time and power consuming scheme than other transcoding, because of the fact that syntax conversion does not involve any computational intensive transformation between the pixel and frequency domain, or any motion estimation and compensation process. In [80, 81], Shanableh *et al.* presented methods for transcoding pre-encoded MPEG-1, 2 video into H.261/H.263 video with different bit-rates, and spatio-temporal resolutions. In [106], Wu *et al.* used transcoding algorithms to transcode

a Motion JPEG encoded video stream into a MPEG video stream. This can be used in current video editing software. In [28, 29], Dogan *et al.* presented a mapping processing between the MPEG-4 simple profile bit-stream syntax and the H.263 bit-stream syntax. In [33, 103], Feamster *et al.* presented a MPEG-2 to H.263 transcoder that accepts an interlaced MPEG-2 bit-stream as the input and produces a lower-bitrate progressive H.263 bit-stream as the output. The potential application scenario is the transmission of a digital television signal, which is encoded by MPEG-2, over a wireless medium. The proposed algorithm exploits the properties of the MPEG-2 and H.263 compression standards to perform interlaced to progressive (field to frame) conversion with spatial downscaling and frame-rate reduction. In [40], Guo *et al.* addressed the problems of transcoding MPEG-2 video into MPEG-4 compliant bit-stream in the compression domain to meet low complexity and low latency real-time applications. In [64], Lin *et al.* presented transcoding techniques to convert multiple layer bit-streams to a single-layer format targeted MPEG-4 FGS-to-Simple Profile transcoding. Other research works related to cross-standards transcoding can also be found in [1, 39].

1.2.7 Object based video transcoding

With the standardization of MPEG-4, arbitrary-shaped objects may be encoded and decoded as separate Video Object Planes (VOPs). At the receiver, these video objects are composed to form compound objects or scenes. For a MPEG-4 compressed video stream with several VOPs, various transcoding techniques can be used. Different bit budgets can be allocated to different VOPs according to their priorities, or certain VOPs can be dropped. In [95], Vetro *et al.* proposed a framework that considers an adaptive means of transcoding each of the objects in the scene based on available bandwidth and complexity of each object. Within the object-based framework, various transcoding strategies can be

employed. [96] dealt with encoding and transcoding multiple video objects at different temporal rates. The authors proposed a solution based on changes in the shape boundaries over time. This shape feature, or shape hint, can be used to vary the temporal resolution of multiple video objects. The newly proposed standard MPEG-7 aims to standardize a set of descriptors and description schemes that can be used to describe multimedia content for a variety of applications, such as content navigation, content management and fast search and retrieval. For video, the descriptors are low-level features of the content, such as color, texture and motion. These low-level features can be organized and referred by higher-level description schemes. The description schemes may refer to various levels of the program, such as a segment or frame, and may also be used to facilitate video summarization, specify user preferences, and support the annotation of video segments. Besides using meta-data for searching and browsing applications, it is also of interest to consider meta-data that would be suitable to guide the operation of a transcoder. Such meta-data are referred to as a Transcoding Hint within the MPEG-7 standard. In [52], Kuhn *et al.* describe several transcoding hints that have been adopted by the MPEG-7 standard. These meta-data can be used for a number of tasks, including anchor frame selection, encoding mode decisions, frame-rate and bit-rate-control, as well as bit-rate allocation among several video objects for object-based MPEG-4 transcoding. In [92], Vetro *et al.* presented an adaptive transcoding system for improved delivery of visual content. The system is adaptive in that it makes use of meta-data to identify key objects in the scene, and, depending on the network conditions, delivers only the most interesting objects. Given the objects to be transmitted, meta-data are also used to improve the bit

allocation among the objects of interest. Some other discussion related to object based transcoding and transcoding hints can be found in [93, 94, 97].

1.3 Summary of Thesis Contributions

The objective of this dissertation is to develop a framework for controlling the bit-rate of transcoded video stream for transmitting over wireless networks. Towards this objective, three major issues are addressed: macroblock layer bit allocation model and algorithm, transcoding rate and buffer analysis, and frame layer rate-control for video transcoding over wireless channels.

1.3.1 Macroblock layer bit allocation model and algorithm

The bit allocation models reported in the literature try to use some statistics of the input source data, such as variance, to describe the input video data [25, 37, 42]. They also try to analyze and model each step of the coding algorithms and formulate an explicit expression of the coding bit-rate. To achieve high coding performance, an efficient coding algorithm must often employ a sophisticated data representation scheme as well as an entropy coding scheme. To model these coding algorithms more accurately, these bit allocation models are getting more and more complex [25, 34, 42, 111], and the estimation and rate-control process becomes increasingly complicated and even unstable with the image-dependent variation [57]. As mentioned in the previous section, the rate-control of the video transcoding is different from that of the standalone video coding, because the coding statistics of the incoming video have been calculated during the first generation encoding and can be reused during the transcoding process. It would be ideal to develop a simple and accurate bit allocation model for video transcoding. Based on this simple model, we could reuse the coding statistics and then develop a simple rate-control algorithm for transcoding. Practically, this simple bit allocation model and control

algorithm based on it would enable us to control the video transcoder accurately and robustly with very low computational complexity and implementation cost. In this research, we developed a new bit allocation model for macroblock layer rate-control based on the experimental results and analysis. It serves as an alternative to the traditional model used in the TMN8 rate-control algorithm for H.263 standard. With Shannon's source coding theorem [26], we give a theoretical proof for this bit allocation model. Based on that model, we develop a rate-control algorithm for H.263 video transcoding. The proposed algorithm has low computational complexity and implementation cost. When compared to the TMN8 rate-control algorithm, our rate-control algorithm provides much more accurate rate-control, buffer regulation, and improved picture quality.

1.3.2 Transcoding rate and buffer analysis

Traditional video transmission systems are designed without using transcoders. When transcoders are introduced into the existing systems, some problems need to be addressed. For transmitting video over wireless channels, the buffer and rate constraints at the encoder, transcoder, and decoder need to be analyzed. At the same time, for real-time applications, the end-to-end delay of a video transmission system with transcoders is also an important issue. In this thesis, we provide a rate and buffer model to analyze the problem in the transcoding system. By using this model, we can derive the conditions for the encoder, transcoder, decoder buffers, and hence the transcoding scaling factor for every frame in order to satisfy these conditions.

1.3.3 Frame layer rate-control for video transcoding over wireless channels

Most previous transcoding research only focuses on transcoding video for constant bit-rate channels. In this case, the frame layer rate-control is relatively simple, because the frame bit budget is only determined by the channel bit-rate (which is constant), frame

rate, and buffer fullness. However, the nature of a video stream is variable bit-rate, considering the visual content of a video itself. Different video texture, motion activity, scene change, etc, will make the video stream to be variable bit-rate. Therefore, in order to get consistent visual quality, video is better to be compressed at the variable bit-rate. As mentioned above, in the case of transmitting video over wireless channel, the channel bandwidth is also changing over time, so the problem becomes how to transcode a variable bit-rate video into another variable bit-rate video matching the channel bandwidth. In this work, we propose several algorithms to reuse the video statistics information, such as scene change, video traffic profile, etc, from the compressed video for the frame layer transcoding rate-control. Our experimental results show that the transcoded video quality can be improved without violating the buffer and delay constraints.

1.3.4 Publications resulting from the thesis

In Refereed Journals

1. Z. Lei and N.D. Georganas, "An Accurate Bit-Rate Control Algorithm for Video Transcoding," Journal of Visual Communications and Image Representation, Vol. 14, Issue 3, September 2003, pp. 321-339.

2. Z. Lei and N. D. Georganas, "A Rate Adaptation Scheme for Real-time Video Transmission over Wireless Channels," Signal Processing: Image Communication, Special issue on multimedia adaptation ,Vol. 18, Issue 8, pp 641-658., 2003.

3. Z. Lei and N. D. Georganas, "Adaptive Video Transcoding and Streaming over Wireless Channels," Journal of Systems and Software, special issue on Adaptive Multimedia Computing.(accepted, to appear)

4. Z. Lei and N. D. Georganas, "A Survey of Video Transcoding Techniques," Submitted to ACM Computing Survey. (under review)

In Refereed Conference Proceedings

1. Z. Lei and N.D. Georganas, "Context Based Media Adaptation in Pervasive Computing," Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) 2001, Toronto, Canada, May 2001.

2. Z. Lei and N.D. Georganas, "Media Transcoding for Pervasive Computing," Doctoral Symposium, Proceedings of ACM Multimedia 2001, Ottawa, Canada, Sept. 2001.

3. Z. Lei and N.D. Georganas, "H.263 Video Transcoding for Spatial Resolution Downscaling," Proceedings of IEEE International Conference on Information Technology: Coding and Computing (ITCC) 2002, Las Vegas, Nevada, U.S.A, Apr. 2002.

4. Z. Lei and N.D. Georganas, "Accurate Bit Allocation and Rate Control for DCT Domain Video Transcoding," Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) 2002, Winnipeg, Canada, May 2002.

5. Z. Lei and N.D. Georganas, "A Frame Layer Bit-Allocation for H.263+ Based Video Transcoding," Proceedings of 21st Biennial Symposium on Communications, Kingston, Canada, June 2002.

6. Z. Lei and N. D. Georganas, "Rate Adaptation Transcoding for Precoded Video Streams," Proceedings of ACM Multimedia 2002, Juan-Les-Pins, France, December 2002.

7. Z. Lei and N. D. Georganas, "Video Transcoding Gateway for Wireless Video Access," Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) 2003. Montreal, May 2003.

8. Z. Lei and N. D. Georganas, "Rate Adaptation Transcoding for Video Streaming over Wireless Channels," Proceedings of International Conference Multimedia Expo. 2003 (ICME2003), Baltimore, July 6-9, 2003.

1.4 Thesis Organization

The thesis is organized as follows:

Chapter 2 first overviews the basic building blocks of the general DCT based video coding systems. Then, typical transcoding architectures are summarized, major issues of developing different transcoders are discussed, and performance and complexity of different transcoder architectures are compared.

Chapter 3 shows that the amount of generated bits during encoding and the number of non-zero AC coefficients in a frame can be modeled by an approximate linear relationship. Based on our extensive experiments, it is shown that the model parameters obtained from the incoming video frame can be reused for the rate-control in the

transcoding process. A theoretical justification for this linear model is provided. The physical meaning of the model parameters is also discussed.

Chapter 4 presents a rate-control algorithm for H.263 video transcoding. Complexity analysis is provided to show that the proposed algorithm has very low computational complexity. The implementation details are discussed. Experimental results and comparisons with other rate-control algorithms are provided.

Chapter 5 first introduces a framework for analyzing the rate, buffer, and delay constraints of transcoders. By using a two-state Markov model, wireless channels are simulated and effective bandwidth is estimated. Based on the channel model and estimated effective channel bandwidth, we propose an algorithm targeted for real-time video transcoding and transmission over wireless channel. In this case, the frame layer bit budget is determined considering the buffer fullness, channel bandwidth and end-to-end delay.

Chapter 6 studies the problem of transcoding and transmitting pre-encoded video over wireless channels. In this case, the frame layer bit budget is determined considering the buffer fullness, scene content of the video, and channel bandwidth.

Chapter 7 presents an implementation of a video transcoding gateway demo system, which serves as a prototype application of video transcoding.

Chapter 8 summarizes the studies presented in this thesis. Concluding remarks are provided, and future research directions are discussed.

Chapter 2. Video Transcoding Techniques

2.1 Overview of Digital Video Coding

2.1.1 Digital video representation

For compression to be meaningful, a standard representation should be defined for the data to be compressed. In order to be processed by computers, analog video that is captured by a light sensor is digitized, which consists of three steps: 1) spatial sampling, 2) temporal sampling, and 3) quantization. After digitization, the continuous video in both the spatial and temporal domains becomes a set of discrete picture elements, commonly called pixels or pels, represented by a fixed number of bits. The following properties are used to characterize the digital representation of the video:

- Color spaces

Representing color requires multiple numbers per sample. There are several alternative systems for representing color, each of which is known as a color space. Most common color spaces for digital image and video representation are RGB (red/green/blue) and YCrCb (luminance/red chrominance/blue chrominance). In the RGB system, each pixel is represented by three numbers indicating the relative proportions of red, green and blue. However, since the human visual system is less sensitive to color than to luminance, in order to take advantage of this characteristic, the YCrCb color system is used more often to represent color efficiently. The key advantage of YCrCb over RGB is that the Cr and Cb components may be represented with a lower resolution than Y because the human visual system is less sensitive to the chrominance components. This reduces the amount of data required to represent the chrominance components without having an obvious effect on visual quality.

- Sampling patterns

In order to represent different color components with different resolution, different sampling patterns are used. As illustrated in Figure 2-1, 4:4:4 means that the three components have the same resolution and hence a sample of each component exists at every pixel position. In 4:2:2 sampling, the chrominance components have the same vertical resolution but half the horizontal resolution. 4:2:0 means that Cr and Cb each have half the horizontal and vertical resolution of Y.

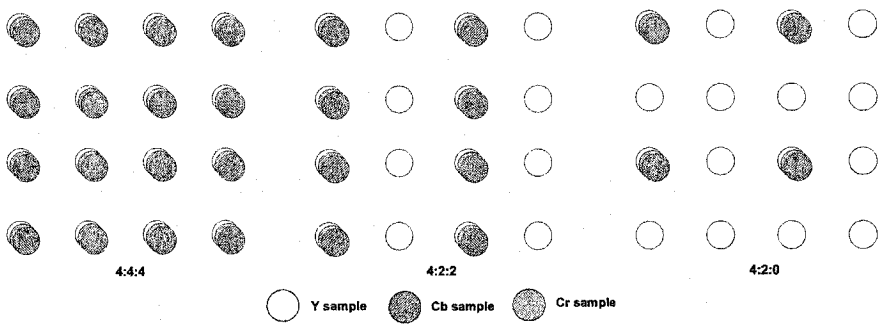


Figure 2-1. Color sampling patterns

- Temporal frequency

A moving video image is formed by sampling the video signal temporally, taking a rectangular snapshot of the signal at periodic time interval. A higher temporal frequency (frame rate) gives a smoother appearance to motion in the video scene but requires more samples to be captured and stored.

To promote the interchange of digital video date, several formats for representing video data have been standardized. Some of the popular standard representations are listed in Table 2-1 [102].

Table 2-1. Digital video formats for different applications

Video Format	Y Size	Color Sampling	Frame Rate	Raw Data (Mbps)
HDTV over air, cable, satellite, MPEG2 video, 20-45 Mbps				
SMPTE 296M	1280×720	4:2:0	24P/30P/60P	265/332/664
SMPTE 295M	1920×1080	4:2:0	24P/30P/60I	597/746/746
Video Production, MPEG2, 15-25 Mbps				
BT.601	720×480/576	4:4:4	60I/50I	249
BT.601	720×480/576	4:2:2	60I/50I	166
High quality video distribution (DVD, SDTV), MPEG2, 4-8 Mbps				
BT.601	720×480/576	4:2:0	60I/50I	124
Intermediate quality video distribution (VCD, WWW), MPEG1, 1.5 Mbps				
SIF	352×240/288	4:2:0	30P/25P	30
Video conferencing over ISDN/Internet, H.261/H.263, 128-384 Kbps				
CIF	352×288	4:2:0	30P	37
Video telephony over wired/wireless modem, H.263, 20-64 Kbps				
QCIF	176×144	4:2:0	30P	9.1

2.1.2 Transform coding

Transform coding has become a dominant approach for image and video compression. In transform coding, an image is divided into blocks of pixels. Each block is mathematically transformed into a different representation, which is further quantized and coded to achieve compression. The mathematical transform is chosen so as to redistribute most of the image information into a small set of less correlated coefficients. Usually, the transform is linear and orthogonal. Through quantization, most of the "unimportant" coefficients are 0 and can be ignored, while the "important" coefficients are retained. In the decoder, an inverse quantization process is followed by an inverse transformation.

The motivation for why transform coding has relatively good capability for compression comes from the following:

- Not all transform coefficients need to be retained to maintain good image quality.
- Coefficients that are coded need not to be represented with full accuracy.

As the transform coefficients are generally related to the spatial frequencies in the image, compression techniques can take advantage of the psychovisual property of the human visual system by quantizing more coarsely the higher frequency coefficients.

For image and video, the two-dimensional discrete cosine transform (2D-DCT) is the popular block transform that is used in most standards. The 8×8 2D-DCT transforms a block $\{x(n,m)\}_{n,m=0}^{7}$ in the spatial domain into a block of frequency components $\{X(k,l)\}_{k,l=0}^{7}$ according to the following equation

$$X(k,l)=\frac{c(k)}{2}\frac{c(l)}{2}\sum_{n=0}^{7}\sum_{m=0}^{7}x(n,m)\cos(\frac{2n+1}{16}\cdot k\pi)\cos(\frac{2m+1}{16}\cdot l\pi) \tag{2-1}$$

where $c(k)=\begin{cases}\frac{1}{\sqrt{2}}, & k=0\\ 1, & k>0\end{cases}$. The inverse transform is given by

$$x(n,m)=\sum_{k=0}^{7}\sum_{l=0}^{7}\frac{c(k)}{2}\frac{c(l)}{2}X(k,l)\cos(\frac{2n+1}{16}\cdot k\pi)\cos(\frac{2m+1}{16}\cdot l\pi) \tag{2-2}$$

In a matrix form, define the 8×8 DCT matrix $S=\{s(k,n)\}_{k,n=0}^{7}$, where

$$s(k,n)=\frac{c(k)}{2}\cos(\frac{2n+1}{16}\cdot k\pi) \tag{2-3}$$

Then,

$$X=SxS^{t} \tag{2-4}$$

where the superscript t denotes matrix transposition. Similarly, let the superscript $-t$ denotes transposition of the inverse. Then,

$$x=S^{-1}XS^{-t}=S^{t}XS \tag{2-5}$$

where the second equality follows from the unitarity of S. In the 8×8 2D-DCT, the coefficient $X(0,0)$ is known as the DC coefficient of the block and is 8 times the average of the input block samples. Other coefficients of the transform are known as the AC coefficients.

2.1.3 Motion estimation and motion compensation

Successive frames in a video sequence are typically highly correlated, especially for scenes where there is little or no motion. Motion estimation and motion compensation are used to take advantage of this temporal redundancy that exists between frames. Motion estimation creates a model of the current frame based on available data in one or more previously encoded frames ("reference frames"). Using this model, the encoder performs motion estimation to determine the motion that exists between reference frames and the current frame. These reference frames may be "past" frames or "future" frames. The current frame is motion compensated by subtracting the model from the frame to produce a motion-compensated residual frame, which is coded and transmitted, along with the information required for the decoder to recreate the model (typically a set of motion vectors). At the same time, the encoded residual is decoded and added to the model to reconstruct a decoded copy of the current frame (which may not be identical to the original frame because of the coding losses). The reconstructed frame is stored to be used as a reference for further predictions. The whole process is illustrated in Figure 2-2.

As can be seen from this figure, in order to correctly reconstruct the encoded frame at the decoder side, both the encoder and decoder must have the same reference frames. Since the original frames are not available at the decoder side, therefore, the encoder usually includes a decoding loop to reconstruct the encoded frame. Then, the previously reconstructed frames, instead of the original frames, are used as references frames in the encoder. In this way, both the encoder and the decoder will have the same reference frames. However, if there is a mismatch between the reference frames in the encoder and decoder, drift error will happen. Since the distorted reconstructed picture is also used for the future prediction, the drift error propagates to the future reconstructed pictures.

Therefore, even a small mismatch at one frame can cause significant quality degradation over time. In this case, quality of the reconstructed video will degrade rapidly.

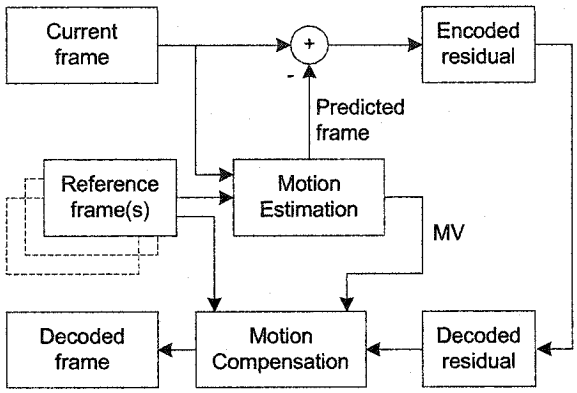


Figure 2-2. Motion estimation and compensation block diagram.

In the popular video coding standards, motion estimation and compensation are carried out on 8×8 or 16×16 blocks in the current frame. Motion estimation of complete blocks is known as block matching. For each block of luminance samples in the current frame, the motion estimation algorithm searches a neighboring area of the reference frame for a "best matching" block. The best match is the one that minimizes the difference between the current block and the corresponding block in the neighboring area of the reference frame. The area in which the search is carried out may be centered on the position of the current block, because (a) there is likely to be a good match in the immediate area of the current block due to the high similarity between subsequent frames and (b) it would be computationally intensive to search the whole reference frame.

Figure 2-3 illustrates the block-based motion estimation process. A current frame is divided into blocks. The location of each block is given by the (m, n) coordinates of its left-top corner. To predict a block in a current frame, a search window centered at the current block is identified in the reference frame. The size of the search window is $(N+2w)\times(N+2w)$. This region is then searched for the best matching block. Let

$(m+i,\ n+j)$ be the location of the best match. The vector from $(m\ n)$ to $(m+i,\ n+j)$ is referred to as the motion vector associated with that block. Typically, the motion vector is expressed in relative coordinates as (i, j).

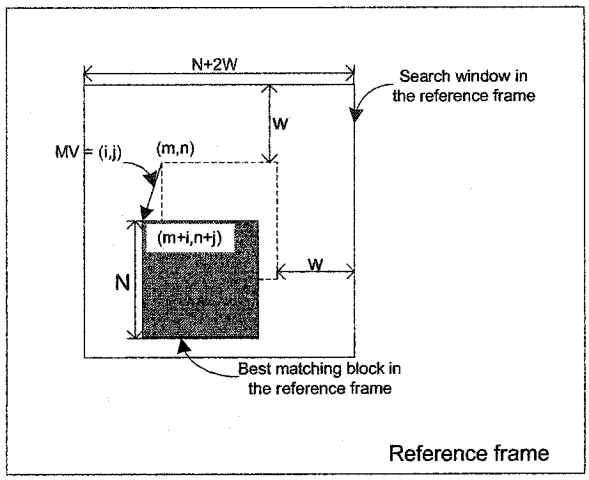


Figure 2-3. Block-based motion estimation process.

For each block in a current frame, the corresponding best matching block in the reference frame is calculated by minimizing a cost function. Two widely used cost functions are the Mean Square Error (MSE) defined as:

$$MSE(i,j) = \frac{1}{N \times N} \sum_{k=1}^{N} \sum_{l=1}^{N} \left[S_c(m+k, n+l) - S_r(m+k+i, n+l+j) \right]^2 \qquad (2\text{-}6)$$

and the Mean Absolute Difference (MAD) defined as:

$$MAD(i,j) = \frac{1}{N \times N} \sum_{k=1}^{N} \sum_{l=1}^{N} \left| S_c(m+k, n+l) - S_r(m+k+i, n+l+j) \right| \qquad (2\text{-}7)$$

where S_c and S_r are pixel values of $N \times N$ blocks in current and reference frames, respectively, and $-w \le i, j \le w$.

In order to find the best match with a maximum displacement of w, an exhaustive search would require $(2w+1)^2$ evaluations of the cost function. If MSE is used, every evaluation associated with a $N \times N$ block corresponds to two additions and one multiplication for the N^2 elements of each block. Therefore, the total complexity for a block is

$(2w+1)^2(3N^2-1)$. To reduce processing cost, MAD is preferred to MSE, and hence is used in all the video codecs. However, for each block of N^2 pixels we still need to carry out $(2w+1)^2$ tests, each with almost $2N^2$ additions and subtractions. In fact, studies have shown that in the H.261 and MPEG-1 codecs, motion estimation comprises almost 60-70% of the total computation in a video codec. Hence fast motion estimation techniques are highly desirable. In the past two decades a number of fast search methods for motion estimation have been introduced to reduce the computational complexity. The basic principle of these methods is that the number of search points can be reduced, by selectively checking only a small number of specific points, assuming that the distortion measure monotonically decreases towards the best matched point.

So far, we have assumed that the best match can be found at a region offset from the current block by an integer number of pixels. Thus, the motion vector would be pixel or pel-accurate. Since the true frame-to-frame displacements are unrelated to the sampling grid, and a moving object will not necessarily move by an integral number of pixels between successive video frames, thus, one would expect improved prediction if displacement estimates were obtained at a finer resolution. This implies that we need to determine motion vectors with fractional or sub-pixel accuracy. In fact, for many blocks a better match can be obtained by searching a region interpolated to sub-pixel accuracy, such as in H.263 and MPEG-2 standards, half-pixel precision motion estimation is used to get more accurate motion vectors, and hence, higher compression. In this case, the search algorithm is extended as follows:

- Interpolate between the samples of the search area in the reference to form a higher-resolution interpolated region.

- Search full-pixel and sub-pixel locations in the interpolated region and find the best match.
- Subtract the samples of the matching region (whether full- or sub-pixel) from the samples of the current block to form the difference block.

Half-pixel interpolation used in the H.263 and MPEG2 is illustrated in Figure 2-4. The original integer pixel positions are shown in black. Samples a and b are formed by linear interpolation between pairs of integer pixels vertically and horizontally, and samples c are interpolated between four integer pixels.

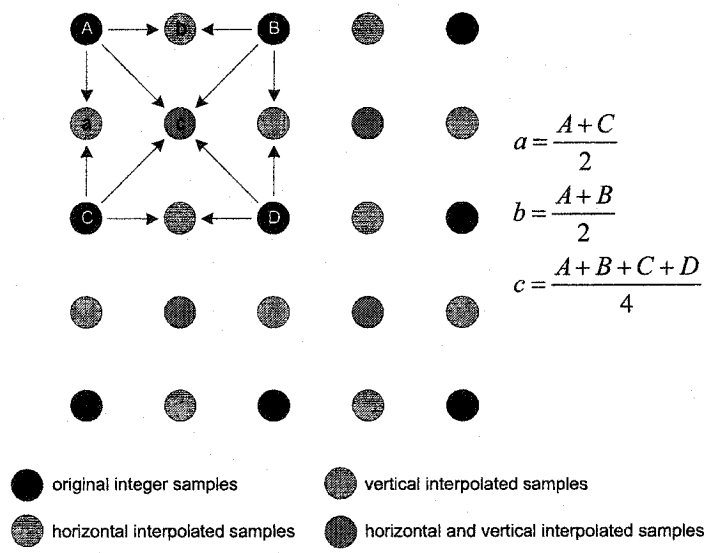


Figure 2-4. Half-pixel interpolation.

In order to obtain the half-pixel interpolated block from the original block of samples, the following processes are applied on the original block. We define x as the original block, x_v, as the vertically interpolated block, x_h as the horizontally interpolated block, and x_{vh} as the block with both vertical and horizontal interpolation. Then we have

$$x_v = f_v \cdot x, \quad x_h = x \cdot f_h, \quad x_{vh} = f_v \cdot x \cdot f_h \tag{2-8}$$

where

$$f_v = \frac{1}{2}\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad f_h = \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

As we can see, searching on a sub-pixel grid obviously requires more computations than the integer searches described earlier. Some simplified methods have been proposed, but such computations are not completely avoidable [15]. In order to limit the increase in complexity, it is a common practice to find the best matching integer position and then to carry out a search at half-pixel locations immediately around this position [76]. Despite the increased complexity, sub-pixel motion estimation and compensation can significantly outperform integer motion estimation/compensation.

2.1.4 Quantization

In a transform-based video codec, the transform stage is usually followed by a quantization stage. Theoretically, the transforms described earlier are reversible, i.e., applying the transform following by its inverse to image data results in the original image data. This means that the transform process does not remove any information; it simply represents the information in a different form. However, the quantization stage removes less "important" information, i.e., information that does not have a significant influence on the appearance of the reconstructed image, make it possible to compress the remaining data. In block-based video coding standards, such as H.263 and MPEG-2, the quantization stage maps the values of the DCT transformed coefficients to a smaller range of values. The class of quantizer is based on the so-called Uniform Threshold

Quantizer (UTQ). It has equal step sizes with reconstruction values pegged to the center of the steps as illustrated in Figure 2-5.

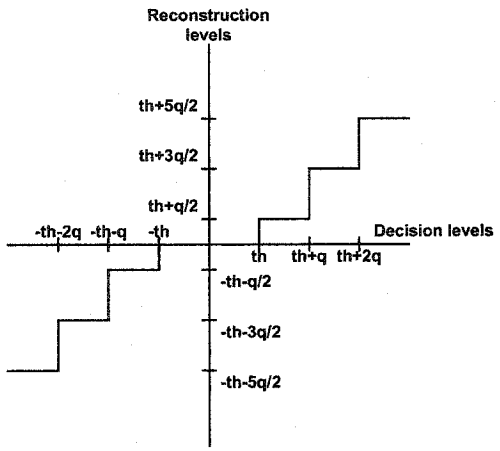


Figure 2-5. Quantization characteristics

A further two sub-classes of UTQ can be identified within the standard codecs, namely those with and without a dead zone and will be hereafter abbreviated as UTQ-DZ and UTQ respectively. They are illustrated in Figure 2-6. The term dead zone commonly refers to the central region of the quantizer, where the coefficients are quantized to zero. Typically, UTQ is used for quantizing intraframe DC coefficients, while UTQ-DZ is used for the AC and DC coefficients of interframe prediction error. This is intended primarily to cause more non-significant AC coefficients to become zero, so increasing the compression. In UTQ, coefficients *F(u,v)* are quantized by their division to the quantizer step size q with rounding towards the nearest integer as $I(u,v) = Round\left(\frac{F(u,v) \pm \frac{q}{2}}{q}\right)$.

Such a ratio, *I(u,v)*, is called the quantization index, and is sent to the decoder. At the decoder, the reconstructed coefficients, $F^q(u,v)$, after inverse quantization are given by $F^q(u,v) = I(u,v) \times q$. Quantization indices employing UTQ-DZ are obtained by using

integer division, which will truncate the resultant towards zero as $I(u,v) = \left\lfloor \frac{F(u,v)}{q} \right\rfloor$. In the inverse quantization, depending on the polarity of the index, an addition or subtraction of half the quantization step is required to deliver the central representation as $F^q(u,v) = \{I(u,v) \pm 1\} \times q$.

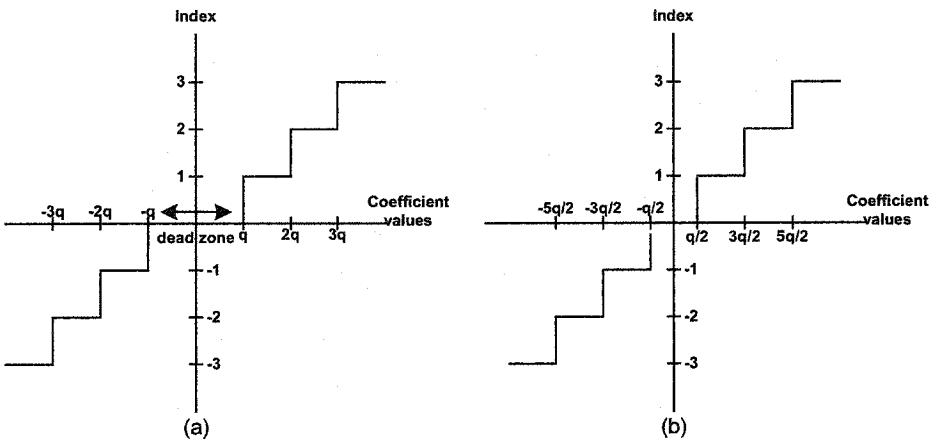


Figure 2-6. Uniform quantizer with (a) and without (b) dead zone.

The design of the quantization process has an important effect on compression performance and image quality. In order to support compatibility between encoders and decoders, the image and video coding specify the *levels* produced by the encoder and the set of reconstructed coefficients. However, they do not specify the quantization process, i.e., the map between the input coefficients and the set of levels. This gives the encoder designer flexibility to design the quantizer to give optimum performance for different applications [76]. In H.263 standard, a quantization scheme based on UTQ is used. The following equations show the quantization and inverse quantization process performed by the H.263 encoder and decoder, respectively. *COF* is the transformed coefficient to be quantized, *LEVEL* is the absolute value of the quantized coefficient and COF' is the

reconstructed coefficient after inverse quantization. QP is called the quantization parameter, and $2 \times QP$ is the actual quantization step size [77].

Quantization

$$LEVEL = \begin{cases} \left\lfloor \frac{COF}{8} \right\rfloor, & INTRA\ DC\ coefficient \\ \left\lfloor \frac{|COF|}{2 \times QP} \right\rfloor, & INTRA\ AC\ coefficient \\ \left\lfloor \frac{|COF| - \frac{QP}{2}}{2 \times QP} \right\rfloor, & INTER\ coefficient \end{cases}$$

Inverse quantization

- *INTRA DC coefficient*:
  $COF' = LEVEL \times 8$
- *INTRA AC or INTER coefficients*:

$$|COF'| = \begin{cases} 0, & if\ LEVEL = 0 \\ 2QP \times LEVEL + QP & if\ LEVEL \neq 0, QP\ is\ odd \\ 2QP \times LEVEL + QP - 1 & if\ LEVEL \neq 0, QP\ is\ even \end{cases}$$

The sign of COF is then added to obtain $COF' = sign(COF) \times |COF'|$.

As we can see, the quantization for INTER coefficients and INTRA AC coefficients are different. The design of this quantization process is based on the distribution of the DCT coefficients. In the INTER frame, most DCT coefficients are clustered about zero, therefore, in this quantization design, the quantizer "bias" the reconstructed coefficients towards zero for INTER coefficients; this means that, on average, the reconstructed values will be close to the original values. This in turn reduces the error introduced by the quantization process. Depending on the source video material, the transform coefficient may have different distributions. It is possible to redesign the quantizer to suit a particular set of input data and achieve optimum image quality [76].

2.1.5 Entropy coding

A video encoder contains two main functions: a source model that attempts to represent a video scene in a compact form that is easy to compress and an entropy encoder that compresses the output of the model prior to storage and transmission. The source model is matched to the characteristics of the input data, whereas the entropy coder may use general-purpose statistical compression techniques that are not necessarily unique in their application to image and video coding. In a typical transform-based video codec, the data to be encoded by the entropy coder fall into three main categories: transform coefficients, motion vectors and syntax information. The method of coding syntax information depends on the standard. Motion vectors can often be represented compactly in a differential form due to the high correlation between vectors for neighboring blocks or macroblocks. Transform coefficients can be represented efficiently with run-length coding, exploiting the sparse nature of DCT coefficient array.

An entropy encoder maps input symbols to a compressed data stream. It achieves compression by exploiting redundancy in the set of input symbols, representing frequently occurring symbols with a smaller number of bits and infrequently occurring symbols with a larger number of bits. The two most popular entropy coding methods used in video coding standards are Huffman coding and arithmetic coding. Huffman coding represents each input symbol by a variable-length codeword containing an integral number of bits. It is relatively straightforward to implement, but cannot achieve optimal compression. Arithmetic coding maps an input symbol into a fractional number of bits, enabling greater compression efficiency at the expense of higher complexity.

Entropy coding used in the H.263 standard consists of three main steps:

Reordering: the non-zero quantized coefficients are clustered around the top left of the block and this step groups these coefficients together. The optimum method of reordering depends on the distribution of the non-zero coefficients. If the original data is evenly distributed in the horizontal and vertical directions, then the significant coefficients will tend to be evenly distributed about the top left corner of the block. In this case a zigzag reordering should group together the non-zero coefficients.

Run-length coding: this step attempts to find more efficient representation for the large number of zeros. The output of the reordering process is a linear array of quantized coefficients. Non-zero coefficients are mainly grouped together near the start of the array and the remaining values in the array are zero. Long sequences of zeros can be represented as a (*run, level*) code, where *run* indicates the number of zeros preceding a non-zero value and level indicates the sign and magnitude of the non-zero coefficient. For the final run of zeros in a block, a special code symbol, "end of block" or EOB, is inserted after the last (*run level*) pair to indicate that there are no non-zero coefficients remaining. In the H.263 standard, a "last" flag is encoded with each (*run, level*) pair; thus, each code now represents three values (*run, level, last*). This "last" flag signifies the final (*run, level*) pair in the block and indicates to the decoder that the rest of the block should be filled with zeros.

Huffman coding: this step maps each (*run, level, last*) symbol into a variable length codeword, which should be uniquely decodable. Ideally, the design of the Huffman coding map should be based on the probability distribution of the symbols in the specific video. However, this has two disadvantages for a practical video codec. First, the decoder must have the same probability distribution table to be able to decode the bit stream.

Second, calculating the probability distribution for a large video sequence is a significant computational overhead. For these reasons, the H.263 standard defines sets of codewords based on the probability distribution of a large range of video material. Since the codeword table is "generic", compression efficiency is lower than that obtained by pre-analyzing the data to be encoded, especially if the sequence statistics differ significantly from the "generic" probability distributions.

2.1.6 Block diagram of the H.263 codec

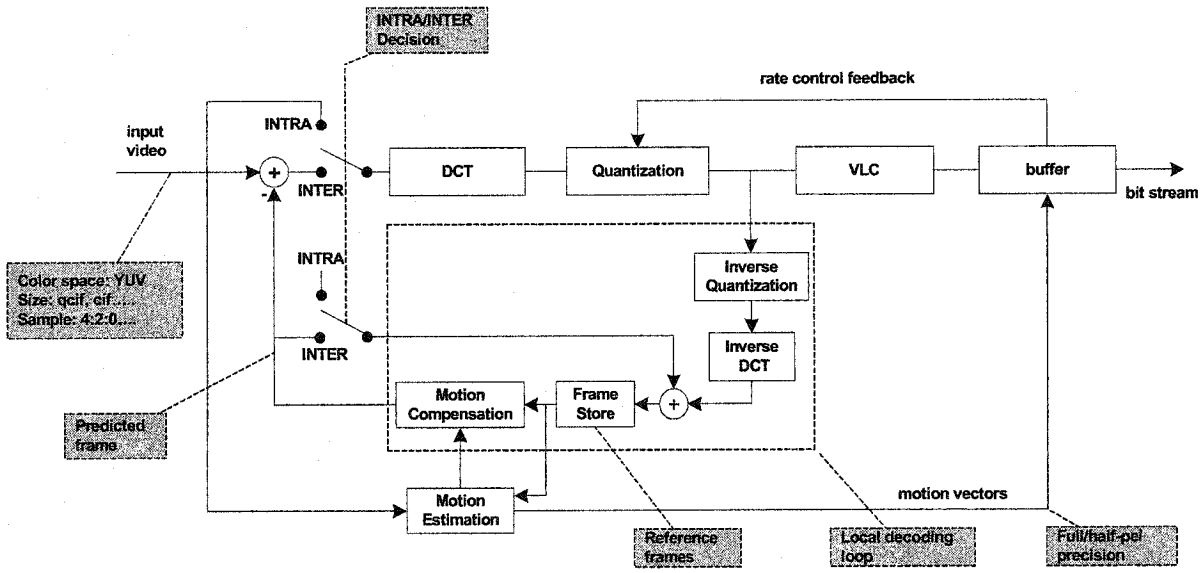


Figure 2-7. Block diagram of H.263 video codec.

