Low Bit Rate DCT-Based Video Coding in Error-Prone Environments

by

Guy Côté

B. Eng. (Electrical Engineering) Royal Military College of Canada, 1993

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

Department of Electrical and Computer Engineering

We accept this thesis as conforming
to the required standard

The University of British Columbia

December 1999

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical and Computer Engineering.

The University of British Columbia
Vancouver, Canada

Date 19 Nov. 1999.

Abstract

Video compression, which is a necessary process for video communication over low bit rate networks, removes spatial and temporal redundancies contained in video sequences. However, when transmitted over error prone networks, compressed video streams are very sensitive to bit errors and packet losses, which lead to inevitable spatial and temporal error propagation.

This thesis presents efficient video encoding methods for transmission over error prone networks. The resulting video communication systems exploit the video input statistics, the channel condition information, and the encoder's knowledge of the video decoder's error handling capabilities to optimize the encoded bit stream for its transmission over error prone networks. We first propose a rate-distortion optimized synchronization marker placement algorithm which can effectively determine the frequency and location of synchronization markers, subject to a known decoder concealment method. Next, we present a semi-fixed length coding method for the efficient entropy coding of motion vectors. With the use of unequal error protection techniques, the proposed method provides superior bit stream and spatial synchronization, compared to variable length coding methods. A new algorithm for temporal

error resilience employing optimal intra block updating is then presented. Using a new distortion measure that takes into account the effects of temporal error propagation, the proposed method chooses the amount of non-predictive coding necessary to achieve temporal synchronization.

We combine the proposed methods and evaluate the performance of two resulting video communication systems in error prone environments, using packet loss and bit error models for the Internet and mobile networks, respectively.

Contents

List of Tables

List of Figures

List of Abbreviations

ACK	Acknowledgment
ARQ	Automatic Repeat Request
ATM	Asynchronous Transfer Mode
BCH	Bose-Chaudhuri-Hocquenghem
BER	Bit Error Rate
BSC	Binary Symmetric Channel
CDF	Cumulative Distribution Function
CDMA	Code Division Multiple Access
CIF	Common Intermediate Format
CRC	Cyclic Redundancy Check
DCT	Discrete Cosine Transform
DIFFSERV	Differentiated Services
DPCM	Differential Pulse Code Modulation
EC	Error Concealment
EEP	Equal Error Protection
EREC	Error Resilient Entropy Code
FEC	Forward Error Correction

FLC	Fixed Length Code
fps	frames per second
GMDC	Generalized Multiple Description Coding
GOB	Group of Blocks
HD	Header Duplication
HEC	Header Extension Code
HVS	Human Visual System
IDCT	Inverse Discrete Cosine Transform
IEC	International Engineering Consortium
IP	Internet Protocol
ISD	Independent Segment Decoding
ISDN	Integrated Services Digital Network
ISO	International Standards Organization
ISP	Internet Service Provider
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
KLT	Karhunen-Loeve Transform
LAN	Local Area Network
MB	Macroblock
MC	Motion Compensation
MD	Multiple Description
MDC	Multiple Description Coding
MPEG	Moving Picture Expert Group

MPL	Maximum Payload Length
MSE	Mean Squared Error
MTU	Maximum Transfer Unit
MV	Motion Vector
NACK	Negative Acknowledgment
PDU	Protocol Data Unit
PLR	Packet Loss Rate
PSNR	Peak Signal-to-Noise Ratio
PSTN	Public Switched Telephone Network
QCIF	Quarter Common Intermediate Format
QP	Quantizer Parameter
QoS	Quality of Service
RCPC	Rate-Compatible Convolutional Code
RD	Rate-Distortion
RFC	Request for Comments
RVLC	Reversible Variable Length Code
RPS	Reference Picture Selection
RR	Receiver Report
RTCP	RTP Control Protocol
RTP	Real-Time Transport Protocol
SAD	Sum of Absolute Difference
SDU	Service Data Unit
SER	Slice Error Rate

SNR	Signal-to-Noise Ratio
SRP	Scalable Reservation Protocol
SS	Slice Structure
SSE	Sum of Squared Error
TCP	Transmission Control Protocol
TEC	Temporal Error Concealment
TMN	Test Model Near-Term
UDP	User Datagram Protocol
UEP	Unequal Error Protection
VLC	Variable Length Code
VRC	Video Redundancy Coding
WCDMA	Wide-band Code Division Multiple Access

Acknowledgements

First, my greatest debt of gratitude goes to my supervisor, Professor Faouzi Kossentini, who has introduced and guided me into this exciting field of research, and provided sound advice and invaluable critical feedback at every stage of this project. Without his support, this thesis could not have been written.

Second, I want to thank my family for their encouragement and support. I also want to thank my wife, Susannah, for her patience and support over the past three years. I would also like to thank Dr. Stephan Wenger of T.U. Berlin for his valuable input during my thesis, as well as is constructive comments on the final thesis. Finally, I want to thank all the members of the Signal Processing and Multimedia Group for their valuable discussions, suggestions, and moral support.

GUY CÔTÉ

The University of British Columbia

October 1999

Chapter 1

Introduction

1.1 Introduction

In recent years, interest in multimedia has generated a lot of research in the area of video coding in academia and industry alike [1, 2, 3, 4, 5] and several successful standards have emerged, e.g. ITU-T H.261, H.263, ISO/IEC MPEG-1, MPEG-2 and MPEG-4. These standards address a wide range of applications having different requirements in terms of bit rate, picture quality, complexity, error resilience and delay. While the demand for digital video communication applications such as video conferencing, video e-mailing and video telephony has increased considerably, transmission rates over the public switched telephone networks (PSTN) and wireless networks are still very limited. This requires compression performance and channel error robustness levels that can only be achieved with efficient source compression and error resilience coding methods.

Video compression algorithms remove unnecessary redundancies, without considering potential loss of information during transmission. However, some level of redundancy is necessary to recover lost information when transmitting video over an error-prone network. Shannon's separation theorem states that video compression (source coding) and transmission optimizations (channel coding) can be performed independently, and the combination will be as efficient as anything designed by considering both problems together. However, this is valid only if infinite coding delays are assumed. Most video communication applications require relatively high reliability and small delays. Thus, joint source-channel coding is necessary to achieve high error resilience and low delay in video communication, simultaneously [6].

In an error prone environment, video optimized error resilience techniques are necessary due to the nature of compressed video bit streams, which are very sensitive to bit errors and packet losses. Such errors may lead to the loss of synchronization at the receiver which can, in turn, lead to the loss of many video blocks and (possibly) frames. For example, all current video coding standards employ variable length codes (VLCs), as well as spatial prediction, to achieve high compression efficiency. However, bit errors in VLCs can lead to the loss of synchronization, and spatial prediction techniques lead to spatial error propagation. Moreover, the predictive coding process of motion compensation will cause errors to propagate in the temporal direction.

Many of the existing video error resilience techniques are adapted from data error recovery algorithms. Examples of such algorithms are Forward Er-

ror Correction (FEC) and Automatic Repeat Request (ARQ). When using a feedback channel, very effective error recovery mechanisms can be implemented [7]. However, feedback channels may introduce additional delay and complexity, and they are usually also affected by channel errors. Moreover, they may not be available in many scenarios such as video broadcasting or multi-point communication.

Most of the media-based error resilience methods that have been proposed in the literature are mainly designed to improve the bit error robustness of the entropy coding. Example methods include self-synchronized VLCs, reversible VLCs [8, 9], the error resilient entropy code (EREC) [10], and fixed length coding [11, 12]. These methods improve the decoder's synchronization with the video bit stream, but they do not guarantee spatial synchronization, i.e., a wrongly received codeword may lead to the reconstruction of video blocks at the wrong spatial location. Such spatial synchronization is possible with the use of synchronization markers carefully placed in the bit stream. Moreover, these techniques are not useful in a packet-based network such as the Internet. Multiple Description video coding methods have been proposed recently to improve spatial error resilience in a packet-based network [13, 14, 15]. Finally, containing the effects of errors due to temporal error propagation without decoder feedback is still an open and challenging problem.