Figure 2-7 illustrates a block diagram of H.263 video codec. The INTRA/INTER decision module determines a video frame should be encoded as an INTRA (I) frame or an INTER (P or B) frame. Motion estimation and motion compensation are only applied on INTER frames. For INTER frames, after motion estimation and compensation, the motion information and prediction error are transmitted to the decoder, which reconstructs the predicted frame from the motion information and the decoded reference frames. In this figure, the dashed box encloses the local decoding loop in the encoder.

The function of this decoding loop is to ensure that both the encoder and the decoder will have exactly the same reference frame.

2.2 Video Transcoding Architectures

In this section, we will review several typical transcoding architectures and compare their performance. These architectures are mainly used for bit-rate adaptation. For other purposes, such as spatial size downscaling, error resilience, etc., extra modules need to be inserted into these architectures.

2.2.1 Open-loop transcoding

For fast scaling video bit-rates, some open-loop transcoding architectures have been proposed in the literature. In these architectures, the incoming video streams are variable-length decoded and dequantized without further decoding. Two approaches can be used in the open-loop transcoder for bit-rate scaling:

- Cutting high frequency AC codewords.

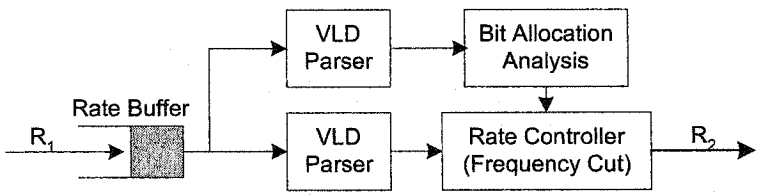


Figure 2-8. Open loop transcoder by cutting high frequencies

As illustrated in Figure 2-8, the incoming precoded stream enters a decoder rate buffer. A variable length decoder is used to parse the bits for a frame in the buffer and identify all the variable length codewords that correspond to AC coefficients used in that frame. No bits are removed from the rate buffer. The codewords are not decoded, but just simply parsed by the VLD parser to determine codeword lengths. The bit allocation analyzer accumulates these AC bit-counts for every macroblock in the frame and creates an AC bit usage profile. A second VLD parser accesses and removes all codeword bits from the

buffer and delivers them to a rate-controller. The rate-controller has memory to store all coefficients associated with the current macroblock. The rate-controller only keeps first part of the AC codewords according to the AC bit usage profile and scaling factor. The other higher frequency AC codewords will be dropped.

- Increasing Requantization step.

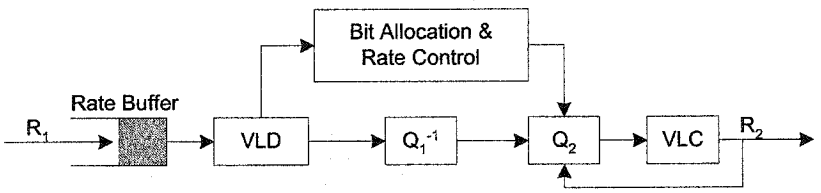


Figure 2-9. Open loop transcoder by increasing the requantization step.

As illustrated in Figure 2-9, this method of bit-stream scaling is based on increasing the requantization step. This method requires an additional dequantizer/quantizer and variable length coding as compared to the first approach. Like the first approach, it also makes a VLD pass on the bit stream and obtains a similar scaled profile of target cumulative codeword bits versus macroblock count to be used for rate-control. After the VLD is made on the bit-stream, quantized DCT coefficients are dequantized. A block of finely quantized DCT coefficients is obtained as a result and is requantized with a coarser quantizer scale. The value used for the coarser quantizer scale is determined adaptively by making adjustments after every macroblock according to the target bit-rate. In both the above approaches, the motion vectors, coding mode and other syntax elements from the incoming video stream are reused by the transcoder.

2.2.2 Pixel domain transcoding

The most straightforward structure for video transcoding is a cascaded pixel domain video transcoder (CPDT) [48], which connects a standard decoder with a standard encoder together as illustrated in Figure 2-10. At the decoder, the motion-compensated

image from the previous frame is added to the decompressed prediction error, which was encoded at a bit-rate R_1 to reconstruct the original image. This reconstructed image is sent to the encoder for compression at a different bit-rate R_2.

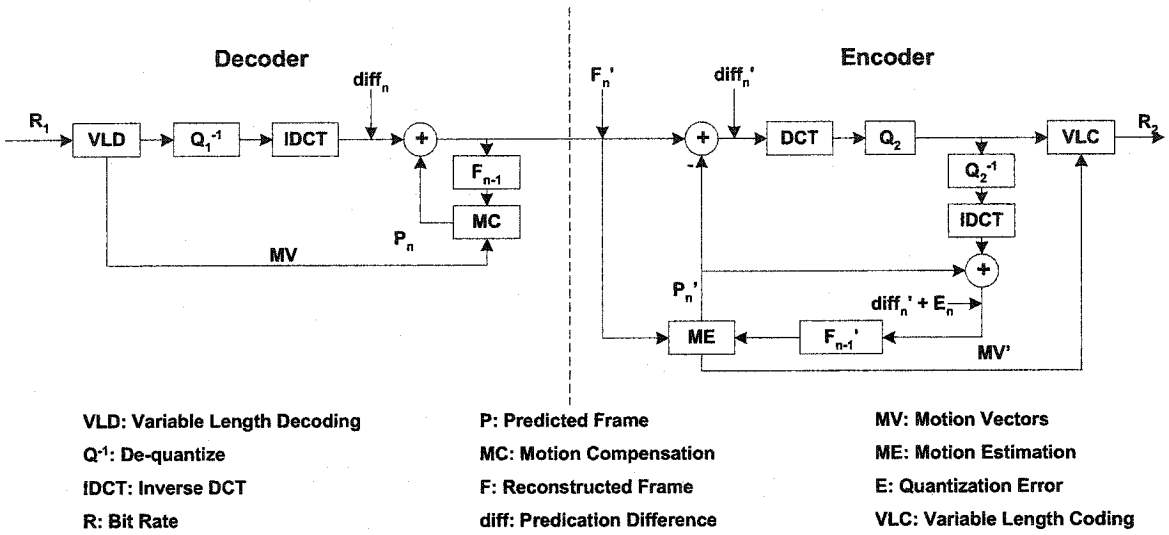


Figure 2-10. Cascaded pixel domain transcoder

If the picture types are the same before and after transcoding in this cascaded architecture, a significant simplification can be achieved by the following steps:

- If the frames are not skipped at the encoder, the decoded motion vectors can be directly reused so that the motion estimation module (ME) in the CPDT can be removed. Since motion estimation is the most time-consuming module in the encoder, removing it will significantly reduce the computational complexity of the transcoder.
- Motion compensation is a linear operation. Assuming *MC(x)* is the function of performing motion compensation to a variable x of a macroblock, then we have

$$MC(x+y) = MC(x) + MC(y) \tag{2-9}$$

Referring to Figure 2-10, we can simplify the architecture as follows:

$$
\begin{aligned}
diff'_{n+1} &= F'_{n+1} - P'_{n+1} = diff_{n+1} + MC(F_n) - MC(P'_n + diff'_n + E_n) \\
&= diff_{n+1} + MC(F_n) - MC(F_n - diff'_n + diff'_n + E_n) \\
&= diff_{n+1} + MC(F_n) - MC(F_n) - MC(E_n) \\
&= diff_{n+1} - MC(E_n)
\end{aligned}
\tag{2-10}
$$

In this equation, n represents the current frame number and $n+1$ represents the following frame number. Thus, the motion compensation module and the frame buffer at the decoder can be discarded. At the encoder, only the second quantization error, E_n, needs to be stored, and motion compensation on quantization error needs to be performed for the following frame.

After the above two steps, the simplified architecture is illustrated in Figure 2-11.

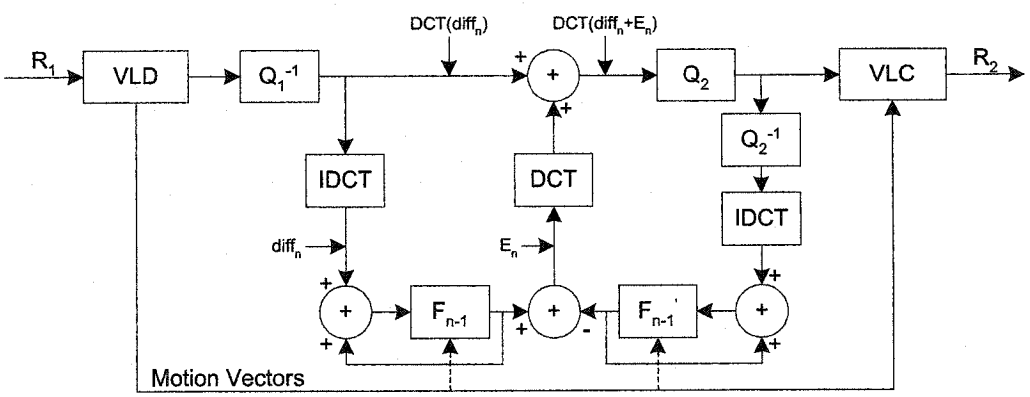


Figure 2-11. Simplified CPDT

In this architecture, we still need to decode the incoming video frame F_{n-1} and keep a copy of a local decoded frame F_{n-1}'. Since the motion estimation is no longer needed and the decoders are much simpler than encoders, the scheme of Figure 2-11 is much simpler than the one depicted in Figure 2-10 [5].

- In Figure 2-11, F_{n-1}' is inverted, hence transcoded DCT coefficients can be inverted prior to IDCT and the resultant $-F_{n-1}'$ added to F_{n-1}. Therefore, both incoming DCT coefficients and inverted transcoded DCT coefficients follow the same decoder and hence they can share it. Since the DCT and IDCT are also linear operations, the

decoder can be further simplified. The simplified architecture is illustrated in Figure 2-12. Please notice here the content of buffer F'_{n-1} is not the reconstructed frame, but the reconstructed requantization error $diff'_n$ [118].

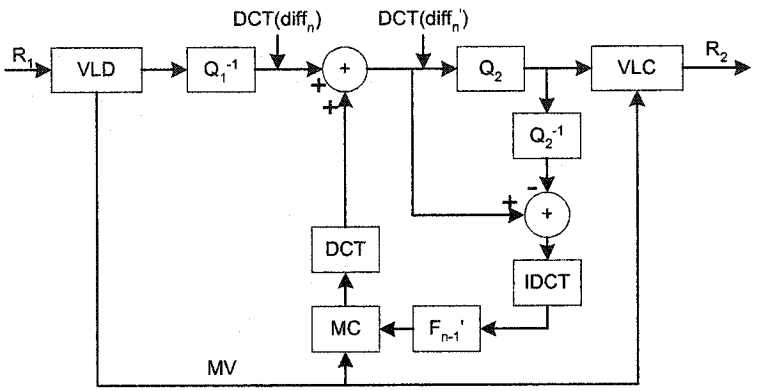


Figure 2-12. Fast pixel domain transcoder

In the above architectures, all other syntax information, such as frame header, passes through the transcoder without any change.

2.2.3 DCT domain transcoding

As can be seen in the simplified architecture derived in the previous section, the whole transcoding process is performed in the DCT domain except the motion compensation loop. In this simplified architecture, the most time-consuming part is in the DCT and IDCT modules. It is highly desirable that these two modules be removed from the architecture.

2.2.3.1 Motion compensation in the DCT domain

The basic motion compensation consists of predicting each 8×8 spatial domain block x of the current frame by a corresponding reference block $\hat{x}$ from a previous frame and encoding the resulting prediction error block $e = x - \hat{x}$ by using the DCT. The best matching reference block $\hat{x}$ may not be aligned to the original 8×8 blocks of the reference frame. As illustrated in Figure 2-13, in general, the reference block may

intersect with four neighboring spatial domain blocks, denoted as x_1, x_2, x_3, and x_4, that together form a 16×16 macroblock.

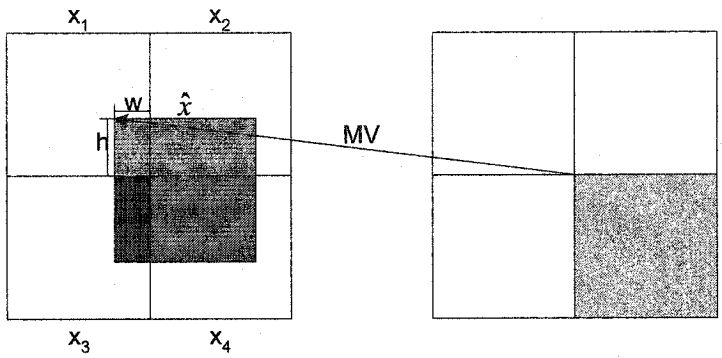


Figure 2-13. illustration of motion compensation

The goal of DCT domain motion compensation is to compute the DCT X of the current block $x = \hat{x} + e$ from the given DCT E of the prediction error e, and the DCT's $X_1, \ldots, X_4$ of $x_1, \ldots, x_4$, respectively. Since $X = \hat{X} + E$, $\hat{X}$ being the DCT of $\hat{x}$, the main problem that remains is that of calculating $\hat{X}$ directly from $X_1, \ldots, X_4$.

Let the intersection of the reference block $\hat{x}$ with x_1 form a $h \times w$ rectangle, where $1 \le h \le 8$ and $1 \le w \le 8$. This means that the intersections of $\hat{x}$ with $x_2, x_3, and\, x_4$ are rectangle of sizes $h \times (8-w), (8-h) \times w$, and $(8-h) \times (8-w)$, respectively. Therefore, the MC block $\hat{x}$ comprises four pixel subblocks, one from each intersected block x_i. These subblocks can be extracted from the respective blocks by multiplying the latter with appropriate matrices that perform window and shift operation as described in [18, 19, 65, 66]. In [19], Chang and Messerschmitt firstly used some geometrical transforms to derive the relationship among the elements of the DCT blocks $\hat{X}$ and $X_1, \ldots, X_4$ from the spatial geometric relationship among the associated pixel-domain blocks $\hat{x}$ and $x_1, \ldots, x_4$ so that the elements of the target block $\hat{X}$ can be computed from the elements of the

neighboring blocks $X_1,....,X_4$ directly in the DCT domain without the need of full decompression of the DCT blocks into the pixel domain. As illustrated in Figure 2-14, in the pixel domain, the target pixel domain block, $\hat{x}$, can be represented as the sum of four component blocks which are denoted as x_{14}, x_{23}, x_{32}, *and* x_{41} corresponding to the four adjacent pixel blocks respectively.

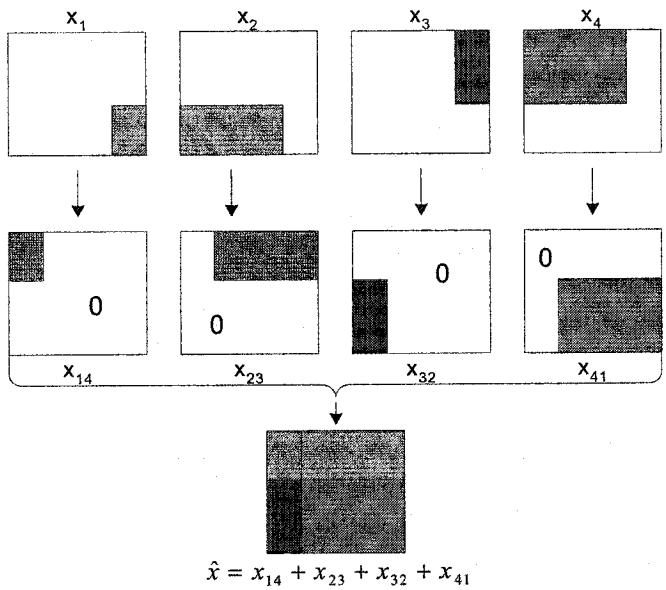


Figure 2-14. Formation of the target block $\hat{x}$ from parts of $x_1 \sim x_4$ through geometrical transform.

Here the pixel block $x_{14}, x_{23}, x_{32}, x_{41}$ are the windowed and shifted version of x_1, x_2, x_3, x_4, i.e., the operation, in the pixel domain, is given by

$$\hat{x} = \sum_{i=1}^{4} h_{hi} \cdot x_i \cdot h_{wi}, \; 1 \le h, w \le 7. \tag{2-11}$$

The matrices h_{hi}, h_{wi} have the structure of u_h, l_w

$$h_{h1} = h_{h2} = u_h = \begin{bmatrix} 0 & I_h \\ 0 & 0 \end{bmatrix}$$

$$h_{w1} = h_{w3} = l_w = \begin{bmatrix} 0 & 0 \\ I_w & 0 \end{bmatrix}$$

and

$$h_{h3} = h_{h4} = u_{8-h}$$

$$h_{w2} = h_{w4} = l_{8-w}$$

where I_h and I_w are identity matrices of size $h \times h$ and $w \times w$ respectively. By applying the distributive property of matrix multiplication with respect to DCT, i.e., $DCT(AB) = DCT(A)DCT(B)$, one can use the DCT matrices $H_{hi} = DCT(h_{hi})$, $H_{wi} = DCT(h_{wi})$ to extract the DCT block $\hat{X} = DCT(\hat{x})$ directly from the DCT blocks $X_i = DCT(x_i)$ of the reference frame, i.e.,

$$\hat{X} = \sum_{i=1}^{4} H_{hi} \cdot X_i \cdot H_{wi}, \quad 1 \le h, w \le 7. \tag{2-12}$$

where H_{hi} and H_{wi} are constant so they can be pre-computed and stored in a memory.

2.2.3.2 Half-pixel precision motion compensation

When an MV with half-pixel precision is used, either two or four pixels are needed to calculate the actual prediction of one single pixel. In terms of blocks, this is equivalent to computing the average, for each pixel, of either two or four blocks. In the DCT domain, this needs the extraction of two or four blocks from the reference frame; hence, Eq.(2-12) is applied either twice or four times to obtain the final predicted block. Since the blocks involved in half-pixel prediction are displaced from each other only by one pixel, the extraction of four different DCT blocks can be avoided by applying a linear filter to the MC-DCT MB reconstructed with the integer component of the motion vector. Vertical

filtering should be used when half-pixel accuracy exists in the vertical direction, while horizontal filtering is used when half-pixel accuracy exists in the horizontal direction.

Let us consider the DCT blocks X_i, i=1,...,4 of an MC-DCT luminance MB ordered according to their location within the MB (top left: X_1, top right: X_2, bottom left: X_3, bottom right: X_4). Suppose also that all of these blocks are spatially adjacent in the pixel domain. The horizontally filtered blocks X_i^h, in the DCT domain, are obtained from the Eq. (2-13), while the vertically filtered blocks X_i^v are obtained from Eq. (2-14).

$$X_i^h = \begin{cases} X_i F_1^h + X_{i+1} F_2^h, & i = 1,3 \\ X_i F_3^h, & i = 2,4 \end{cases} \tag{2-13}$$

$$X_i^v = \begin{cases} F_1^v X_i + F_2^v X_{i+2}, & i = 1,2 \\ F_3^v X_i, & i = 3,4 \end{cases} \tag{2-14}$$

The filter coefficient matrices F_i^h, F_i^v of these equations are the DCT of pixel-domain filters defined in Eq. (2-8). The details of DCT domain motion compensation used in transcoders can be found in [10], including filters for interlaced video. In the case of both horizontal and vertical half-pixel prediction, vertical filtering should follow the horizontal filtering or vice versa.

2.2.3.3 Structure of the DCT domain transcoder

Follow the DCT-domain motion compensation procedures described in the previous section, the DCT and IDCT can be removed from the transcoding architecture in Figure 2-12. The simplified transcoding architecture is illustrated in Figure 2-15.

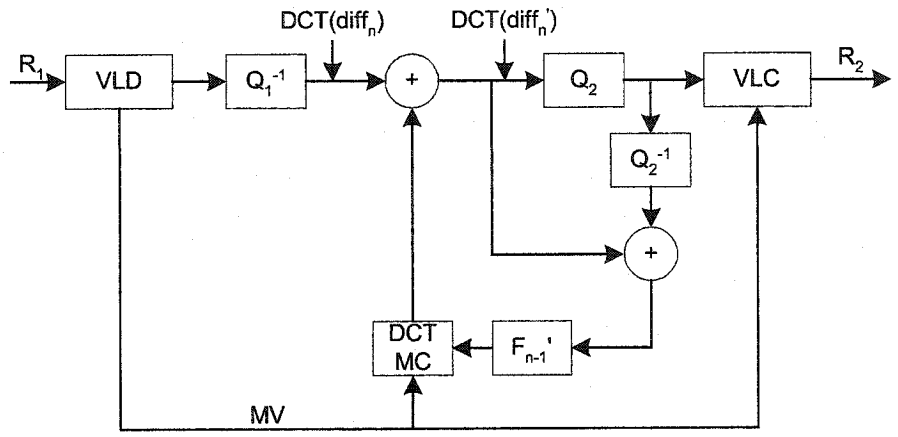


Figure 2-15. DCT domain transcoder

2.2.4 Performance comparison of various transcoding architectures

2.2.4.1 Drift error and visual quality

For bit-rate adaptation, the objective of video transcoding is to convert the video stream into low bit-rate while maintaining low complexity and achieving the highest quality possible. Ideally, the quality of the reduced rate bit-stream should have the quality of a bit-stream directly generated at the reduced rate. This is somewhat difficult, but very close approximations have been reported. In open-loop transcoding architectures, simple transcoding of compressed video into lower bit-rates can be done by the requantization of DCT coefficients or just dropping some of them. However, dropping high frequency coefficients and requantization error will cause the visual quality to degrade. At the same time, drift error is also a severe problem of OLT. The closed-loop architecture eliminates the mismatch between predictive and residual components by introducing a motion compensation loop in the transcoder. However, [120] shows that, in real implementation, small drift error still exists due to several reasons, such as clipping function, precision of half pixel motion compensation, and finite precision of DCT/IDCT transforms, etc. Generally speaking, in terms of drift error, CPDT usually has better visual quality than that of FPDT, DDT and OLT.

In order to compare the performance of different transcoding architectures, we implemented several transcoding architectures as follows:

- **CPDT-ME**: this is a cascaded pixel domain transcoder as illustrated in Figure 2-10. In this architecture, the incoming video stream is fully decoded into the pixel domain. A full-search motion estimation is performed based on the decoded bit stream. Then the decoded bit stream is re-encoded with a new quantization parameter to generate the transcoded bit stream.
- **CPDT-ReuseMV:** this architecture is the same as CPDT-ME, except that the motion vectors from the incoming video stream are reused.
- **CPDT-Refine2:** this architecture is the same as CPDT-ReuseMV, except that the original motion vectors are refined in a small search window [-2, 2].
- **DDT-ReuseMV**: this is a DCT domain transcoder. The original motion vectors are reused without refinement.

In all of the above transcoding architectures, the motion vectors are at the half-pixel precision. In our experiments, two test sequences: "foreman" and "carphone" were used for simulation. Each incoming sequence was encoded with QP_1=3 and transcoded with QP_2=5, 10, 15, 20, respectively. The PSNR comparison results of QP_2=5 are illustrated in Figure 2-16. In this figure, "SNR-Q5" standards for the PSNR result obtained by directly encoding the test sequence with QP=5. This PSNR result is used as a benchmark for other PSNR results. As we can see from the figure, "CPDT-ME", "CPDT-ReuseMV", and "CPDT-Refine2" have almost the same PSNR results. The PSNR result from "DDT-ReuseMV" is worse than that of all pixel domain transcoders, but the difference is very small (less than 0.5 dB). All PSNR results obtained by transcoding is close to that

obtained by directly encoding the test sequence with coarser QP, and the difference is less than 1 dB.

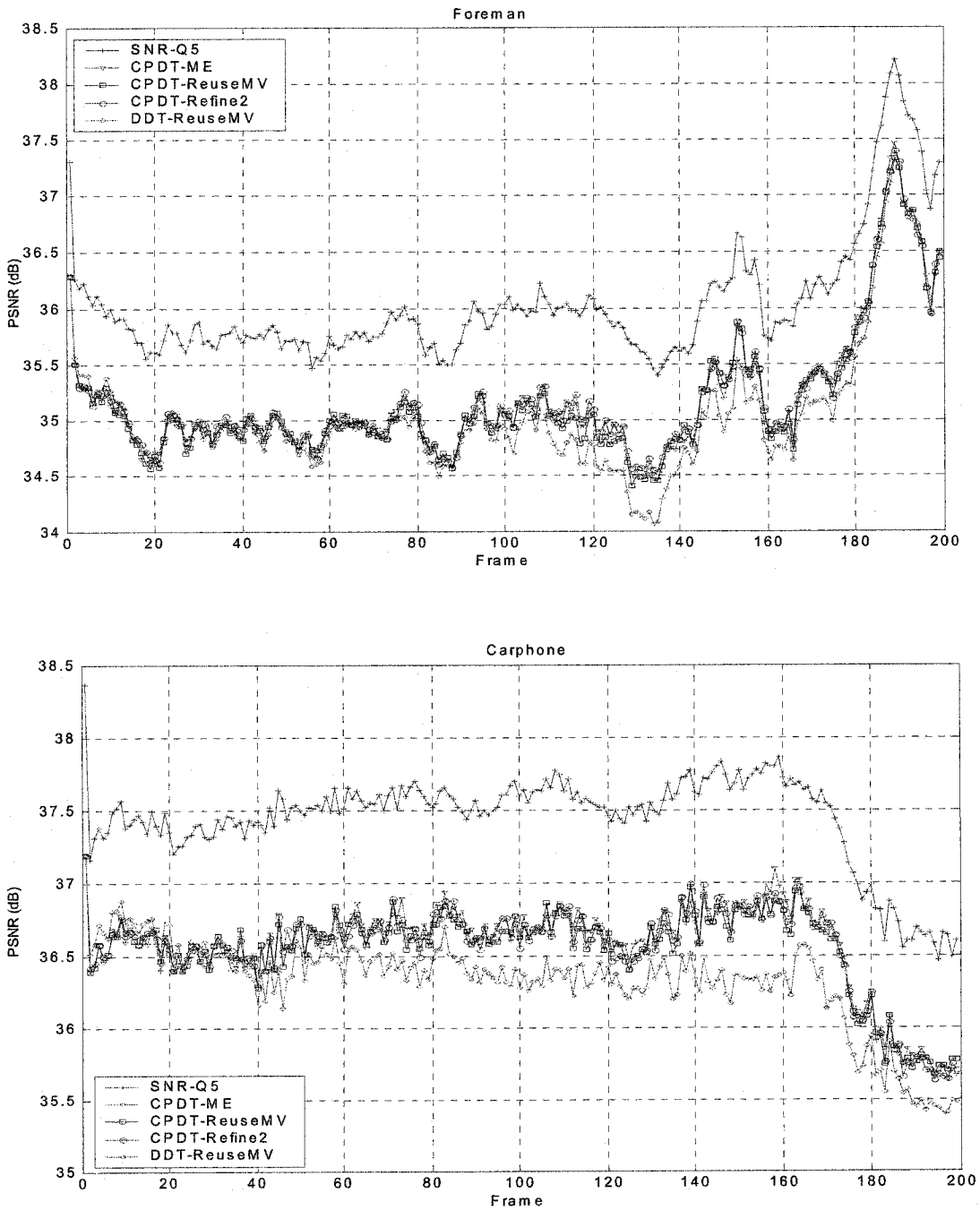


Figure 2-16. PSNR comparison of different transcoding architectures.

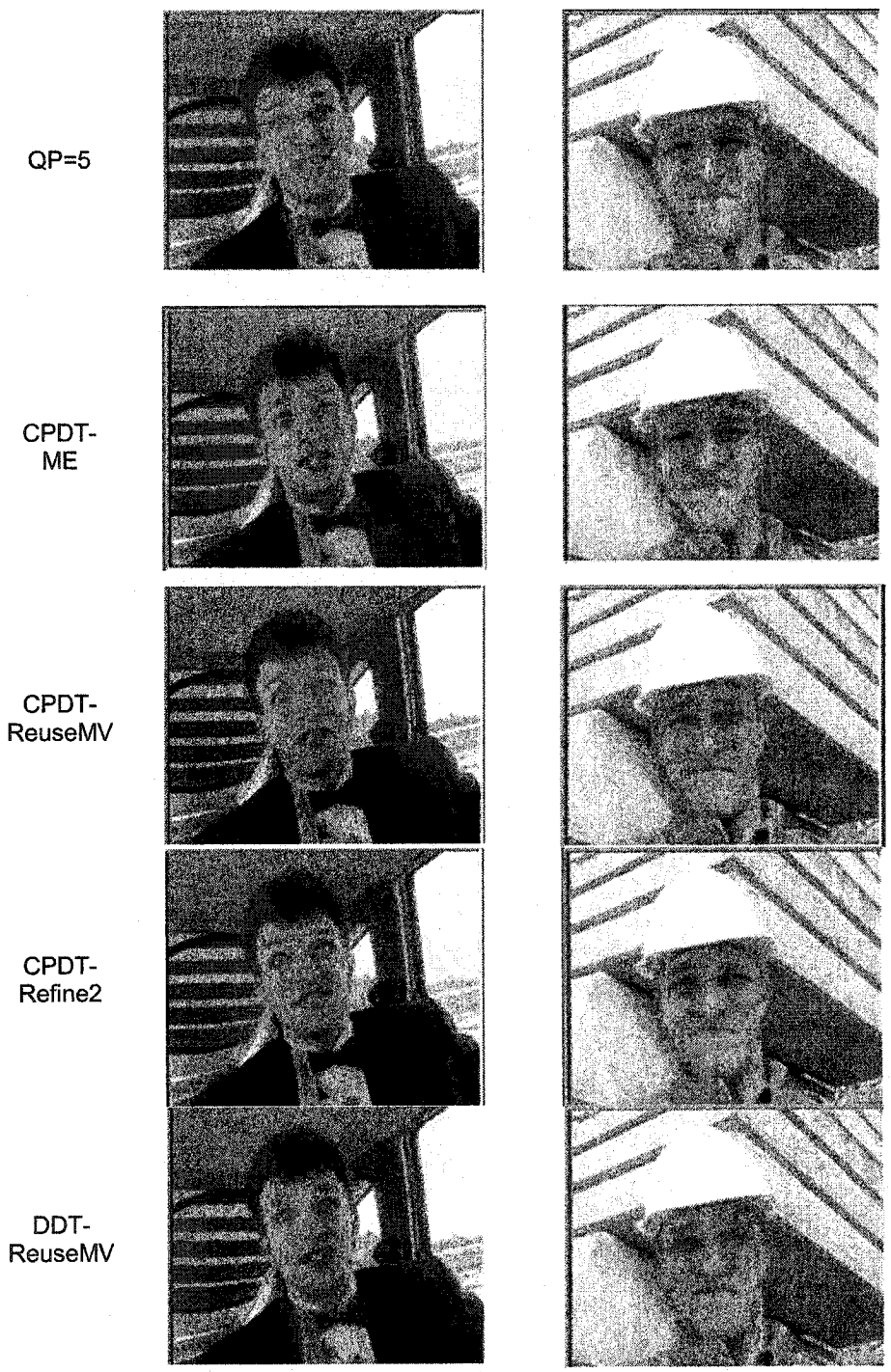


Figure 2-17. Visual quality comparison of encoded and transcoded test sequences.

Figure 2-17 shows the 123th frame of encoded and transcoded "carphone" sequence and the 140th frame of encoded and transcoded "foreman" sequence. As we can see, it is very hard to see any difference between the encoded and transcoded frames with different transcoders.

Table 2-2. Performance comparison of average PSNR and bit-rate with four different transcoders. The incoming sequences were encoded with QP_1=3, and transcoded using QP_2=5, 10, 15, and 20, respectively.

(a) CPDT-ME

Sequence	QP	Average PSNR (dB)	Average Bit-rate (Kbits/s)
Foreman	5	35.17	347.26
	10	31.72	135.41
	15	29.96	81.84
	20	28.72	59.63
Carphone	5	36.57	223.64
	10	32.92	84.51
	15	30.93	49.68
	20	29.56	34.86

(b) CPDT-ReuseMV

Sequence	QP	Average PSNR (dB)	Average Bit-rate (Kbits/s)
Foreman	5	35.16	345.94
	10	31.52	137.70
	15	29.57	83.07
	20	28.13	59.50
Carphone	5	36.54	224.25
	10	32.74	87.89
	15	30.68	53.05
	20	29.17	38.00

(c) CPDT-RefineMV

Sequence	QP	Average PSNR (dB)	Average Bit-rate (Kbits/s)
Foreman	5	35.19	345.39
	10	31.61	135.10
	15	29.75	81.37
	20	28.35	58.44
Carphone	5	36.55	224.26
	10	32.82	86.51
	15	30.82	52.34
	20	29.39	37.46

(d) DDT

Sequence	QP	Average PSNR (dB)	Average Bit-rate (Kbits/s)
Foreman	5	35.04	357.92
	10	31.48	140.75
	15	29.54	84.10
	20	28.14	60.06
Carphone	5	36.32	232.76
	10	32.66	89.82
	15	30.67	54.07
	20	29.19	38.40

Table 2-2 shows the average PSNR values and bit-rate of the transcoded videos using four transcoding architectures for the two test sequences. The resultant PSNR-bit-rate pairs form the plots in Figure 2-18.

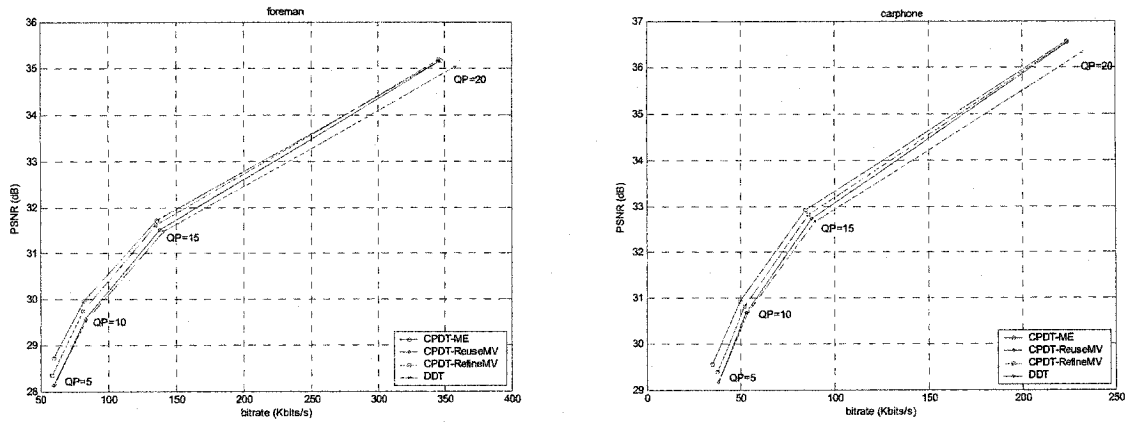


Figure 2-18. Performance comparison of average PSNR with four different transcoders.

2.2.4.2 Complexity

For video transcoding, the complexity of the transcoding architectures is an obvious metric to reduce. From the above introduction of transcoding architectures, we can easily get the idea that the cascaded pixel domain transcoder is the most complicated one in terms of the overall structure. The open loop transcoder is the simplest one. However, if the motion vectors from the incoming video stream are reused for transcoding, a significant part of computation is reduced. From this point of view, reusing motion vectors is the most significant step for reducing transcoders complexity. From the algorithm structure point of view, the DCT domain transcoder has less processing modules than the pixel domain transcoder. Most of the computational complexity of the DCT domain transcoder comes from Eq. (2-12). In fact, the brute-force computation of Eq. (2-12), in the case where the MC block is not aligned in any direction with the block structure, requires six matrix multiplications and three matrix additions. Some methods for reducing the computation involved in Eq. (2-12) have been proposed in [10]. The authors claim that the DCT domain transcoder becomes less complex than its pixel domain counterpart.

2.3 Summary

Video transcoding is an efficient method for rate adaptation and video format conversion in networked video applications. In this chapter, we first overviewed some preliminary knowledge and techniques of video codecs. Then, we discussed various architectures and related issues for implementing video transcoders. A straightforward approach to implement a video transcoder is to cascade a video decoder followed by a video encoder. This cascaded architecture can avoid the drift problem, while its high complexity is unacceptable in real-time applications. Based on a motion vector reuse scheme, a simplified pixel-domain video transcoder achieving significant computation saving can be constructed, though it may not perform as well, in video quality, as the cascaded architecture. The simplified transcoder architecture can also be implemented in the DCT domain without performing the DCT/IDCT computations. Instead, this approach uses an interpolation method to estimate the DCT coefficients of a shifted macroblock. We have implemented several transcoders mentioned above and compared their performance. Our experimental results show that the simplified pixel- and DCT-domain transcoder can achieve computation savings when compared to the cascaded pixel-domain transcoder, while the visual quality of the simplified transcoders is very close to that of the cascaded pixel-domain transcoder. In addition, we have to say that there is always a trade-off between visual quality and complexity. In most cases, the visual quality will be degraded due to simplifying some modules in transcoders. Therefore, choosing which kind of transcoding architectures will be used depends on the dedicated applications and user requirements.

Chapter 3. Model Based Bit Allocation

3.1 Classic Rate-Distortion Theory

Existing video applications require high compression ratios, over an order of magnitude higher than what is typically possible with lossless compression methods. These high levels of compression can be realized only if we accept some loss in fidelity between the uncompressed and compressed representations. There is a natural tradeoff between the size of the compressed representation and the fidelity of the reproduced images. This tradeoff between coding rate and distortion is quantified in rate-distortion theory.

Let D be a measure of distortion according to some fidelity criterion. In classical Rate-Distortion theory, a rate-distortion function, *R(D)*, is defined to be the theoretical lower bound on the best coding rate achievable as a function of the desired distortion D for a given information source, by any compressor. In general, the fidelity criterion can be any valid metric; in practice, Mean Squared Error (MSE) and Mean Absolute Difference (MAD) are often used.

Assuming X is a random variable following a distribution probability density function $f_X(x)$, let $\widetilde{X}$ be the uniform quantized version of X using a quantizer step size Q, and X' be the reconstructed version of X, then the quantization error measured by MSE is:

$$D(Q) = MSE(Q) = E\left[(X - X')^2\right] = \int_{-\infty}^{\infty}(x - x')^2 f_X(x)dx = \sum_{k=-\infty}^{\infty}\int_{kQ-\frac{Q}{2}}^{kQ+\frac{Q}{2}}(x - kQ)^2 f_X(x)dx \quad (3\text{-}1)$$

while the quantization error measured by MAD is:

$$D(Q) = MAD(Q) = E\left[\left|X - X'\right|\right] = \int_{-\infty}^{\infty} |x - x'| f_X(x)dx = \sum_{k=-\infty}^{\infty} \int_{kQ-\frac{Q}{2}}^{kQ+\frac{Q}{2}} |x - kQ| f_X(x)dx \qquad (3\text{-}2)$$

For a discrete source, *R(0)* is simply the entropy of the source and corresponds to lossless coding (*D*=0). In case where the distortion is bounded from above by D_{max}, then $R(D_{max})$=0. Furthermore, it can be shown that *R(D)* is a non-increasing convex function of D [26].

For some specific information sources and distortion measures, closed form expressions for the rate-distortion function have been determined. For example, for a zero-mean i.i.d Gaussian source with variance σ^2, i.e,

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \qquad (3\text{-}3)$$

and a mean squared error distortion measure,

$$R(D) = \begin{cases} \frac{1}{2}\log_2 \frac{\sigma^2}{D}, & if\ 0 \le D \le \sigma^2; \\ 0, & if\ D > \sigma^2. \end{cases} \qquad (3\text{-}4)$$

For a Laplacian source, i.e.,

$$f_X(x) = \frac{\lambda}{2} e^{-\lambda|x|} \qquad (3\text{-}5)$$

and a mean absolute difference distortion measure,

$$R(D) = \log_2\left(\frac{1}{\lambda D}\right) \qquad (3\text{-}6)$$

By combining the *R(D)* and *D(Q)* function, the rate-control algorithm based on rate distortion theory needs to find the quantization parameter *Q*, which minimizes the distortion *D*, while keeping the produced bit-rate lower than *R*. In practice, classical rate-

distortion theory is not directly applicable to complex encoding and decoding systems since sources are typically not well-characterized and *R(D)* is difficult, if not impossible, to determine.

3.2 Rate-Control for Video Coding and Transmission

3.2.1 Model based bit allocation and rate-control

Generic rate-control belongs to the budget-constrained bit allocation problem [46], which is stated below:

> *Given a set of quantizers* $\{Q_1, Q_2, ..., Q_M\}$, *a sequence of N macroblocks, and a target bit budget B, determine an assignment of quantizers* $Q=<Q_1, Q_2, ..., Q_N>$ *to each macroblock that minimizes a distortion measure D and uses* $R \leq B$ *bits.*

In typical transform coding, both *R* and *D* are controlled by quantization parameters of the quantizer. The generic rate-control problem can be separated into the following two steps:

(1) Allocate target bits for each frame according to image complexities and buffer fullness for a given channel bit-rate. This step is usually called *Frame-Layer Rate-control.*

(2) Derive the actual quantization parameter for each macroblock in the picture, and make the number of produced bits meet the bit target. This step is usually called *Macroblock-Layer Bit Allocation.*

In this chapter, we only focus on the macroblock-layer bit allocation problem. For this problem, methods based on Lagrangian optimization [23, 54, 58, 69, 72, 105] or Dynamic Programming [68, 112] have been considered in the literature to solve it. These methods typically perform a pre-analysis of future video frames to measure their Rate-Distortion characteristics before applying a rate allocation strategy. The advantage of the

R-D techniques is that they can achieve better performance in video quality. The drawback is the increase of encoder complexity. If frame dependences are taken into account, the complexity can become very high, as increasing numbers of *R-D* operating points have to be measured, thus making some of these methods only suitable for off-line encoding [59].

An alternative approach for reducing the complexity of *R-D* techniques is to rely on the *R-D* model introduced in the previous section. It is able to predict the compressed bits when a certain quantization step size is in use before the real quantization and VLC are actually applied to. There are two approaches in constructing a bit production model: 1) the analytic approach that constructs a mathematical model based on the information theory and 2) the empirical approach that derives the input/output relationship of a coder based on the observed data.

As introduced in the previous section, for the first approach, the distribution of the *DCT* coefficients has to be assumed known. In [42], the authors have provided the derivation of R-D functions for uniform, Gaussian, Laplacian distributed signals quantized with step size Q. Based on rate distortion theory, different forms have been proposed to approximate the classical *R-Q* formula. In [42], Hang and Chen proposed an exponential *R-Q* model with two parameters based on the formulas of R and D for uniform, Gaussian, and Laplacian distributions. However, those formulas are only accurate for medium and high bit-rates. In [27], Ding and Liu proposed another exponential model with the third parameter to control the curvature of the function. In [87], Tao *et al.* proposed a logarithmic model applying a theoretical *R-D* function. In [22], Chiang and Zhang use a quadratic bit allocation model, etc. The theoretical bit-allocation models based on rate

distortion theory have two assumptions. First, the DCT coefficients are uncorrelated and have a closed form distribution function [42]. Second, the coding bit-rate is approximately equal to the entropy. However, the first assumption is invalid for DCT coefficients in the motion compensated difference frame. Closed-form expressions of entropy and distortion for arbitrary probability distributions are generally unavailable. In transform coding of image or videos, especially at very low bit-rate, there is a large mismatch between the theoretical entropy and the actual coding bit-rate [45]. Additionally, complex formulas would likely make the optimization computationally expensive.

In the second approach, bit production models are derived based on test data. In [71], Puri *et al.* select the quantization step size of an image block according to its activity measure, variance of all DCT coefficients, and a pre-trained table. The entries in this pre-trained table are computed from the statistics of training images coded using the same coders. The activity of a frame can also be measured by the sum of all the DCT coefficients' absolute values with or without DC component, etc. Such as in [20], an empirical first-order bit model indicates that the number of coded bits is approximately proportional to the sum of the AC coefficients absolute values and is inversely proportional to the quantization step size. In [45], a bit allocation model describes the relationship between the produced bits and the percent of zeros in a macroblock.

3.2.2 Standard rate-control algorithms

Current video coding standards, such as H.26x, MPEGx, did not define what rate-control algorithm should be used. This is left open for codec providers to use different rate-control algorithms. However, these standards define some constraints that have to be met by the coded bit-streams. For example, in MPEG2, a Video Buffer Verifier (VBV) was

defined as a hypothetical decoder, which is conceptually connected to the output of an encoder. It is required that MPEG2 compliant coded bit-streams should not cause the VBV buffer to overflow. In the reference software for current coding standards, some well-known rate-control algorithms have been adopted as Test Model rate-control algorithms. In the following, we provide a brief review of three rate-control algorithms and the rate distortion models they used.

3.2.2.1 TM5 rate-control algorithm

The Test Model 5 (TM5) rate-control algorithm is designed for bit-rate-control in MPEG-2 video coding. In the MPEG2 coding standard, the input video sequence is segmented into groups of pictures (GOP). The first frame of each GOP is intracoded and called I-frame. The rest are either predicted coded frames (P-frames) or bidirectionally interpolated frames (B-frames). The TM5 rate-control algorithm consists of two major steps. In the first step, the target bit-rate for each video frame inside the GOP is obtained by a frame layer rate-control scheme. In the second step, the quantization parameter for each macroblock is determined from the buffer status and the spatial activity of the macroblocks.

The whole rate-control algorithm is based on the following assumptions:

1. the distortion D increases linearly with the quantization parameter Q.
2. to maintain a consistent video presentation quality, the average quantization parameters for I, P, B frames, denoted as Q_I, Q_P, Q_B, are related by

$$\frac{Q_I}{1.0} = \frac{Q_P}{K_P} = \frac{Q_B}{K_B} \tag{3-7}$$

   where K_P, K_B are constant and they are set to be 1.0 and 1.4 by default, respectively.
3. the coding bit-rate R is inversely proportional to the distortion D. In other words,

$$R \cdot D = constant$$

At the frame layer, after each picture is coded by using allocated target-bits, the respective complexity (X_I, X_P, X_B) , which is the relative complexity measure of three kinds of previous picture and used in allocating target-bits for the next picture, is calculated as follows.

$$X_I = S_I \cdot Q_I, X_P = S_P \cdot Q_P, X_B = S_B \cdot Q_B \tag{3-8}$$

where S_I, S_P, S_B are the number of bits generated after encoding each picture and Q_I, Q_P, Q_B are average quantization parameters computed by averaging actual quantization values used in coding of all the macroblocks for each type of picture. Target bits for each picture in the GOP are calculated as follows:

$$T_I = max\left\{\frac{R}{1+\frac{N_P \cdot X_P}{X_I \cdot K_P}+\frac{N_B \cdot X_B}{X_I \cdot K_B}}, \frac{bit\ rate}{8 \times picture\ rate}\right\}$$

$$T_P = max\left\{\frac{R}{N_P+\frac{N_B \cdot X_B \cdot K_P}{X_P \cdot K_B}}, \frac{bit\ rate}{8 \times picture\ rate}\right\} \tag{3-9}$$

$$T_B = max\left\{\frac{R}{N_B+\frac{N_P \cdot X_P \cdot K_B}{X_B \cdot K_P}}, \frac{bit\ rate}{8 \times picture\ rate}\right\}$$

where N_P and N_B are the number of P-pictures and B-pictures remaining in the current GOP. R is the bit budget for the remaining frames.

At the macroblock layer, before encoding j-th macroblock, the virtual buffer fullness is computed as follows.

$$d_j^{I,P,B} = d_0^{I,P,B} + B_{j-1} - \frac{T_{I,P,B} \times (j-1)}{MB_{cnt}} \tag{3-10}$$

where $d_0^{I,P,B}$ is the initial virtual buffer fullness for corresponding picture types. B_{j-1} is the number of bits generated after encoding the previous j-1 macroblocks, and MB_{cnt} is the total number of macroblocks in a frame. The buffer fullness $d_j^{I,P,B}$ is then fed back and used to adjust the quantization step size for the j-th macroblock as

$$Q_j = \frac{d_j^{I,P,B} \times 31 \times picture\,rate}{2 \times bit\,rate} \tag{3-11}$$

There is a known problem with TM5. The target bit for an I-picture, P-pictures, and B-pictures are set based upon only the information obtained from encoding of previous pictures. This is not proper for a picture in a new scene that differs significantly from the previous scene. In addition, it can be seen that extremely simplified R-D models are used in the TM5 algorithm. Therefore, it can not achieve accurate rate-control.

3.2.2.2 TMN8 rate-control algorithm

The Test Model Near term 8 (TMN8) rate-control algorithm was designed for H.263 video coding standard. Since H.263 was designed for low delay video communication, I frame and B frames are seldom used, which is different from the MPEG-2 standard. At the frame layer, the target number of bits for the current frame (P frame) is determined as follows.

$$B = \frac{R}{F} - \Delta \tag{3-12}$$

where R and F are the channel and frame rate, respectively. Δ is defined as:

$$\Delta = \begin{cases} \dfrac{W}{F}, & W > Z \cdot M \\ W - Z \cdot M, & otherwise \end{cases} \tag{3-13}$$

where W is the encoder buffer fullness, Z is set to 0.1 by default. Δ is a small value that provides feedback from the buffer fullness W. If W is larger than 10% of the maximum M, the target B is slightly decreased. Otherwise, B is slightly increased. Δ correction will help maintain a small number of bits in the buffer without underflowing the buffer. After the current frame is encoded, the buffer fullness is updated as:

$$W = max(W_{prev} + B' - \frac{R}{F}, 0) \tag{3-14}$$

where B' is the actual number of bits used for encoding the frame, W_{prev} is the previous buffer fullness. If W is larger than or equal to the maximum value M, the encoder skips encoding frames until the buffer fullness is below M. For each skipped frame, the buffer fullness is reduced by an additional R/F bits.

At the macroblock layer, the coding statistics of previous macroblocks are utilized to update the model parameter for the current macroblock. The TMN8 algorithm is based on the following logarithmic R-D model,

$$R(Q) \approx H(Q) = \begin{cases} \frac{1}{2} log_2(2e^2 \frac{\sigma^2}{Q^2}), & if \ \frac{\sigma^2}{Q^2} > \frac{1}{2e}, \\ \frac{e}{ln2} \frac{\sigma^2}{Q^2}, & if \frac{\sigma^2}{Q^2} < \frac{1}{2e}, \end{cases} \tag{3-15}$$

where H(Q) is the empirical entropy of the Q-quantized coefficients, and σ^2 is the variance of the DCT coefficients. This model is a modified version of the classical R-D formula, and assumes that the DCT coefficients of the motion-compensated difference frame are approximately uncorrelated and Laplacian distributed with variance σ^2. The distortion introduced by quantization is modeled as:

$$D(Q) = \frac{1}{N} \sum_{i=1}^{N} \alpha_i^2 \frac{Q_i^2}{12} \tag{3-16}$$

where N is the number of macroblocks in a frame and α_i is the distortion weight of the *i*-th macroblock. By using Largrangian multiplier, the optimized quantization step size for every macroblock can be obtained as:

$$Q_i^* = \sqrt{\frac{AK}{B-ANC}\frac{\sigma_i}{\alpha_i}\sum_{k=1}^{N}\alpha_i\sigma_i}, \qquad i=1,....,N. \tag{3-17}$$

where A is the number of pixels in a macroblocks, K and C are model parameters. The details of TMN8 rate-control algorithm can be found in [25]. Similar to other models based on information theory, this model assumes the DCT coefficients are uncorrelated and Laplacian distributed. Actually, there is usually some correlation left among the DCT coefficients. On the other hand, at the lower bit-rate, the syntax overhead will be a significant part of the frame bit budget. The estimation of the bit budget for encoding syntax overhead information has to be considered.

3.2.2.3 VM8 rate-control algorithm

In [22], Chiang and Zhang proposed a quadratic formation of the rate distortion function. The algorithm based on this model is adopted by MPEG4 in the Verification Model (VM) 8. By assuming the DCT coefficients are Laplacian distributed and using the mean absolute difference as distortion measure, the following rate distortion function is obtained:

$$R(Q) = \frac{X_1 * MAD}{Q} + \frac{X_2 * MAD}{Q^2} \tag{3-18}$$

where MAD is the mean absolute difference between the original frame and the reconstructed frame, which is an indication of encoding complexity, Q is the quantization parameter of the frame, and X_1, X_2 are the first and second order model parameters. After a video frame is encoded, the average quantization parameter and the total coding bit-rate

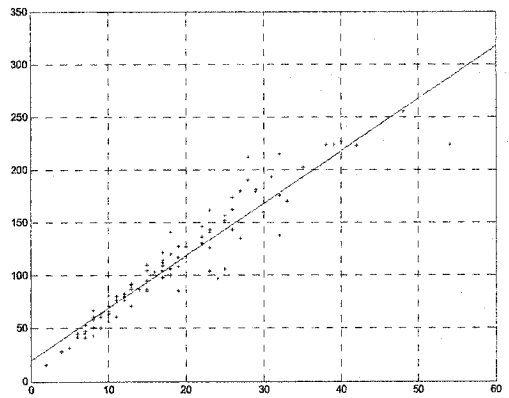
are known. Such coding statistics of a number of previous frames are then used to estimate the model parameters X_1 and X_2 for the current frame. Similar to TM5, VM8 suffers from severe performance degradation at scene changes. Besides this, the VM8 algorithm also suffers from relatively large control error due to the limited accuracy and robustness of its rate model.

3.3 Linear Bit Allocation Model

3.3.1 Experimental studies

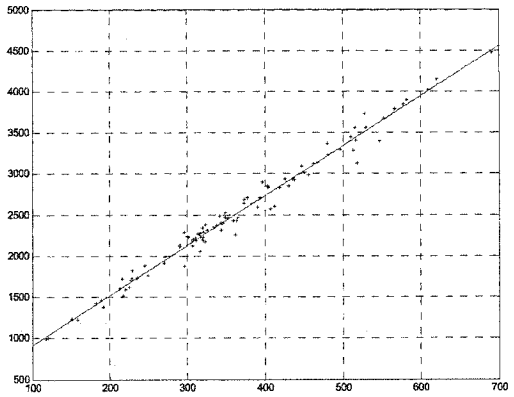
Through our experiments, we found out that the relationship between the produced bit count and the number of VLC code words in a frame or a macroblock can be approximated by a linear function. Based on the quantization scheme in H.263, as we have explained in Chapter 2, only the INTRA AC coefficients (INTRA MBs) and INTER coefficients (INTER MBs) are quantized using quantization parameter QP. INTRA DC coefficients are divided by a constant to generate the quantization indices, which has nothing to do with the selection of the quantization parameter. Therefore, in order to treat the coefficients differently, we define code words as the non-zero quantized indices of INTRA AC and INTER coefficients. In our experiments, we encode several test sequences with a series of quantization parameters (QP=5, 10, 15, 20, 25, 30) with an H.263 video codec [88]. Let $\hat{B}$ be the number of produced bits for encoding codewords in every frame, and C the number of the code words in every frame. In Figure 3-1, we plot $(\hat{B}, C)$ for several test cases. We also use linear curve fitting to obtain the estimated linear functions for every test cases, which are also listed in this figure.

Foreman (QCIF, QP=5, 99 P Frames)



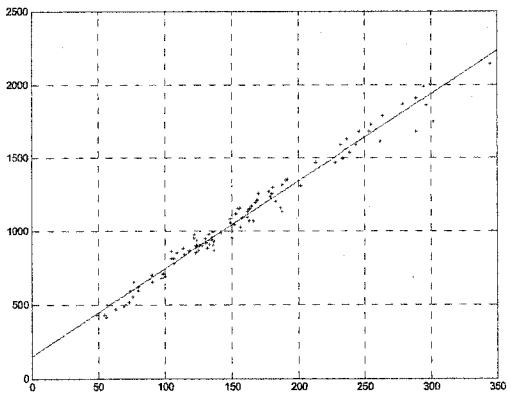
$$\hat{B} = 6.09 \times C + 440$$

Foreman (QCIF, QP=10, 99 P Frames)



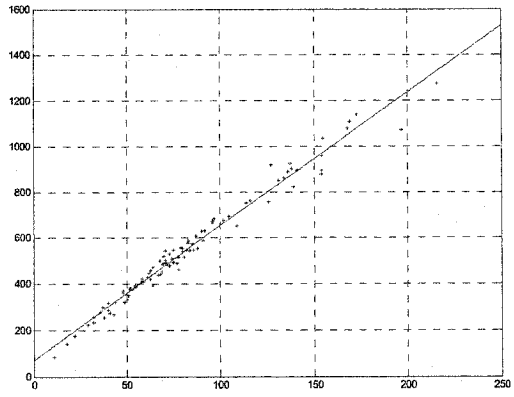
$$\hat{B} = 6.08 \times C + 298$$

Foreman (QCIF, QP=15, 99 P Frames)



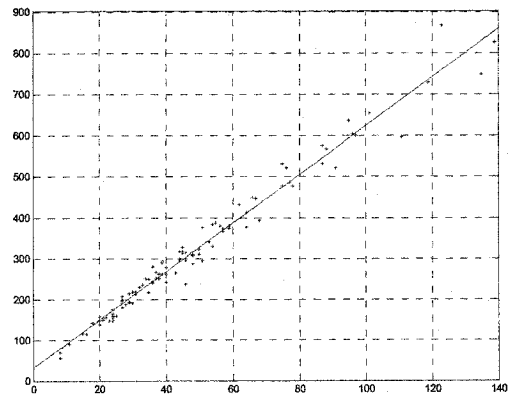
$$\hat{B} = 5.96 \times C + 145$$

Foreman (QCIF, QP=20, 99 P Frames)



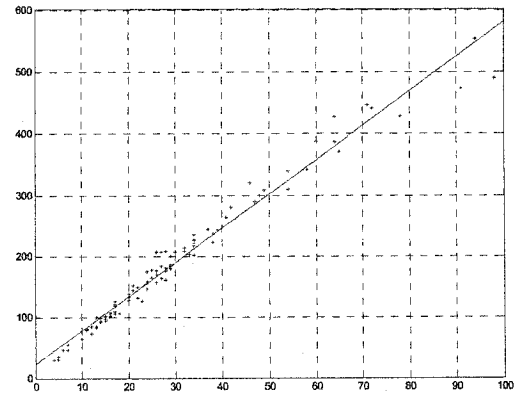
$$\hat{B} = 5.84 \times C + 68.1$$

Foreman (QCIF, QP=25, 99 P Frames)



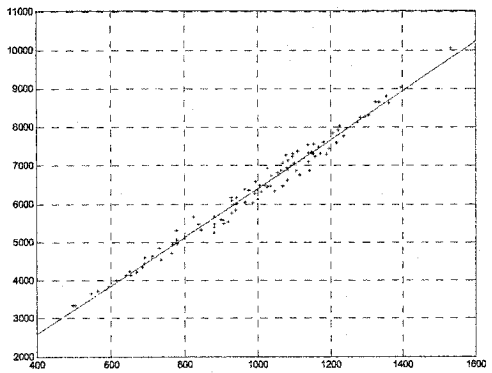
$$\hat{B} = 5.92 \times C + 31.6$$

Foreman (QCIF, QP=30, 99 P Frames)



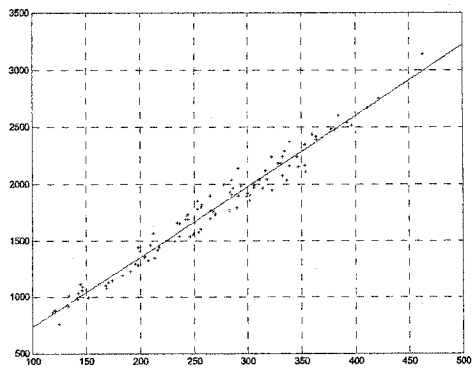
$$\hat{B} = 5.58 \times C + 22.7$$

Carphone (QCIF, QP=5, 99 P Frames)



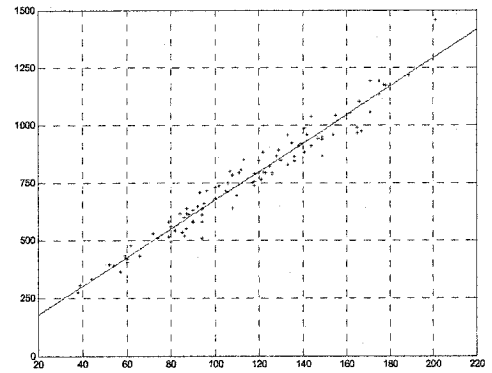
$$\hat{B} = 6.37 \times C + 24.6$$

Carphone (QCIF, QP=10, 99 P Frames)



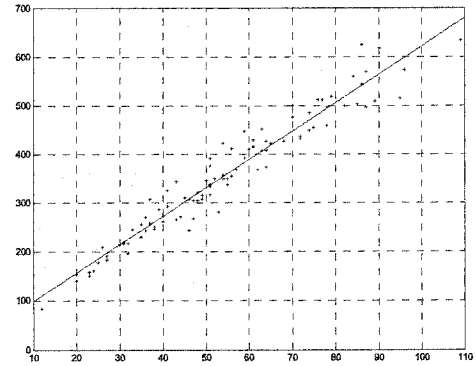
$$\hat{B} = 6.23 \times C + 106$$

Carphone (QCIF, QP=15, 99 P Frames)



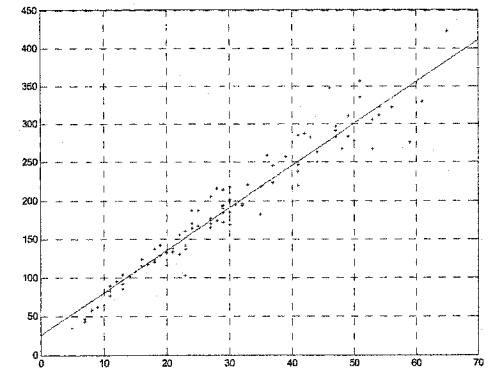
$$\hat{B} = 6.2 \times C + 53.7$$

Carphone (QCIF, QP=20, 99 P Frames)



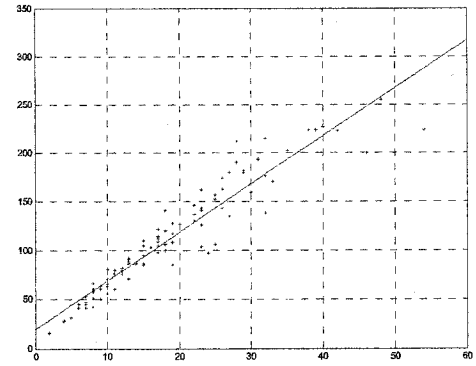
$$\hat{B} = 5.82 \times C + 40.2$$

Carphone (QCIF, QP=25, 99 P Frames)



$$\hat{B} = 5.51 \times C + 25.2$$

Carphone (QCIF, QP=30, 99 P Frames)



$$\hat{B} = 4.97 \times C + 19.3$$

Figure 3-1. the relationship between $\hat{B}$ and C, with different test sequences and quantization parameters. In this figure, the x-axis is the number of code words (x10^3), the y-axis is the number of bits (x10^3) used to encode these code words.

It can be seen that, at the frame layer, $\hat{B}$ is approximately a linear function of C, and this linear relationship is independent of test sequences and quantization parameters. We also notice that, at the lower bit-rate (larger QP), the approximate linear relationship still holds. To examine this approximate linear relationship, we calculate the Pearson's Correlation Coefficient $S(C,\hat{B})$, of C and $\hat{B}$, which is defined as:

$$S(C,\hat{B}) = \frac{\sum_{i=1}^{L}(C_i \cdot \hat{B}_i)}{\sqrt{\sum_{i=1}^{L} C_i^2 \cdot \sum_{i=1}^{L} \hat{B}_i^2}}$$

The value of Pearson's Correlation Coefficient varies from 0 (random relationship) to 1 (perfect linear relationship) or -1 (perfect negative linear relationship). In Table 3-1, we list the Pearson's Correlation Coefficient values of different test cases.

Table 3-1. Correlation coefficients of each test case

	Foreman	Carphone	Container
QP=1	0.9994	0.9997	1.0000
QP=5	0.9997	0.9996	0.9999
QP=10	0.9987	0.9988	0.9997
QP=15	0.9976	0.9978	0.9996
QP=20	0.9970	0.9962	0.9991
QP=25	0.9964	0.9935	0.9986
QP=30	0.9952	0.9875	0.9989

From Table 3-1, we can see that the correlation coefficient is very close to 1, which means the relationship between C and $\hat{B}$ is an approximate linear relationship.

This linear relationship can also be examined by how the DCT coefficients in a macroblock or a frame are quantized and entropy coded. As we know, in the H.263 standard, every macroblock includes 6 blocks — 4 Luminance blocks and 2 Chrominance blocks. After DCT, most energy of a block is compacted into the low frequency part of

the block. The DCT coefficients are further quantized to achieve more compression. The quantization scheme in H.263 is illustrated as follows:

$$LEVEL = \begin{cases} \left\lfloor \frac{COF}{8} \right\rfloor, & INTRA\ DC\ Coefficient \\ \left\lfloor \frac{|COF|}{2 \times QP} \right\rfloor, & INTRA\ AC\ Coefficient \\ \left\lfloor \frac{|COF| - QP/2}{2 \times QP} \right\rfloor, & INTER\ Coefficient \end{cases} \tag{3-19}$$

where *LEVEL* is the quantized indices of DCT coefficients, *QP* is the quantization parameter, and *COF* is the DCT coefficient. For INTRA AC and INTER coefficients, the quantization results are truncated to the nearest integer. During entropy coding, indices of the quantized coefficients in INTER and INTRA mode macroblocks will be variable length coded according to the VLC table, which is designed based on the probability of codewords. In H.263, a 3-dimensional variable-length coding table is used to encode (*LAST, RUN, LEVEL*) of each non-zero coefficient, where RUN is the number of successive zeros preceding the coefficient, LEVEL shows the non-zero value of the quantized coefficient, and LAST indicates if the coefficient is the last non-zero coefficient in a block. The less probable combinations of (*LAST, RUN, LEVEL*) are coded with a 22-bit fixed-length codeword. Part of the variable-length codes is given in Table 3-2 [41].

Table 3-2. Variable-Length code table for transform coefficients

INDEX	LAST	RUN	|LEVEL|	BITS	VLC CODE
0	0	0	1	3	10s
1	0	0	2	5	1111s
2	0	0	3	7	010101s
.....	.....	.....	.....	.....	.....
11	0	0	12	12	00000100000s
12	0	1	1	4	110s
.....	.....	.....	.....	.....	.....
58	1	0	1	5	0111s
59	1	0	2	10	000011001s
60	1	0	3	12	00000000101s
61	1	1	1	7	001111s
.....	.....	.....	.....	.....	.....
102	ESCAPE			7	0000011
Examples of 22-bit fixed-length codeword (7 bits ESCAPE, 1 bit LAST, 6 bits RUN, and 8 bits LEVEL).					
	0	0	13	22	
	0	1	7	22	
	1	0	4	22	

Note: the last bit "s" denote the sign of the level, "0" for positive and "1" for negative

Figure 3-2 illustrates the quantization result for the same DCT coefficients in a INTER block with different quantization parameters, *QP*. Here, *C* is the number of the non-zero codewords, and *B* is the number of produced bits for variable length encoding these codewords.

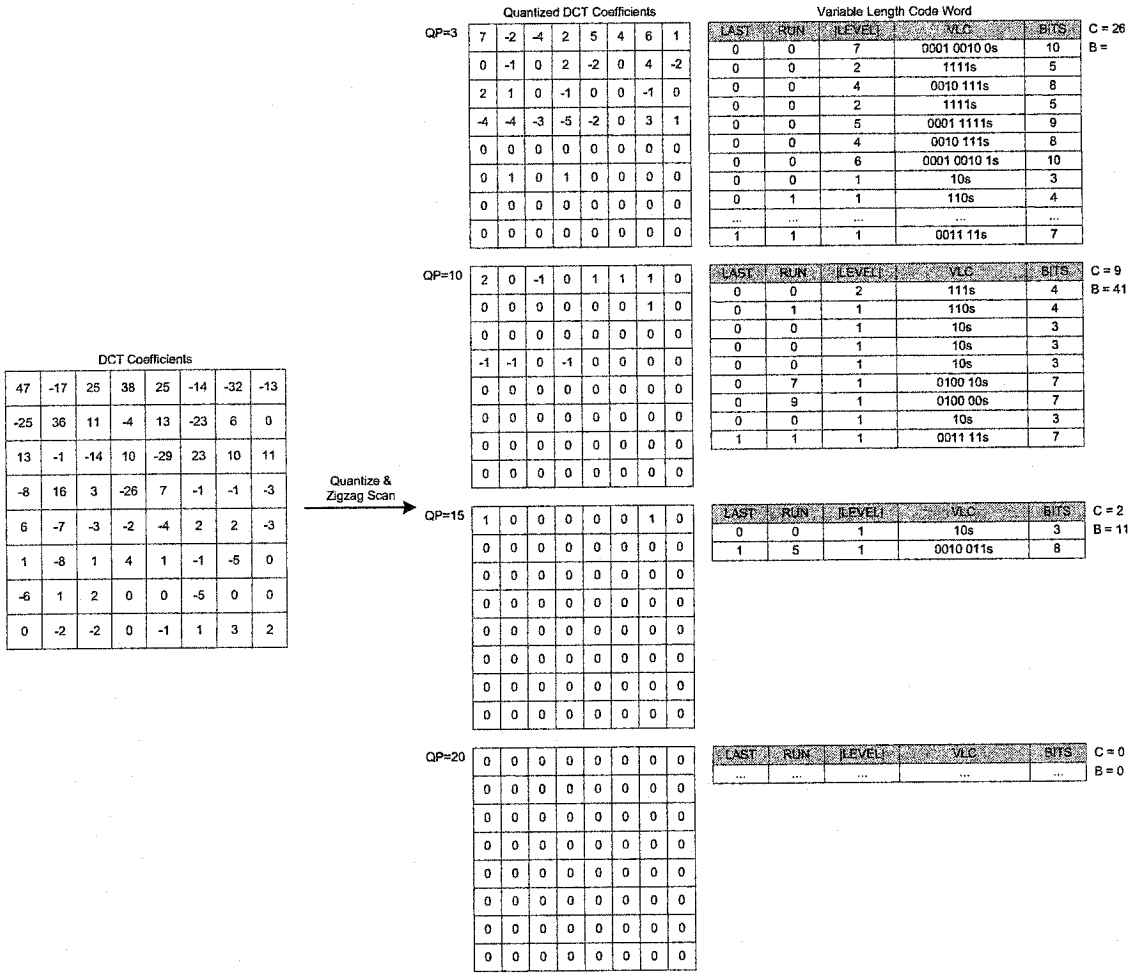


Figure 3-2. Relationship between the number of code words and the generated bits count.

If we know the probability distribution of every possible code word and the bit count for encoding this code word, we can calculate the average length of the Table 3-2, i.e.,

$$\overline{\alpha} = \sum_{i=1}^{N} p_i L_i \tag{3-20}$$

where p_i is the probability of the *i_th* codeword, N is the total number of code words in the VLC table, and L_i is the bit count for encoding the code word [36]. If we define α_i as the average bit count that is used to encode the code words in a macroblock, then the total bit count for encoding a macroblock is:

$$B_i = \alpha_i C_i(QP_i) + \beta_i \tag{3-21}$$

where i is the index of the macroblock, $C_i(QP_i)$ is the number of code words with quantization parameter QP_i , β_i is the bit count for coding the overhead information, which includes the INTRA DC coefficients (if MB is INTRA mode), Motion Vectors, and some other syntax information. Similarly, at the frame level, if we define α as the average bit count for encoding a frame, we can get the total bit count for encoding a frame is

$$B = \alpha C + \beta = \sum_{i=1}^{N} B_i = \sum_{i=1}^{N} [\alpha_i C_i(QP_i) + \beta_i] \tag{3-22}$$

where C is the total number of code words in a frame.

According to Eq. (3-19), for a specific quantization parameter QP, the number of code words in a MB is:

$$C_i(QP) = \sum_{k=1}^{K} \sum_{n=0}^{63} C_i(k,n,QP) \tag{3-23}$$

where

$$C_i(k,n,QP) = \begin{cases} 1, & |COF| \geq 2.5QP \text{ for INTER MB, or } |COF| \geq 2QP \text{ for INTRA AC}; \\ 0, & \text{otherwise}; \end{cases},$$

COF is the n_th AC coefficient in the k_th block of a MB, K (=6) is number of blocks in a macroblock. As mentioned before, the objective of rate-control is to determine the quantization parameter, QP, for every macroblock. As we can see from Eq. (3-23), there exists a one-to-one mapping between the number of code words, C_i, in a MB and the quantization parameter, QP_i, of this MB. Therefore, if we know the target number of code words for a macroblock, we can easily get the quantization parameter for this macroblock by looking up this mapping table. Different from the rate-control models adopted in standards, which directly create a closed form formula between rate and

quantization parameters, the linear bit allocation model creates a relationship between rate and the number of code words. The one-to-one mapping between the number of code words and the actual quantization parameters are then created by pre-analyzing the DCT coefficients of the frame. Because the pre-analysis can get more accurate estimation of the number of produced bits, this linear model is more accurate than other models introduced before.

3.3.2 Theoretical justification

In DCT-based image/video coding, [53] have mathematically derived that the DCT coefficients have a Laplacian distribution given by

$$p(x) = \frac{\lambda}{2} e^{-\lambda|x|} \tag{3-24}$$

Considering a uniform quantizer, for a given quantization step size q and dead zone threshold $0.5q$, the number of code words of a Laplacian source is

$$\begin{aligned} C &= N \cdot \int_{|x|\geq 0.5q} \frac{\lambda}{2} e^{-\lambda|x|} dx = N \cdot \left(1 - \int_{|x|<0.5q} \frac{\lambda}{2} e^{-\lambda|x|} dx \right) = N \cdot \left(1 - 2 \int_{0}^{0.5q} \frac{\lambda}{2} e^{-\lambda x} dx \right) \\ &= N \cdot e^{-\frac{\lambda q}{2}} \end{aligned} \tag{3-25}$$

where N is the number of coefficients.

For a Laplacian source, let us define the distortion measure as $D(x,\tilde{x}) = |x - \tilde{x}|$ (MAE) where x is the input symbol and $\tilde{x}$ is the output symbol of quantizer. According to [26], if a distortion D is allowable, the minimum number of bits to represent a symbol from a Laplacian source is given by

$$R(D) = \log_2 \left(\frac{1}{\lambda D} \right) \tag{3-26}$$

Considering a uniform quantizer, for a given quantization stepsize q, the corresponding distortion is

$$D(q) = 2\int_{0}^{0.5q} p(x)x\,dx + 2\sum_{i=1}^{\infty}\int_{(i-0.5)q}^{(i+0.5)q} p(x)|x - iq|\,dx \tag{3-27}$$

With Eq. (3-24), we have

$$D(q) = \frac{1}{\lambda}\left[1 + \frac{e^{-\lambda q}}{(1-e^{-\lambda q})}\left(2 - e^{0.5\lambda q} - e^{-0.5\lambda q}\right) - e^{-0.5\lambda q}\right] \tag{3-28}$$

With Eq.(3-25), we have

$$D(C) = \frac{1}{\lambda}\frac{N-C}{N+C} \tag{3-29}$$

With Eq.(3-26) and Eq.(3-29), we have

$$R(C) = \log_2\frac{N+C}{N-C} = \log_2\left(\frac{1+\frac{C}{N}}{1-\frac{C}{N}}\right) \tag{3-30}$$

For $\frac{C}{N}$ close to zero, a Taylor expansion of Eq. (3-30) yields

$$R(C) = \frac{2C}{N\ln 2} + O\left(\left(\frac{C}{N}\right)^3\right) \tag{3-31}$$

which is approximately a linear function.

The H.263 quantization scheme is given by Eq. (3-19). If we define $D_I(x)$ and $D_P(x)$ to be the distribution of the code words in the INTRA and INTER macroblocks, respectively, for any quantization parameter QP, the number of non-zero code words in a frame can be obtained as follows:

$$C(QP) = \int_{|x|\geq 2QP} D_I(x)dx + \int_{|x|\geq 2.5QP} D_P(x)dx \tag{3-32}$$

Since in H.263 codec, the DCT coefficients are rounded to integers, $D_I(x)$ and $D_P(x)$ are actually histograms of the DCT coefficients, and Eq.(3-32) becomes

$$C = \sum_{|x|\geq 2QP} D_I(x) + \sum_{|x|\geq 2.5QP} D_P(x) \tag{3-33}$$

3.3.3 Model parameters

The only parameters of the proposed model in Eq. (3-22) are α and β. The physical meaning of the model parameters, α and β, is the average length of the code words in a frame and the bits used for encoding the overhead information in a frame, respectively. In our experiment, we also calculate the model parameters, α and β, of every frame. Figure 3-3, and 3-4 show the results.

In Figure 3-3, we can see that, if we encode the same video sequence with different quantization parameter QP, the model parameter α for different streams is approximately the same. On the other hand, from Figure 3-4, we can see that the model parameter β changes drastically within a stream. However, for different streams encoded with different quantization parameters, β follows the same profile and, at the lower bit-rate ($QP \geq 10$), the model parameter β of a frame is similar to that of the other streams coded with a different QP.

In summary, based on the experiments and analysis, we can reach the following conclusions:

- For each frame in a video stream, the relationship between the number of produced bits and the number of code words in a frame can be approximated by a linear function.
- When encoding the same video sequence with different quantization parameters, the model parameter α for different streams is approximately the same.
- When encoding the same video sequence with different quantization parameters, the model parameter β for different streams is different. At the lower bit-rate, β of the same frame in different streams have a similar value.