In this thesis, we present an error resilience video coding and transmission framework, to address the above problems, that exploits the video input statistics, the channel condition knowledge, and some information about the

video decoder error handling capabilities to optimize the encoded bit stream for its transmission over error prone networks. The proposed algorithms do not employ a feedback channel from the decoder to the encoder. Instead, the encoder employs network feedback, where the effects of delay are insignificant.

The proposed methods are suitable for real-time video communications over packet lossy (e.g., Internet) and bit error prone networks (e.g., mobile networks). The combination of the proposed methods yield systems that can provide acceptable video quality at packet loss rates as high as 20% and at bit error rates (random and bursty) as high as 10^{-3}.

1.2 Outline of the Thesis

We first review low bit rate video coding algorithms and present the H.263 video coding standard in Chapter 2. In Chapter 3, we review current error resilience video coding methods, including H.263 and MPEG-4 compliant methods.

Synchronization markers allow for bit synchronization of the decoder with the bit stream, they reset the spatial location in the decoded frame, and they prevent propagation of spatial prediction errors. However, if used too frequently, synchronization markers will introduce an unnecessary increase in bit rate. We propose in Chapter 4 a rate-distortion (RD) optimized algorithm for the localization of synchronization markers.

In order to provide additional bit stream and spatial synchronization, as well as data recovery, we present, in Chapter 5, a semi-fixed length coding

method for motion vectors. Motion vectors are a very important part of the bit stream at low bit rates, and the probability of correctly received motion vector information can be significantly increased using the proposed coding method. When semi-fixed length codes are used, we can employ unequal error protection (UEP) to apply more channel error protection to the critical part of the motion vector code, thus reducing delay and overhead.

Temporal synchronization is obtained by coding video blocks in the intra mode, which stops the propagation of errors due to motion compensated predictive coding. However, intra coding can be expensive in terms of bit rate. In order to only intra code blocks that cannot be appropriately concealed at the decoder, we propose, in Chapter 6, an RD optimized intra updating method.

Finally, we present experimental results of the proposed error resilience methods for the Internet and mobile networks, in Chapters 7 and 8, respectively. In the Internet case, we present an efficient packetization method suitable for low delay video communications. We evaluate the performance of the packetization method and the RD optimized mode selection algorithm for different packet loss rates. In the mobile case, we introduce the transport protocol and evaluate the performance of the RD optimized synchronization marker placement, semi-fixed-length motion vector coding, and the RD optimized intra updating algorithm for random bit error conditions and a WCDMA network. We also evaluate the effects of mismatch between the parameters used by the video encoder and the actual channel conditions and decoder error handling capabilities.

Chapter 2

Low Bit Rate Video Compression and the ITU-T H.263 Recommendation

2.1 Introduction

Video compression is necessary for transmission of digital video over today's bandlimited networks. For example, the transmission of a 24 bit per pixel raw video sequence sampled at 176×144 spatial resolution and 30 frames per second (fps) temporal resolution[1] would require a bit rate of 18.25 Mbps! Compression of digital video without significant quality degradation is usually possible because video sequences contain a high degree of 1) spatial redundancy, due

[1]This is the typical resolution used in videophone applications over the PSTN, which can nowadays support 33.6 kbps duplex communication.

to the correlation between neighbouring pixels, 2) spectral redundancy, due to correlation among the colour components, 3) temporal redundancy, due to correlation between video frames, and 4) psycho-visual redundancy, due to properties of the human visual system (HVS).

Many algorithms have been proposed for the compression of still images. Recent techniques include vector quantization, fractal compression, subband coding, and wavelets. Object-oriented methods are also becoming popular. Vector quantization refers to quantization and coding of a block of pixels or transform coefficients at once, rather than element by element [16, 17]. Fractal compression can be considered as an extension of vector quantization with a virtual codebook [18, 19]. In subband coding, the image is decomposed into several non-overlapping frequency bands [20, 21, 22]. The number of bits allocated for encoding each of the subbands depends on the fidelity requirements. Wavelet transform coding is closely related to subband coding in that it provides an alternative method for filter design [23]. Instead of explicitly applying filters to an image to create subbands, wavelet transforms expand the image onto a set of wavelet basis functions representing different frequency components. In object-oriented methods, the image is segmented into some primitives whose shape and texture are encoded [24].

Removal of temporal redundancies in video signals accounts for a significant percentage of the achieved compression. Therefore, advanced techniques for the coding of the residual signal usually provide little additional compression as compared to traditional techniques, and the additional complexity does

not justify this improvement. Moreover, extension of the advanced image compression methods to 3-dimensional video signals still do not outperform the classical hybrid motion compensated and transform coding method. Consequently, this method is employed in all current video coding standards (e.g. H.261, H.263, MPEG-1, MPEG-2, MPEG-4).

In this chapter, we first describe the block-based hybrid motion compensated and transform video coding method. Distortion measures employed in video coding are then reviewed. Next, rate-distortion optimizations for video coding are introduced. Finally, the H.263 video coding recommendation is briefly described. We present the H.263 optional modes that are not specifically designed for error resilience but may be useful in an error prone environment. Error resilience modes are discussed in the next chapter.

2.2 Low Bit Rate Video Coding

In the hybrid motion compensated and transform video coder, motion compensated prediction first reduces temporal redundancies. Transform coding is then applied to the corresponding difference frame to reduce spatial redundancies. For highly correlated sources, such as natural images, the compaction ability of the Discrete Cosine Transform (DCT) is very close to that of the optimal transform, the Karhunen-Loeve Transform (KLT). Moreover, the DCT, unlike the KLT, is data independent. This has made the DCT the most popular transform for image coding, and has thus become part of the JPEG [25] still image international standard. Moreover, although motion compensated

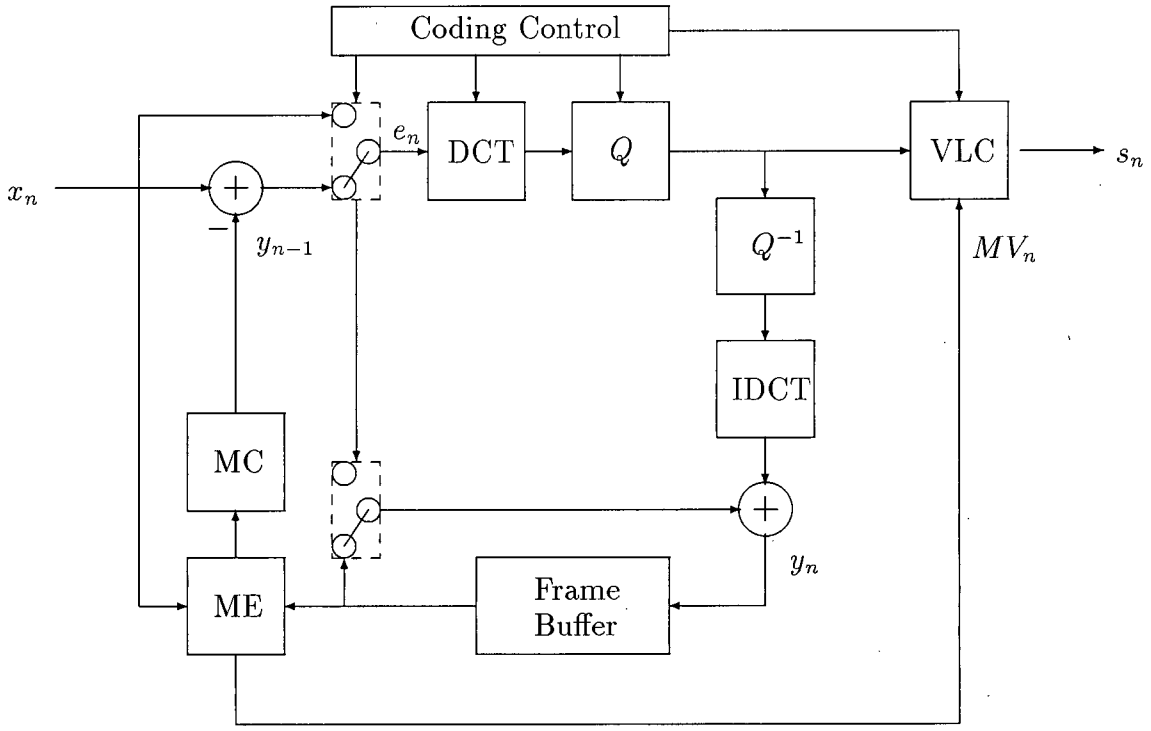


Figure 2.1: Block diagram of the video encoder.

prediction difference frames are poorly correlated, the DCT is still the most popular transform for coding such frames. In fact, the DCT is employed in all current video coding standards.