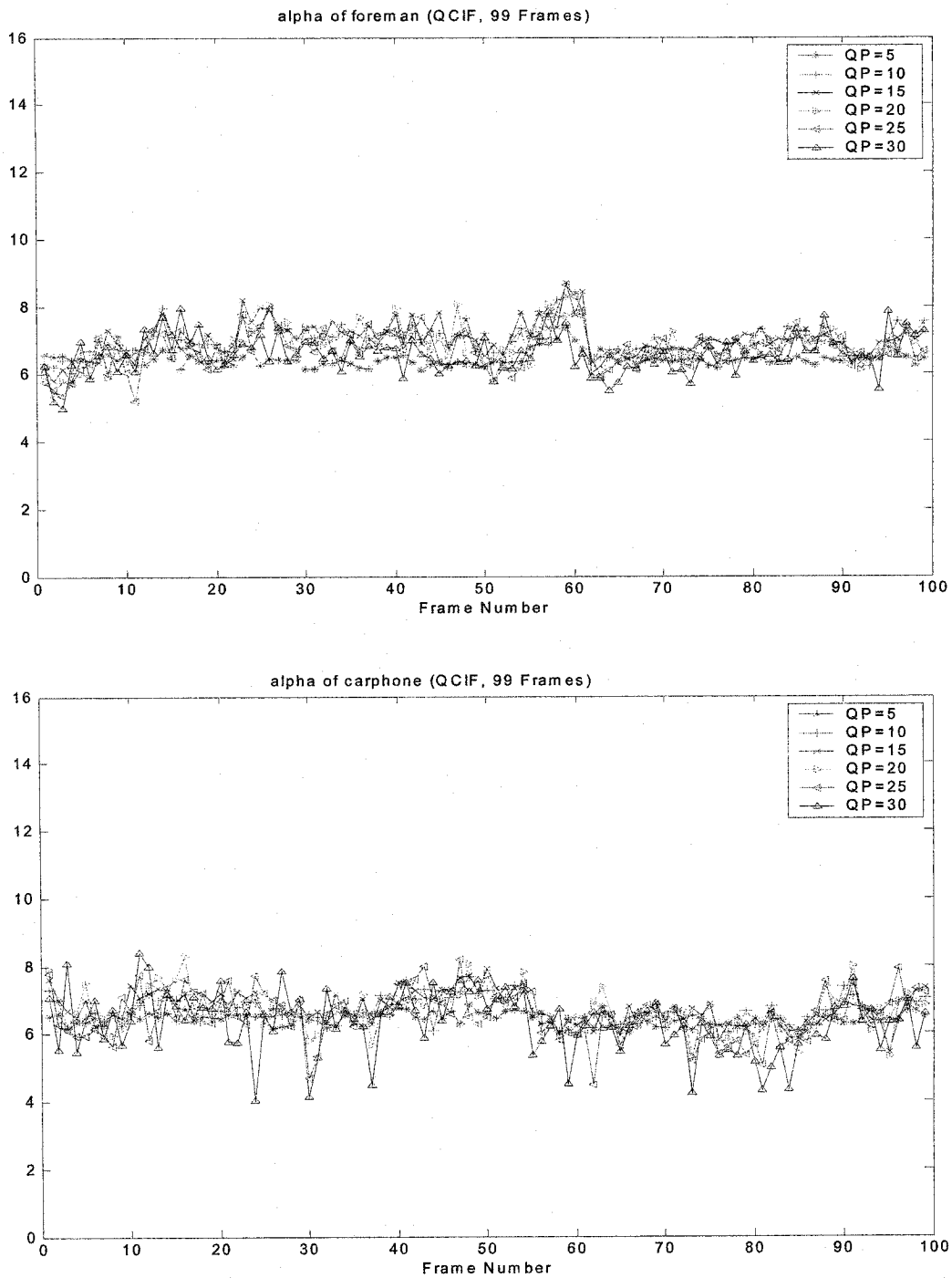


Figure 3-3. α of different test cases. The x-axis is the frame number, and the y-axis is the value of α.

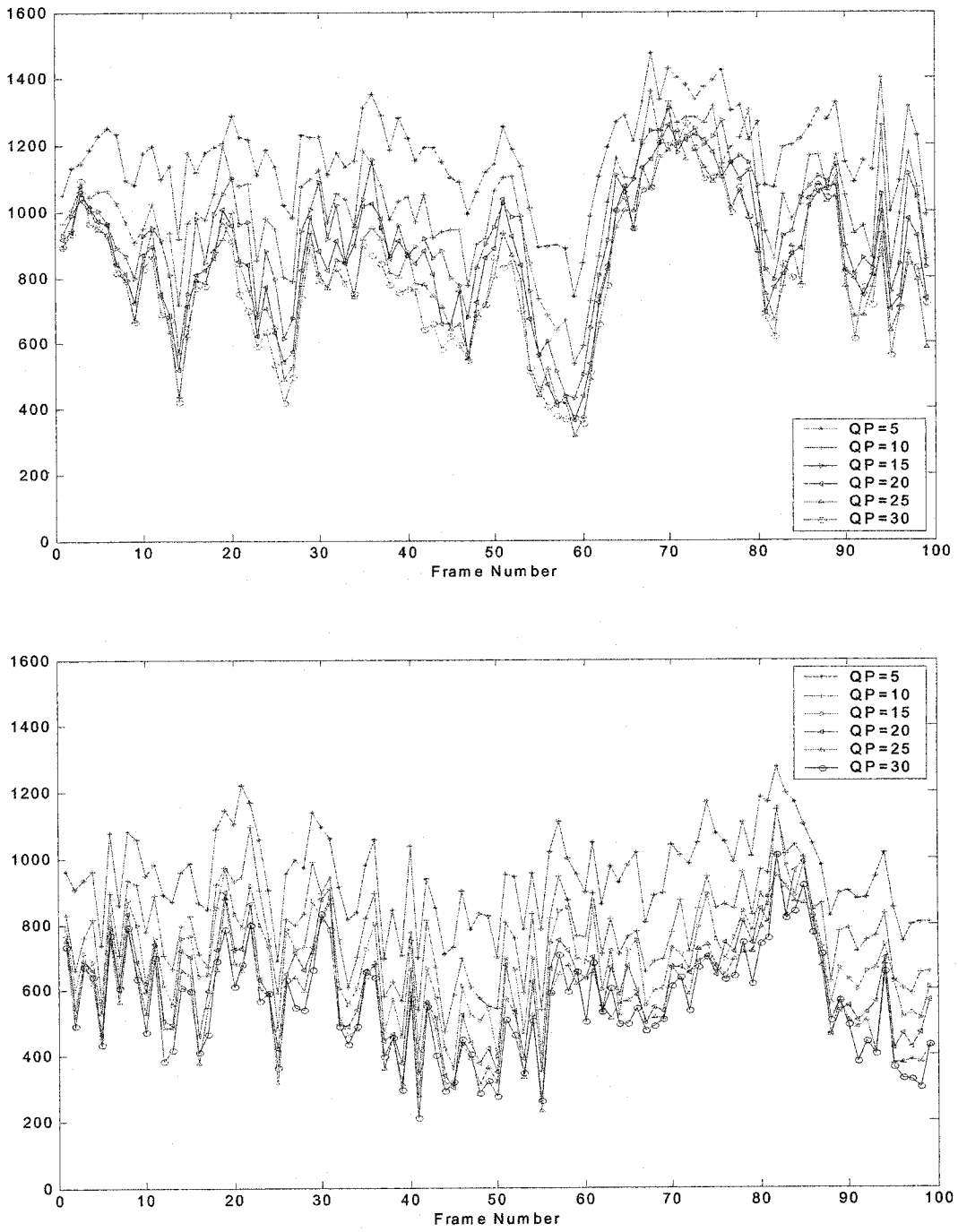


Figure 3-4. β of different test cases. The x-axis is the frame number, and the y-axis is the value of β ($x10^3$).

Based on these results, the produced bits for encoding a frame can be estimated from the number of code words in this frame. For the same video, the model parameters from the incoming video stream can be reused for the transcoder bit allocation and rate-control. The detailed algorithm is described in the next chapter.

3.4 Summary

In this Chapter, we first reviewed the problem of video rate-control and some existing model-based solutions. Based on our extensive experiments and analysis, we developed a new bit allocation model for macroblock layer rate-control. With Shannon's source coding theorem, we provided a theoretical proof for this bit allocation model. We have also discussed the physical meaning of the model parameters, and how these parameters can be reused in transcoding. In the next chapter, based on this model, a rate-control algorithm for H.263 video transcoding will be developed.

Chapter 4. Rate-control for Video Transcoding

4.1 Macroblock Layer Bit Allocation Algorithm

4.1.1 Algorithm based on the linear bit allocation model

An efficient transcoder should be able to reuse as much information from the incoming video as possible. Such information includes motion vectors, modes of macroblocks, and other syntax information. Based on analysis in Chapter 3, we prove that the model parameters, α and β, from the incoming video stream can be reused to control the output bit-rate of the video transcoder. In the video transcoder, after decoding an incoming video frame, we know the number of bits that are used for encoding the whole frame, B, and bits that are used for encoding frame header information, code words, motion vectors, and other syntax information. In this work, what we are interested in is the number of code words, C, and the number of bits for encoding these code words, $\hat{B}$. Then we can calculate the model parameters, α and β, of an incoming frame as

$$\alpha = \frac{\hat{B}}{C} = \frac{B - \beta}{C} \tag{4-1}$$

Based on our analysis in the previous chapter, we know that α is independent of the quantization parameters. Therefore, during the encoding phase, parameter α from the incoming video stream can be reused as a reference to estimate the produced bit count of every frame. In order to estimate the produced bit count by using Eq. (3-22), we have to know the number of code words for every possible quantization parameter, $\{C_i(QP_i), 0 \le i \le N-1, 1 \le QP_i \le 31\}$, where i is the macroblock index, QP_i is the quantization parameter, and N is the number of macroblocks within a frame. According

to Eq. (3-23), after transcoding and obtaining a new set of DCT coefficients, we can analyze the DCT coefficients and get the table of $C_i(QP_i)$.

Let $C(QP)$ be the total number of code words for a frame, i.e. $C(QP) = \sum_{i=0}^{N-1} C_i(QP)$, and B and B_i $(i=0,1,...,N-1)$ the target bit number to be allocated to a frame and the i_th Macroblock respectively. If we use the same frame layer rate-control scheme as TMN8, B is determined by the target bit-rate R, the target frame rate F, and the buffer fullness $B=R/F-\Delta$ [24], and B_i is adaptively determined for each macroblock. The goal of macroblock layer bit allocation is to match the sum of B_i to B. From the incoming video stream, we can get the average α as in Eq. (4-1). Here, we first assume that we can know the total bit count, β, for encoding the overhead information of the output frame. Actually, this information can only be obtained after encoding the whole frame. In a later part, we will introduce a heuristic method for estimating β. We define $\hat{B}$ as the bit target for encoding the total code words in a frame, i.e.

$$\hat{B} = B - B_{header} - \overline{\beta} \cdot N \tag{4-2}$$

where B_{header} is the number of bits for encoding the picture header, which can be known before encoding a frame, $\overline{\beta}$ is the average bit number for encoding the overhead information in a macroblock and N is the total number of macroblocks in a frame.

Then, initially, we can estimate the total number of the code words in the output frame by:

$$C(QP) = \frac{\hat{B}}{\alpha} \tag{4-3}$$

By searching the table $\{C_i(QP_i)\}$, we can get a reference quantization parameter QP_{ref} as

$$QP_{ref} = \arg\min\left|C(QP_i) - C(QP)\right| \tag{4-4}$$

and the reference code words number of the output video frame $C(QP_{ref})$ as

$$C(QP_{ref}) = \sum_{i=0}^{N-1} C_i(QP_{ref}) \tag{4-5}$$

QP_{ref} is used as the reference quantization parameter for the frame and is a syntax element in the frame header. QP_{ref} is also used for encoding the first macroblock. After encoding the *i_th* macroblock, we can know the produced bit number, $\hat{B}_i$, for encoding the code words, C_i, in this macroblock. Then we can calculate the bit budget for encoding the remaining macroblocks, $\hat{B}_{i+1}^R$, as

$$\hat{B}_{i+1}^R = \hat{B}_i^R - \hat{B}_i, \quad i = 0 \cdots N-1, \quad \hat{B}_0^R = \hat{B} \tag{4-6}$$

If we define $\hat{B}_i^P$ as the total produced bits for encoding the code words in the previous i macroblocks, and C_i^P as the number of code words in the previous i macroblocks, we can update the model parameter, α_{i+1}, for the remaining macroblocks as

$$\alpha_{i+1} = \frac{\hat{B}_i^P}{C_i^P} \tag{4-7}$$

where $\hat{B}_i^P = \hat{B}_{i-1}^P + \hat{B}_i$, $\hat{B}_0^P = 0$, $C_i^P = C_{i-1}^P + C_i$, $C_0^P = 0$, $\alpha_0 = \alpha \quad i = 0 \cdots N-1$.

Then the estimated number of code words in the remaining macroblocks, C_{i+1}^R, can be calculated as

$$C_{i+1}^R = \frac{\hat{B}_{i+1}^R}{\alpha_{i+1}} \tag{4-8}$$

Using a similar method as in Eq. (4-4) and Eq. (4-5), we can calculate QP_{i+1} for encoding the *(i+1)_th* macroblock as

$$QP_{i+1} = \arg\min \left| C(QP_r) - C_{i+1}^R \right|, \quad r = i+1, \cdots, N \tag{4-9}$$

After selecting the QP_{i+1}, the $(i+1)\_th$ macroblock will be quantized and entropy coded. In this scheme, the quantization parameter QP_i is only determined by the bit allocation algorithm. However, in order to keep a smooth video quality, the H.263 standard requires that the difference between quantization parameters for two successive macroblocks not be greater than 2 [41], i.e., $|QP_{i+1} - QP_i| \le 2, \quad 0 \le i \le N-1$. Therefore, if the difference between the quantization parameter obtained from Eq. (4-9) and the previous quantization parameter is greater than 2, the new quantization parameter has to be adjusted. This constraint can also help us save some computational cost. After encoding a macroblock with quantization parameter QP_i, the quantization parameter for encoding the next macroblock could only be in $\{QP_i\text{-}2,\ QP_i\text{-}1,\ QP_i,\ QP_i\text{+}1\ QP_i\text{+}2\}$, so, in Eq. (4-9), we do not have to search the number of the non-zero coefficients for the other quantization parameters. In this scheme, after encoding every macroblock, we adaptively update the model parameter α according to the statistical information from previous encoded macroblocks to ensure the linear relationship between the bit budget and number of code words for the remaining macroblocks, which makes the number of produced bit meet the bit budget accurately.

In the above analysis, we assume that, before encoding a frame, we already know the produced bits, β, for encoding the overhead information, which includes frame header, motion vectors, DC coefficients, and other macroblock-layer syntax information. However, β can only be known after encoding the whole frame. For example, in the H.263 basic option, if the quantization parameter of a macroblock is different from that of the previous macroblock, there is a 5-bit penalty for changing the quantization parameter.

Specifically, 2 bits are used in the syntax element *DQUANT* to indicate the difference between the values of two quantization parameters, and three additional bits are needed on average in the syntax element *MCBPC* to indicate that the value of *QP* in the current macroblock is different from that in the previous macroblock [25]. In this case, before getting the quantization parameter, *QP*, for the macroblock, we can not know if a *DQUANT* field is required in the macroblock header. At a high bit-rate (smaller *QP*), the amount of β is only a small part of the total bit budget, but at a lower bit-rate (larger *QP*), β is a significant part of total bit budget. For example, in our experiments, if we set the target bit-rate to 20Kbps, frame rate to 10 frame/s, then the bit budget for encoding every P frame is about 2000 bits. After encoding, the average β is about 1000 bits, which is half of the total bit budget. Therefore, estimation of β has a significant effect on the bit allocation. In this work, we use a heuristic method based on linear regression to estimate the model parameter β. Based on the analysis in Chapter 3, we know that, at the lower rate, β of the same frame in different video streams has a similar value. Therefore, we can reuse the β information from the incoming video stream. At the higher rate, we can pre-assign an average value as an initial β. Before encoding a frame, we can know the total bit number for encoding the picture header, so we only consider β as the bit number for encoding the overhead information of the macroblocks in a frame. Then, before encoding a frame, we can calculate the average β of every macroblock as

$$\overline{\beta} = \frac{\beta}{N} \tag{4-10}$$

After encoding the *i_th* macroblock, we can get the actual bit number of β_i. We define β_i^P as the total bit number for encoding the overhead information in the previous i macroblocks, i.e.

$$\beta_i^P = \beta_{i-1}^P + \beta_i, i = 1 \cdots N, \beta_0^P = 0 \tag{4-11}$$

Then we can calculate the average β_i for the previous i macroblocks as

$$\overline{\beta}_i^P = \frac{\beta_i^P}{i} \tag{4-12}$$

and calculate the average β_i for the remaining (N-i) macroblocks as

$$\overline{\beta}_i^R = \frac{i \times \overline{\beta}_i^P + (N-i) \times \overline{\beta}}{N} \tag{4-13}$$

Finally, $\overline{\beta}_i^R$ is used to estimate the total bit number for encoding the overhead information in the remaining (N-i) macroblocks as

$$\beta_i^R = \overline{\beta}_i^R \times (N-i) \tag{4-14}$$

The above procedure for the proposed bit-rate-control algorithm is summarized as follows:

Step1: Decode the incoming P frame into DCT coefficients; calculate the model parameters α and β of the frame. Perform transcoding and get the new DCT coefficients of the frame;

Step 2: Analyze the DCT coefficients of every MB and calculate the table of the code words $\{C_i(QP), 1 \le QP \le 31, \quad 0 \le i \le N-1\}$ as in Eq. (3-23); Calculate the target bit count, $\hat{B}$, reference quantization parameter QP_{ref}, and reference code words count, $C(QP_{ref})$ as in Eq. (4-2), Eq. (4-3), and Eq. (4-4); Calculate the average β for all macroblocks as in Eq. (4-10);

Step 3: Calculate the model parameter α_i, quantization parameter QP_i as in Eq. (4-7) and Eq.(4-9). Transcode the current macroblock.

Step 4: Let β_i and $\hat{B}_i$ be the number of bits produced for encoding overhead information and code words in the current macroblock. Update $\hat{B}_i^R$ and $\overline{\beta}_i^R$ as in Eq. (4-6) and Eq.(4-13).

Step 5: Let $i=i+1$. If $i<N$, go to step 3, otherwise go to step 1 for the next frame.

4.1.2 Algorithm implementation and complexity

In this algorithm, the model parameters, α and β, are updated after encoding every macroblock. The only extra complexity introduced is the calculation of the code word table, $\{C_i(QP_i),\ 0\le i\le N-1,\ 1\le QP_i\le 31\}$, by pre-analyzing the DCT coefficients. As indicated in Eq. (3-23), we only need to compare the absolute value of DCT coefficients, |COF|, to $2.5QP$ *or* $2QP$, but do not have to really quantize the coefficients. Moreover, due to the non-increasing property (for the same DCT coefficients set in a MB, quantization with a larger QP will always have fewer or equal number of code words) of quantization, the pre-analysis can be optimized in the real implementation. When we find that the code word number at a certain QP_i is zero, then, instead of calculating the code word number at other QPs that are larger than QP_i, we can directly set them to zero. The algorithm for initializing the code word table is described as follows.

```
InitCodewordTable()

N: number of MBs in a frame;
block_count: number of blocks in a MB;
mb[k]: the k-th macroblock;
MB_codeword: codeword table;
frame_codeword: number of codeword in a frame;
coeff: dct coefficients

for(k=0;k<N;k++){
   if(mb[k] is INTRA_MB){
      for(l=0;l<block_count;l++){
         for(m=1; m<64; m++){
            x=abs(coeff[k][l][m]);
            if(x==0)
               continue;
```

```
            for(QP=1;QP<=31;QP++){
              if(x>=2*QP){
                MB_codeword[k][QP]++;
                frame_codeword[QP]++;
              }
              else
                break;
            } } } }
else{ //INTER MB
    for(l=0;l<block_count;l++){
      for(m=0;m<64;m++){
    x=abs(coeff[k][l][m]);
    if(x==0)
      continue;
    for(QP=1;QP<=31;QP++){
      if(x>=2.5*QP){
        MB_codeword[k][Q]++;
        frame_codeword[Q]++;
      }
      else
        break;
    } } } } }
```

As we can see from the algorithm, since in INTER frames, DCT coefficients are zero or very close to zero, most comparisons in the algorithm can be skipped. Since in most low bit rate encoding cases, the smaller QPs are never used, the algorithm can be further optimized by reducing the QP candidate set. In addition, step 3 searches the code word table and finds a QP_i that can make the produced bits close to the bit budget. As mentioned above, due to the constraint of the QP range in the H.263, we only need to search $\{QP_{i-1}-2, QP_{i-1}-1, QP_{i-1}, QP_{i-1}+1, QP_{i-1}+2\}$, instead of searching the whole set of quantization parameters.

4.1.3 Performance evaluation and analysis

The proposed rate-control scheme has been implemented in a H.263 video transcoder based on [88]. Comparisons between results achieved by our rate-control method and by the TMN8 rate-control method are discussed in this section.

For our experiments, several test sequences in QCIF format are used. The test sequences are first encoded with unified quantization parameter QP=1. Every test sequence is

encoded with only 1 I frame followed by P frames without any rate-control scheme. Table 4-1 shows the average PSNR, achieved average output bit-rate, and average frame bit count of these test sequences.

Table 4-1, Test sequences for transcoding.

	Frames	PSNR(Y, dB)	Bit-rate (Kbps)	Bits/frame
Carphone	1 I, 250 P	48.50	2002.69	66,756
Foreman	1 I, 250 P	48.81	2366.82	78,893
Salesman	1 I, 250 P	48.34	1192.54	39,751
Suzie	1 I, 149 P	48.54	1442.68	48,089

Then, the resulting H.263 test sequences are sent into the video transcoder, which uses TMN8 and the proposed rate-control schemes. The transcoding output frame rate is set to 10 fps.

First, we want to examine the bit allocation based on the proposed model and the model of the TMN8. Figure 4-1 shows the bit allocation at different channel bandwidths with the proposed allocation scheme and the TMN8. For a fair comparison, we did not use any frame-layer rate-control scheme in this experiment, so the bit budget for every frame is constant, R/F, in both the TMN8 and the proposed bit allocation scheme. With the proposed algorithm, the number of bits produced by each frame is well matched to the target bit-rate or the channel bandwidth. Especially at the lower bit-rate, the bit allocation with the proposed scheme is more accurate than that with the TMN8.

To measure rate-control performance, we define the normalized deviation as $\frac{Actual\ bit\ counts - Bit\ budget}{Bit\ budget} \times 100\%$. The normalized deviation with different rate-control schemes is plotted in Figure 4-2. It can be seen that when compared to the TMN8

rate-control algorithm the proposed algorithm yields a much smaller deviation at all target rate levels.

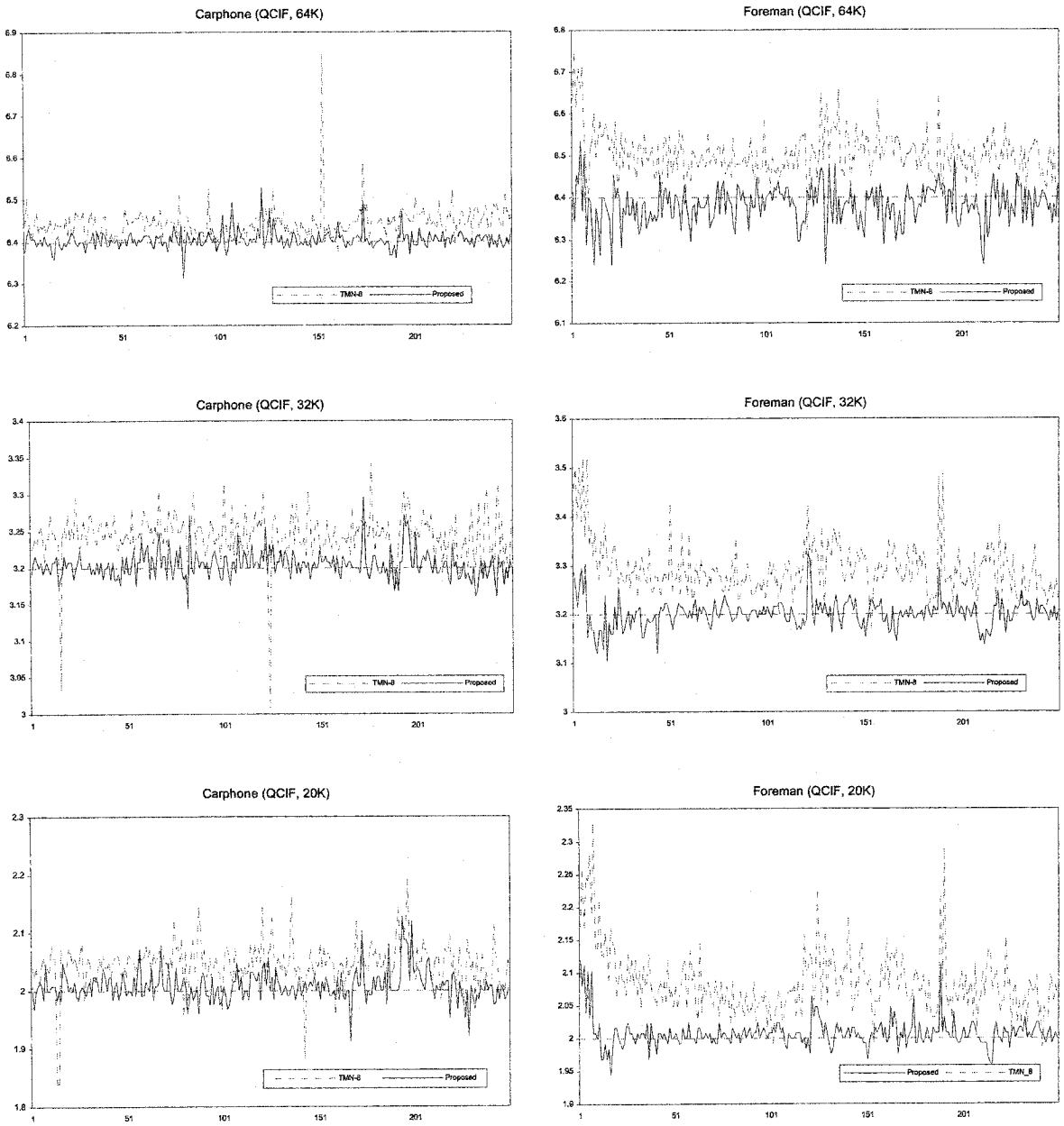


Figure 4-1. Bit allocation result at different channel bandwidth with the proposed scheme and TMN8. In this figure, the x-axis is the number of frames, and the y-axis is the number of produced bits ($x10^3$).

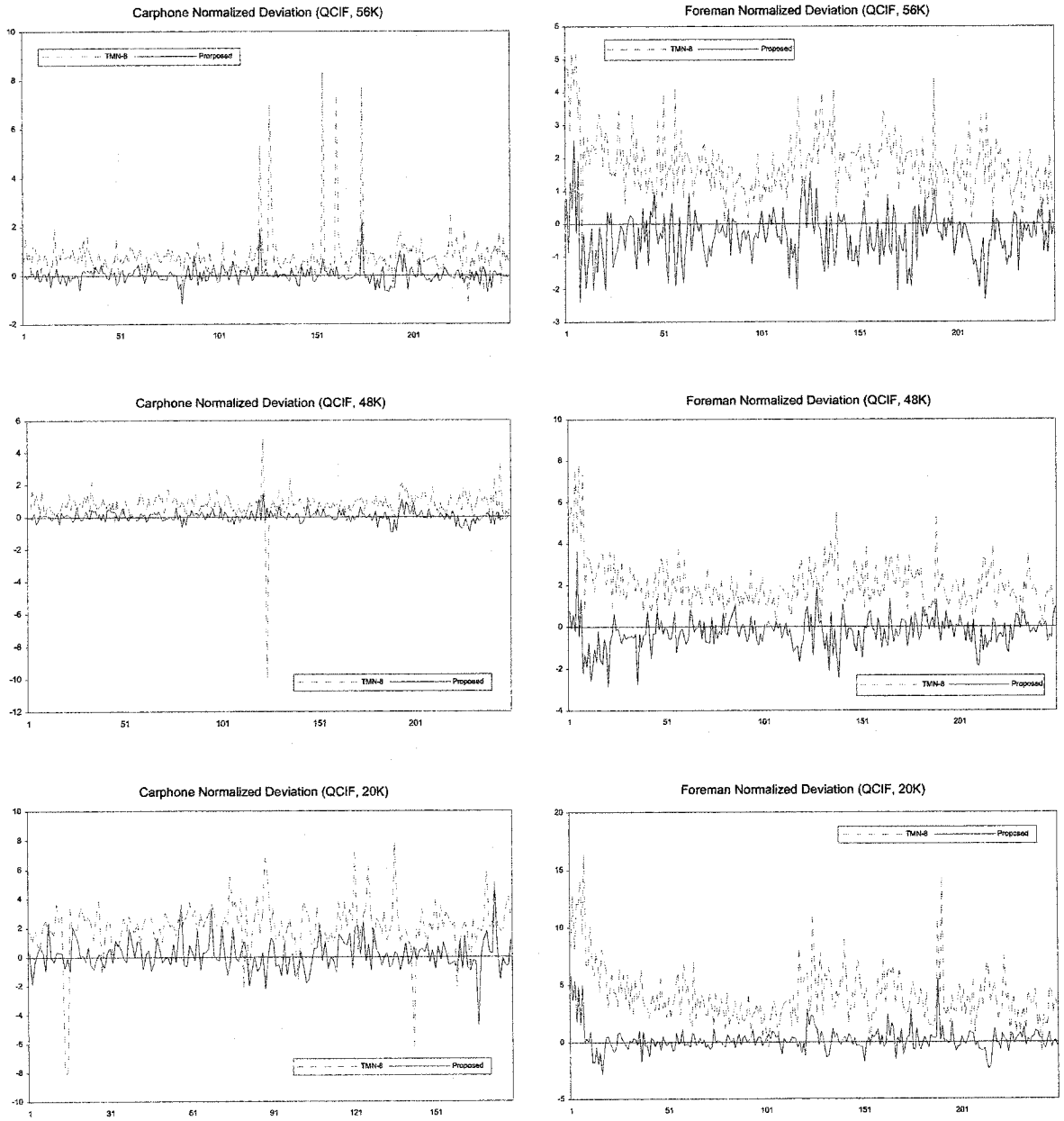


Figure 4-2. The normalized deviation between the target and actual bit count. In this figure, the x-axis is the number of frames, and the y-axis is the value of normalized deviation (%).

In Figure 4-3, we plot the number of bits in the buffer for each transcoded frame when the proposed rate-control algorithm and the TMN8 algorithm are applied. From these

experimental results, we can see that the proposed rate-control algorithm can keep a stable buffer occupancy level.

In this experiment, we assume the maximum buffer delay is one frame. When the buffer occupancy is larger than the size of one frame R/F, the following frame will be skipped.

In the "Salesman, 20K" test case, when using the TMN8 rate-control scheme, there are two frames which are skipped: frame 169 and frame 178. However, with the proposed rate-control, no frame is skipped at these two points.

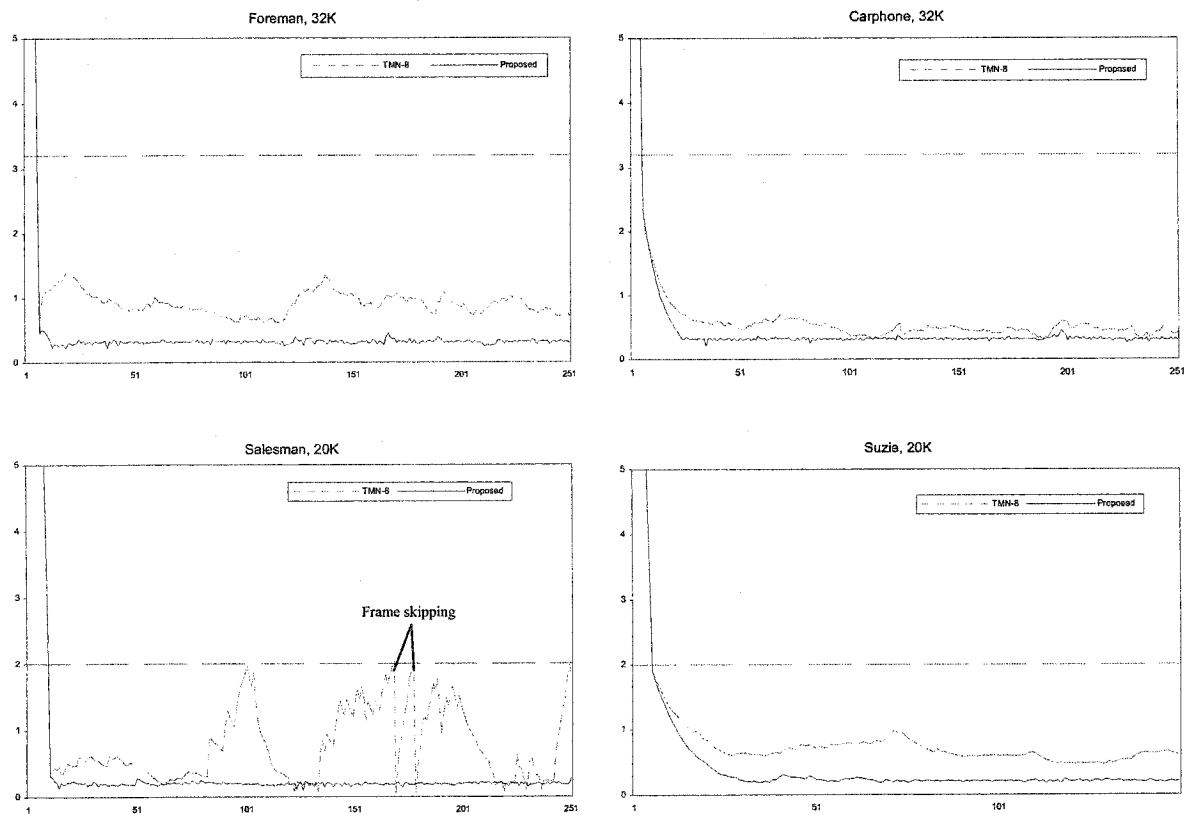


Figure 4-3. Comparison of the number of bits in the buffer. In this figure, the x-axis is the number of frames, and the y-axis is the number of bits in the buffer (x10^3). The dash line is the threshold (*R/F*) of skipping frame.

Table 4-2 summarizes the result of the Average PSNR, the achieved bit-rates, and average normalized deviation for our proposed method and the TMN8.

Table 4-2. Result of average PSNR, bit-rate, and normalized deviation

	Target bit rate (Kbps)	Bit budget /frame	TMN8			Proposed		
			Average PSNR (dB)	Achieved Bit-rate (Kbps)	Average Normalized Deviation(%)	Average PSNR (dB)	Achieved Bit-rate (Kbps)	Average Normalized Deviation(%)
Carphone	128	12800	38.60	128.80	0.74	38.87	128.09	0.15
	96	9600	37.47	96.48	0.58	37.51	96.06	0.18
	64	6400	35.74	64.44	0.65	35.76	64.05	0.20
	56	5600	35.19	56.49	0.84	35.19	56.05	0.22
	48	4800	34.58	48.40	0.86	34.56	48.06	0.23
	32	3200	33.04	32.44	1.39	33.03	32.04	0.41
	20	2000	31.29	20.46	2.40	31.34	20.11	1.05
	16	1600	30.51	16.49	4.13	30.59	16.33	2.87
Foreman	128	12800	36.41	128.88	0.69	36.59	127.74	0.48
	96	9600	35.41	96.83	0.97	35.38	95.78	0.51
	64	6400	33.87	65.03	1.58	33.80	63.81	0.61
	56	5600	33.38	57.05	1.82	33.32	55.84	0.63
	48	4800	32.81	49.01	2.04	32.76	47.91	0.59
	32	3200	31.40	32.93	2.81	31.33	32.01	0.65
	20	2000	29.64	20.85	4.09	29.70	20.07	0.68
	16	1600	28.75	16.80	4.78	28.88	16.24	1.63

From Table 4-2, we can see that for all test cases, the proposed algorithm can provide accurate bit allocation and improved video quality. In all test cases, the achieved bit-rate of the proposed algorithm is lower than that of the TMN8, but the corresponding PSNR value of the proposed algorithm is higher than that of the TMN8. At low bit-rate, the average normalized deviation of the proposed algorithm is much smaller than that of the TMN8.

4.2 Scene Context Based Frame Layer Rate-control

4.2.1 Scene context based frame layer rate-control for transcoding

In the previous rate-control algorithms adopted by standards, the frame layer rate-control did not consider the visual content of a video sequence, or only use a simple measure, such as variance or complexity, to adjust the bit allocation. These rate-control strategies work well for normal video sequences. However, they can not handle the non-stationary

nature of digital video efficiently. In most existing rate-control schemes, the information obtained from previously coded pictures is utilized for estimating the target bits for the current picture. However, if a scene change occurs, information obtained from previously coded pictures is no more useful and even can cause visual quality degradation in the pictures following the scene change [70]. A scene change represents any distinctive difference between two adjacent video frames. It includes rapid motion of moving objects as well as changes to different visual content. Saw [78] has classified scene variations into different categories, including general scenes, rapid motion, panning, zooming, and scene cuts. Different metrics have been introduced in the literature to describe the non-stationary nature and scene variation of digital video scene.

In order to measure the visual content of a video frame, different measurements are used, such as complexity in MPEG-2, variance in H.263, and average motion vectors, etc. Similar as in [25], in this work we use the average variance of a frame as the energy measure of a frame, which is defined as

$$ave\_\mathrm{var}(k) = \frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2 = \frac{1}{6\times 64\times N}\sum_{i=0}^{N-1}\sum_{j=1}^{6\times 64}(p_j - \bar{p})^2 \qquad (4\text{-}15)$$

where p_j is the pixel value of the luminance and chrominance components in a macroblock, $\bar{p}$ is the mean value of the macroblock, and N is the number of macroblocks in a frame. Because the DCT is an orthogonal transformation, σ_i^2 is also the variance of the DCT coefficients of a macroblock. By using the energy preservation property of DCT transformation, we can easily calculate σ_i^2 in the DCT domain. Figure 4 shows the average variance of three test video sequences with different visual content and movement.

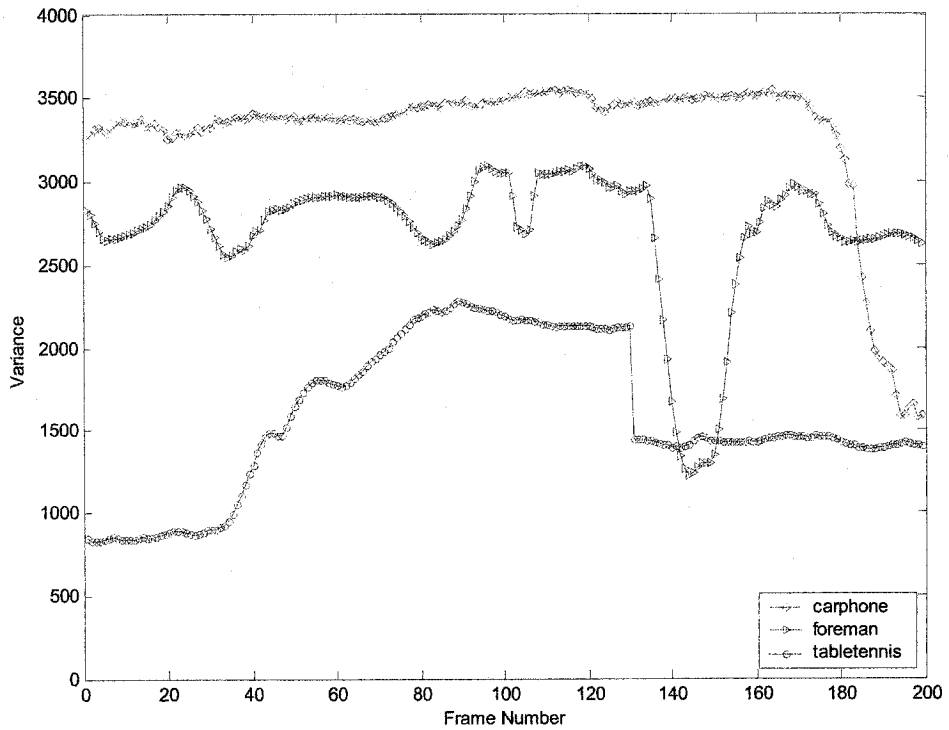


Figure 4-4. average variance of different test sequences.

Besides visual content, scene change is another critical element that affects video quality. In [78], the variance between two pictures is used to detect the scene change. It is defined as:

$$\text{var\_}dif(k) = \frac{\sum_{i=0}^{ROW-1}\sum_{j=0}^{COL-1}(dif(k,i,j)-\mu_{dif})^2}{ROW \times COL} \tag{4-16}$$

where the indices i and j represent the pixels in the row and column directions respectively, μ_{dif} is mean value of the difference picture, *dif(k)*, which is obtained by subtracting the previous picture from the current picture on a pixel-by-pixel basis. ROW and COL represent the number of pixels of a row and column. In [70], the signed

difference of mean absolute difference (MAD) is used to detect the scene change. It is defined as:

$$SDMAD(k) = MAD(k) - MAD(k-1)$$
$$= \frac{\sum_{i=0}^{ROW-1}\sum_{j=0}^{COL-1}(dif(k,i,j) - dif(k-1,i,j))}{ROW \times COL} \quad (4\text{-}17)$$

In these two schemes, we must have the picture pixel value to detect scene changes. However, in DCT domain video transcoder, we can only get the DCT coefficients of the incoming video. In this work, we simply use the percent of INTRA mode blocks in an INTER frame to detect the scene change, i.e.

$$SCD = \frac{Number\ of\ INTRA\ Coded\ Macroblocks}{Number\ of\ Macroblocks\ in\ a\ Frame} \times 100\% \quad (4\text{-}18)$$

SCD can be calculated after variable length decoding the incoming video. In our experiment, we cascaded several different test sequences, and use this scene change detection method. Detection results are plotted in Figure 4-5.

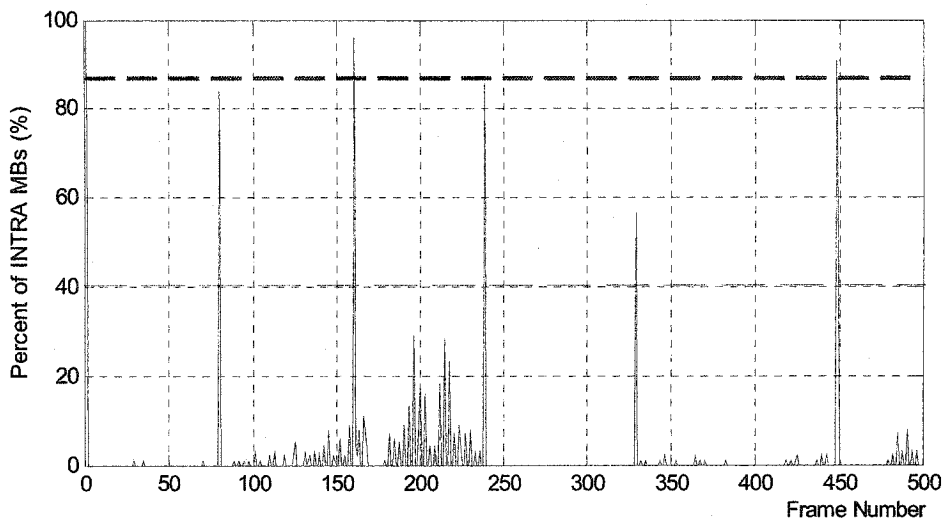


Figure 4-5. SCD of a typical H.263 coded video sequence.

In this work, we select 40% as the scene change detection threshold, i.e., if the percent of INTRA block in a frame is higher than 40%, a scene change occurs. By utilizing the scene change detection and frame variance, we can allocate bits to every frame based on its variance and type. The detailed algorithm will be introduced in the following section.

4.2.2 Frame-layer bit allocation for H.263 video transcoding

For video transcoding, the INTRA (I) frames are very important for the quality of reconstructed frames because motion compensation of the following frames is dependent on the quality of INTRA frames. Considering that INTRA frames appear seldom and irregularly, we should transcode the INTRA frames at high quality. In the proposed scheme, if the incoming video frame is an INTRA frame, we use unified quantization parameter (QP=5) to transcode it. Since encoding INTRA frames will produce large number of bits, physical size of the buffer must be larger than the bit number of the INTRA frames. To keep lower buffer delay, several frames after the INTRA frame may have to be skipped.

After variable length decoding the incoming video frames, we calculate the percent of INTRA blocks in the INTER frames. If the percent of INTRA blocks is greater then 40%, a scene change occurs. In this case, we encode the first INTER frame after the scene change as INTRA frame. Therefore, the quality of frames after the scene change is not affected.

At the frame layer, instead of making the bit target of every frame meet channel bandwidth, as in the TMN8, we make the bit budget of a group of picture (GOP) meet the channel bandwidth. Different from the GOP structure in the TM5, the length of a GOP N

is predefined and a GOP only contains INTER frames. Then, the bit budget for encoding a GOP is defined as:

$$B_{GOP} = (R/F) \times N + \Delta \tag{4-19}$$

where Δ is the number of left bits from encoding the previous GOP. Here, we should know that the B_{GOP} includes bits for encoding DCT coefficients in a frame and bits for encoding frame header, motion vectors and syntax information. For simplification, we only consider B_{GOP} as bit target for encoding all DCT coefficients in a GOP. Inside a GOP, the bit budget is allocated according to the energy and frame types of individual frames. Inside a GOP, P frames are allocated more bits then B frames. In this work, the bit budget for the first B frame is half of that of the first P frame. Based on this rule, we can know the bit budget for all P frames and B frames in a GOP respectively. For the frames with the same type, bits are allocated to every frame according to its energy, which is measured by the average variance of a frame. Then, for P frames, we have

$$\hat{B}_i = \frac{B_{GOP}^P \times ave\_\text{var}(i)}{\sum_{i=1}^{N_P} ave\_\text{var}(i)} = \frac{B_{GOP}^P}{N_P} \cdot \frac{ave\_\text{var}(i)}{\overline{ave\_\text{var}}} \tag{4-20}$$

where $\hat{B}_i$ is the bit budget for encoding DCT coefficients of the i-th P frame in a GOP, and $\overline{ave\_\text{var}}$ is the average variance of all P frames in a GOP. In the proposed scheme, we define $B^P_{GOP(i)}$ as the total bit number for encoding the first i P frames in the GOP, i.e.,

$$\begin{aligned} &B_{GOP}^P(i) = B_{GOP}^P(i-1) + B_i, \\ &B_{GOP}^P(0) = 0, \quad i = 0,1,2\cdots N \end{aligned} \tag{4-21}$$

Because the average variance of the GOP is unknown, we use the average variance of the previous i P frames with the same type as $\overline{ave\_var}$ and updated it after encoding every frame, i.e.,

$$\overline{\text{ave\_var}} = \frac{1}{i}\sum_{j=1}^{i} \text{ave\_var}(j). \tag{4-22}$$

Then, we have the bit budget for the $(i+1)$_th P frame as

$$\hat{B}_{i+1} = \frac{B_{GOP}^{P}(i)}{i} \cdot \frac{\text{ave\_var}(i+1)}{\overline{\text{ave\_var}}} \tag{4-23}$$

The bit allocation equation for B frames can be obtained similarly. Similarly as in TMN8, we use the average of the previous frames to estimate the number of bits used for encoding the overhead information. In order to keep the buffer delay, when the buffer occupancy exceeds a predefined threshold, which is used to control the maximum buffer delay, the following frame is skipped until the buffer occupancy is lower than the threshold.

4.2.3 Performance evaluation and analysis

The proposed rate-control scheme has been implemented in a video transcoder. The structure of the transcoder is illustrated in Figure 4-6.

In this transcoder, after variable length decoding the incoming frame, the percent of INTRA blocks in the INTER frames is calculated in the Scene Change Detection (SCD) module. After de-quantizing the code words and getting DCT coefficients, we calculate the average variance of the video frame as the frame variance. Frame layer rate-control module (FRC) combines the scene change, frame energy, and buffer feedback to allocate the bit budget for every frame. At the macroblock layer, proposed algorithm in the previous section is used. Comparisons between results achieved by our bit allocation

method and by the bit allocation in the TMN8 rate-control method are discussed in this section.

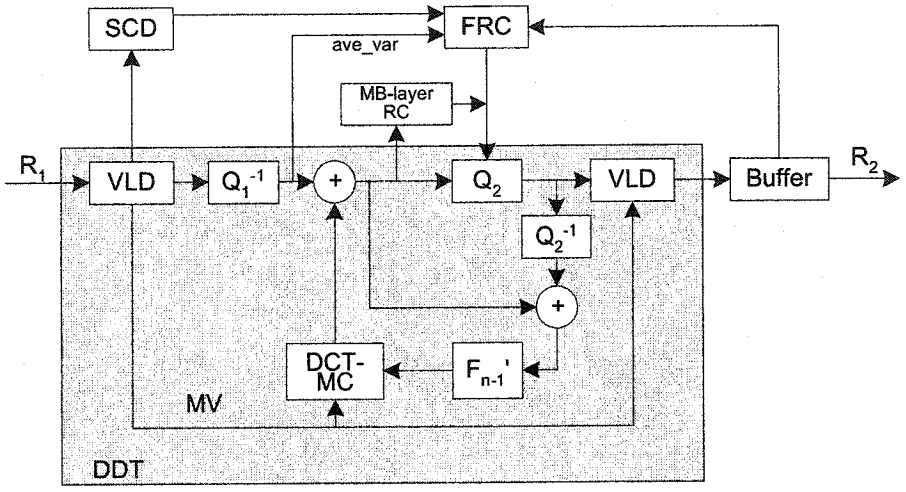


Figure 4-6. Structure of the rate adaptation transcoder.

In our experiments, several test sequences with different scene variations in QCIF format are used. The test sequences are first encoded with unified quantization parameter QP=3. Every test sequence is encoded with only 1 INTRA frame followed by INTER frames (B, P) without any rate-control scheme. The results are summarized in Table 4-3.

Table 4-3. Test sequences

Sequence Name	Frame Patterns	Achieved PSNR (dB)	Achieved Bit-rate (kbps)
Carphone	IBBPBBP..	40.89	432.02
Foreman	IBBPBBP..	39.55	754.04
Table tennis	IPPPPPP..	39.31	786.01
Cascaded	IPPPPPP..	41.25	274.02

The resulting H.263 test sequences are sent into the video transcoder, which uses TMN8 and the proposed bit allocation schemes. If the incoming frame is an INTRA frame or an INTER frame after the scene change, it is transcoded as an INTRA frame with QP=5.

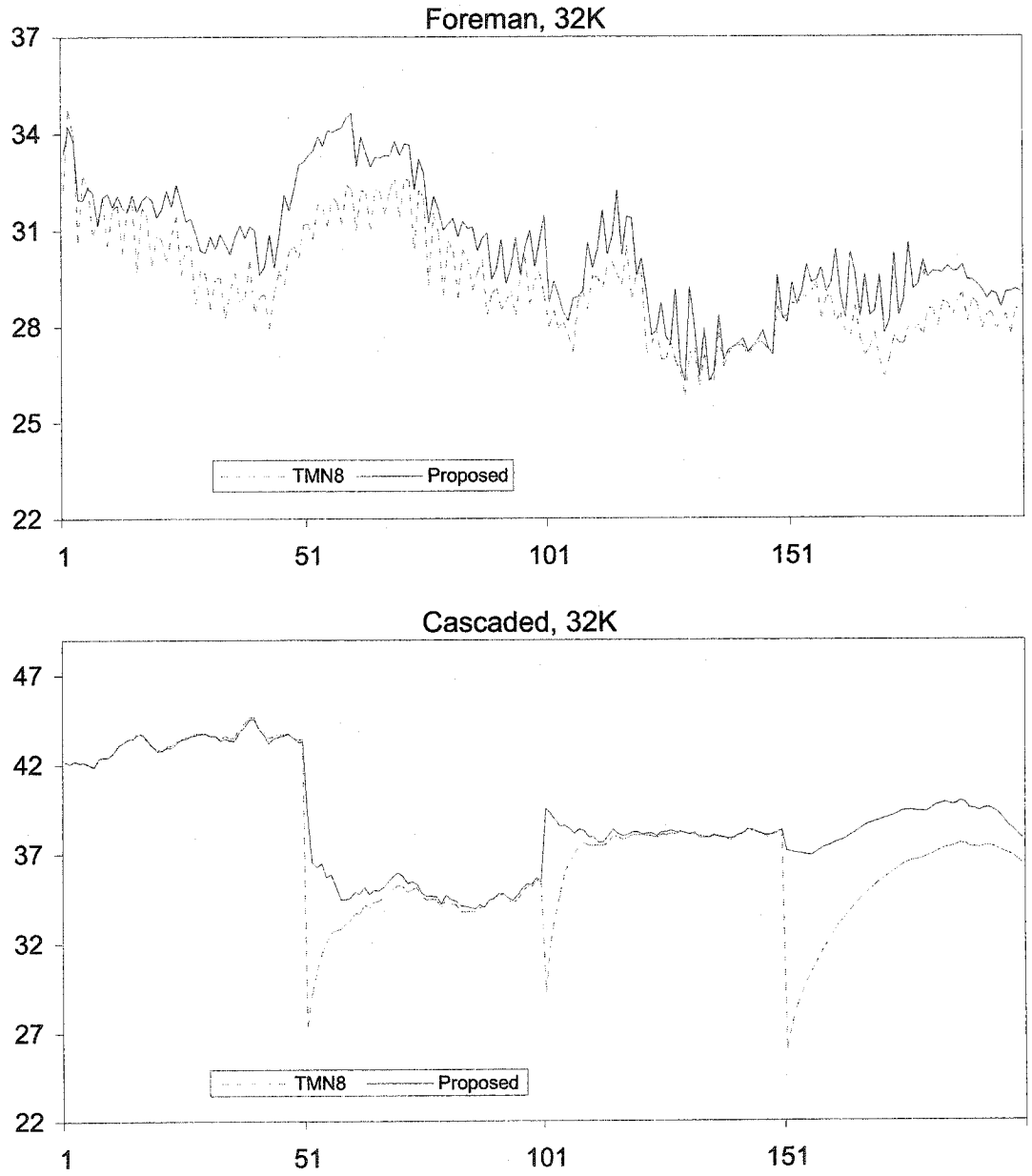


Figure 4-7. Comparison of transcoding video quality. In this figure, the x-axis is the frame number, the y-axis is the PSNR (dB).

Figure 4-7 plots the achieved PSNR per frame for two of the test cases. We can see that the proposed rate-control can obtain better visual quality than the bit allocation scheme in the TMN8. In the "Cascaded" test case, the proposed scheme detects scene changes and

encodes the first INTER frame after the scene change as INTRA frame. The visual quality after scene change is much better than that of the TMN8.

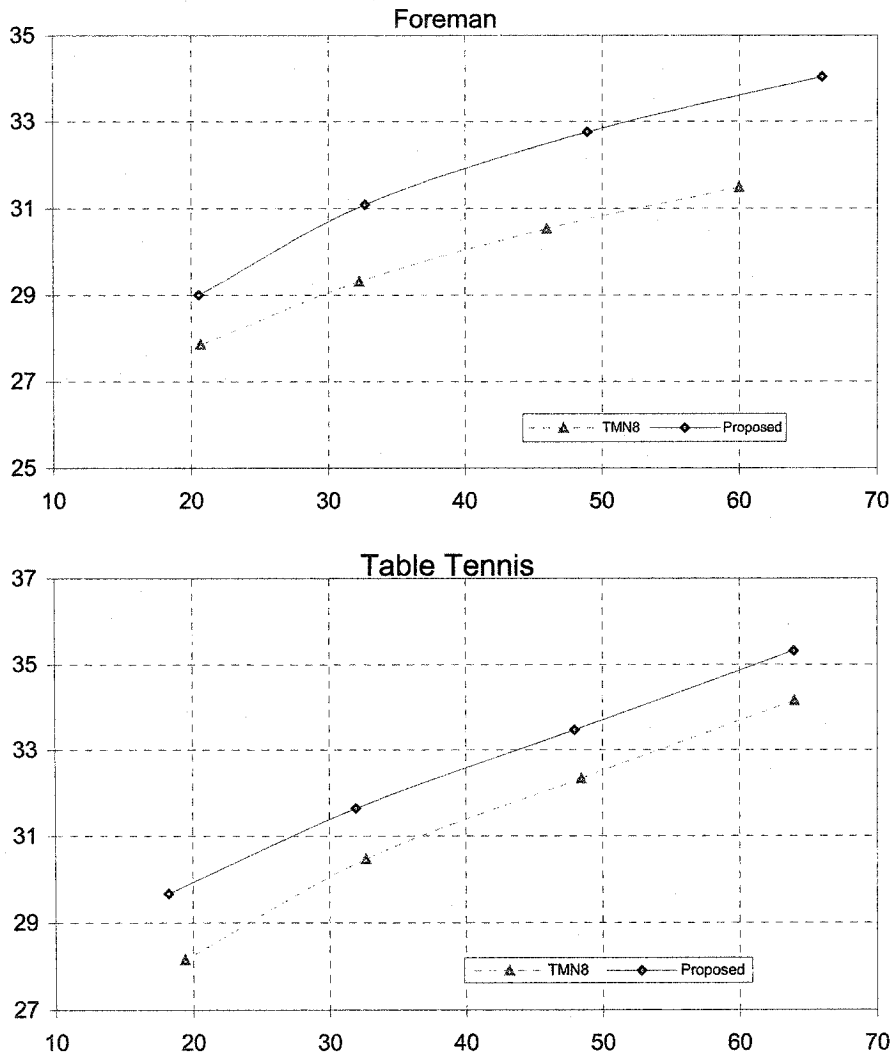


Figure 4-8. Rate-Distortion of different test cases. In this figure, the x-axis is the achieve bit-rate (Kbps), the y-axis is the achieved average PSNR(dB) for INTER frames.

Figure 4-8 illustrates the approximate Rate-Distortion curve of different test cases and rate-control schemes. Because the proposed rate-control allocates more bits to the frames with higher scene variations and frames at scene changes, and allocates fewer bits to frames with lower scene variations, the achieved PSNR is higher than that of TMN8 at the same achieve bit-rate.

Table 4-4. Result of average PSNR, bit-rate, and normalized deviation.

	Target Bit-rate (kbps)	TMN8			Proposed		
		PSNR (dB)	Achieve Bit-rate (kbps)	Average Normalized Deviation	PSNR (dB)	Achieved Bit-rate (kbps)	Average Normalized Deviation
Carphone	64	36.26	61.94	0.87	39.07	62.20	0.17
	48	35.00	46.74	1.01	37.43	46.77	0.22
	32	33.36	31.14	1.48	35.54	31.34	0.38
Foreman	64	31.49	59.97	1.19	34.04	65.97	0.53
	48	30.53	45.94	1.52	32.76	48.91	0.52
	32	29.32	32.29	2.49	31.09	32.72	0.84
Table Tennis	64	34.15	64.01	1.76	35.32	63.97	0.50
	48	32.34	48.45	2.10	33.47	47.97	0.63
	32	30.47	32.71	3.40	31.64	31.98	1.62
Cascaded	64	43.76	64.64	1.21	46.52	63.99	0.46
	48	41.81	48.98	2.07	44.40	48.00	0.41
	32	39.04	33.23	3.83	41.98	32.00	0.47

Table 4-4 summarizes the average PSNR, the achieved bit-rate and average normalized deviation of the proposed rate-control scheme and the TMN8. For all test cases, the proposed algorithm can provide accurate bit allocation and improved video quality. Especially for test sequences with higher scene variations, the average gain of PSNR is more than 1dB.

4.3 Summary

In this chapter, we have presented a new macroblock layer bit allocation algorithm based on the bit allocation model proposed in the previous chapter. We also proposed a heuristic method to estimate the coding syntax overhead. Furthermore, assuming a CBR target channel, we proposed a frame layer rate-control algorithm, which allocates more bits to frames with higher scene variations and frames after scene changes. Experimental results show that the proposed algorithm effectively alleviates visual quality degradation of frames with higher variations and frames after scene changes. For the macroblock layer rate-control, the proposed scheme can provide accurate control and stable buffer occupancy. Because all information required by frame bit allocation and macroblock

layer rate-control can be calculated from the incoming video on the DCT domain, the proposed scheme can be used for DCT domain video transcoders for transmitting digital video over heterogeneous networks.

Chapter 5. Real-Time Video Transcoding and Transmission over Wireless Channels

5.1 Rate-control for VBR Transcoding

Due to the variety of different networks comprising the present communication infrastructure, a connection from the video source to the end user may be established through links of different characteristics and capacities. In the interactive video application, the bit-rate of the compressed video was usually adjusted by the source encoder to match the available capacity of the most stringent link used in the connection. The visual quality has to be sacrificed because the encoded video bit-rate has to match the "weakest link". Recently, the increasing demand for mobile communications has resulted in the extensive use of wireless communication technology. Transmission of real time video over wireless networks is challenging because of the delay constraints involved, and because of the negative impact of channel errors on the perceptual quality of video at the decoder. Uncorrected channel errors may result in significant quality degradation at the decoder. To achieve high video quality at the decoder requires error control techniques to be used for video communications. These techniques can be roughly categorized into open-loop (e.g., forward error correction, FEC) and close loop (e.g., automatic repeat request, ARQ). FEC codes can be chosen to guarantee certain error rate requirements for the worst channel conditions. However, this causes unnecessary overhead and wastes bandwidth when the channel is in a good state. In addition, for bursty error channels, i.e., errors tend to occur in clusters during fading periods, FEC can not correct the error if the whole packet is corrupted. Using ARQ error control for video communications over wireless channels has been selected as an

alternative to a purely FEC-based approach, because retransmission is only required during periods of poor channel conditions.

In the current wireless video communication systems, the wireless link is usually an extension of the wired network, i.e. the wireless access point is a node of the wired network and mobile terminals connected to the access point through a point-to-point link. Moreover, the video source encoder is not generally located right at the wireless access point or base station. Through a video transcoder located at the wireless access point, the video bit-rate can be adapted at the wireless access point without the delay that could be incurred in transmitting the channel information back to the source encoder. In this chapter, we concentrate on how a real time wireless video system can be supported by an ARQ error control scheme and rate adaptation transcoding techniques. To take full advantage of the error control capabilities of the ARQ scheme, we propose to combine the ARQ feedback mechanism with the transcoding mechanism at the video transcoder, by which one can achieve an intuitively appealing result: the rate for the transcoded video is reduced during the periods of poor channel conditions.

5.1.1 Delay and buffer constraints on transcoding

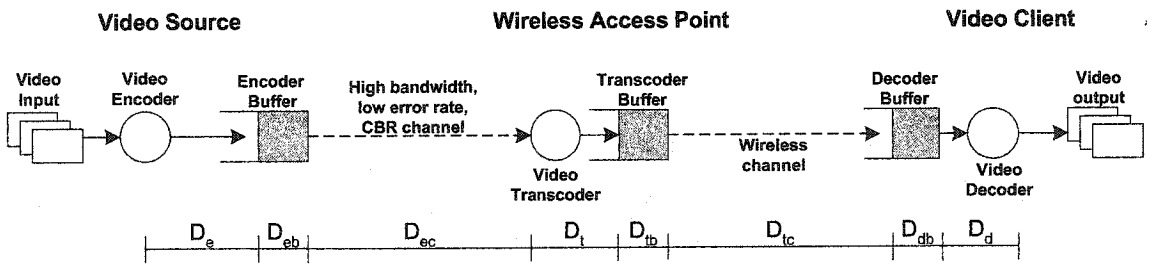


Figure 5-1. Wireless video communication system with transcoder

In this chapter, we define a wireless real time video transmission system with a transcoder as illustrated in Figure 5-1. In this system, the video source is connected to the wireless access point through a high bandwidth, low error rate, CBR channel, therefore,

the transmission between the video source and access point is assumed to be error-free. The video client is connected to the wireless access point through a wireless channel. The video transcoder, which is located at the access point, is responsible for dynamically adapting the video transmission rate from the high bit-rate channel to the lower bit-rate channel. For real time captured video, each frame is captured, processed and finally displayed in real time within a predefined delay interval. Such delay interval is referred to as the end-to-end delay of video transmission. We assume here video frames are played at the same frame rate as they are encoded at the video source. Under these conditions, the end-to-end delay per frame, D, will have to be constant. Each frame captured at the video source at time t will have to be decoded and available for display before time $t+D$. In the defined system, an ARQ error control scheme is used on the wireless channel. The effective channel rate is variable due to the retransmission when the channel is in a bad state. Therefore, a buffer is also needed at the transcoder to smooth out the mismatch between the transcoding rate and the effective wireless channel rate. Intuitively, the transcoder should adjust its transcoding ratio according to the effective channel rate in order to keep the buffer without overflow. Traditionally, rate-control has been studied from the point of view of memory, i.e., rate-control was required to avoid overflowing the available buffers at the encoder and decoder. However, in what follows we will assume that sufficient physical memory is available and formulate the problem only from the point of view of end-to-end delay. In the defined system, the video transcoder will drop a frame, if it arrives at the decoder too late to be decoded. Clearly, frame skipping at the transcoder results in quality loss, especially when motion compensated video coding is used.

Let us assume that a video is encoded and transmitted at a fixed frame rate F. At the decoder side, the video is decoded and displayed at the same frame rate F. As illustrated in Figure 5-1, the end-to-end delay each frame experiences (from the time it is obtained from the video input to the time it is placed in the video display) consists of several delay components as in Eq (5-1).

$$\begin{aligned} D = {} & D_e\,(\textit{Encoding Processing Delay}) \\ & + D_{eb}\,(\textit{Encoder Buffer Delay}) \\ & + D_{ec}\,(\textit{CBR Channel Delay}) \\ & + D_t\,(\textit{Transcoding Processing Delay}) \\ & + D_{tb}\,(\textit{Transcoder Buffer Delay}) \\ & + D_{tc}\,(\textit{VBR Channel Delay}) \\ & + D_{db}\,(\textit{Decoder Buffer Delay}) \\ & + D_d\,(\textit{Decoder Processing Delay}) \end{aligned} \tag{5-1}$$

As introduced in the previous section, in the above equation, the processing delay in the encoder, transcoder or decoder depends on the computer power, and the transcoding complexity can be much lower than that of the encoding process. Therefore, we can assume the processing delay components are constant and can be neglected. Since we assume that the link between the video source and wireless access point is a high bandwidth, CBR channel, and the link between the access point and the video client is a point-to-point wireless channel, the variation of the channel delay is comparably small. Thus, we assume the channel delay components are also constant and can be neglected. Under these assumptions, the end-to-end delay will only consist of the buffering delay components as in Eq. (5-2).

$$D = D_{eb} + D_{tb} + D_{db} \tag{5-2}$$

After the first frame is encoded, it will be sent into the encoder buffer, which is empty at this time. In general, it is possible for encoder buffer underflow to occur if transmission

starts at the same time as the encoder puts the first bit into the buffer. On the other hand, at the decoder buffer, which is also empty for the first frame, underflow will occur if decoding starts at the same time as the first bit of the first frame arrives. In practice, this is prevented by starting the transmission and decoding after a certain initial delay. The same initial delay is also necessary for the transcoder buffer. Therefore, the end-to-end delay for the first frame is given by the sum of the initial delays. To simplify the analysis, we define the system initial delay D, i.e., the first frame will be encoded at time t and will be decoded and displayed at time $t+D$. The overall delay that any following video frame will experience has to remain constant as D. Under this constraint, the system will function normally as long as the decoder buffer does not overflow and underflow, which prevents the video data from being lost and guarantees that the decoder has received the data of a video frame before it is scheduled to be displayed, respectively. Since, in the most general case, the decoder is capable of removing bits from its buffer faster than bits arrive, therefore, decoder buffer overflowing will not occur; hence, the goal of the system will be to avoid decoder buffer underflow. The decoder buffer is said to underflow when the decoder must display a new frame (which happens for real-time video applications), but no new frame has been decoded, i.e., the end-to-end delay of this frame exceeds the required delay.

5.1.2 Buffer analysis of VBR transcoders

In this section, we will analyze the conditions that the encoder buffer and transcoder buffer have to meet in order to prevent the decoder buffer from underflowing within the end-to-end delay constraints. In [73], Reibman *et al.* have proven that for a CBR video system without transcoder, it is possible to prevent the decoder buffer from overflowing or underflowing simply by ensuring that the encoder buffer never underflows or

overflows. Therefore, when the decoder buffer underflow does not occur, i.e., $B_d(t+D) \geq 0$, the encoder buffer occupancy should be $B_e(t) \leq B_{\max}^e$. On the other hand, when the decoder buffer overflow does not occur, i.e., $B_d(t+D) \leq B_{\max}^d$, the encoder buffer occupancy should be $B_e(t) \geq 0$. Therefore, the decoder buffer underflow and overflow can be prevented by simply controlling the encoder buffer occupancy such that $0 \leq B_e(t) \leq B_{\max}^e$ at any time t. Note that the maximum buffer occupancy $B_{\max}^e$ is determined by the end-to-end delay, D, rather then by physical memory considerations, given that there is sufficient memory available at the encoder and decoder. In [13], Assunção *et al.* have analyzed the effect of inserting a transcoder along the CBR transmission path. For transcoders with either fixed or variable compression ratio, they proved that the encoder buffer size can be maintained as if no transcoder existed while the decoder has to modify its own buffer size according to both the bit-rate conversion ratio and transcoder buffer size. They also derived the conditions that both the encoder and transcoder buffers have to meet in order to prevent the decoder buffer from overflowing or underflowing. However, if the target channel is a VBR channel, the buffer constraints of the encoder and transcoder need to be analyzed. In this section, we will derive the corresponding buffer constraints for the encoder and transcoder. Different from [13], in which the encoder and decoder buffer size is a major concern, in this work, we focus on the delay involved by inserting a transcoder and the transcoder buffer constraints for preventing the decoder buffer from underflowing.

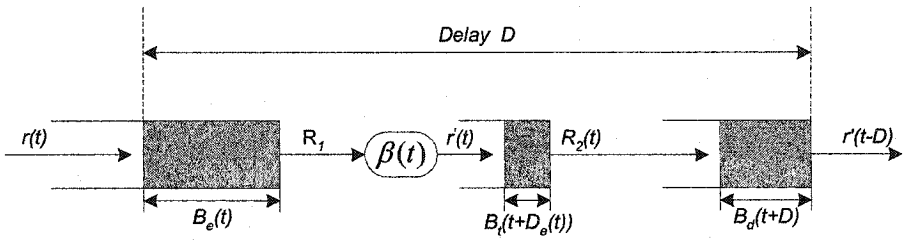


Figure 5-2. Buffering with variable transcoding ratio for VBR channel

As illustrated in Figure 5-2, the encoded video is sent into the encoder buffer at rate *r(t)*, and the video data in the encoder buffer are transmitted to the transcoder at the constant rate R_1. At the transcoder, the transcoding is modeled as a scaling function $\beta(t)$ which is multiplied by *r(t)* and produces the transcoded video at rate $r'(t)$, i.e.,

$$r'(t) = \beta(t) \cdot r(t) \tag{5-3}$$

The effect of multiplying $\beta(t)$ by $r(t)$ can be seen as equivalent to reducing the number of bits used in the video frame encoded at time *t*. The output of the decoder buffer consists of a delayed version of $r'(t)$. In the system of Figure 5-2, transcoding is performed on the CBR R_1 which consists of the video data units of *r(t)* after the encoder buffering delay, defined as the delay $D_e(t)$ that a video frame encoded at time *t* waits in the encoder buffer before being transmitted.

If we define the initial delay of the transcoder buffer as D_T, the buffer occupancies of the encoder, transcoder, and decoder buffers are given by

$$B_e(t) = \int_0^t r(\tau)d\tau - R_1 \cdot t \tag{5-4}$$

$$B_t(t) = \begin{cases} \int_0^t r'(\tau)d\tau, & t < D_T \\ \int_0^{t-D_e(t)} r'(\tau)d\tau - \int_0^t R_2(\tau)d\tau, & t \geq D_T \end{cases} \tag{5-5}$$

$$B_d(t) = \begin{cases} \int_0^t R_2(\tau)d\tau, & t < D \\ \int_0^t R_2(\tau)d\tau - \int_0^{t-D} r'(\tau)d\tau, & t \geq D \end{cases} \tag{5-6}$$

Let us now verify that under normal conditions, where neither of the three buffers underflows or overflows, the total delay between encoder and decoder is still constant. A video data unit entering the encoder buffer at time t will arrive at the transcoder at $t+D_e(t)$ and will be decoded at $t+D$. Since the processing delay in the transcoder is neglected, $t+D_e(t)$ is also the time at which the video data unit is transcoded and put into the transcoder buffer. Therefore, in order to calculate the total delay of the system, the encoder, transcoder and decoder buffers should be analyzed at instants t, $t+D_e(t)$, and $t+D$ respectively, which are given by.

$$B_e(t) = \int_0^t r(\tau)d\tau - R_1 t \tag{5-7}$$

$$B_t(t + D_e(t)) = \int_0^t r'(\tau)d\tau - \int_0^{t+D_e(t)} R_2(\tau)d\tau \tag{5-8}$$

$$B_d(t + D) = \int_0^{t+D} R_2(\tau)d\tau - \int_0^t r'(\tau)d\tau \tag{5-9}$$

From Eq. (5-9), we can derive the following relation:

$$\begin{aligned} B_d(t + D) &= \int_0^{t+D} R_2(\tau)d\tau - B_t(t + D_e(t)) - \int_0^{t+D_e(t)} R_2(\tau)d\tau \\ &= \int_{t+D_e(t)}^{t+D} R_2(\tau)d\tau - B_t(t + D_e(t)) \end{aligned} \tag{5-10}$$

Since the decoder buffer must not underflow, then we have the following result:

$$\begin{aligned} B_d(t + D) > 0 \Rightarrow & \int_0^{D-D_e(t)} R_2(\tau)d\tau - B_t(t + D_e(t)) > 0 \\ \Rightarrow & B_t(t + D_e(t)) < \int_{t+D_e(t)}^{t+D} R_2(\tau)d\tau \end{aligned} \tag{5-11}$$

Therefore, the condition that transcoder buffer occupancy has to meet is

$$0 < B_t(t + D_e(t)) < \int_{t+D_e(t)}^{t+D} R_2(\tau)d\tau \tag{5-12}$$

Since the connection between video source and transcoder is a CBR channel, according to [84], the encoder buffer at the video source must meet the following condition to prevent the transcoder buffer from underflow:

$$B_e(t) < \int_0^{D_T} R_1 d\tau = R_1 \cdot D_T \tag{5-13}$$

If we define D_E as the maximum encoder buffer delay and use it to substitute $D_e(t)$ in Eq. (5-11), we can have a more strict condition for the transcoder buffer as

$$0 < B_t(t + D_e(t)) < \int_{t+D_E}^{t+D} R_2(\tau)d\tau \tag{5-14}$$

In summary, for live captured video, in order to keep the end-to-end delay less than D, and decoder buffer does not underflow, the encoder buffer and transcoder buffer must satisfy the following conditions:

$$0 < B_t(t + D_e(t)) < \int_{t+D_E}^{t+D} R_2(\tau)d\tau \tag{5-15}$$

$$0 < B_e(t) < R_1 \cdot D_T \tag{5-16}$$

As we can see, if we know the end-to-end delay and the effective channel bit-rate $R_2(t)$, then the transcoding compression ratio $\beta(t)$ can be selected according to Eq. (5-8) and Eq. (5-15).

In this work, we only consider the VBR link between the wireless access point and video client as in Figure 5-2, because it is the real target channel for the video transcoder. Let us assume that a video sequence with a total of M frames is transmitted at a fixed frame rate F. For simplification purpose, time is measured discretely in units of number of

frames. Let B_i be the number of bits assigned to the i-th frame by the transcoder and $t_d=0$ be the time at which the first bit of the video sequence is received by the decoder. Thus, $t_d=i$ corresponds to i frame intervals having passed, where one frame lasts δ_f seconds, i.e., $\delta_f = \frac{1}{F}$. Let C_i be the number of bits received correctly by the decoder buffer during the i-th frame interval. For this specific ARQ channel, C_i is the number of bits that are transmitted and acknowledged during the i-th frame interval. Note that C_i does not correspond necessarily to compressed data for the i-th frame. This is because data for a particular frame could be transmitted over several frame intervals, depending on bandwidth conditions and the number of bits used to code the frame. We will assume that the end-to-end delay, D, is ΔN_d frame intervals, and the maximum encoder delay, D_E, is ΔN_e frame intervals. Then, we define $\Delta N_t = \Delta N_d - \Delta N_e$ as the delay between the transcoder and the decoder. (Here, the end-to-end delay, ΔN_d, is determined by the applications, while ΔN_t is calculated according to the end-to-end delay and the maximum encoding buffer delay.) Thus, the first frame will be decoded and displayed at time $t_d = \Delta N_d$, assuming that frames are instantaneously decoded and displayed. Since we assume that frames are played back at a constant rate, it follows that the i-th frame must be available at time $i-1+\Delta N_d$, i.e., $(i-1+\Delta N_d)\cdot\delta_f$ seconds. In order to prevent the decoder from losing frames the transmitter has to ensure that all the information corresponding to frame i has arrived to the decoder before $t_d = i-1+\Delta N_d$. This is equivalent to the channel rates having been sufficient to transmit all the necessary data,

$$\sum_{k=1}^{i-1+\Delta N_d} C_k \geq \sum_{k=1}^{i} R_k, \qquad \forall i \tag{5-17}$$

i.e., the total channel rate used up to time $i-1+\Delta N_d$, has been enough to transport the first i frames. Parallel to Eq. (5-15), we can get the condition that the transcoder buffer has to satisfy as

$$0 < B_t(i+\Delta N_e) < \sum_{j=i+1}^{i+\Delta N_t} C_j \tag{5-18}$$

We can introduce the concept of effective buffer size, $B_{eff}(i)$, which we define as the maximum level of buffer occupancy that the transcoder can reach at time i such that the channel rate is adequate to transport all the bits without violating the end-to-end delay constraint. From Eq. (5-18) the effective buffer size is

$$B_{eff}(i) = \sum_{j=i+1}^{i+\Delta N_t} C_j \ . \tag{5-19}$$

The effective buffer size depends on the frame interval, i, and is equal to the sum of the future ΔN_t channel rates. We can guarantee that if the transcoder buffer fullness $B_t(i)$ is always smaller than $B_{eff}(i)$, then the decoder buffer will not underflow.

If there is no knowledge about the channel, very little can be done, other than perhaps assume the worst case behavior. On the other hand if we have an *a priori* model of the channel and/or some online observation of its current state then it is possible to attach a likelihood of achieving a certain channel rate.

Let us call $P_i(c) = P_i(c \,|\, \text{a priori channel model, channel observation})$, the probability of the available channel rate being c at time i. Note that this model could be discrete or

continuous, and will depend on specific channel characteristics. Given the model $P_i(c)$, the effective transcoder buffer size is given by

$$B_{eff}(i) = E\left[\sum_{j=i+1}^{i+\Delta N_i} C_j\right] \tag{5-20}$$

where E[] indicates the expectation with respect to the probabilities $P_i(c)$.

In a VBR channel operating at a channel rate $R_2(t)$, which is not deterministically known, the key difficulty in dealing with this scenario is that the transcoder has to make a decision on the transcoding ratio, by taking into account the random behavior of the channel. The channel behavior is such that the data may arrive to the receiver but, due to the need of retransmission, they may arrive at the decoder too late to be decoded in time for display. Thus, while the transcoder cannot control the channel behavior, it can use whatever knowledge of the channel is available in order to avoid decoder buffer underflow.

In the following section, we will show how to make use of a probabilistic model of the channel and the observation of the current channel state in the context of this rate-control problem.