A block diagram of a typical motion compensated prediction and DCT video encoder is presented in Fig. 2.1 and the corresponding decoder is presented in Fig. 2.2. In the next sections, we describe the building blocks of this video coder.

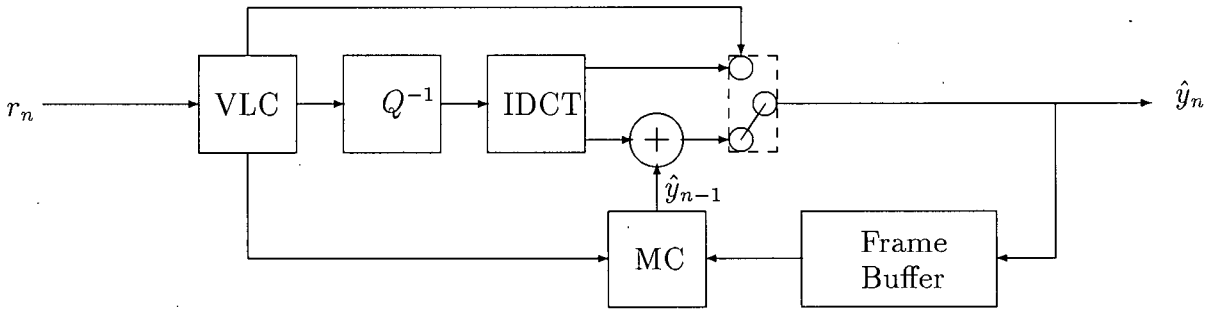


Figure 2.2: Block diagram of the video decoder.

2.2.1 Motion estimation and compensation

Each video frame is divided into blocks of equal size. Motion compensated prediction assumes that a block of pixels within the current picture can be modeled as a translation of a block from a previous picture [26], as shown in Fig. 2.3. Each block is normally predicted from the previous frame. This implies an assumption that each pixel within the block undergoes the same amount of translational motion. This motion information is represented by two-dimensional displacement vectors or motion vectors. Due to the block-based picture representation, many motion estimation algorithms employ block-matching techniques, where the motion vector is obtained by minimizing a cost function measuring the mismatch between a candidate block and the current block. Although several cost measures have been introduced, the most widely used in motion estimation algorithms is the sum-of-absolute-differences (SAD), presented in Section 2.3. Here, we have

$$SAD = \sum_{k=1}^{N} \sum_{l=1}^{N} \mid B_{i,j}(k,l) - B_{i-u,j-v}(k,l) \mid, \tag{2.1}$$

standards allow both horizontal and vertical components of the motion vectors to be of half pixel accuracy. The search window used in motion estimation is often limited by the range of representable motion vector values. A positive value of the horizontal or vertical component of the motion vector represents a block spatially to the right or below the block being predicted, respectively. Motion estimation is usually performed on block sizes of 16x16 or 8x8.

2.2.2 Transform

The purpose of the 8×8 DCT employed in all current video coding standards is to decorrelate the 8×8 blocks of original pixels or motion compensated difference pixels and compact their energy into as few coefficients as possible. Besides its relatively high decorrelation and energy compaction capabilities, the 8×8 DCT is simple, efficient, and amenable to software and hardware implementations [30]. The most common algorithm for implementing the 8×8 DCT is that which consist of 8-point DCT transformation of the rows and the columns, respectively. The 8×8 DCT is defined by

$$C_{m,n} = \alpha(m)\beta(n) \sum_{i=1}^{8} \sum_{j=1}^{8} B_{i,j} \cos(\frac{\pi(2i+1)m}{16}) \cos(\frac{\pi(2j+1)n}{16}), \quad 0 \leq m,n \leq 7$$

where

$$\alpha(0) = \beta(0) = \sqrt{\frac{1}{8}}, \quad and \quad \alpha(m) = \beta(n) = \sqrt{\frac{1}{4}} \quad for \quad 1 \leq m,n \leq 7.$$

Here, $B_{i,j}$ denotes the $(i,j)^{th}$ pixel of the 8×8 original block and $C_{m,n}$ denotes the coefficients of the 8×8 DCT transformed block. The original 8×8 block

of pixels can be recovered using an 8×8 inverse DCT (IDCT) given by

$$B_{i,j} = \sum_{m=1}^{8} \sum_{n=1}^{8} C_{m,n} \alpha(m) \cos(\frac{\pi(2m+1)i}{16}) \beta(n) \cos(\frac{\pi(2n+1)j}{16}), \quad 0 \leq i,j \leq 7.$$

Although exact reconstruction can be theoretically achieved, it is often not possible using finite-precision arithmetic. While forward DCT errors can be tolerated, inverse DCT errors must meet a minimum level of precision in order to avoid IDCT mismatch between the reconstructed frames at the encoder and decoder.

2.2.3 Quantization

The human viewer is more sensitive to reconstruction errors related to low spatial frequencies than those related to high frequencies [31]. Slow linear changes in intensity or colour (low frequency information) are important to the eye. Sharp, high frequency changes can often not be seen and may be discarded. For every element position in the DCT output matrix, a corresponding quantization value is computed using the equation

$$C^q_{m,n} = \frac{C_{m,n}}{Q_{m,n}}, \quad 0 \leq m,n \leq 7,$$

where $C_{m,n}$ is the $(m,n)^{th}$ DCT coefficient and $Q_{m,n}$ is the $(m,n)^{th}$ integer quantization step size. The resulting real numbers are then rounded to their nearest integer values. The net effect is usually a reduced variance between quantized coefficients as compared to the variance between the original DCT coefficients, as well as a reduction of the number of non-zero coefficients in $C^q_{m,n}$.

For low bit rate video coding, quantization is usually performed using the same step size within a block (i.e, using a uniform quantization matrix). For example, in H.263, even quantization levels in the range from 2 to 62 are allowed, except for the first coefficient (DC coefficient) of an intra block, which is uniformly quantized using a step size of 8. The quantizers consist of equally spaced reconstruction levels with a dead zone centered at zero. For higher bit rate, high quality applications (e.g. MPEG-2), a quantization matrix exploiting the characteristics of the HVS can also be employed. After the quantization process, the reconstructed picture is stored so that it can be later used for prediction of the future picture.

2.2.4 Entropy coding

Entropy coding is usually performed by means of variable length codes (VLCs)[2]. VLCs can be generated using Huffman codes [32]. The VLCs are usually a subset of the complete Huffman tree, since some codewords are not employed in order to avoid the emulation of synchronization words by a combination of VLCs. Arithmetic coding [33, 34] may also be employed as means of entropy coding.

Motion vectors are usually predicted by setting their components values to median values of those of neighbouring motion vectors already transmitted; the motion vectors of the blocks to the left, above and above right of the current block. The difference motion vectors are then VLC coded.

[2]Some header information may be coded by means of fixed length codes (FLCs).

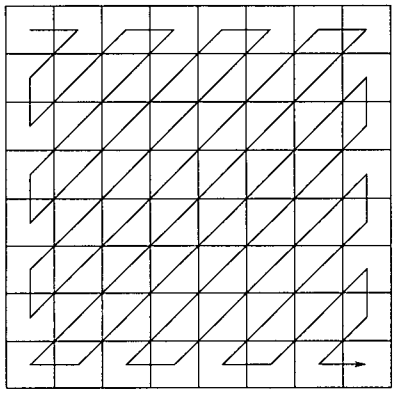


Figure 2.4: Example of zig-zag scan pattern to re-order DCT coefficients from low to high frequencies.

Prior to entropy coding, the quantized DCT coefficients are arranged into a one-dimensional array by scanning them in zig-zag order. This re-arrangement places the DC coefficient first in the array and the remaining AC coefficients are ordered from low to high frequency. An example of the scan pattern for baseline H.263 is illustrated in Fig. 2.4. The re-arranged array is coded using a 3-dimensional run-length VLC table, representing the triple (LAST, RUN, LEVEL). The symbol RUN is defined as the distance between two non-zero coefficients in the array. The symbol LEVEL is the non-zero value immediately following a sequence of zeros. The symbol LAST is equivalent to the End-of Block flag also employed in 2-dimensional run-length coding, where "LAST = 1" means that the current code corresponds to the last coefficient in the coded block. This coding method produces a compact representation of the 8×8 DCT coefficients, as a large number of the coefficients are normally quantized to zero and the re-ordering results (ideally) in the grouping of long

runs of consecutive zero values. Other information such as prediction types and quantizer indication is also entropy coded by means of VLCs.

2.2.5 Coding control

The two switches in Fig. 2.1 represent the intra/inter mode selection. Such a selection is made at the macroblock level [3]. The performance of the motion estimation process, usually measured in terms of the associated distortion values, can be used to select the coding mode. The coding mode where temporal prediction is used is called the *inter* mode. This mode is selected if the motion compensation process is effective, and only if the prediction error macroblock - the difference between original macroblock and the motion compensated predicted macroblock - need be encoded. If temporal prediction is not employed, the corresponding coding mode is called the *intra* mode. If a macroblock does not change significantly with respect to the reference picture, an encoder can also choose not to encode it, and the decoder will simply repeat the macroblock located at the subject macroblock's spatial location in the reference picture. This coding mode is referred to as *skip*. More sophisticated coding mode selection algorithms based on rate-distortion (RD) optimization methods can also be employed, as discussed in Section 2.4.

[3] A macroblock consists of four 8 pixels by 8 lines of luminance Y blocks and the spatially corresponding 8 pixels by 8 lines of colour difference blocks C_b and C_r.

2.3 Distortion Measures

In order to evaluate the performance of lossy video coding algorithms, and to optimize the rate-distortion tradeoffs, measures for distortion are necessary. The distortion measures should be both physically meaningful and analytically tractable. However, this is not always possible and, especially in video coding, one has to settle for a less meaningful but more tractable distortion measure. Research in the area of measurement scales for the assessment of the reproduced video quality is ongoing [35, 36, 31, 37]. For our purposes, we present in this section most commonly used distortion measures in video coding.

Distortion measures are usually performed on pixel values, where original pixel values and reconstructed pixel values are compared. The sum of absolute difference (SAD) is widely used in video coding and is defined as

$$SAD = \sum_{k=1}^{N} \sum_{l=1}^{N} \mid o_{i,j}(k,l) - r_{i,j}(k,l) \mid, \tag{2.2}$$

where $o_{i,j}(k,l)$ represents the $(k,l)^{th}$ pixel of a $N \times N$ block from the original picture at the spatial location (i,j), and $r_{i,j}(k,l)$ represents the $(k,l)^{th}$ pixel of a $N \times N$ block from the reconstructed picture at the spatial location (i,j). Another popular cost measure is the sum-of-squared-errors (SSE), defined by

$$SSE = \sum_{k=1}^{N} \sum_{l=1}^{N} (o_{i,j}(k,l) - r_{i,j}(k,l))^2 . \tag{2.3}$$

The average peak signal-to-noise ratio (PSNR) is usually used as a distortion measure to evaluate the reproduction quality at the decoder, and is given by

$$PSNR \;=\; 10 \log \frac{255^2}{MSE}, \quad \text{where} \tag{2.4}$$

$$MSE = \frac{1}{N^2} SSE, \tag{2.5}$$

for 8 bits/pixel samples, where the MSE represents the mean-squared-error. The average PSNR of all encoded blocks and pictures is employed when comparing reproduction quality of a video sequence. Even though the MSE does not always correlate well with human perception, it is the most widely accepted distortion measure in image and video coding. Surprisingly, during the JPEG-2000 standardization process, image coders minimizing the MSE were reported to perform best in both perceptual and objective tests [38].

2.4 Rate-Distortion Optimized Video Coding

A key component in high-compression lossy video coding is the operational control of the encoder, through the motion estimation process, quantization step size selection, and the video coding mode selection. The process of selection between different possible representations with varying rate-distortion efficiencies can be optimized using Lagrangian minimization techniques [39] based on rate-distortion theory [40], which are reviewed in this section. At the source coding level, rate-distortion theory sets limits on the achievable output distortion for a given coder output rate, or conversely, sets limits on achievable output rate for a given output distortion. Shannon first described the rate-distortion function in [41], and a thorough discussion of rate-distortion theory is available in [40].

In video coding, the coding modes of operation are generally associ-

ated with signal-dependent rate-distortion characteristics, and rate-distortion tradeoffs are inherent in the coding parameters selection process. The optimization task is to choose, for each image block, the most efficient coded representation in the rate-distortion sense. This task is complicated by the fact that the various coding options show varying efficiency at different bit rates and with different scene content. For example, inter coding is efficient in representing key changing content in image sequences. However, intra coding may be more efficient in a situation where the block-based translational motion model cannot accurately represent the image sequence changes. For low activity regions of the video sequence, simply using the skip mode may be preferred. By allowing multiple modes of operation, we expect improved rate-distortion performance if the mode selection method is applied judiciously to different spatio-temporal regions of a video sequence.

The goal of the video communication system is to achieve the best fidelity (or the lowest distortion D) given the capacity of the transmission channel, subject to the coding rate constraint $R(D)$. This optimization process can be solved using the Lagrangian multiplier method [39] where the distortion term is weighted against a rate term. The Lagrangian formulation of the minimization problem is such that we minimize

$$J = D + \lambda R, \tag{2.6}$$

for a particular Lagrangian parameter λ. Each solution to Eq. (2.6) for a given value of the Lagrangian parameter λ should correspond to a locally optimal solution for a given rate constraint. A given value of λ represents a

specific point on the operational rate-distortion curve. It is possible to obtain an approximate relation between the quantizer step size Q[4], which controls the output bit rate, and the optimal value of λ. This is particularly useful when employing a rate control method to achieve a particular video encoder bit rate. In [42], a relationship between Q and λ was obtained as

$$\lambda = c\left(\frac{Q}{2}\right)^2, \tag{2.7}$$

where c is a constant that depends on the different coding parameters. This relationship is obtained by recording the quantizer step size Q that minimizes J for a given value of λ. For example, in an H.263 video coder with all possible coding parameters and error free conditions considered in [42], $c = 0.85$.

RD optimized video mode selection for error free transmission has been studied extensively [42, 43, 44, 45]. In this thesis, we propose an RD optimization framework for error prone environments. The proposed framework will consider the network condition and the decoder error handling capabilities during the mode selection process.

2.5 The ITU-T H.263 Video Coding Recommendation

The most popular video coding standard for video communication at low bit rates is H.263 [46]. This is due to its superior compression performance, flexible frame and bit stream structure allowing transmission over a wide range of

[4]For low bit rate video coding, a fixed quantizer step size in a block is usually employed.

circuit and packet switched networks, and excellent complexity/performance tradeoffs achievable through the use of different optional modes. Therefore, H.263 is employed for comparison purposes and as a basis for the proposed video communication framework. In this section, we first describe the video frame structure of H.263. Then, we discuss the optional modes of H.263 and H.263 Version 2 (also referred to as H.263+) that are not specifically designed for error resilience but may be useful in an error prone environment. Error resilience modes are presented in Chapter 3. A very detailed discussion of all coding modes of H.263 and H.263+ can be found in [47]. The coding modes not discussed in this thesis are also important, as they improve significantly the compression efficiency of H.263+. Finally, we present the rate control method developed for H.263 and employed in this thesis.