5.2 Wireless Channel Model

5.2.1 Wireless channel behavior and statistical model

Errors in the wireless environments occur both as random bit errors and as clusters of errors in long and short bursts. These errors are generally attributed to phenomena such as multipath fading, shadowing, ground wave path loss, noise, and interference from other users [14]. To achieve high video quality at the decoder requires a robust transmission scheme. The classic technique to combat transmission errors is Forward Error Correction (FEC). The concept of FEC is to add redundancy bits to the block of

video data for the purpose of error detection and correction. However, in order to correct long bursts of errors, a significant amount of redundancy bits need to be added. This reduces the effective channel bandwidth for the video data, causes unnecessary overhead and waste of channel bandwidth when the channel is in good condition. On the other hand, closed-loop error techniques like ARQ have been shown to be more effective than FEC and successfully applied to wireless video transmission [49]. Previously, the throughput for different ARQ schemes has been investigated [3, 32, 50, 56]. From the video transmission point of view, when ARQ is used, the channel becomes a variable bit-rate channel with throughputs depending on the channel conditions. When the channel becomes poor, the retransmissions use up bandwidth and thus reduce the effective channel rate (the effective channel rate is defined as the rate of the information that is correctly transmitted). For the purpose of simulation, a Markov chain was used to model bursty errors during transmission based on collected network traffic traces [14, 38, 51, 123]. Previous studies show that a first-order Markov chain, such as a two-state Markov model or a finite-state model provides a good approximation in modeling the error process at the packet level in wireless channels [122]. Using these models, one can dynamically estimate the situation of the specified channel and use this knowledge to help designing corresponding algorithms. At the same time, one can generate artificial network traces for the network under study and use the traces to simulate, and thus, better understand the performance of existing and new protocols and applications.

In [4], a two-state Markov model was used to estimate the channel states, and an adaptive rate-control algorithm was proposed based on the channel estimation. A more general N-

state Markov model was also used for channel estimation. To simplify the analysis, here we use a two-state Markov model, similar to [4], to emulate the process of packet errors. The specific channel under consideration is a wireless CDMA spread spectrum system for a mobile transmission environment, where the wireless channel consists of two radio links, namely uplink (mobile-to-base) and downlink (base-to-mobile). The transcoded video stream is packetized into constant-size packets for transmission. In our scheme, a selective repeat ARQ is used, where the reception of a packet is acknowledged by the receiver by sending either an acknowledgment (ACK) or a negative acknowledgment (NAK) to the transcoder. Only the erroneous packets are retransmitted. A time-out mechanism is used so that, if the feedback information is corrupted, data are retransmitted anyway. Packets that have been sent are stored in the ARQ buffer until they are acknowledged. Packets awaiting transmission are stored in the transcoder buffer. Since video transmission is subject to a delay constraint as discussed in the previous sections, the retransmission of any packet is attempted only while it's decoding and playing time has not been exceeded. Data loss occurs during the channel fading intervals whenever the data cannot be retransmitted before their decoding and playing time.

In the channel model under study, the channel switches between a "good state" and a "bad state," S_0 and S_1, as illustrated in Figure 5-3.

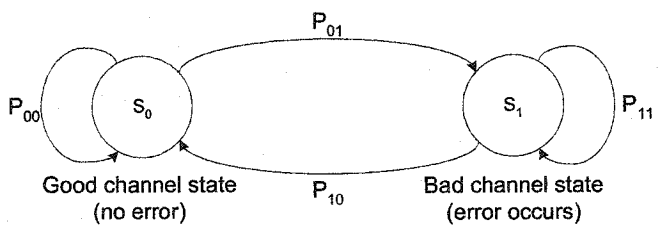


Figure 5-3. Two-state Markov channel model

A packet is transmitted correctly when the channel is in state S_0, and errors occur when the channel is in state S_1. P_{ij} for $i, j \in \{0,1\}$ are the transition probabilities. The packet-error statistics will vary according to the values of the transition probabilities. The transition probabilities can be calculated from the collected channel statistics (such as the average packet-error-burst length and the packet error-rate) generated from the wireless channel simulator. The channel state-transition probability matrix for this channel model can be set up as

$$P = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} = \begin{bmatrix} 1-P_{01} & P_{01} \\ P_{10} & 1-P_{10} \end{bmatrix} \tag{5-21}$$

The transition probability, P_{01} and P_{10}, can be derived using the assumption of Gilbert's Markov model. The run lengths of error-bursts have a geometric distribution with mean $1/P_{10}$. Thus, we have

$$P_{10} = \frac{1}{Mean\_Burst\_Length} \tag{5-22}$$

We can obtain the mean-burst-length statistics from the packet error-pattern obtained from the wireless channel simulator. According to [123], the average packet error rate is given by

$$PER = \frac{P_{01}}{P_{01} + P_{10}} \tag{5-23}$$

Therefore, P_{01} can be derived from two parameters, P_{10} and packet error rate (PER), as

$$P_{01} = \frac{P_{10} \times PER}{(1-PER)} \tag{5-24}$$

For example, in this work, we use a transition matrix which has been calculated based on a set of transceiver's parameters in [4]. The transition probabilities are P_{00}=0.9909,

P_{01}=0.0091, P_{10}=0.0526, P_{11}=0.9474. This corresponds to an average error-burst length of 19 packets and a packet-error-rate of 0.1475.

5.2.2 Effective channel bandwidth of ARQ protocol

Based on the channel feedback information from the client and the channel model introduced in the previous section, the future effective channel rate can be estimated. From the transitional probability matrix and a given initial state, the expected future channel throughput, i.e., the average of the probability of the correct transmission in the next *i* packets, can be calculated as in [4]. We use this information to adjust the target number of bits in the video rate-control algorithm. In the following discussion, all the time periods motioned are normalized with the time to transmit a packet. From the ACK/NACK at time *t*, the channel state at time *t-b*, *S(t-b)*, is known, where *b* is the round-trip delay. Based on the transmission probability matrix *P*, we define the state-probability vector at time *k*

$$\pi(k \mid S(t-b)=S_n)=[\pi_0(k \mid S(t-b)=S_n),\ \pi_1(k \mid S(t-b)=S_n)],\quad n \in \{0,1\} \tag{5-25}$$

as a row vector formed by the two-state probabilities, i.e., the probabilities for the channel to be in state S_0 and S_1 at time *k*, respectively, given that it was observed to be in state S_n at time *t-b*. Note that *t*, *b*, and *k* are all discrete values. The initial state probability $\pi(t-b \mid S(t-b)=S_n)$ at time *t-b* is set as

$$i,j \in \{0,1\},\ \pi_i(t-b \mid S(t-b)=S_j)=\begin{cases}1, & when\ i=j\\ 0, & otherwise.\end{cases} \tag{5-26}$$

The state probabilities at time *k* can be derived from the state probabilities at time *k-1*

$$\pi(k \mid S(t-b)=S_n)=\pi(k-1 \mid S(t-b)=S_n)*P \tag{5-27}$$

By recursively using Eq. (5-26), the channel state probabilities at time k, where $k > t-b$, can be calculated from the initial state probability and the transition probability matrix

$$\pi(k \mid S(t-b) = S_n) = \pi(t-b \mid S(t-b) = S_n) * P^{k-t+b} \tag{5-28}$$

In this channel model, packets are transmitted correctly ($\overline{C}$ bits are transmitted) when the channel is in state S_0, while errors occur (0 bits are transmitted) when the channel is in state S_1. The expected channel rate $E[C(k) \mid S(t-b)]$ given the observation of channel state *S(t-b)* can be calculated as

$$E[C(k) \mid S(t-b)] = \overline{C} \times \pi_0(k \mid S(t-b)) \tag{5-29}.$$

5.3 Video Transcoding for Real Time Transmission over Wireless Channel

5.3.1 Channel adaptive transcoding algorithm

The frame-layer rate-control uses the channel model to estimate the future channel condition and adjust the frame target. We propose the following scheme for the frame-layer bit-allocation.

Step 1: Calculate the transcoding buffer occupancy. If we define C as the number of bits which are successfully transmitted during the previous frame interval, then the buffer occupancy is given by

$$W = \max\left(W_{prev} + B_{prev} - C,\ 0\right). \tag{5-30}$$

In Eq. (5-30), C can be calculated according to the channel feedback information (ACK/NAK) during the previous frame interval.

Step 2: Calculate the encoding buffer delay, $D_e(t)$, that is experienced by the incoming video frame according to Eq. (5-4). Then the maximum transcoding buffer delay for the current frame is determined by

$$D_t^{\max}(t) = D - D_e(t) \tag{5-31}$$

If the estimated effective channel rate of the next frame interval is *R(t)*, then the frame skipping threshold, *M*, is given by

$$M = D_t^{\max}(t) \cdot R(t) \tag{5-32}$$

Step 3: If the buffer occupancy, *W*, is larger than the threshold *M*, the current frame is skipped. Go to Step 1; otherwise, go to step 4.

Step 4: Determine the current channel states. Before we start transcoding each frame, we first determine the current channel state. If we based it on only a single packet feedback information to determine the channel state, it will not be reliable. Instead, we calculate the average ratio of successfully transmitted bits to the average number of transmitted bits in the past *N* frame-intervals. If the ratio is greater than a threshold *H*, we decide that the channel is in the good state, S_0; otherwise, the current channel is in the bad state, S_1. The window-size parameter *N* affects the tradeoffs between the response time of the algorithm to changes in the channel conditions and the reliability of the prediction. In our simulations, *N* is chosen to be 3, which corresponds to the average burst-length statistics estimated from the wireless channel error-patterns. The threshold *H* is selected empirically based on the fact that the higher the threshold, the faster the encoder will react to prepare for the bad state in advance, in order to prevent buffer overflow and frame skipping. A value of 0.7 is used in the simulations.

Step 5: Calculate the estimated effective channel rate of the next frame interval. Depending on the current channel-state information, we can use the Markov model to find the probability of the correct transmission in the next frame interval.

In order to have a more stable estimation, we actually calculate the probability of the correct transmission in the next L frame intervals. Since the probability parameters we have derived before were at the packet level, we need to translate the L frame intervals into the equivalent number of packets. The average number of packets per frame interval can be calculated as

$$P = \frac{R}{F * PSize} \tag{5-33}$$

where *PSize* is the packet size. We can then estimate the average number of successfully transmitted bits per frame interval as

$$SBits(i) = \frac{R}{F} * p(i \mid Current\_State = S_n) \tag{5-34}$$

where $p(i|Current\_State=S_n)$ is the probability of the successful transmission in the next L frame intervals. In the simulations, we choose L depending on the transcoder buffer-size. Since a buffer-size corresponding to three frame periods is used, we choose $L=3$. The effective channel rate of the next L frame intervals is given by

$$R_{eff} = \frac{\sum_{i=1}^{L} SBit(i)}{L * (R/F)} * R \tag{5-35}$$

Step 6: Calculate the frame bit target B for the current frame. The condition that the transcoder buffer must meet is:

$$(no\ underflow)\ \ 0 < W + B - \frac{R_{eff}}{F} < D_t(t) * R_{eff}\ \ (no\ delay\ violation) \tag{5-36}$$

Then, we can have the condition that the frame bit target, B, must meet as given by

$$\underbrace{\max\left(\frac{R_{eff}}{F}-W,0\right)}_{B_{\min}} < B < \underbrace{D_t(t)*R_{eff}+\frac{R_{eff}}{F}-W}_{B_{\max}} \quad (5\text{-}37)$$

By default, the frame bit target is set to $B=\frac{B_{\min}+B_{\max}}{2}$.

Step 7: Transcode the incoming video frame. Allocate B to every macroblock according to the macroblock layer bit allocation algorithm, which will be introduced in the next section.

Step 8: If there are more frames to be encoded, go to step 1, otherwise, stop.

5.3.2 Performance evaluation

In order to show the effectiveness of the proposed rate adaptation transcoding algorithms, we implemented them and performed simulations using some channel error traces generated by the channel state model described in Section 4. 200 frames of the test video sequences including "Foreman," "Carphone," "Salesman," and "Table tennis" in QCIF format (176×144 pixels/frame) are used in our experiments. In this section, we will introduce the experiment setup and performance of the proposed algorithms.

5.3.2.1 Simulation approaches

At the encoder side, the test sequences are encoded at R_I=128kbps, which is the target bandwidth of the CBR channel, with a frame rate of 10 fps. The default TMN8 rate-control algorithm is used, in which the frame skipping threshold, M, is set to the average size of a frame, i.e., $M=R_I/F$, and Z is set to 0.1 as described in Section 5.1. The first frame of the sequences is encoded as an I frame with uniform QP=10. The transcoder will transcode the I frame with uniform QP=15. All the other frames are encoded as P frames, and the encoder and transcoder calculate the frame bit target and QP according to

their rate-control algorithms. In order to simulate the wireless channel with the ARQ protocol, we use a random number generator to generate a random number r, which is uniformly distributed in $\{0, 1\}$. Based on the introduced Markov model, if the current state is S_0, when r is less than P_{01}, the transition from state S_0 to S_1 occurs. If the current state is S_1, when r is less than P_{10}, the transition from S_1 to S_0 occurs. In this way, we generate 4 channel error traces with 20,000 packets for each trace. Two of the traces start with the "good" state; while the other two traces start with the "bad" state. The performance of rate-control algorithms is evaluated by the average of the results on these four traces.

In the transcoder, the proposed rate-control scheme is used at the frame layer and macroblock layer. We do not consider channel coding overhead, and define the packet payload size as 40 bits. We assume the target wireless channel bandwidth is 32 kbps, in which the selective ARQ scheme is used as transport protocol.

5.3.2.2 Performance of the proposed algorithms

First, we want to evaluate the effectiveness of the rate adaptation transcoder with channel model knowledge. In one case, the transcoder will transcode the incoming video assuming the effective channel bandwidth is 32kbps, and the TMN8 rate-control scheme is used in the transcoder. In the other case, where the effective channel bandwidth is calculated based on the channel observation and statistical model, the proposed rate-control scheme is used. The end-to-end delay is assumed to the period of one frame interval, i.e., 100ms.

Table 5-1 shows the comparison of the number of frames skipped. Since the first frame of every test sequence is encoded as I frame and transcoded also as I frame, which has much higher bit number than P frames, so several P frames after the first frame are skipped in

order to meet the end-to-end delay requirement. As we can see, with the help of the channel model, we are able to save more than 60% of the frames skipped for all the test sequences.

Table 5-1. Comparison of the number of skipped frames in different algorithms

Video Sequence	TMN8 without Channel Model	Proposed Scheme with Channel Model	Percentage of Frames Skipped Reduction
"Carphone"	24	8	66.7%
"Foreman"	27	8	70.4%
"Salesman"	29	7	75.9%
"Table tennis"	27	10	63.0%

Second, we want to assess the resulting performance degradation, when the channel model is inaccurate. We still use the error patterns generated by the transition probability matrix P as defined in Section 4, i.e.,

$$P = \begin{bmatrix} 0.9909 & 0.0091 \\ 0.0526 & 0.9474 \end{bmatrix}, Packet\ Error\ Rate = 0.1475$$

However, the transcoder uses inaccurate two-state models that are different from the correct model in transition probabilities as

$$P_{high} = \begin{bmatrix} 0.9909 & 0.0091 \\ 0.0053 & 0.9947 \end{bmatrix}, Packet\ Error\ Rate = 0.6319$$

$$P_{low} = \begin{bmatrix} 0.9998 & 0.0002 \\ 0.0526 & 0.9474 \end{bmatrix}, Packet\ Error\ Rate = 0.0038$$

P_{high} corresponds to a channel with higher packet error rate than that of P, while P_{low} corresponds to a channel with lower packet error rate than that of P. The rate-control scheme uses the mismatched probability matrix to control the output bit-rate. Based on the packet size and frame numbers, we can calculate the average channel capacity of the four channel traces. The result is 27.39Kbps. We then compare the number of frame skipping, the number of underflows occurred in the transcoder buffer and the actually generated bit-rate. The result is illustrated in Table 5-2.

Table 5-2. Comparison of the number of skipped frames with different probability matrices

Video Sequences	P			P_{high}			P_{low}		
	Skipped frames	underflow	Bit-rate (Kbps)	Skipped frames	underflow	Bit-rate (Kbps)	Skipped frames	underflow	Bit-rate (Kbps)
"Carphone"	8	1	28.46	7	172	20.05	8	1	28.63
"Foreman"	8	1	28.51	8	164	20.79	9	1	28.68
"Salesman"	7	1	28.42	7	172	19.77	8	1	28.59
"Tabletennis"	10	2	28.82	11	155	21.68	12	2	29.15

From Table 5-2, we can see that when P_{high} is used in the rate-control scheme, the transcoder is pessimistic about the channel. Therefore, a lot of underflow occurs in the transcoder buffer, and the generated bit-rate is much lower than the channel capacity, because the transcoder shoots fewer bits than what can actually be transmitted by the channel. While when P_{low} is used, the transcoder is optimistic about the channel. The generated bit-rate is close to the channel capacity, but more frames are dropped than when P and P_{high} are used because more bits are generated than what can actually be transmitted by the channel. From the above experimental results, we can see that the accuracy of the channel model directly affects the performance of the proposed rate-control scheme.

5.4 Summary

In this chapter, we proposed to use a video rate adaptation transcoder in a retransmission based real-time wireless video system. By analyzing buffering implications of inserting a video transcoder within the real-time wireless video system, we derived the conditions that the transcoder buffer has to meet in order to prevent the decoder buffer from underflowing. Based on these conditions, we proposed a new rate-control scheme for transcoding and transmitting a H.263 encoded real-time video stream over a wireless channel with bursty channel errors. In the proposed algorithm, the effective channel bandwidth of the target VBR channel is estimated based on a Markov model and channel

observation. At the frame layer, the target bit for every frame is determined considering the estimated channel bandwidth and end-to-end delay requirements. At the macroblock layer, the frame bit budget is allocated to every macroblock through the bit allocation algorithm proposed in Chapter 4. Experimental results have shown that the proposed algorithm can effectively determine the frame bit target according to the variable channel bandwidth. Therefore, fewer frames are skipped than the TMN8 scheme. The proposed algorithm can be integrated in a wireless video transmission system and provide flexible control over the bit-rate of compressed video streams.

Chapter 6. Joint Source and Channel Transcoding and Streaming over Wireless Channels

In this chapter, we concentrate on using rate adaptation transcoding techniques and ARQ error control scheme for transmitting pre-encoded video over wireless channels. Different from real time video applications, in networked playback applications, the video source is already encoded and stored on the server side. Therefore, more information, such as video bit profile, scene variance, etc., is available after the video is originally encoded. By sending this information to the transcoder, it can be used to get better transcoding quality and control. In addition, in this kind of application, a longer initial delay is usually acceptable, which gives the transcoder more flexibility.

6.1 Buffer and Rate Analysis

In this chapter, we define a pre-encoded video streaming system in which the pre-encoded high quality video program is transcoded, transmitted, decoded and displayed within some delay interval. A block diagram of the whole system is illustrated in Figure 6-1. In this section, we will analyze the effect of the delay and buffer constraints on the transcoding ratio.

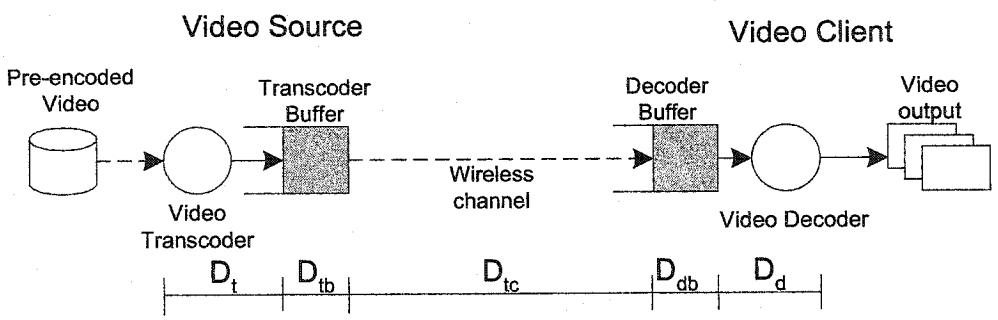


Figure 6-1. Basic component of the defined video streaming system

In the defined system as illustrated in Figure 6-1, we assume that a video is transcoded and transmitted at a fixed frame rate F. At the decoder side, the video is decoded and displayed at the same frame rate F. Similar to the analysis in the previous chapter, in the studied system, we are primarily concerned the delays introduced by the transcoder buffer and decoder buffer, because they can be much larger than the transmission delays and the amount of buffering in the video system can strongly affect the video quality and end-to-end delay.

After the first frame is transcoded, it will be sent into the transcoder buffer, which is empty at this time. In general, it is possible for transcoder buffer underflow to occur if transmission starts at the same time as the transcoder puts the first bit into the buffer. On the other hand, the decoder buffer, which is also empty for the first frame, underflow will occur if decoding starts at the same time as the first bit of the first frame arrives. In practice, this is prevented by starting the transmission and decoding after a certain initial delay. Therefore, the end-to-end delay for the first frame is given by the sum of the initial delays. To simplify the analysis, we define the system initial delay D, i.e., the first frame will be transcoded at time t and will be decoded and displayed at time $t+D$. Then, the maximum delay that any following video frame will experience has to remain constant as D. In real-time interactive applications, the end-to-end delay, D, must be kept less than a certain limit, such as 100ms in video conferencing application. In precoded video streaming applications, the end-to-end delay can be much longer. However, short initial delay is still a desired target. At the same time, long initial delay is equivalent to larger decoder buffer, which we want to avoid. Under this constraint, the system will function normally as long as the decoder buffer does not overflow and underflow, which prevents

the video data from being lost and guarantee that the decoder has received the data of a video frame before it is scheduled to be displayed, respectively.

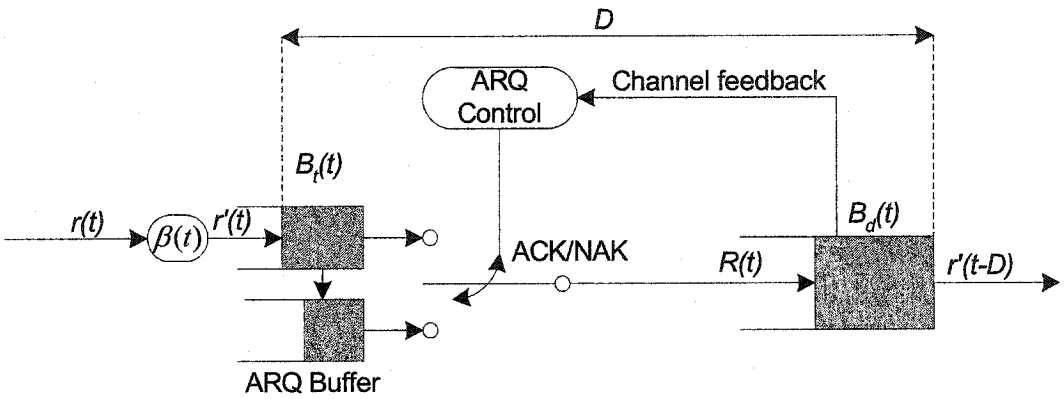


Figure 6-2. Buffering with variable transcoding ratio for VBR channel

As illustrated in Figure 6-2, the precoded video is sent into the transcoder at the source coding rate *r(t)*. Actually, because all data of the pre-encoded video are available at the server side, the transcoder can work much faster than F frames per second. To simplify the analysis, here, we assume that the transcoder only transcodes F frames in one second. Similar as in the previous chapter, the transcoding is modeled as a scaling function $\beta(t)$ which is multiplied by *r(t)* produces the transcoded video at rate $r'(t)$, i.e.,

$$r'(t) = \beta(t) \cdot r(t) \tag{6-1}$$

The effect of multiplying $\beta(t)$ by $r(t)$ can be seen as equivalent to reducing the number of bits used in the video frame transcoded at time t. We define B_i^t, and B_i^d as the instantaneous occupancy of the transcoder and decoder buffers, respectively. First, we discretize the problem by defining E_i (i=1, 2, ...) to be the number of bits generated by the transcoder in the i-th frame interval $[(i\text{-}1)T, iT)$, where T is the duration of one frame interval. Therefore,

$$E_i = \int_{(i-1)T}^{iT} r'(t)dt. \tag{6-2}$$

Similarly, let R_i be the number of bits that are transmitted during the i-th frame interval:

$$R_i = \int_{(i-1)T}^{iT} R(t)dt. \tag{6-3}$$

Assuming the transcoder buffer is empty at time t=0, then the transcoder buffer occupancy after transcoding the i-th frame is

$$B_i^t = \sum_{j=1}^{i} E_j - \sum_{j=1}^{i} R_j = B_{i-1}^t + E_i - R_i. \tag{6-4}$$

After the decoder begins to receive data, it waits D frame intervals before starting to decode and play. Then, the decoder buffer occupancy after the i-th frame interval is

$$B_i^d = \begin{cases} \sum_{j=1}^{i} R_j, & i \le D \\ \sum_{j=1}^{i} R_j - \sum_{j=1}^{i-D} E_j = B_{i-1}^D + R_i - E_{i-D}, & i > D \end{cases} \tag{6-5}$$

The transcoder can calculate the decoder buffer fullness if D is predetermined or sent explicitly as a decoder parameter. The system will function normally as long as the decoded buffer does not underflow within the end-to-end delay constraint. Therefore, B_i^d should be greater than zero at any time.

6.2 Transcoding Ratio Constraints

We now combine equations from section 6.1 to obtain conditions necessary to prevent transcoder and decoder buffers underflow. To prevent transcoder buffer underflow, from Eq. (6-4), we have

$$\begin{aligned} & B_i^t = B_{i-1}^t + E_i - R_i > 0 \\ & \Rightarrow E_i > R_i - B_{i-1}^t \end{aligned} \tag{6-6}$$

which is a constraint on the number of bits of every transcoded frame.

In order to prevent the decoder buffer underflow, from Eq. (6-5), we have

$$\begin{aligned} & B_i^d = B_{i-1}^d + R_i - E_{i-D} \geq 0 \\ & \Rightarrow E_{i-D} \leq B_{i-1}^d + R_i, \quad i > D \\ & \Rightarrow E_i \leq B_{i+D-1}^d + R_{i+D}, \qquad i > 1 \\ & \Rightarrow E_i \leq \sum_{j=1}^{i+D-1} R_j - \sum_{j=1}^{i-1} E_j + R_{i+D} = \sum_{j=1}^{i+D} R_j - \sum_{j=1}^{i-1} E_j = \sum_{j=1}^{i-1} R_j + \sum_{j=i}^{i+D} R_j - \sum_{j=1}^{i-1} E_j \end{aligned} \tag{6-7}$$

As we can see from Eq. (6-7), $\sum_{j=1}^{i-1} R_j$ is the total bits that have been transmitted in the past (i-1) frame intervals, and $\sum_{j=1}^{i-1} E_j$ is the total bits of the past (i-1) transcoded frames. These two terms are known by the transcoder. However, $\sum_{j=i}^{i+D} R_j$ is the total bits that will be transmitted in the future (D+1) frame intervals, which is not available at the transcoder if the channel bandwidth is variable.

Therefore, the condition that the bit number of the i-th transcoded frame has to satisfy is

$$\mathbf{max}\left(0\ ,\ R_i - B_{i-1}^t\right) < E_i < \sum_{j=1}^{i-1} R_j + \sum_{j=i}^{i+D} R_j - \sum_{j=1}^{i-1} E_j \tag{6-8}$$

Under this condition, there will be maximum D frames stored in the whole system, because the maximum end-to-end delay is D frame intervals. Then, we have

$$B_i^t + B_i^d \leq \sum_{j=i-D+1}^{i} E_j, \quad \forall i > D \tag{6-9}$$

6.3 Adaptive Transcoding Based on Video Content

6.3.1 Frame types and source video traffic

Current video coding standards, such as MPEG-x and H.26x, use motion compensation to reduce the temporal redundancy between successive frames. Usually, in MPEG-x, one

Group of Pictures (GOP) contains one INTRA (I) frame and several INTER (P, or B) frames in a certain pattern. An I frame has no motion compensation performed on it but it is very important because motion compensation of the following frames is dependent on the quality of the I frame. P frames use the previous I, or P frames as reference for motion compensation and also are used as reference frames for other INTER frames. B frames use both the previous and successive I, or P frames as references for motion compensation but B frames are not used as reference frames for INTER frames. Because of this hierarchy in coding, visual quality for different frame types has different effect on the quality of the whole video stream with the priority being $I > P > B$. Therefore, an effective rate-control scheme should treat different types of frames unequally, in order to improve the quality of the whole video.

For wireless applications, the H.263 standard is more popular due to their focus on low bit-rate. Different from MPEG-1, 2, in which I frames are periodically used mainly for indexing, in the H.263 standard, I frames are seldom used, just to refresh the visual quality. When a fixed quantization parameter is used for coding all frames in a video sequence, the visual objective quality (PSNR) will tend to be near constant while the generated bit-rate will be liable to fluctuate due to the scene content and different frame types. Thus, if the pre-encoded bit-stream generated by the encoder is kept very close to a constant rate, usually the purpose of doing so is for easy transmission over a CBR channel, there will be a penalty in terms of quality. In one of our experiments, we encoded a video sequences with unified quantization parameters for all I, P, B frames. Only the first frame is coded as an I frame and all other frames are coded as B, and P frames in a certain pattern (IBBP). Figure 6-3 shows the number of bits per frame of this

stream. As we can see, I frames usually have the largest amount of bits, B and P frames have much fewer bits than an I frame. We can also notice that when scene changes happen, the bit-rate of the first INTER frame (we called it anchor frame) after the scene change is very close to that of the I frame. This is because when a scene change happens, motion estimation can not find the best matching blocks in the previous frame, thus most MBs are encoded as INTRA blocks. We also found that the average bit number for B frame is usually half of that for P frame within a scene segment.

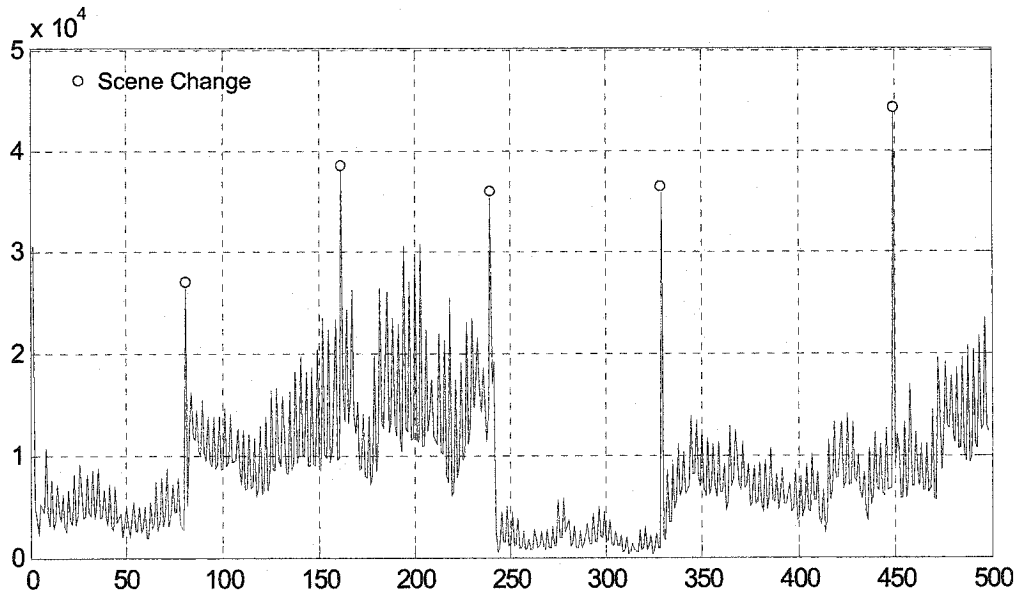


Figure 6-3.Source coding rate of a typical H.263 coded video.

The source bit-rate can be recorded when the original video is encoded and saved with the encoded video as side information. Some research [25] has shown that for the same type of frame encoded with a unified quantization parameter, the generated frame size indicates the scene variations and motion activities. Therefore, in the proposed scheme, we directly use the source bit-rate as a parameter to measure the video content and to calculate the transcoding bit budget for every frame.

As pointed out by [13], in transcoding of precoded video, the bit target for every frame should not be simply calculated as the product of the number of bits of the incoming frame and the ratio between the output and input bit-rates of the transcoder. In one of our experiment, we encoded the same video sequence with two different quantization parameters and the bit-rates of the generated bit streams are 263kbps and 667kbps. The obtained bits/frame ratios are illustrated in Figure 6-4. The scaling factor α of the two video streams is about 0.4, however, as we can see from Figure 6-4, the number of bits B_1^i, B_2^i used to encode the corresponding picture of each video stream are not related through the same scaling factor α, i.e., $N_2^i \neq \alpha N_1^i$.

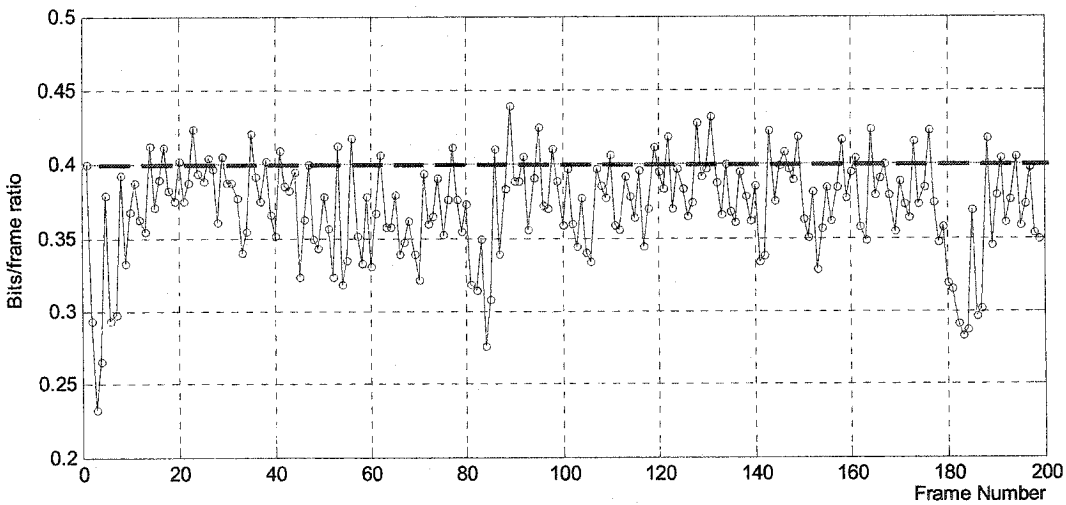


Figure 6-4. Relation between the bits/frame of two bit streams encoded at two different rates.

If the objective of transcoding is to make the quality of pictures to be near constant at the reduced rate, then the scaling factor of such a transcoder should follow the curve shown in Figure 6-4.

6.3.2 Adaptive frame layer rate-control

Different from real time interactive video applications, such as video conferences, in which the end-to-end delay of each frame must be kept within certain limit, the requirement for end-to-end delay in the streaming and playing precoded video application is relaxed. In this type of application, an initial startup delay is allowed and a buffer at the decoder will smooth the delay jitter. Therefore, as long as the decoder buffer is not underflowing, the end-to-end delay for every frame can be different.

The current frame layer rate-control schemes in the H.263 standard, such as TMN8, are dedicated for transmitting real time video in low delay over a constant bit-rate channel. The frame budget for every frame is only determined by the buffer occupancy and channel bandwidth. Therefore, nearly constant bit budget is allocated to each frame and each frame will experience similar end-to-end delay. Different from the frame layer rate-control scheme in TMN8, the proposed scheme determines the transcoded frame bit budget considering the scene change, frame types and source coding rate of the pre-encoded video. In the design of the bit-rate-control algorithm, since an exact curve such as the one in Figure 4 is not available to the transcoder, the following rules are adopted to determine the transcoded frame bit budget.

- If the incoming frame is an INTRA coded frame, it will be transcoded as an INTRA frame with unified quantization parameter used for every MB.
- P and B frames are treated differently by the transcoder. The bit budget for B frames will be half of that for P frames, on the average.
- For the same type of INTER frames, the frame bit budget is determined considering the effective channel bandwidth, end-to-end delay and the original source traffic.

- If a scene change is detected, the anchor frame will be transcoded as an INTRA frame with unified quantization parameter for every MB.

At the frame layer, we first determine the bit budget for a GOP, which has a fixed number of frames. Since the INTRA frames are seldom used in H.263, we assume here that the GOP only includes INTER frames (P and B) in a certain pattern. The bit budget for encoding a GOP is defined as:

$$B_{GOP} = \frac{(N + \alpha D) \times R}{F} + \Delta$$
$$\alpha = \begin{cases} 0.7, & first\ GOP \\ 0, & otherwise \end{cases} \tag{6-10}$$

where N is the number of frames in a GOP, R is the effective channel bandwidth, F is the frame rate, D is the required end-to-end delay measured in frame intervals, and Δ is the bit budget unused from the previous GOP. In order to take full advantage of the initial delay D, we introduce parameter α, which is empirically set to 0.7 for the first GOP. As we can see in Eq. (6-10), for the first GOP, N frames will have more bit budget than that the channel can actually transmit in the first N frame intervals. Then, when the end-to-end delay is relaxed by the application, the visual quality will be improved. For other following GOPs, Eq. (6-10) assumes that N frames in a GOP will be transmitted within N frame intervals. The calculation of the effective channel bandwidth, R, will be introduced in the following section. Inside a GOP, the frame bit budget is determined according to frame types and the original source bit-rate following the above rules. Based on the above rules and the pattern of P, and B frames within a GOP, we can know the bit budget for all P frames and B frames in a GOP respectively as follows:

$$B_{GOP-B} = \frac{N_B}{2N - N_B} B_{GOP}, \quad B_{GOP-P} = \frac{2N_p}{N_p + N} B_{GOP} \tag{6-11}$$

where N_P, N_B are the numbers of P, and B frames in a GOP. For the same type of frames, bits are allocated to every frame according to the original source bit-rate. Then, for P frames, we have

$$B_P(i) = \frac{\overline{B}_P(i)}{\sum_{j=0}^{N_P-1} \overline{B}_P(j)} B_{GOP-P}, \quad B_B(i) = \frac{\overline{B}_B(i)}{\sum_{j=0}^{N_B-1} \overline{B}_B(j)} B_{GOP-B} \tag{6-12}$$

where $B_P(i), B_B(i)$ are the bit budgets for the i-th P and B frames, $\overline{B}_P(i), \overline{B}_B(i)$ are the bit-rates of the i-th P and B frames in the precoded video, respectively. When the frame bit budget is determined, the macroblock layer bit allocation algorithm is responsible for calculating the quantization parameter for every MB and make the generated bits as close as possible to the bit budget. However, we believe that the varying quality produced by varying quantization parameters is far less than that produced by varying the frame bit budget. Therefore, in this work, we directly use the macroblock layer bit allocation algorithm in TMN8. Details about the macroblock layer bit allocation in TMN8 can be found in [25].

6.3.3 Adaptive frame skipping

As indicated in Eq. (6-9), at any given time, the sum of transcoder buffer and decoder buffer occupancy must be greater than the bits for the D frames that have been transcoded but not yet decoded. Otherwise, some frames will experience longer end-to-end delay than D. The reason of violating Eq. (6-9) is as follows. First, due to the control error of the macroblock layer bit allocation algorithm, the generated frame size may not be exact as the bit budget. Second, due to the frame type change, such as transcoding P frames to I

frames, the generated frame size may be much greater than the frame bit budget. Third, because the algorithm has to estimate the effective channel bandwidth of the future D frame intervals, the accuracy of the estimation will also affect the result of Eq. (6-10). In order to guarantee that the end-to-end delay of every frame is less than D, in the proposed algorithm, after transcoding every frame, we will evaluate Eq. (6-9). If there are more than D frames stored in the transcoder and decoder buffer, the next frame may be dropped according to its type.