2.5.1 Video frame structure

H.263 supports five standardized picture formats: sub-QCIF, QCIF, CIF, 4CIF and 16CIF. The luminance component, Y, is sampled at the original resolution while the two colour difference components, Cb and Cr, are down-sampled by 2 in both the horizontal and vertical directions. The picture structure is shown in Fig. 2.5 for the QCIF resolution. Each picture in the input video sequence is divided into macroblocks, consisting of four Y blocks of 8 pixels by 8 lines followed by one Cb block and one Cr block, each consisting of 8 pixels by 8 lines. A Group of Blocks (GOB) is defined as an integer number of macroblock rows, a number that is dependent on picture resolution. For

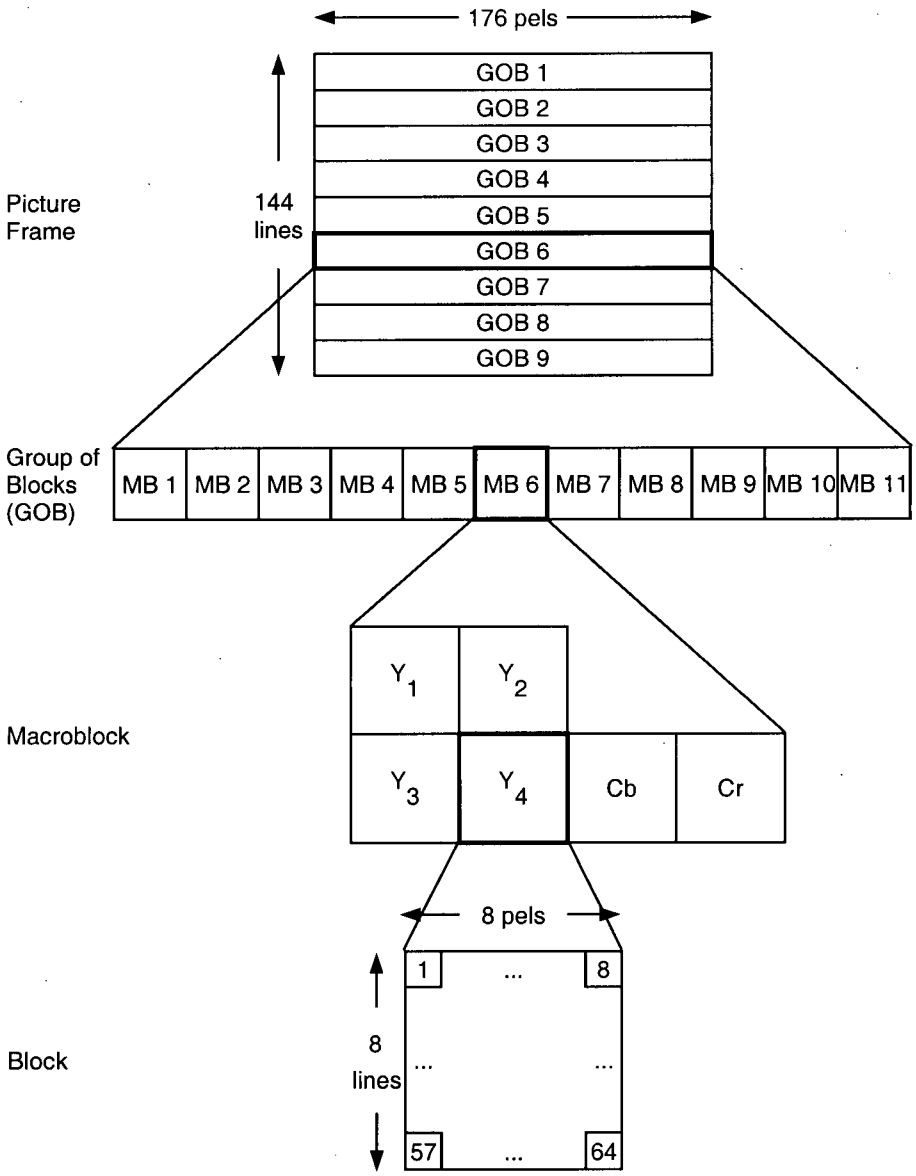


Figure 2.5: H.263 picture structure at QCIF resolution.

example, a GOB consists of a single macroblock row at QCIF resolution.

2.5.2 H.263 optional modes

In addition to the general encoding and decoding algorithms described above, H.263 includes four negotiable advanced coding modes: Unrestricted Motion Vectors, Advanced Prediction, PB-frames and Syntax Based Arithmetic Coding. The first two modes are used to improve inter picture prediction. The PB-frames mode improves temporal resolution with little bit rate increase. When the Syntax Based Arithmetic Coding mode is enabled, arithmetic coding replaces the default VLC coding where applicable.

The objective of H.263+ is to broaden the range of applications and to improve compression efficiency. H.263+ offers many improvements over H.263. It allows the use of a wide range of custom source formats, as opposed to H.263, wherein only five video source formats defining picture size, picture shape and clock frequency can be used. This added flexibility opens H.263+ to a broader range of video scenes and applications, such as wide format pictures, resizeable computer windows and higher refresh rates. Moreover, picture size, aspect ratio and clock frequency can be specified as part of the H.263+ bit stream. Another major improvement of H.263+ over H.263 is scalability, which can improve the delivery of video information in error-prone, packet-lossy, or heterogeneous environments by allowing multiple display rates, bit rates, and resolutions to be available at the decoder. Furthermore, picture segment[5]

[5]A picture segment is defined as a slice or any number of GOBs preceded by a GOB header.

dependencies may be limited, likely reducing error propagation. An additional 12 optional modes are also introduced. These optional modes allow developers to tradeoff between compression performance and complexity.

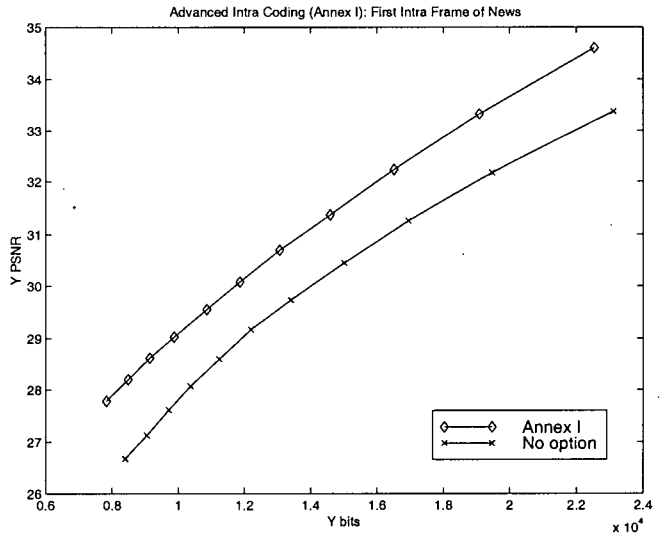
In remaining of this thesis, we refer to H.263 as the latest version of the Recommendation, i.e., H.263 Version 2 or H.263+ [46]. Next, we describe two compression efficiency modes useful in error prone environments, namely the Advanced Intra Coding mode and the Deblocking Filter mode.

Advanced Intra Coding mode (Annex I)

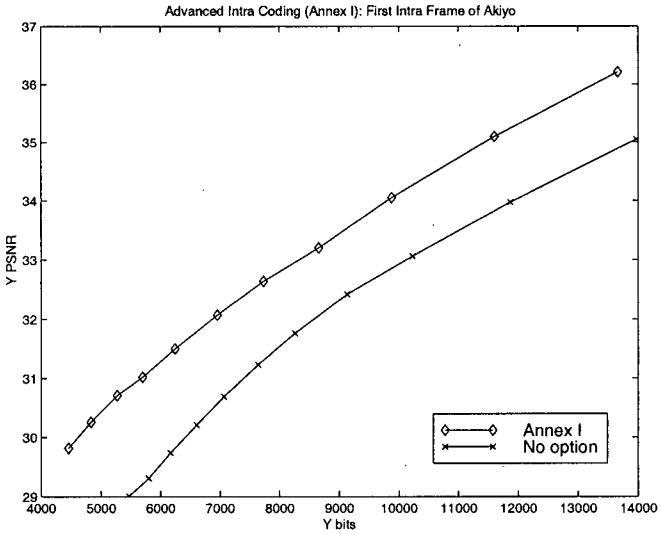
This mode improves compression performance when coding intra macroblocks. In this mode, intra-block prediction from neighbouring intra blocks, a modified inverse quantization of intra DCT coefficients and a separate VLC table for intra coded coefficients are employed. Block prediction is performed using data from the same luminance or chrominance components (Y, Cr or Cb). As illustrated in Fig. 2.6, one of three different prediction options can be signaled: DC only, vertical DC & AC, or horizontal DC & AC. In the DC only option, only the DC coefficient is predicted, usually from both the block above and the block to the left, unless one of these blocks is not in the same picture segment or is not an intra block. In the vertical DC & AC option, the DC and first row of AC coefficients are vertically predicted from those of the block above. Finally, in the horizontal DC & AC option, the DC and first column of AC coefficients are horizontally predicted from those of the block to the left. The option that yields the best prediction is applied to all blocks of the subject intra macroblock.

The difference coefficients, obtained by subtracting the predicted DCT coefficients from the original ones, are then quantized and scanned differently depending on the selected prediction option. Three scanning patterns are used: the basic zig-zag scan for DC only prediction, the alternate-vertical scan (as in MPEG-2) for horizontally predicted blocks or the alternate-horizontal scan for vertically predicted blocks. The main part of the standard employs the same VLC table for coding all quantized coefficients. However, this table is designed for inter macroblocks and is not very effective for coding intra macroblocks. In intra macroblocks, larger coefficients with smaller runs of zeros are more common. Thus, the Advanced Intra Coding mode employs a new VLC table for encoding the quantized coefficients, which is optimized to global statistics of intra macroblocks.

The use of this mode substantially increase the compression performance of intra coded macroblock. Prediction lowers the number of bits required to represent the quantized DCT coefficients, while quantization without a dead zone improves the picture reproduction quality. We present some coding efficiency results in Fig. 2.7 for the first intra frames (i.e. where all the macroblocks are intra coded) of the video sequences **NEWS** and **AKIYO**. Compression improvements of 15-25% are achieved when using the advanced intra coding mode. Based on our implementation, the associated encoding time increases by 5% on average, due to the prediction method selection process. This mode requires slightly more memory to store the reconstructed DCT coefficients, needed for intra prediction. The increase in decoding time



(a)



(b)

Figure 2.7: Advanced Intra Coding rate-distortion performance for (a) NEWS and (b) AKIYO.

is negligible, as only a few additions are required to predict an intra coded macroblock.

Since intra coding is an efficient technique to stop temporal error propagation while being expensive in term of bit rate, the use of the Advanced Intra Coding mode is an important tool in error resilience video coding. Moreover, spatial prediction of coefficients is not allowed across picture segment boundaries, avoiding potential spatial error propagation[6].

Deblocking Filter mode (Annex J)

This mode introduces a deblocking filter inside the coding loop. Unlike in post-filtering, predicted pictures are computed based on filtered versions of the previous ones. A filter is applied to the edge boundaries of the four luminance and two chrominance 8×8 blocks. The filter is applied to a window of four edge pixels in the horizontal direction and it is then similarly applied in the vertical direction. The weight of the filter's coefficients depend on the quantizer step size for a given macroblock, where stronger coefficients are used for a coarser quantizer. This mode also allows the use of four motion vectors per macroblock, as specified in the Advanced Prediction mode of H.263, and also allows motion vectors to point outside picture boundaries, as in the Unrestricted Motion Vector mode. The above techniques, as well as filtering, result in better prediction and a reduction in blocking artifacts. The computationally expensive overlapping motion compensation operation of the Advanced

[6] While spatial error propagation can be avoided across picture segments, coding efficiency will be reduced because predictive coding would not be employed.

Prediction mode is here not used in order to keep the additional complexity of this mode minimal.

The Deblocking Filter mode improves subjective quality by removing blocking and mosquito artifacts common to block-based video coding at low bit rates. Many applications also make use of a post filter to reduce blocking and ringing artifacts. This post filter is usually present at the decoder and is outside the coding loop. Therefore, prediction is not based on the post filtered version of the picture. We present here subjective results for the deblocking filter and a post-filtering in Fig. 2.8. In our simulations, we used the post filter described in the TMN-8 model [48] for comparison with the deblocking filter. In the figure, results are shown for the sequence **FOREMAN** decoded using the deblocking filter alone, TMN-8 post filter alone and both the deblocking and post filters at 24 kbps and 10 fps. The reconstructed picture for frame number 75 is shown. The deblocking filter alone reduces blocking artifacts significantly, mainly due to the use of four motion vectors per macroblock. The filtering process provides smoothing, further improving subjective quality. The effects of the post filter are less noticeable, and adding the post filter may actually result in blurriness. Therefore, the use of the deblocking filter alone is usually sufficient.

Like the Advanced Prediction mode of H.263, the Deblocking Filter mode involves using four motion vectors per macroblock. This requires additional motion estimation, increasing the computational load and resulting in a 5-10% additional encoding time. However, if the Advanced Prediction

mode is employed, the additional computational requirements associated with the Deblocking Filter mode are quite small. The Advanced Prediction mode already involves using four motion vectors, and only some additional filtering operations are required.

In an error prone environment, the deblocking filter improves the subjective quality by filtering artifacts caused by both block coding and error concealment. Therefore, the use of this mode is advisable when channel errors are expected.

2.5.3 Bit rate control

Rate control methods are necessary for constant bit rate video communication applications. These methods usually vary the quantization step size at the macroblock level in order to achieve the bit rate budget. Moreover, frame level rate control methods may also be employed to adjust the encoded frame rate according to the available bit rate.

The H.263 Test Model Version 8, TMN-8 [48], describes two rate control algorithms suitable for low delay videophone applications. Both methods use a buffer regulation scheme at the frame level in which a target bit rate is chosen and frames are skipped until the buffer reaches a limit below the number of bits required to transmit the next frame. Since encoding delays are directly related to buffer fullness, large variations in buffer content will produce undesirable variable delays.

The most recent Test Model rate control method is based on a model

that chooses an "optimal" quantizer for every macroblock in a given picture. The details and theory underlying this technique can be found in [49], while a step-by-step implementation example is available in the H.263 Test Model TMN-8 [48]. We describe next this rate control method, which consists of a frame level rate control and a macroblock level rate control.

Frame level rate control

In this section, we describe the algorithm employed for the frame level rate control [48]. The following definitions are used in describing the algorithm:

- B' - Number of bits occupied by the previous encoded frame.
- R - Target bit rate in bits per second (e.g., 10000 bps, 24000 fps, etc.).
- G - Frame rate of the original video sequence in frames per second (e.g. 30 fps).
- F - Target frame rate in frames per second (e.g., 7.5 fps, 10 fps, etc.). G/F must be an integer.
- M - Threshold for frame skipping. By default, set $M = R/F$. (M/R is the maximum buffer delay.)
- A - Target buffer delay is AM sec. By default, set $A = 0.1$.

The number of bits in the encoder buffer is $W = \max(W + B' - R/F, 0)$.

The *skip* parameter is set to 1, $skip = 1$.