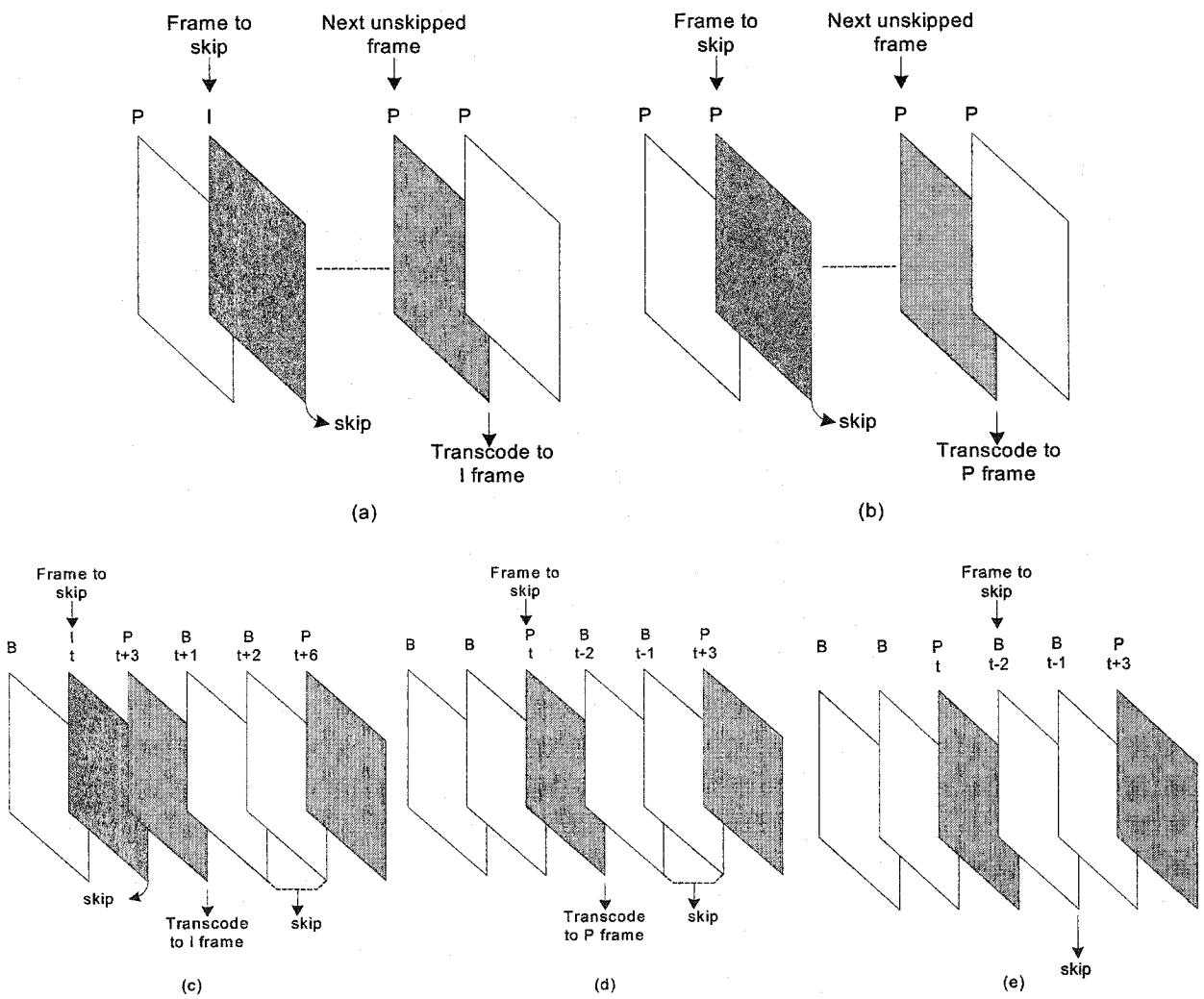


Figure 6-5. Adaptive frame skipping

For example, if the original video was encoded in IPPP pattern and the frame that needs to be skipped is an I frame, then this I frame will be skipped and the next unskipped frame (P) will be transcoded to an I frame with unified quantization parameter as in Figure 6-5 (a). If the frame that needs to be skipped is a P frame, then this P frame will be skipped and the next unskipped frame will be transcoded to a P frame as in Figure 6-5 (b). If the original video was encoded in IBBP pattern, we have to consider the frame reordering that will happen at the decoder. As illustrated in Figure 6-5 (c), suppose the I frame will be displayed at time t, then the following two frames that need to be displayed will be B frames at time $t+1$ and $t+2$. However, when the original video is encoded, the P frame, which is scheduled to be displayed at time $t+3$, is encoded and stored prior to the two B frames. When the decoder decodes this video stream, it will first decode the P frame, and then the following two B frames can be decoded and displayed. Therefore, in our transcoder, if the frame that needs to be skipped is an I frame, we will skip this frame and transcode the following P frames as an I frame. At the same time, the following two B frames will also be skipped as in Figure 6-5 (c). If the frame that needs to be skipped is a P frame, then instead of skipping this P frame, we keep this frame but skip the following two B frames as in Figure 6-5 (d), because, as we mentioned before, the bit-rate for a P frame is usually two times of that of a B frame. If the frame that needs to be skipped is a B frame, we then simply skip this frame as in Figure 6-5 (e). When the frame scheduled to be skipped happens to be an anchor frame, based on the same idea, the following rules in Table 6-1 will be applied for frame skipping.

Table 6-1. Adaptive frame skipping decision*

Frame Type	Scheduled to Skip?	SCD>=40%?	Action
I	Yes	Yes	Keep the I frame, skip the next PBB frames
I	Yes	No	N/A.**
I	No	Yes	Keep the I frame.
I	No	No	N/A.**
P	Yes	Yes	Transcode the P frame as I frame, skip the next BB frames.
P	Yes	No	Keep the P frame, skip the next BB frames.
P	No	Yes	Transcode the P frame as I frame.
P	No	No	Keep the P frame.
B(1)	Yes	Yes	Skip the 2 B frames, transcode the next P frame as I frame.
B(1)	Yes	No	Skip the B frames.
B(1)	No	Yes	Skip the 2 B frames, transcode the next P frame as I frame
B(1)	No	No	Keep the B frame
B(2)	Yes	Yes	Skip the B frame, transcode the next P frame as I frame
B(2)	Yes	No	Skip the B frame.
B(2)	No	Yes	Skip the B frame, transcode the next P frame as I frame
B(2)	No	No	Keep the B frame.

*: Assume the frame pattern is IPBBPBBPBB... B(1), B(2) stand for the first and the second B frame in a PBBP pattern, respectively.
**: when the frame type is I, the SCD will be definitely greater than 40%.

6.3.4 Algorithm

Combining the ideas in the previous sections, we propose the joint source-channel rate adaptation algorithm. In this section, we summarize the whole process of the algorithm. The following notations are used:

B_{GOP} target number of bits assigned to a GOP;

F target frame rate in frames per second;

N frame number in a GOP;

$\overline{N_P}$ number of uncoded P frames in a GOP;

$\overline{N_B}$ number of uncoded B frames in a GOP;

$\overline{B_{GOP}}$ target number of bits left for the uncoded frames in a GOP;

$\overline{B_{GOP-P}}$ target number of bits left for the uncoded P frames in a GOP;

$\overline{B_{GOP-B}}$ target number of bits left for the uncoded B frames in a GOP;

$B_P(i)$ bit target for the i-th P frame;

$B_B(i)$ bit target for the i-th B frame;

The detailed algorithm is as follows:

Step 1: *Determine the current channel state*: we calculated the average ratio of successfully transmitted bits to the average number of total transmitted bits in the past L frame intervals. If the ratio is greater than a threshold H, we decide that the channel is in the good state. In our simulation, we set L as 10 and H as 0.9 empirically.

Step 2: *Calculate the estimated channel bandwidth*: Depending on the current channel state, we can use the Markov model to find the probability of the correct transmission in the next D frame intervals. Then we can calculate the estimated channel bandwidth, R, for the next D frame intervals as introduced in the previous chapter.

Step 3: *Calculate the bit budget for a GOP.* At the beginning of every GOP, we first calculate the bit budget for the GOP as in Eq. (6-10). In our experiments, we set N to 30. At the beginning of a GOP, the bit budget for the uncoded frames in the GOP, $\overline{B_{GOP}}$, is equal to B_{GOP}.

Step 4: *Calculate the bit budget for every frame within a GOP according to frame types, scene context and source coding rate.* If the incoming frame will be transcoded as INTER frame, the frame bit target for this frame will be calculated according to the source coding rate. Since the original source rate and frame pattern are known when the original video is coded, the frame bit target can be calculated by applying the rules in section 4.3. Similarly as Eq.(4-19) and (4-20), we can have

$$\overline{B_{GOP-P}} = \frac{2 \times \overline{N_P} \times \overline{B_{GOP}}}{2 \times \overline{N_P} + \overline{N_B}}, \quad \overline{B_{GOP-B}} = \frac{\overline{N_B} \times \overline{B_{GOP}}}{2 \times \overline{N_P} + \overline{N_B}} \tag{6-13}$$

$$B_P(i) = \frac{\overline{B}_P(i)}{\sum_{j=i}^{\overline{N_P}-1} \overline{B}_P(j)} \overline{B_{GOP-P}}, \quad B_B(i) = \frac{\overline{B}_B(i)}{\sum_{j=i}^{\overline{N_B}-1} \overline{B}_B(j)} \overline{B_{GOP-B}} \tag{6-14}$$

***Step 5**: Adjust the frame bit budget.* By using Eq. (6-9), the frame bit budget needs to be adjusted considering the buffer and end-to-end delay constraints.

***Step 6**: Transcoding the incoming frame.* In this work, the TMN8 macroblock layer bit allocation algorithm is adopted to calculate the quantization parameter of every MB.

***Step 7**: Update.* Transcoder and decoder buffer occupancy is updated as in Eq. (6-5) and (6-6). Other parameters, such as $\overline{B_{GOP}}$, $\overline{N_P}$, $\overline{N_B}$ are also updated. Eq. (6-10) will be evaluated to see if the total number of frames stored in the transcoder and decoder buffer is greater than the maximum end-to-end delay, D, or not. If there are more than D frames stored in the system, the successive frame is scheduled to be skipped. The action will be decided according to Table 1. If there are more frames that need to be transcoded, go to *Step 1*, otherwise, stop.

6.3.5 Performance evaluation

In this work, we implement the rate adaptation transcoder and the proposed algorithm based on the public domain software for H.263 [88]. In our experiments, the test sequence as in Figure 6-3 is encoded with using unified quantization parameters, QP=5, for every frame with a frame rate of 10 fps. Only the first frame of the sequences is encoded as an I frame and other frames are encoded as B or P frames. This sequence is

created by cascading several test sequences with different scene content and serves as a high quality pre-encoded video.

In order to simulate the wireless channel with the ARQ protocol, we use a random number generator to generate a random number r, which is uniformly distributed in {0, 1}. Based on the Markov model and transition matrices introduced in the previous chapter, if the current state is S_0, when r is less than P_{01}, the transition from state S_0 to S_1 occurs. If the current state is S_1, when r is less than P_{10}, the transition from S_1 to S_0 occurs. In this way, we generate 6 packet level channel error traces with length of 50,000 packets, as shown in Table 6-2. Every trace represents a possible channel with a specific packet error rate. We do not consider channel coding overhead, and define the packet payload size as 40 bits and the mean burst length is 18 packets. In our experiments, we will use these transition matrices to estimate the effective channel rate.

Table 6-2. Summary of the transition matrices used in our experiments

Mean_Burst_Length = 18 Packets Packet Size = 40 bits				
Average Packet Error Rate (%)	P_{00}	P_{01}	P_{10}	P_{11}
5	0.9971	0.0029	0.0556	0.9444
10	0.9938	0.0062	0.0556	0.9444
15	0.9902	0.0098	0.0556	0.0944
20	0.9861	0.0139	0.0556	0.9444
25	0.9815	0.0185	0.0556	0.9444
30	0.9762	0.0238	0.0556	0.9444

In the transcoder, the proposed joint source-channel adaptive transcoding scheme is used to transcode the original video according to the estimated channel conditions. We assume the channel bandwidth for video payload is 64kbps, 48kbps and 32kbps. Based on the channel feedback and the same channel model that is used to generate the error trace, the transcoder will estimate the effective channel bandwidth after every frame interval. We

simulate the channel coder, transmitter and decoder buffer. Whether a video packet is transmitted successfully to the decoder will be determined by the error trace. A simplified simulation scenario is shown in Figure 6-6.

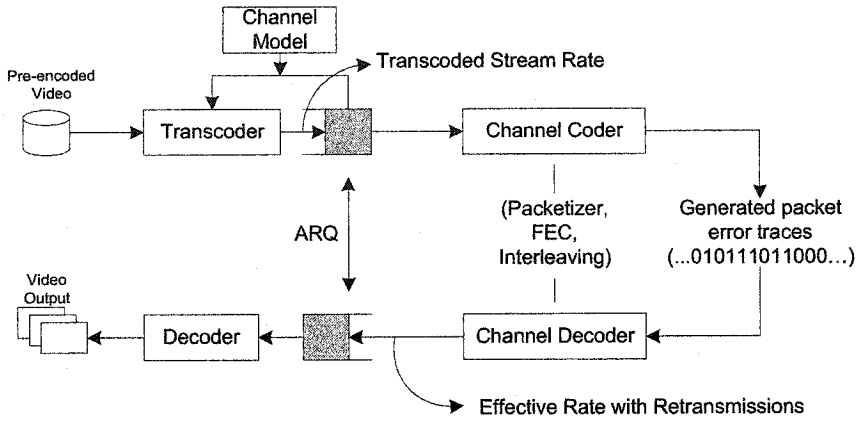
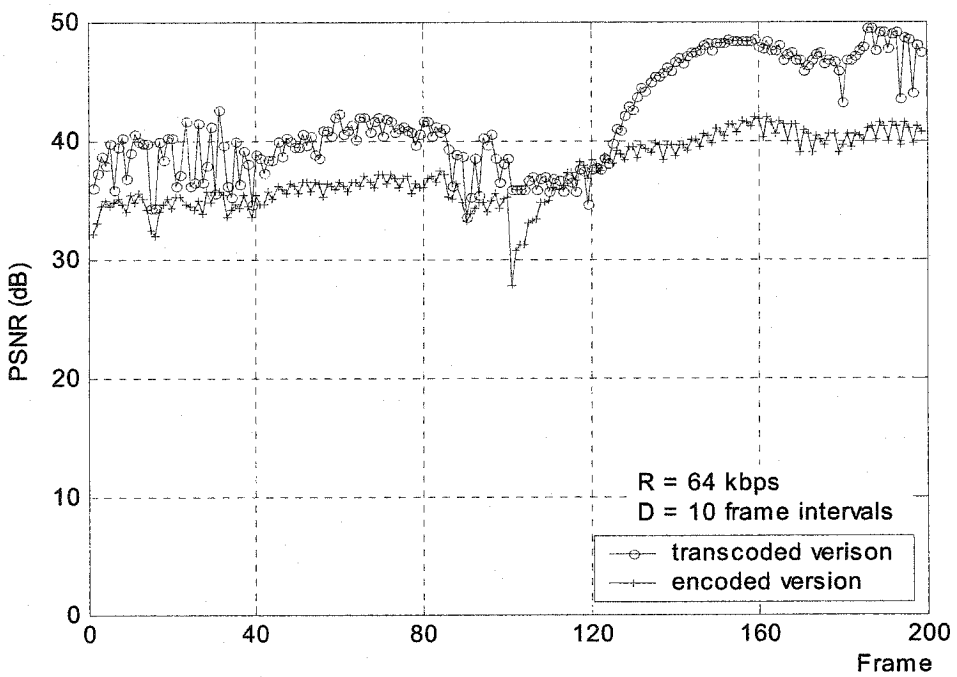
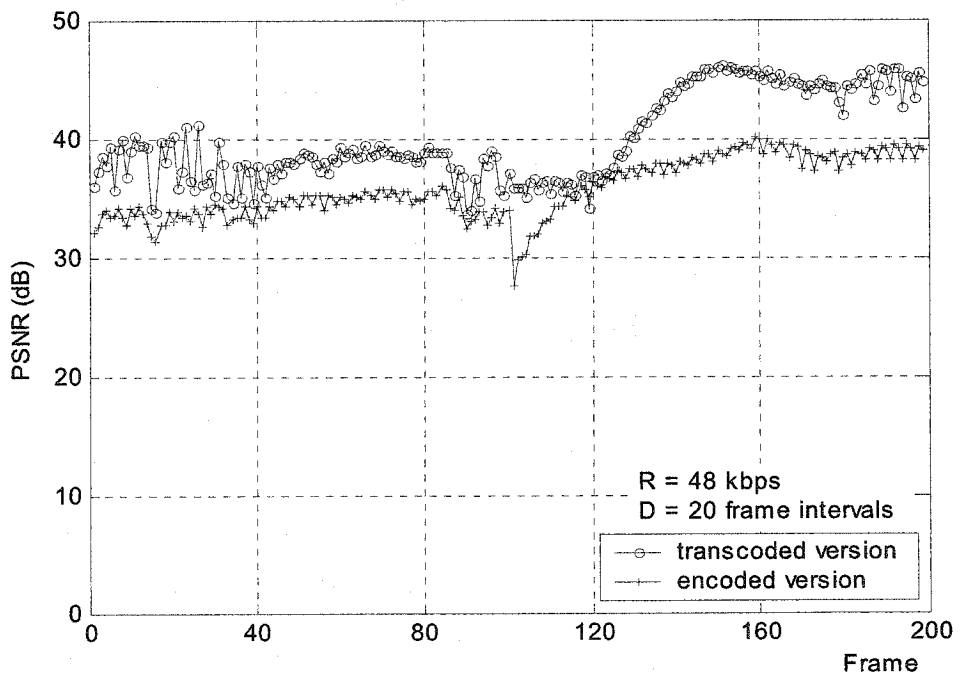


Figure 6-6. System diagram of the proposed ARQ-based transcoding and streaming scheme.

In order to compare the video quality, we also encode the source video sequence at the same rates of 64kbps, 48kbps and 32kbps by using the TMN8 rate-control scheme. At the frame layer, TMN8 allocates near constant bit budget to every frame without considering the frame types and scene context. A frame is skipped if the number of bits accumulated in the buffer after encoding the previous frame is greater than a threshold. To get a fair comparison, we change this threshold to allow longer end-to-end delay, and then there are will be no frame skipping in the encoded video.

The visual quality is measured by calculating the *Peak Signal Noise Ratio* (PSNR) of the transcoded video and the encoded video. The PSNR comparison result for different effective channel rate and end-to-end delay is illustrated in Figure 6-7.


PSNR (dB)
Frame
R = 64 kbps
D = 10 frame intervals
transcoded verison
encoded version


PSNR (dB)
Frame
R = 48 kbps
D = 20 frame intervals
transcoded version
encoded version

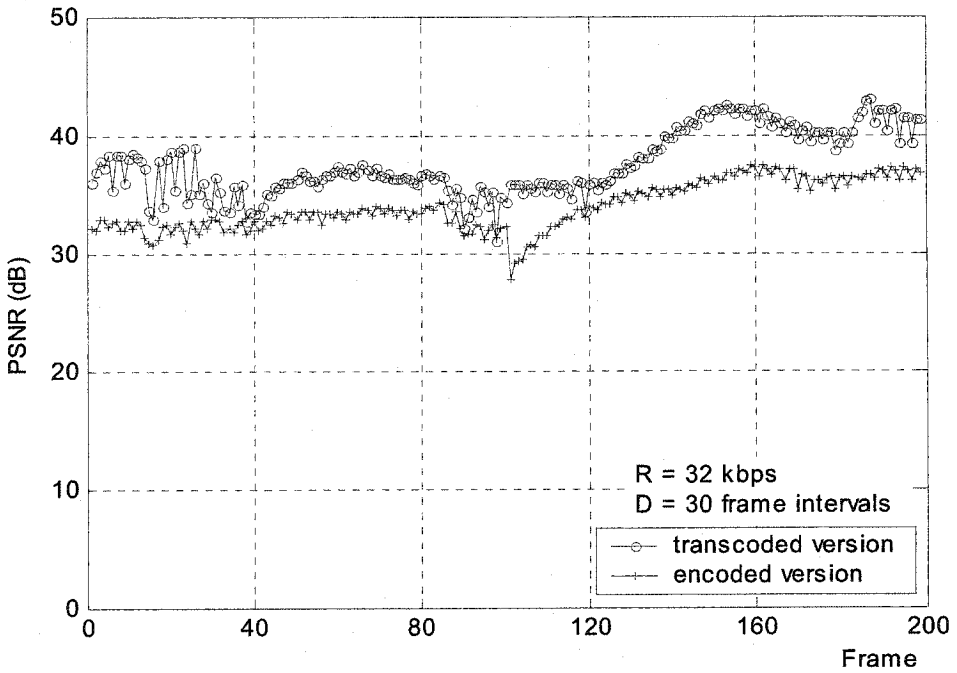


Figure 6-7. PSNR comparison of transcoded video and encoded video at different channel bandwidth and end-to-end delay.

As shown in Figure 6-7, by controlling the number of bits assigned to each frame considering the frame types, we can obtain higher PSNR result as compared with encoding the original video at the same low channel bandwidth, and maximum end-to-end delay. Especially when a scene change happens at the 100-th frame, because the INTER frame at the scene change is transcoded to an I frame, the visual quality of the following frame is much higher than the encoded version.

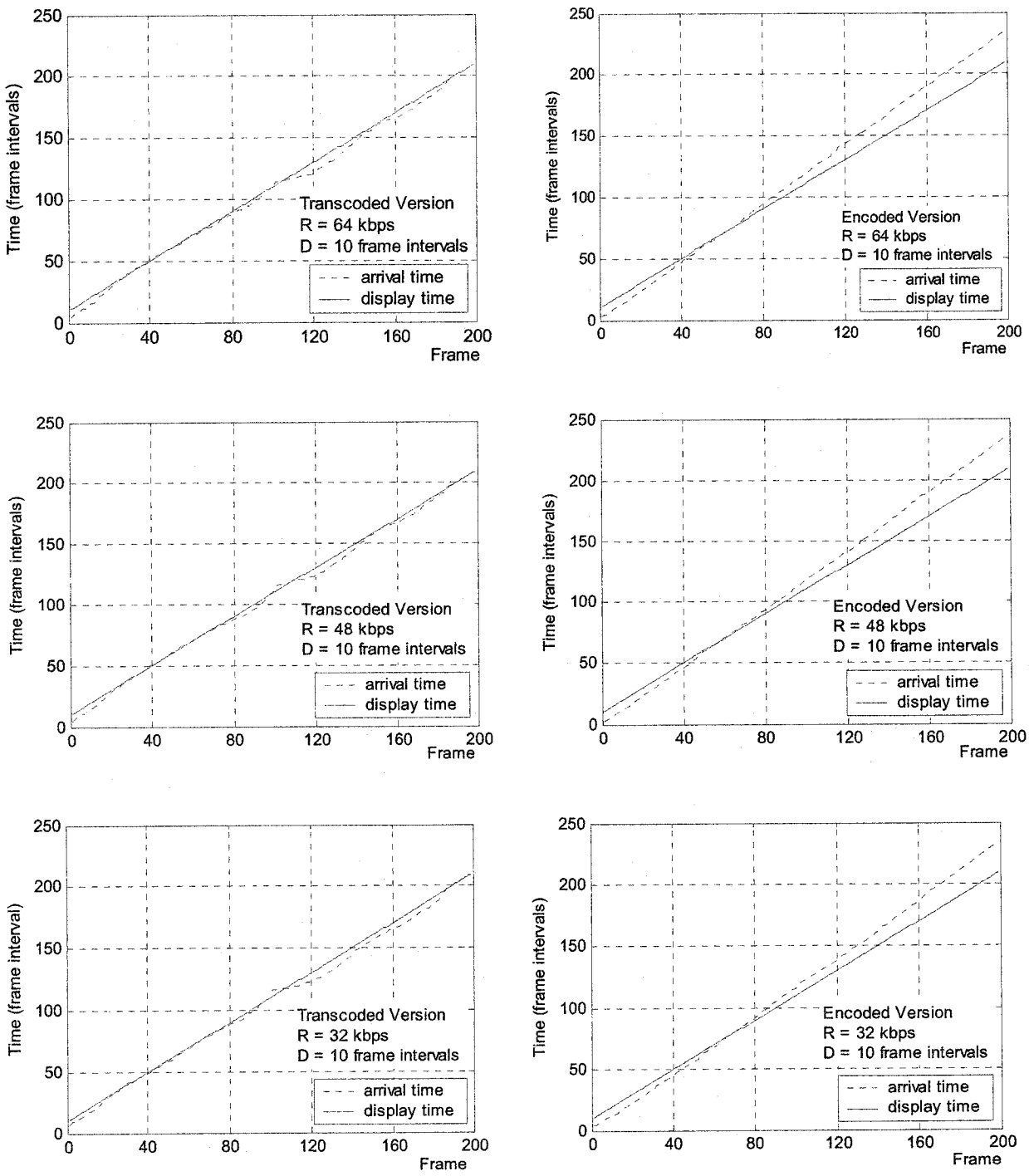


Figure 6-8. Comparison of frame arrival time and scheduled display time.

We also calculate the time of every frame arriving at the decoder buffer and scheduled to be displayed. The comparison result is illustrated in Figure 6-8. At the right side, we also

calculate the result of the encoded video being transmitted over the same channel. As we can see, by rate adaptation transcoding, the arrival time of every frame is very close to the display time. The decoder will not have to suspend decoding and displaying for a long time. However, if the test sequence is encoded with low bit-rate (64kbps, 48kbps, 32kbps) as the target bit-rate, since the effective channel rate is changing over time, after several frames the frame arrival time will be late than its display time, as we can see in Figure 6-8. Therefore, the decoder will have to wait for the next frame coming.

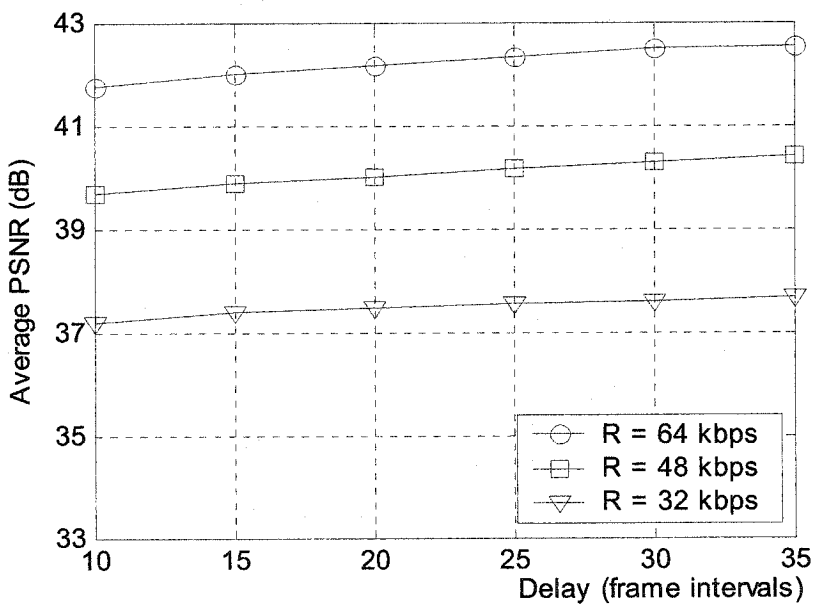


Figure 6-9. Average PSNR vs. end-to-end delay and channel bandwidth.

Figure 6-9 shows the relation among the average PSNR of the whole sequence, the end-to-end delay and the target channel rate. As we can see, for the same target channel rate, when the end-to-end delay is relaxed, the average PSNR increases. Therefore, the proposed trasncoding scheme will be able to take full advantage of the end-to-end delay. However, if we just encode the test sequence at the target channel rate, the video quality

will be fixed after encoding. No matter what the end-to-end delay will be, the quality of the decoded video sequence will not change.

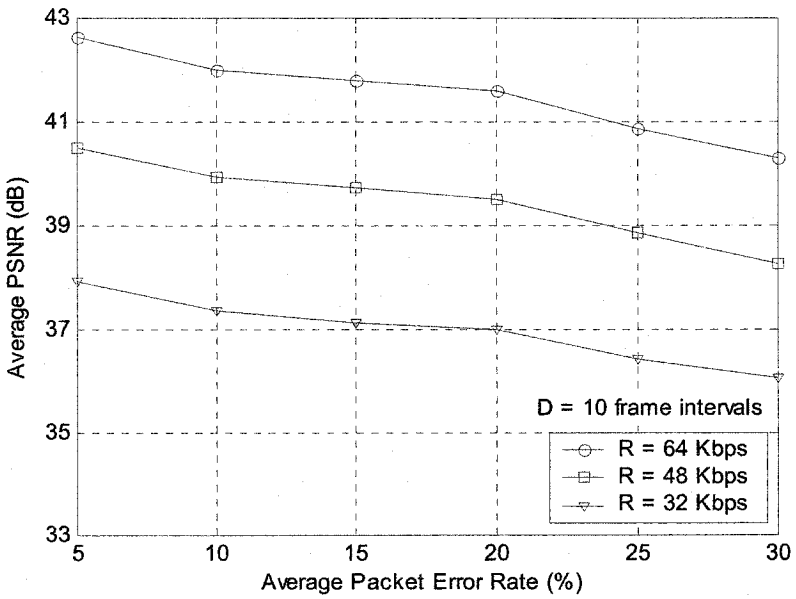


Figure 6-10. Average PSNR vs. average PER and channel bandwidth.

Figure 6-10 shows the relation between the packet error rate and the average PSNR at the same end-to-end delay.

6.4 Summary

In this chapter, we investigated the impact of a video rate adaptation transcoder in a retransmission based wireless video system. By analyzing buffering implications of inserting a video transcoder within the wireless video system, we derived the conditions that the transcoder buffer has to meet in order to prevent the decoder buffer from underflowing and overflowing. In the proposed algorithm, at the frame layer, the target bit for every frame was determined considering the estimated channel bandwidth, end-to-end delay requirements, and visual content. Experimental results have shown that the proposed algorithm can effectively determine the frame bit target according to the

variable channel bandwidth. Therefore, for pre-encoded video transmission scenario, effective channel bandwidth is allocated according to the visual content of the video, hence the transcoded video quality is improved.

Chapter 7. Implementation of a Video Transcoding Gateway Demo System

In order to enable users to access video through wireless networks and handheld devices, we propose using a transcoder as a gateway to dynamically convert the video according to user devices and network connections. In this approach, a content provider provides only one video stream, which is pre-encoded at high quality. The video transcoding gateway performs transcoding dynamically for each user and provides converted video streams. The architecture of the video transcoding system is illustrated in Figure 7-1. In this system, the transcoder is integrated into the video streaming server. It can also be placed as an intermediate node along the transmission path.

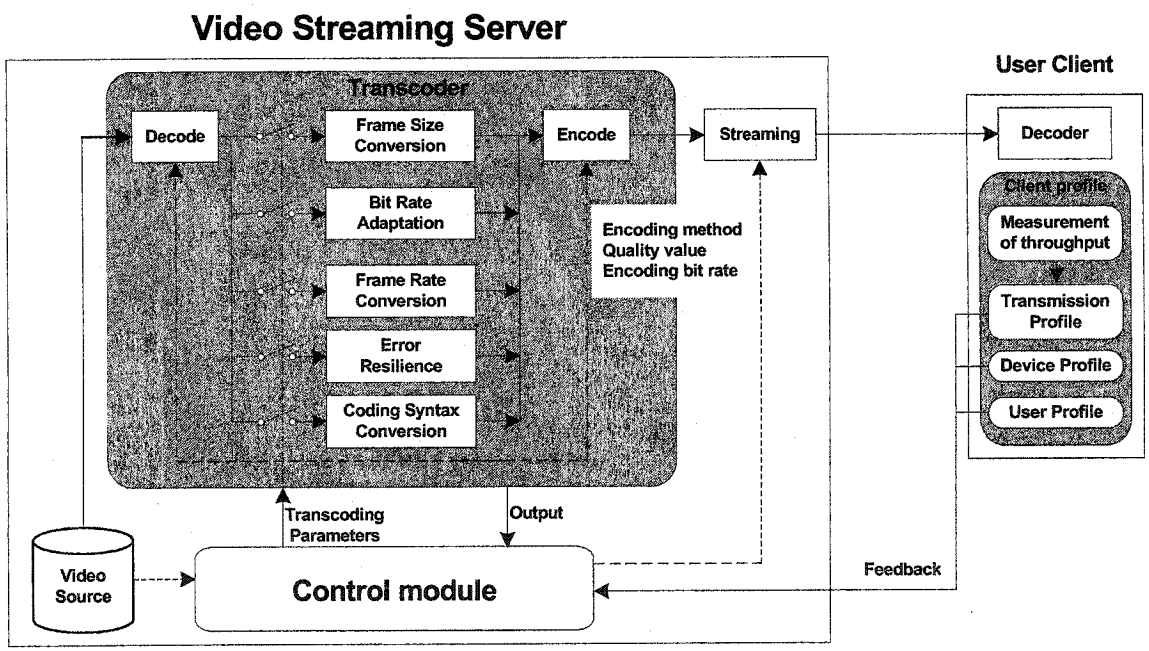


Figure 7-1.Video transcoding system architecture

7.1 System Modules and Features

7.1.1 Client profile

The client profile maintains the following components:

- Device profile includes hardware and software information on the devices, such as display size, processing power and decoder information.
- User profile includes user preference information.
- Transmission profile is responsible for monitoring the dynamic condition of the transmission channel, such as effective channel bandwidth, channel error rate, etc.

The information included in the device profile and user profile will be transmitted to the video stream server before the start of the session. The information included in the transmission profile will be sent to the stream server periodically to control the bit rate adaptation in the transcoder. All of the information in the client profile will be used as parameters for the transcoding process.

7.1.2 Video transcoder

The video transcoder is the actual conversion engine of a video stream. It decodes a video stream, which is pre-encoded at high quality and stored in the video source, and then performs transcoding and re-encoding. Multiple conversions can be realized simultaneously, including frame size conversion, bit rate adaptation, frame rate conversion, error resilience, coding syntax conversion, etc. When fast transcoding architectures are used, it is possible to execute transcoding in real time.

7.1.3 Control module

The control module is responsible for creating a transcoding plan according to the client profile and other information. The transcoding plan will include some transcoding parameters. In order to decide appropriate transcoding parameters, decision must be made

by considering all the factors intelligently. For example, when connection throughput is low, the bit rate of the video needs to be converted. At the same time, in order to ensure the video quality, the frame rate of the video also needs to be reduced. In this way, every frame will have enough bit budget to maintain acceptable visual quality.

7.2 System Design

Based on the system architecture, we implemented the video transcoding gateway demo system. In the implemented system, the transcoding gateway is implemented together with the streaming server, where the pre-encoded videos are stored. The frame size conversion (downscale by 2, from 4CIF to CIF, and CIF to QCIF), frame rate conversion (from 30 fps to 30, 25, 20, 15, 10, 5 fps), and bit rate adaptation modules are implemented. Moreover, we also implemented the feature of color-to-grayscale conversion module, which will convert a color video into black-and-white video. The clients connect to the server through TCP/IP. The TCP is used as the transport protocol considering its reliability. The clients based on desktop and handheld devices (iPaq with WinCE PocketPC 2002) are implemented for decoding and displaying videos.

7.2.1 Server design

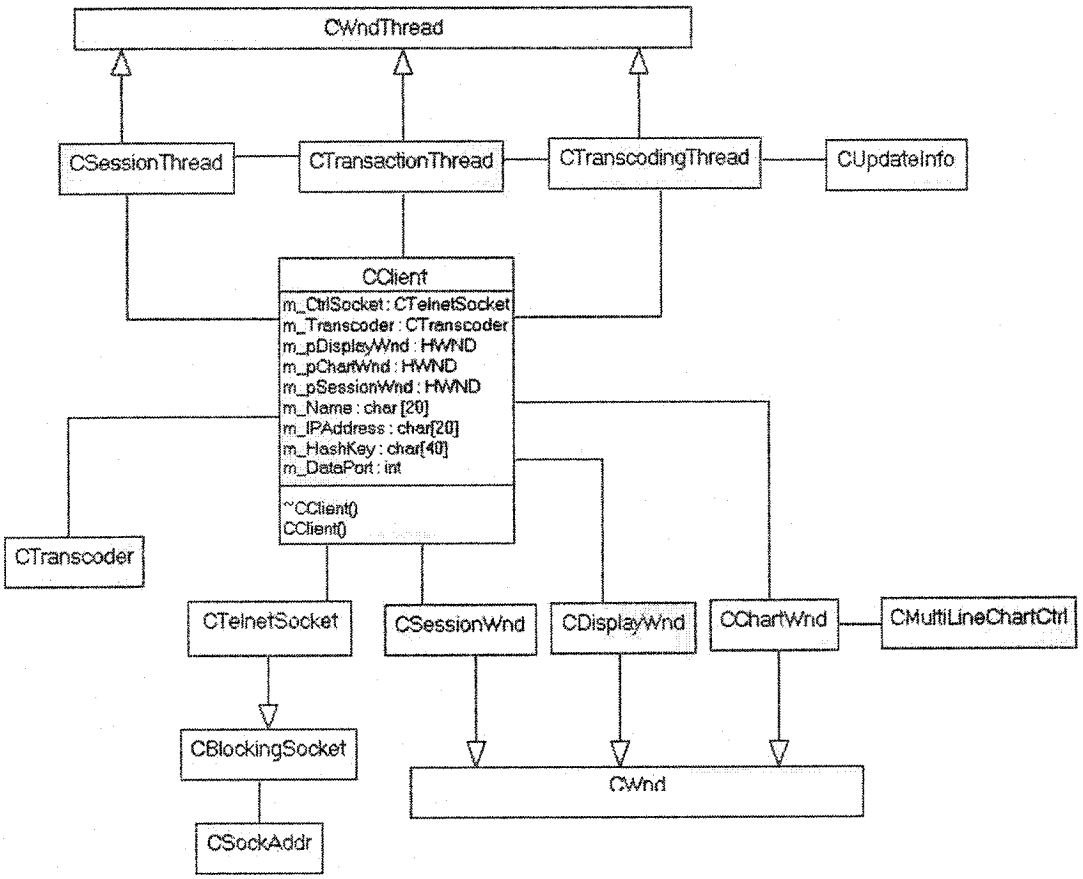


Figure 7-2. Server module class diagram

The main classes at the server side are illustrated in Figure 7-2. The transcoding and streaming server is a multithread socket server. Through a set of threads, multiple client requests are dispatched and processed by calling different transcoders. A set of socket classes are responsible for streaming the transcoded video to different clients. The user interfaces are implemented by a set of window classes.