While $W > M$

{

$W = \max(W - R/F, 0)$

$skip++$

}

Skip encoding the next $(skip \times (G/F) - 1)$ frames of the original video sequence. The target number of bits per frame is $B = (R/F) - \delta$, where

$$\delta = \begin{cases} \frac{W}{F}, & W > A \times M \\ W - A \times M, & \text{otherwise} \end{cases} \tag{2.8}$$

For fixed frame rate applications, skip is always equal to 1. The frame skip is constant and equal to $(G/F) - 1$. Also, when computing the target number of bits per frame, A should be set to 0.5.

Macroblock level rate control

First, the variances of all macroblocks in the motion compensated picture are calculated. Based on these variances, and the remaining bits available for encoding the current picture, model parameters are updated. These parameters are then used to find an "optimal" quantizer for each macroblock. One of the model parameters allows for the weighting of macroblocks based on perceptual importance. The test model describes a simple method to calculate this parameter where a macroblock with high spatial activity (higher variance) is assigned a finer quantizer. The remaining of this section describes step-by-step the TMN-8 macroblock level rate control method [48].

Step 1. Initialization It is assumed that the motion vector estimation has already been completed. σ_k^2 is defined as the variance of the luminance and chrominance values in the k^{th} macroblock. If the k^{th} macroblock is intra coded, set $\sigma_k^2 = \sigma_k^2/3$. Let:

$i = 1$ and $j = 0$

$\tilde{B}_1 = B$, the target number of bits for the frame.

$N_1 = N$, the number of macroblocks in a frame.

$K = K_1 = K_{prev}$, the initial value of the model parameters.

$C = C_1 = C_{prev}$

$S_1 = \sum_{k=1}^{N} \alpha_k \sigma_k$

where

$$\alpha_k = \begin{cases} 2\frac{B}{16^2 N}(1 - \alpha_k) + \alpha_k, & \frac{B}{16^2 N} < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{2.9}$$

Step 2. Compute Optimized Quantizer for i^{th} macroblock If $L = \tilde{B}_i - 16^2 N_i C) \leq 0$ (running out of bits), set the quantizer, Q_i to the largest step-size possible. Otherwise, compute:

$$Q_i = \sqrt{\frac{16^2 K \sigma_i}{L \alpha_i}} S_i. \tag{2.10}$$

Step 3. Update Counters Let B_i' be the number of bits used to encode the i^{th} macroblock, compute:

$$\tilde{B}_{i+1} = \tilde{B}_i - B_i', S_{i+1} = S_i - \alpha_i \sigma_i, \quad \text{and} \quad N_{i+1} = N_i - 1. \tag{2.11}$$

Step 5. Update Model Parameters K and C The model parameters measured for the i^{th} macroblock are:

$$\hat{K} = \frac{B'_{LC,i}(Q_i)^2}{16^2\sigma_i^2}, \quad \text{and} \quad \hat{C} = \frac{B'_i - B'_{LC,i}}{16^2}. \tag{2.12}$$

where $B'_{LC,i}$ is the number of bits spent for the luminance and chrominance of the macroblock.

Next, measure the average of the $\hat{K}$'s and $\hat{C}$'s computed so far in the frame. If ($\hat{K} > 0$ and $\hat{K} < \pi \log_2 e$), set $j = j + 1$ and compute $\tilde{K}_j = \tilde{K}_{j-1}(j-1)/j + \hat{K}/j$. Compute $\tilde{C}_i = \tilde{C}_{i-1}(i-1)/i + \hat{C}/i$.

Finally, the updates are a weighted average of the initial estimates, K_1, C_1, and their current average:

$$K = \tilde{K}_j(i/N) + K_1(N-i)/N, \tag{2.13}$$
$$C = \tilde{C}(i/N) + C_1(N-i)/N.$$

Step 6. If $i = N$, stop (all macroblocks are encoded), and set $K_{prev} = K$ and $C_{prev} = C$. Otherwise, let $i = i + 1$, and go to Step 2.

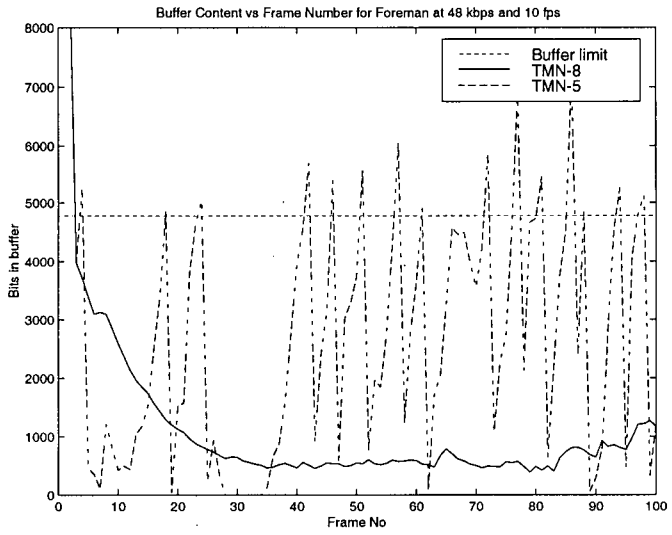
Comparison of rate control methods

The alternate rate control method described in the Test Model, also known as TMN-5 rate control method, uses a simpler technique for adapting the quantizer. In this method, the quantizer is changed every macroblock row based on a simple relation between the number of bits remaining for the current picture and the quantizer step size. This method is simpler to implement than

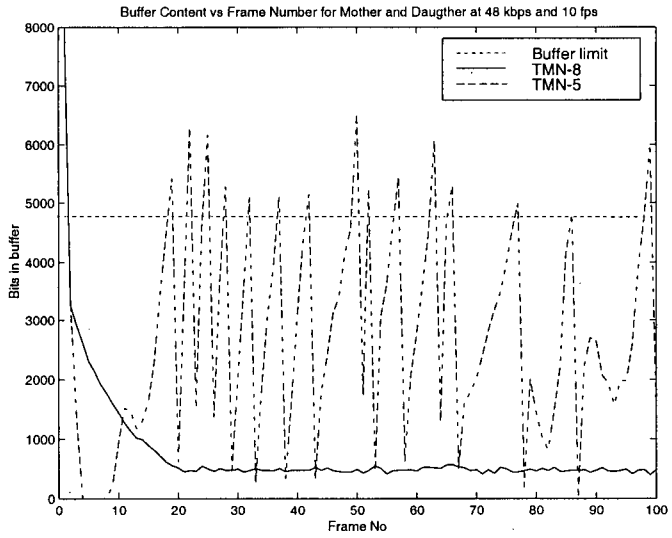
the one previously described, but it also does not provide accurate quantizer selection making it less effective.

Both Test Model (TMN) bit rate control algorithms were implemented. In general, for a given bit rate, the two methods usually achieve similar PSNR level. However, TMN-8 bit rate control method [48, 49] achieves the target bit rate more accurately. Moreover, it keeps a buffer content well below the maximum level thus reducing frame skipping as well as delay. If it is assumed that the decoder simply repeats the previous frame to replace a skipped frame, then the new bit rate control method performs better, in terms of PSNR, for a given bit rate. It has also been reported in [49] that the TMN-8 rate control method can outperform the TMN-5 method by as much as 1.2 dB when the same number of frames are coded.

Fig. 2.9 illustrates buffer fullness per frame for the video sequences FOREMAN and MOTHER AND DAUGHTER at 48 kbps and 10 fps. Whenever buffer content reaches the model limit, frames are repeatedly skipped at the encoder until the buffer content is below the limit. In the case of TMN-5 rate control, many frames are skipped, reducing temporal resolution, which can be critical in applications such as sign language and/or lip reading. Furthermore, the buffer content varies substantially from frame to frame (i.e. exhibiting high variance), introducing variable delays at the decoder. Finally, as buffer underflow occurs quite frequently, the available bandwidth is often not fully utilized. On the other hand, TMN-8 rate control maintains a desirable, constant buffer fullness, offering a low and, more importantly, near-constant delay.



(a)



(b)

Figure 2.9: Comparison of Test Model rate control methods based on encoder buffer regulation for (a) FOREMAN and (b) MOTHER AND DAUGHTER at 48 kbps and 10 fps

Furthermore, the available bandwidth is fully utilized by avoiding buffer underflow.