7.2.2 Client design

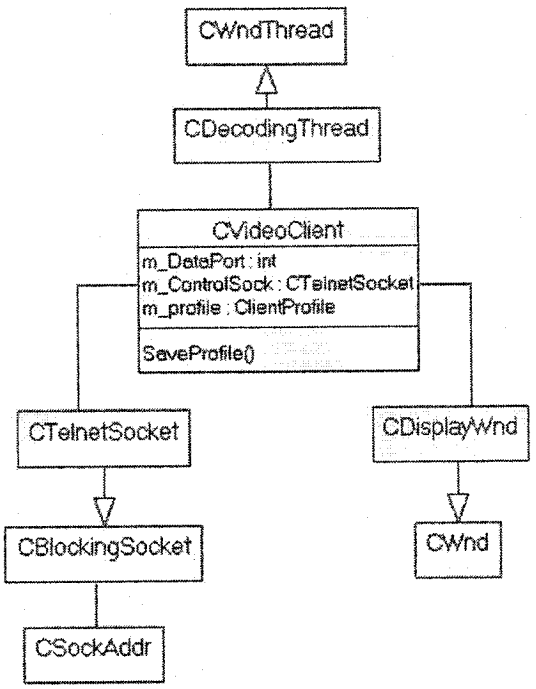


Figure 7-3. Client module class diagram

The classes at the client side are illustrated in Figure 7-3. Similarly, the socket class is responsible for communicating with the server and receiving the transcoded video data. Decoding thread is responsible for decoding the received video and window class is responsible for displaying the decoded video.

7.3 System Screenshots

Figure 7-4 shows a server screen shot when two videos are requested and streamed to two clients. At the lower left corner, a list shows information for all pre-encoded video on the server. The upper left part lists all information about every session, which includes the client IP address, socket ID, video that is requested, current frame number and frame bit rate. By pressing the "Transcoding" button, the transcoding parameters will be displayed. By pressing the "Channel" and "Error Control" buttons, the different channels can be

simulated and different error control parameters can be applied. Please note that this part of function has not been implemented yet. In the session statistics window, the frame bit rate and PSNR will be plotted in two charts. For every session, a page will be inserted into the session statistics window. At the right side, the requested pre-encoded video will be decoded and displayed.

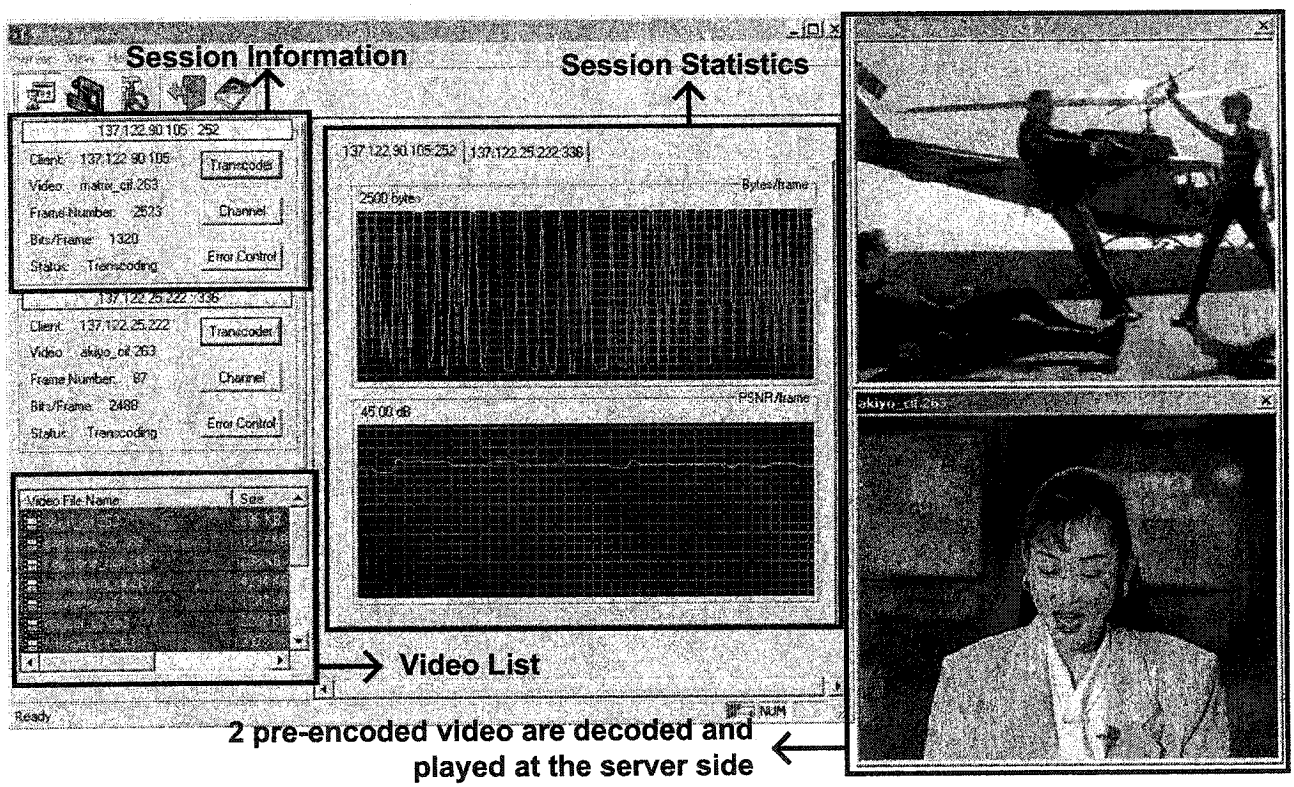


Figure 7-4.Server screenshot with two pre-encoded video are transcoded and streamed

At the client side, the transcoded video is decoded and played. The desktop client screenshot is illustrated in Figure 7-5. The handheld client screenshots are illustrated in Figure 7-6, in which the same video is transcoded into different formats.

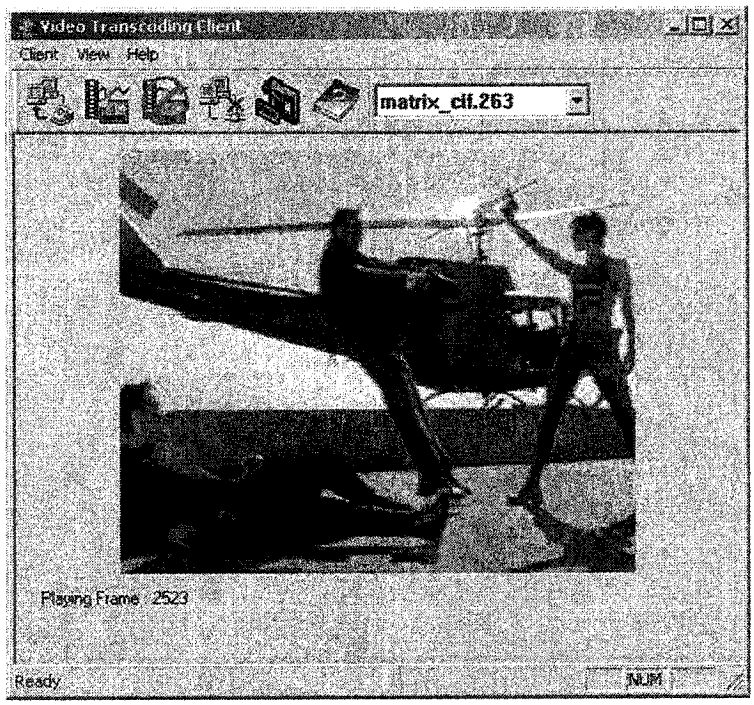
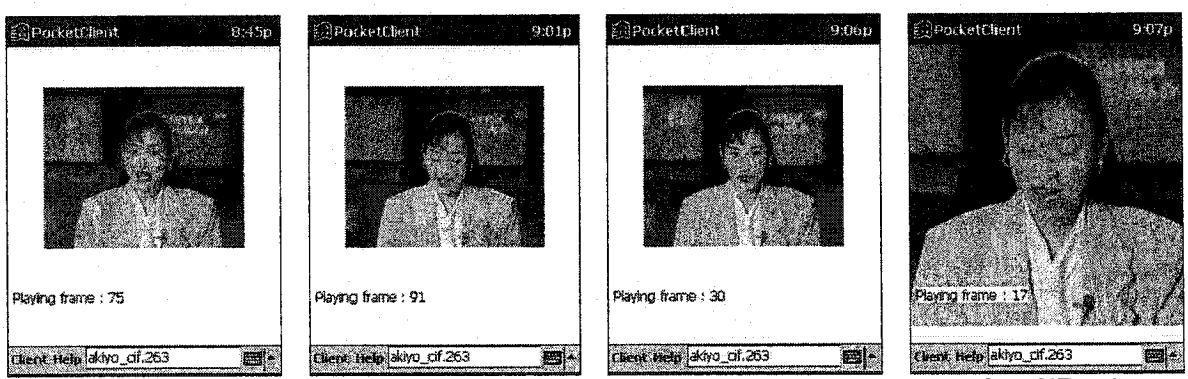


Figure 7-5. Desktop client decoding and playing a video stream.



30 fps, QCIF, color | 10fps, QCIF, color | 30fps, QCIF, Grayscale | 10fps, CIF, color

Figure 7-6. Handheld client decoding and playing a video stream with different formats

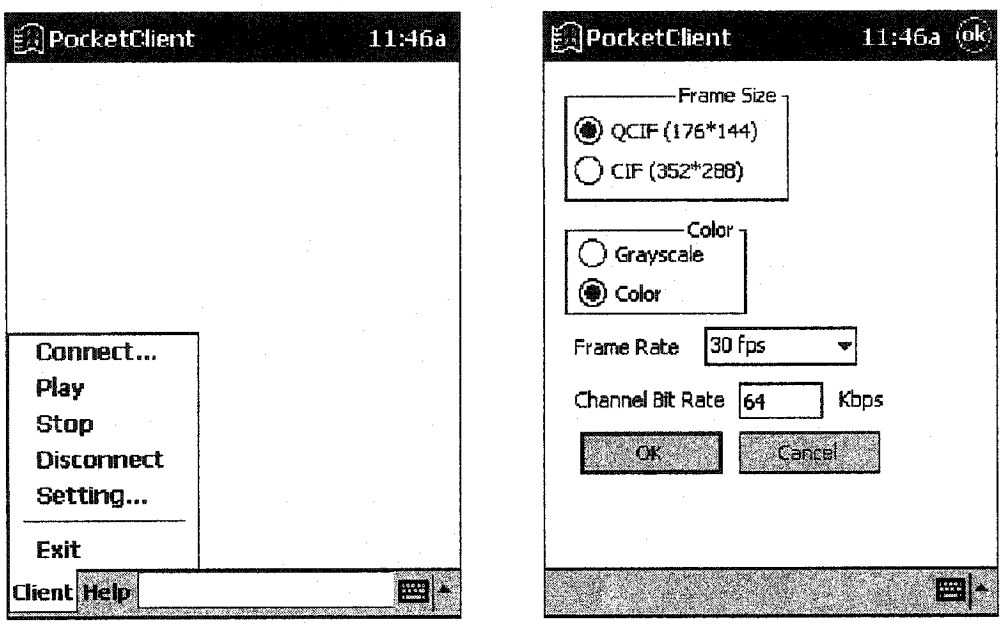


Figure 7-7. Handheld client menu and setting interface.

The user menu and client profile setting interface of the handheld client is illustrated in Figure 7-7.

This demo system can be downloaded from http://www.discover.uottawa.ca/demos.

Chapter 8. Conclusions and Future Research Directions

8.1 Thesis Conclusions

With the increasing number of applications using compressed digital video for transmission over heterogeneous networks, transcoding is becoming an important issue for bit-rate adaptation to meet channel capacities. Transcoding provides flexible bit-rate adaptation, especially if the bit-rate is only known at transmission time and the bit-rate is changing over time. This flexibility is very important when the coded video is transmitted over channels with different capacities. Moreover, wireless video transmission, transcoding is a key technology for providing dynamic adaptation of the bit-rate of compressed video to the effective channel bandwidth.

Motivated by the importance of transcoding in current video technologies, this thesis mainly investigated rate adaptation issues related with transcoding. First, Chapter 2 overviewed typical video transcoding architectures, their performance and complexity. Considering low bit-rate of wireless channels, the bit-rate-control at the macroblock layer must be accurate and tight. Chapter 3 presented an approximate linear bit allocation model based on extensive experimental results. This model indicates that there is an approximate linear relationship between the number of non-zero quantized AC coefficients in a frame and the generated bits. We also provided a theoretical proof of the proposed model. Based on this model, the bit allocation statistics from the incoming compressed video can be reused for the transcoding bit allocation. In Chapter 4, a novel bit allocation algorithm based on the linear bit allocation model was implemented. The

implementation issues were addressed and performance was compared. Considering the visual content and scene variations, we also proposed a frame layer rate-control algorithm for transcoding video based on scene variation and scene changes for CBR channel. The main focus of this thesis was on the problem of transcoding and transmitting of compressed video over wireless channels. In Chapter 5, we first presented a framework for analyzing buffer and rate constraints when a transcoder is introduced in a wireless video transmission system. Based on a simplified Markov model, we presented a methodology for estimating the effective wireless channel bandwidth. Considering the buffer fullness, end-to-end delay requirements, effective channel bandwidth, we proposed a rate-control algorithm for transcoding real-time video over wireless channel. In Chapter 6, for pre-encoded video distribution, we proposed another rate-control algorithm, which dynamically adjusts the frame bit budget considering the original video scene variations, end-to-end delay, and buffer fullness. Chapter 7 presents the implementation of a video transcoding gateway demo system, which serves as a prototype application of video transcoders.

8.2 Future Research Directions

8.2.1 Rate-Distortion optimized multi-functional transcoding

A number of architectures and techniques for video transcoding have been introduced in this thesis. However, given delivery constraints, such as bandwidth limitation, display size constraint, and/or processing speed available on a give client, there may be multiple transcoding methods available that satisfy the constraint. For example, for transcoding video to meet certain bandwidth constraint, we have multiple choices. We can use simple rate adaptation transcoding techniques. If the bandwidth is too low, directly using rate adaptation transcoding will not get acceptable quality. We can use a down-sampling

transcoder to change the video resolution, which will reduce the bit-rate significantly. At the decoder side, the decoded lower resolution video can be up-sampled to get the full resolution video. Furthermore, if the video quality is still not good enough, we can skip some frames and down-sample the video temporally. At the decoder side, the decoded video can be interpolated to get full resolution video. Consequently, it is desirable to develop a means to examine the performance of each transcoding option. The ultimate goal is to select appropriate transcoding strategy based on the requirements imposed by the server, network, client, and to maintain the best possible end-to-end video quality. To solve this problem, we need an analytical model to describe the relationship among rate, distortion and quantization parameters of video transcoding output. Under the given delivery constraint, and optimal transcoding method can then be selected, where optimality is defined in the rate-distortion sense.

8.2.2 Transcoding to H.264

H.264 (MPEG-4 AVC) is the emerging video coding standard. It offers significant higher efficiency than earlier standards, including other profiles of MPEG-4 and MPEG-2. However, the improved compression efficiency comes at a price. The computational complexity of an H.264 encoder is much higher than other standards. The increase in the computational complexity comes mainly from the new prediction modes introduced in H.264: multiple reference frames, variable block-sizes, and various intra prediction modes, etc. For transcoding, e.g., from MPEG-2 to H.264, it is interesting to investigate how the input coding statistics can be intelligently utilized to calculate the new prediction modes with minimal computational requirement but can achieve optimal video quality. Intuitively, the various motion refinement techniques should be used to perform the multiple reference frame prediction in the H.264.

References

[1] S. Acharya, and B. Smith, "Compressed Domain Transcoding of MPEG," *Proceedings of International Conference on Multimedia Computing and Systems*, 1998.

[2] O. Alshaykh, and H. Chen, "Minimum-Drift Digital Video Down-Conversion," *Proceedings of ACM Multimedia'99*.

[3] M. E. Anagnostou and E.N. Protonotarios, "Performance Analysis of the Selective Repeat ARQ protocol," *IEEE Transactions on Communications*, Vol. 34, pp.127-135, February 1986.

[4] S. Aramvith, I.-M. Pao, and M.-T. Sun, "A Rate-Control Scheme for Video Transport over Wireless Channels," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 5, pp.569-580, May 2001.

[5] P. Assunção and M. Ghanbari, "Post-Processing of MPEG-2 Coded Video for Transmission at Lower Bit Rates," *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'96*, Vol.4, pp. 1998-2001, May 1996.

[6] P. Assunção and M.Ghanbari, "Optimal Transcoding of Compressed Video," *IEEE International Conference on Image Processing, ICIP'97*, vol.1, pp.739-742, USA, October 1997.

[7] P. Assunção, and M. Ghanbari, "Transcoding of Single-Layer MPEG Video into Lower Rates," IEE Proceedings on Vision, Image and Signal Processing, 144(6):377-383, December 1997.

[8] P. Assunção, and M. Ghanbari, "Optimal Bit Rate Conversion of MPEG-2 Bit Streams", IEE Electronics Letters, 33(8):675-677, April 1997.

[9] P. Assunção, and M. Ghanbari, "Congestion Control of Video Traffic with Transcoders," *IEEE International Conference on Communications, ICC 97'*, Montreal, 1997.

[10] P. Assunção and M. Ghanbari, "A Frequency-Domain Video Transcoder for Dynamic Bit-Rate Reduction of MPEG-2 Bit Streams," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.8, no.8, pp. 953-967, December 1998.

[11] P. Assunção and M. Ghanbari, "Bit Rate Control of Internetworking Video Transcoders," *Proceedings of IEEE International Conference on Electronics, Circuits and Systems*, ICECS'98, vol.2, pp. 247-250, Portugal, September 1998.

[12] P. Assunção, M. Ghanbari, "A Buffer Control Algorithm for CBR Video Transcoding," *IEEE International Conference on Image Processing, ICIP'98*, USA, October 1998

[13] P. Assunção and M. Ghanbari, "Buffer Analysis and Control in CBR Video Transcoding", *IEEE Transactions on Circuits and Systems for Video technology*, vol.10, no.1, pp.83-92, February 2000.

[14] P. Bahl, I. Chlamtac, "H.263 Based Video Codec for Real-Time Visaul Communications Over Wireless Radio Networks," *Proceedings of IEEE International Conference on Universal Personal Communications*, 1997.

[15] V. Bhaskaran, K. Konstantinides. *Image and Video Compression Standards: Algorithms and Architectures*. Kluwer Academic Publishers, 1995.

[16] N. Bjork and C. Christopoulos, "Transcoder Architectures for Video Coding," *IEEE Transactions on Consumer Electronics*, Vol. 44, No.1, pp.88-98, February 1998.

[17] N. Bjork and C. Christopoulos, "Video Transcoding for Universal Multimedia Access," *Proceedings of ACM Multimedia'2000*, USA, October 2000.

[18] S.-F. Chang and D. Messerschmitt, "A New Approach to Decoding and Compositing Motion Compensated DCT-Based Images," *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'93*, vol. 5, pp. 421-424, April 1993.

[19] S.-F. Chang, and D. Messerchmitt, "Manipulation and Compositing of MC-DCT Compressed Video," *IEEE Journal of Selected Area in Communications, Special Issue on Intelligent Signal Processing in Communications*, Vol. 13 Issue: 1, pp, 1-11, Jan. 1995.

[20] J.-B. Cheng and H.-M. Hang, "Adaptive Piecewise Linear Bits Estimation Model for MPEG Based Video Coding," *Journal of Visual Communication and Image Representation*, Vol. 8, No. 1, pp. 51-67, March 1997.

[21] M.-J. Chen, M.-C. Chu, and C.-W. Pan, "Efficient Motion-Estimation Algorithm for Reduced Frame-Rate Video Transcoder," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 4, pp. 269-275, Apirl, 2002.

[22] T. Chiang, and Y.-Q. Zhang, "A New Rate Control Scheme Using Quadratic Rate Distortion Model," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 1, pp.246-250, February 1997.

[23] J. Choi and D. Park, "A Stable Feedback Control of The Buffer State Using The Controlled Langrange Multiplier Method," *IEEE Transaction on Image Processing*, Vol. 3, pp. 546–558, Sept. 1994.

[24] J. R. Corbera and S. Lei, "Rate Control for Low Delay Video Communications," *ITU-Telecommunications Standardization Sector, Video Coding Experts Group, Document Q15-A-20*, Portland, 24-27 June 1997.

[25] J. R. Corbera, and S. Lei, "Rate Control in DCT Video Coding for Low-Delay Communications," *IEEE Transaction on Circuit and System for Video Technology*, Vol.9, No. 1, pp.172-185, Feb. 1999.

[26] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, New York, 1991.

[27] W. Ding, and B. Liu, "Rate Control of MPEG Video Coding and Recording by Rate-Quantization Modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 1, pp.12-20, February 1996.

[28] S. Dogan, A.H. Sadka and A.M. Kondoz, "Tandeming/Transcoding Issues between MPEG-4 and H.263," *Mobile and Personal Satellite Communications 3 - Proceedings of the Third European Workshop on Mobile/Personal Satcoms (EMPS'98)*, Springer-Verlag, Great Britain, 1999, ISBN: 1-85233-045-7, pp. 339-346.

[29] S. Dogan, A.H. Sadka and A.M. Kondoz, "Efficient MPEG-4/H.263 Video Transcoder for Interoperability of Heterogeneous Multimedia Networks," *IEE Electronics Letters*, Vol. 35, No. 11, pp. 863-864, May 1999.

[30] S. Dogan, A. Cellatoglu, A.H. Sadka and A.M. Kondoz, "Error-Resilient MPEG-4 Video Transcoder for Bit Rate Regulation," *Proceedings of the Fifth World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'2001)*, Vol. XII, Part. II, Orlando, Florida, USA, 22-25 July 2001, pp. 312-317.

[31] S. Dogan, A.H. Sadka and A.M. Kondoz, "MPEG-4 Video Transcoder for Mobile Multimedia Traffic Planning," *Proceedings of the IEE Second International Conference on 3G Mobile Communication Technologies (3G'2001), IEE Conference Publication* No. 477, London, UK, 26-28 March 2001, pp. 109-113.

[32] G. Fayolle, E.Gelenbe, and G. Pujolle, "An Analytic Evaluation of the 'send and wait'," *IEEE Transactions on Communications*, Vol. 26, pp.313-319, March 1978.

[33] N. Feamster and S.J. Wee, "An MPEG-2 to H.263 Transcoder," *SPIE Voice, Video, and Data Communications Conference*, Boston, MA, September 1999.

[34] E. D. Frimout, J. Biemond, and R. L. Lagendik, "Forward Rate Control for MPEG Recording," *Proceedings of SPIE Visual Communications and Image Processing*, Cambridge, MA, pp. 184-194, November 1993.

[35] K.-T. Fung, Y.-L. Chan and W.-C. Siu, "Dynamic Frame Skipping for High-Performance Transcoding," *Proceedings of the International Conference on Image Processing* (ICIP2001).

[36] M. Ghanbari, *Video Coding: An Introduction to Standard Codecs*, Institution of Electrical Engineering Press, London, 1999.

[37] H. Gish and J. N. Pierce, "Asymptotically Efficient Quantizing," *IEEE Transactions on Information Theory*, vol. IT-14, pp. 676-683, September 1968.

[38] J. Gomez, and A.T.Campbell, "A Channel Predictor for Wireless Packet Networks," *Proceedings of IEEE International Conference on Multimedia and Expo*, July 2000.

[39] Z. Guo, O. Au, K. Letaief, "Parameter Estimation for Image/Video Transcoding," *Proceeding of IEEE International Symposium on Circuits and Systems*, Vol.2, 2000.

[40] W. Guo, L. Lin, and W. Zheng, "Mismatch MB Retrieval for MPEG-2 to MPEG-4 Transcoding," *Proceedings of IEEE Pacific Rim Conference on Multimedia*, Beijing, 2001.

[41] ITU-T Draft H.263, "Video Coding for Low Bit Rate Communication," Jan. 27, 1998.

[42] H.-M. Hang and J.-J. Chen, "Source Model for Transform Video Coder and Its Application – Part I: Fundamental Theory," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 287 – 298, April 1997.

[43] M.Hashemi, L.Winger, and S. Panchanathan, "Compressed Domain Motion Vector Resampling for Downscaling of MPEG Video," *IEEE International Conference on Image Processing*, Vol.4, 1999.

[44] M. Hashemi, L. Winger, and S. Panchanathan, "Macroblock Type Selection for Compressed Domain Down-sampling of MPEG Video," *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering*, Vol. 1, 1999.

[45] Z. He, Y. K. Kim, and S. K. Mitra, "Low-Delay Rate Control for DCT Video Coding via ρ – Domain Source Modeling," *IEEE Transaction on Circuits and Systems for Video Technology*, Vol. 11, No. 8, pp.928-940, August 2001.

[46] D. T. Hoang and J. S. Vitter, *Efficient Algorithms for MPEG Video Compression*, Wiley-Interscience Inc. Sept. 2001.

[47] J.-N. Hwang, T.-D. Wu, and C.-W. Lin, "Dynamic Frame Skipping in Video Transcoding," *Proceedings of IEEE Workshop on Multimedia Signal Processing*, USA, Dec. 1998.

[48] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding of MPEG Bitstreams," *Signal Processing: Image Communication*, pp481-500, 1996.

[49] M. Khansari, A. Jalali, E. Dubois and P. Mermelstein, Low Bit-rate Video Transmission over Fading Channels for Wireless Microcellular Systems, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 1, pp.1-11, February 1996.

[50] S. R. Kim and C.K.Un, "Throughput Analysis for Two ARQ schemes Using Combined Transition Matrix," *IEEE Transactions on Communications*, Vol. 40, No. 11, pp.1679-1683, November 1992.

[51] A. Konrad, B.Y. Zhao, A.D.Joseph, and R. Ludwig, "A Markov-Based Channel Model Algorithm for Wireless Networks," *Proceedings of Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2001)*, May 2001.

[52] P. Kuhn, T. Suzuki and A. Vetro, "MPEG-7 Transcoding Hints for Reduced Complexity and Improved Quality," *Proceedings of International Packet Video Workshop*, Kyongju, Korea, April 2001.

[53] E. Y. Lam and J. W. Goodman, "A Mathematical Analysis of the DCT Coefficient Distributions for Image," *IEEE Transactions on Image Processing*, vol. 9, pp. 1661-1666, October 2000.

[54] J. Lee and B. W. Dickinson, "Joint Optimization of Frame Type Selection and Bit Allocation for MPEG Video Encoders," *Proceedings of International Conference on Image Processing '94*, Vol. II, pp. 962–966, Austin TX, 1994.

[55] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard," *IEEE Transactions on Circuit and Systems for Video Technology*, Vol. 11, No. 3, pp.301-317, March 2001.

[56] S. Lin and P.S. Yu, "An Effective Error Control Scheme for Satellite Communications," *IEEE Transactions on Communications*, Vol. 28, pp.395-401, March 1980.

[58] D. W. Lin, M.-H. Wang, and J.-J. Chen, "Optimal Delayed-coding of Video Sequences Subject to A Buffer-size Constraint," *Proceedings of SPIE Visual Communication and Image Processing '93*, pp.223–234, Cambridge, MA, Nov. 1993.

[59] L.-J. Lin, and A. Ortega, "Bit-Rate Control Using Piecewise Approximated Rate-Distortion Characteristics," *IEEE Transactions on Circuit and Systems for Video Technology*, Vol. 8, No. 4, pp. 446-459, August 1998.

[60] C.-W.Lin, Y.-C. Chen, "Video Transcoding for Multipoint Video Conferencing," Ph.D Thesis.

[61] C.-W. Lin, T.-J. Liou, and Y.-C. Chen, "Dynamic Rate Control in Multipoint Video Transcoding," *IEEE International Symposium on Circuits and Systems, ISCAS 2000*, Geneva, Switzerland, May 2000.

[62] C.-W. Lin, Y.-C. Chen, and M.-T. Sun, "Dynamic Region of Interest Transcoding for Multipoint Video Conferencing," *Proceedings of International Computer Symposium and Workshop on Computer Networks, Internet, and Multimedia*, Chiayi, Taiwan, Dec. 2000.

[63] C.-W. Lin, "Video Transcoding Techniques for Multipoint Video Conferencing," *Ph.D Dissertation*, Department of Electrical Engineering, National Tsing Hua University, Jan 2000.

[64] Y.-C. Lin, C.-N. Wang, T. Chiang, A. Vetro and H. Sun, "Transcoding of MPEG-4 FGS to Simple Profile," *Proceedings of IEEE International Conference on Consumer Electronics*, Los Angles, CA, June 2002.

[65] N. Merhav and V. Bhaskaran, "A Fast Algorithm for DCT-Domain Inverse Motion Compensation," *HP Labs Technical Report, HPL-95-17*, Sept. 1995.

[66] N. Merhav and V. Bhaskaran, "Fast Inverse Motion Compensation Algorithms for MPEG-2 and for Partial DCT Information," *HP Labs Technical Report, HPL-96-53*, Apr. 1996.

[67] G. Morrison, "Video Transcoders with Low Delay", *IEICE Transactions on Communications*, June 1997.

[68] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal Trellis-based Buffered Compression and Fast Approximations," *IEEE Transaction on Image Processing*, Vol. 3, pp. 26–40, Jan. 1994.

[69] I.-M. Pao, and M.-T. Sun, "Encoding DCT Coefficients Based on Rate-Distortion Measurement," *Journal of Visual Communication and Image Representation*, Vol. 12, pp. 29-43, 2001.

[70] Sanggyu Park, Youngsun Lee, and Hyunsik Chang, "A New MPEG-2 Rate Control Scheme Using Scene Change Detection," *ETRI Journal*, Vol. 18, July 1996.

[71] A. Puri and R. Aravind, "Motion-compensated Video Coding with Adaptive Perceptual Quantization," *IEEE Transaction on Circuits and Systems foe Video Technology*, Vol. 1, pp. 351-361, Dec. 1991.

[72] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit Allocation for Dependent Quantization with Application to Multiresolutions and MPEG Video Coders," *IEEE Transaction on Image Processing,* Vol.2, pp.533-545, Sept. 1994.

[73] A.R. Reibman and B.G. Haskell, "Constraints on Variable Bit-rate Video for ATM Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.2, No. 4, pp.361-372, December 1992.

[74] G. Reyes, A. Reibman, J. Chuang, and S.-F. Chang, "Video Transcoding for Resilience in Wireless Channels," *Proceedings of International Conference on Image Processing' 98*, USA, October 1998.

[75] G. Reyes, A.R. Reibman, S.-F. Chang, and J.C. Chuang, "Error-Resilient Transcoding for Video over Wireless Channels," *IEEE Journal on Selected Areas in Communications*, Vol. 18, No.6, pp.1063-1074, June 2000.

[76] Iain E.G. Richardson. *Video Codec Design: Developing Image and Video Compression Systems*. John Willey & Sons, Ltd. 2002.

[77] Abdul H. Sadka. *Compressed Video Communications*. John Willey & Sons, Ltd. 2002.

[78] Yoo-Sok Saw, *Rate-Quality Optimized Video Coding*, Kluwer Academic Publishers, 1999.

[79] Y.Senda, and H. Harasaki, "A Realtime Software MPEG Transcoder Using A Novel Motion Vector Reuse And A SIMD Optimization Techniques," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 4, 1999.

[80] T. Shanableh and M. Ghanbari, "Heterogeneous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Formats," *IEEE Transactions on Multimedia*, vol.2, no.2, June 2000.

[81] T. Shanableh, and M. Ghanbari, "Transcoding of Video into Different Encoding Formats," *Proceeding of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 4, 2000.

[82] B. Shen, I. Sethi, and B. Vasudev, "Adaptive Motion-Vector Resampling for Compressed Video Downscaling", *IEEE Transactions on Circuits and Streams for Video Technology*, vol.9, no.6, pp.929-936, September 1999.

[83] H. Sorial, W. Lynch, A. Vincent, "Joint Transcoding of Multiple MPEG Video Bitstreams," *IEEE International Symposium on Circuits and Systems (ISCAS'99)*, Orlando, Florida, pp. 251--254, vol.4, May 1999.

[84] M.-T. Sun and A. Reibman, *Compressed Video over Networks*, Marcel Dekker, Inc. 1997.

[85] M.-T. Sun, T.-D. Wu, and J.N. Hwang, "Dynamic Bit Allocation in Video Combining for Multipoint Conferencing," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 45, No.5, pp.644-648, May 1998.

[86] R. Swann, and N. Kingsbury, "Transcoding of MPEG-2 For Enhanced Resilience to Transmission Errors," Ph.D Thesis.

[87] B. Tao, B. W. Dickinson, and H. A. Peterson, "Adaptive Model-Driven Bit Allocation for MPEG Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 1, pp.147-157, February 2000.

[88] TMN Version 3.0, Department of Electrical Engineering, University of British Columbia CANADA. May 27, 1997

[89] A. Vetro, H. Sun, J. Bao and T. Poon, "Frequency Domain Down-Conversion Using an Adaptive Motion Compensation Scheme," *Proceedings of IEEE International Conference on Image Processing, ICIP'97*, Santa Barbara, CA, Oct. 1997.

[90] A. Vetro and H. Sun, "Frequency Domain Down-Conversion Using an Optimal Motion Compensation Scheme," *International Journal of Imaging Systems and Technology*, vol. 9, no. 4, Aug. 1998.

[91] A. Vetro, H. Sun, J. Bao, and T. Poon, "Frequency Domain Down-Conversion of HDTV Using Adaptive Motion Compensation," *Proceedings of International Conference on Image Processing*, 1997.

[92] A. Vetro, A. Divakaran, H. Sun and T. Poon, "Adaptive Transcoding System Based on MPEG-7 Meta-data,"' *Proceedings of IEEE Pacific-Rim Conference on Multimedia*, Sydney, Australia, Dec 2000.

[93] A. Vetro, H. Sun and Y. Wang, "An MPEG-4 Object-Based Transcoder: Architecture and Applications," *Proceedings of IEEE International Conference on Consumer Electronics*, Los Angles, CA, June 2000.

[94] A. Vetro, H. Sun and Y. Wang, "Object-Based Transcoding for Scalable Quality of Service," *Proceedings of IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, May 2000.

[95] A. Vetro, H. Sun and Y. Wang, "Object-Based Transcoding for Adaptable Video Content Delivery," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no.3, pp. 387-401, March 2001.

[96] A. Vetro and H. Sun, "Encoding and Transcoding of Multiple Video Objects with Variable Temporal Resolution," *Proceedings of IEEE International Symposium on Circuits and Systems*, Sydney, Australia, May 2001.

[97] A. Vetro, "Object-Based Encoding and Transcoding," *Ph.D. Dissertation*, Polytechnic University, Brooklyn, NY, June 2001

[98] A. Vetro, A. Divakaran and H. Sun, "Providing Multimedia Services to a Diverse Set of Consumer Electronic Devices," *Proc. IEEE International Conferenceon Consumer Electronics*, Los Angeles, CA, June 2001.

[99] A. Vetro, H. Sun and Y. Wang, "Rate-Distortion Optimized Video Coding with Frameskip," *Proceedings of IEEE International Conference on Image Processing*, Thessaloniki, Greece, Oct. 2001.

[100] A. Vetro. T. Hata, N. Kuwahara and H. Kalva, "Complexity-Quality Analysis of MPEG-2 to MPEG-4 Transcoding Architectures," *Proceedings of IEEE International Conference on Consumer Electronics*, Los Angles, CA, June 2002.

[101] A. Vetro, P. Yin, B. Liu and H. Sun, "Reduced Spatio-Temporal Transcoding Using an Intra Refresh Technique," *Proceedings of IEEE International Symposium on Circuits and Systems*, Scottsdale, AZ, May 2002.

[102] Y. Wang, J. Ostermann and Y.-Q. Zhang, *Video Processing and Communications*, Prentice-Hall, 2002.

[103] S.J. Wee, J.G. Apostolopoulos, and N. Feamster, "Field-to-Frame Transcoding with Temporal and Spatial Downsampling," *IEEE International Conference on Image Processing, ICIP'99*, Kobe, Japan, October 1999.

[104] J. Wong, and A. Tourapis, "Predictive Motion Estimation for Reduced-Resolution Video from High-resolution Compressed Video," *Proceedings of IEEE International Conference on Image Processing 1998, ICIP'98*, USA, October 1998.

[105] S.-W. Wu and A. Gersho, "Rate-Constrained Optimal Block-adaptive Coding for Digital Tape Recording of HDTV," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 1, pp. 100–112, Mar. 1991.

[106] J.-L. Wu, S.-J. Huang, Y.-M. Huang, C.-T. Hsu, and J. Shiu, "An Efficient JPEG to MPEG-1 Transcoding Algorithm," *IEEE Transactions on Consumer Electronics*, Vol. 42, No. 3, pp.447-457, Aug. 1996.

[107] T.-D. Wu, J.-N. Hwang, "Dynamic Bit Rate Conversion in Multipoint Video Transcoding," *Proceedings of IEEE International Conference on Image Processing*, Vol. 3, 1999.

[108] J. Xin, M.-T. Sun and K.S. Kan, "Bit Allocation for Joint Transcoding of Multiple MPEG Coded Video Streams," *IEEE International Conference on Multimedia & Expo*, Tokyo, Japan, 2001

[109] J. Xin and M.T. Sun, "Joint Transcoding of MPEG Video Streams Using Adaptive Picture Complexity Measure," ICME 2001.

[110] J. Xin, M.-T. Sun and K. Chun, "Bit Allocation for Transcoding Pre-encoded Video Streams," *Proceedings of SPIE: Visual Communication and Image Processing*, San Jose, California,2002.

[111] A. Y. K. Yan and M. L. Liou, "Adaptive Predictive Rate Control Algorithm for MPEG Videos by rate Quantization Method," *Proceedings of Picture Coding Symposium*, Berlin, Germany, pp. 619-624, September 1997.

[112] Y. Yang and S. S. Hemami, "Generalized Rate-Distortion Optimization for Motion-Compensated Video Coders," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 6, pp.942-955, Sept. 2000.

[113] N. Yeadon, N. Davies, A. Friday, and G.Blair, "Supporting Video in Heterogeneous Mobile Environments," *Proceedings of ACM Multimedia'98.*

[114] P. Yin, M. Wu, and B. Liu, "Video Transcoding by Reducing Spatial Resolution," *Proceedings of IEEE Conference on Image Processing'2000*, Canada, 2000.

[115] P. Yin, A. Vetro, H. Sun, and B. Liu, "Drift Compensation Architectures and Techniques for Reduced Resolution Transcoding," *Visual Communication and Image Processing, SPIE'2002.*

[116] J. Youn, M.-T. Sun, and C.-W. Lin, "Motion Estimation for High Performance Transcoding," *IEEE Trans. Consumer Electronics*, vol. 44, pp. 649-658, Aug. 1998.

[117] J.Youn, M.-T. Sun, and C.-W. Lin, "Motion Vector Refinement for High-Performance transcoding," *IEEE Transactions on Multimedia*, vol.1, no.1, pp.30-40, March 1999.

[118] J. Youn, M.-T. Sun, and J. Xin, "Video Transcoder Architectures for Bit Rate Scaling of H.263 Bit Streams," *Proceedings of ACM Multimedia'99*, USA, October 1999.

[119] J.Youn and M.T. Sun, "A Fast Motion Vector Composition Method for Temporal Transcoding," *Proceedings IEEE International Symposium on Circuits and Systems (ISCAS'99)*, May 1999.

[120] J. Youn, and M.-T. Sun, "Video Transcoding with H.263 Bit-Streams," *Journal of Visual Communication and Image Representation*, Vol. 11, 2000.

[121] W. Zhu, K.Yang, and M. Beacken, "CIF-to-QCIF Video Bit Stream Down-Conversion in the DCT Domain," *Bell Labs technical Journal*, Vol.3, No.3, July 1998.

[122] M. Zorzi, R.R.Rao, L.B. Milstein, "On the Accuracy of A First-order Markov Model for Data Transmission on Fading Channels," *Proceedings of the IEEE ICUPC'95*, November 1995.

[123] M. Zorzi and R. R. Rao, "On the Statistics of Block Errors in bursty Channels," *IEEE Transactions on Communications*, Vol. 45, No. 6, pp.660-667, June 1997.