While TMN-8 rate control method is superior in terms of buffer fullness control performance, the number of computations increases, due mainly to variance calculations. In our implementation, this increases encoding time by approximately 5%. However, better computation-performance tradeoffs may be obtained by using, instead of variances, SAD values. Moreover, the calculation of variances may be incorporated in the motion estimation process further reducing computational complexity. In the remainder of this thesis, we always employ TMN-8 rate control method to achieve a target video bit rate.

Chapter 3

Error Resilience Video Coding

3.1 Introduction

Up to now, we have discussed video coding algorithms when error free transmission is assumed. Video transmission over error prone environments requires that the video coding process takes into account the possibility that the bit stream will be affected by channel errors. In this chapter, we first describe the error propagation process in the video coder. Next, we present a review of error resilience video coding methods that have been proposed in the literature. A brief introduction of video packetization for the transport of video bit streams over networks is then introduced. Finally, error concealment, which is a necessary process in a video communication system for error prone networks, is discussed.

scribed as follows. As illustrated in Fig. 2.1 and 2.2, the input video block x_n at time n is first predicted from the motion compensated block in the previous frame, y_{n-1}, resulting in an error block e_n. Assuming an error free channel, $r_n = s_n$, $\hat{y}_n = y_n$ (i.e. the reproduction block at the encoder is identical to the one at the decoder) and the only lossy stage is quantization. Thus the error between the encoded signal and the decoded signal is $\hat{y}_n - x_n = q_n$, where q_n is the quantization error. However, if a bit error occurs during transmission, the received block at the decoder, r_n, will be corrupted and concealed during the decoding process. In this case, $\hat{y}_n = c_n$, where c_n is the block used for concealment. Thus, the error for the next block predicted from the previously concealed one, assuming this next block is received correctly, will be $\hat{y}_n - x_n = q_n + (\hat{y}_{n-1} - y_{n-1}) = q_n + (c_{n-1} - y_{n-1})$, where $(c_{n-1} - y_{n-1})$ is the concealment error. Therefore, when computing the total error due to coding and transmission, we can consider quantization errors, and errors due to prediction from a concealed block, separately.

Until the prediction process based on erroneous data is stopped, the reconstructed frames at the receiver will be unsynchronized with those at the encoder. Prediction errors will occur at the decoder, even if future information is received without errors, since the reconstructed frame used for prediction at the decoder is different than the corresponding frame used in the encoding process. When a feedback channel is available, the encoder can estimate the error propagation based on information about the status of received blocks at the decoder and take action to minimize error propagation. However, when a

feedback channel is not available, it is very difficult to assess the effects of error propagation of random, and often time-varying, errors. In such a situation, it is still possible to characterize such error propagation effects using statistical models, as proposed later in this thesis.

In the remainder of this section, we discuss the effects of single bit errors and packet losses on spatio-temporal error propagation.

3.2.1 Effects of bit errors

In bit oriented networks, bit errors can be detected at the transport level or at the source decoding level, depending on the adopted protocol environment and system design method. If a video decoder decides to drop a video segment that contains bit errors, then the impact of errors is equivalent to the loss of the complete segment. If transport level error detection is not available or if the system decides to knowingly process incorrect data, bit errors will be present at the video decoder. Such errors can still be detected using syntactic or semantic violations of the video bit stream during the error handling process at the decoder, as discussed in Section 3.6. Since the decoder almost always loses synchronization with the bit stream and fails to locate the bit errors while decoding corrupted VLCs, data between two synchronization (sync) words is usually discarded, as illustrated in Fig. 3.2. Decoding of information that is corrupted by errors lead to annoying artifacts. Rather, error concealment is usually applied to the missing blocks corresponding to the missing bits between the two synchronization markers. It may be possible to estimate the

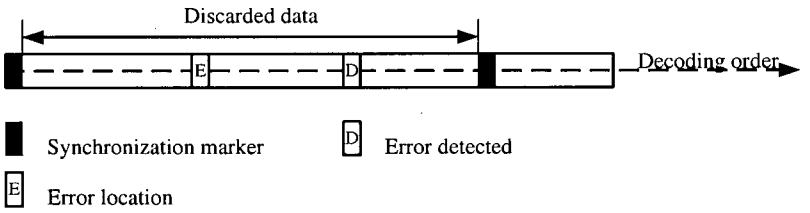


Figure 3.2: Error detection cannot locate a bit error when decoding VLCs: the data between two synchronization words is usually discarded.

location of the bit error based on the first known syntactic or syntax violation in the bit stream. All bits between the last sync marker and this estimated bit error location may be used for the reconstruction of blocks in question. This strategy has the advantage recovering more information, but the disadvantage of possibly decoding blocks with errors. It may also be possible to recover additional information using alternative entropy coding methods that allow the decoder to stay synchronized with the bit stream, as discussed in Section 3.3.2.

3.2.2 Effects of packet losses

Packet losses occur in packet networks, such as ATM networks or the Internet. Moreover, as discussed above, bit errors may be seen as packet losses at the video decoder, depending on how such errors are handled. Packet sizes, spatio-temporal location of packets within the video sequence and the extent of predictive coding will determine the impact of packet losses on error propagation. Depending on the size of packets, a packet loss may affect only a small spatial location of a video frame, or it can lead to the loss of complete

coded video frame(s). If the packet loss occurs within a limited spatial location within a video frame, error concealment may be applied to the missing video data. If complete frames are lost, the decoder can choose to repeat the last correctly received frame, or it can perform temporal frame interpolation to maintain an adequate frame rate. Nevertheless, when any type of errors occur in a predictive coding system, proper action is necessary at the encoder in order to limit the effects of spatio-temporal error propagation.

3.3 Error Resilience Video Coding Techniques

In this section, we describe current video error resilience techniques. We first discuss channel coding techniques adapted for video data. Spatial error resilience coding techniques are described next. Finally, temporal error resilience coding techniques are discussed.

3.3.1 Channel-based error recovery techniques

Forward error correction (FEC) techniques and Automatic repeat request (ARQ) techniques [50] are used extensively for channel error recovery in data communication [51]. We describe next their application in a video communication system.

Forward Error Correction

FEC techniques can be used effectively in a video communication system for the detection and correction of bit errors. In this open-loop method, the

recovery from errors is the responsibility of the decoder. The transmitter adds redundant parity bits which can be used, to a certain extent, to detect and (possibly) recover lost information. FEC methods add some amount of delay since the information is grouped into blocks of bits before adding parity information. Using FEC techniques, recovery from single bit errors is possible but protection from long bursts of errors would require the addition of an excessive amount of parity bits and, consequently, a substantial increase in bandwidth and delay.

The performance of FEC algorithms supported in current video coding standards is often limited, due mainly to the strict computational complexity and delay constraints set on practical video communication systems. For example, in H.263, the BCH (511,492) FEC code may be employed in a low bit error environment. This code adds 19 parity bits to a source block of 492 bits, and allows detection of 2-bit errors, and the correction of single bit errors. Introduced for ISDN, this FEC code is rarely employed in highly error prone environments such as mobile networks, due to its poor performance under such conditions.

When applying FEC to blocks of fixed sizes, it can be very difficult to align the start of an FEC block with the start of a video frame or slice. Most likely, the start of a video frame/slice will lie somewhere in the middle of the FEC block, and the loss of one FEC block may lead to the loss of more than one video frame/slice. It is possible to align frames/slices with the start of FEC blocks using stuffing bits. However, this overhead may be significant and

it may not be justified in some low bit rate applications.

In order to be more effective, FEC techniques can take into account the importance of the different parts of a video bit stream, thereby applying unequal error protection (UEP) [52]. The video bit stream is first partitioned into classes of different sensitivities. Important information such as frame size, temporal location of a frame, spatial location indication, coding mode, and quantization information has to be protected to ensure a minimum level of video reproduction quality. Errors in these header bits will not only affect the video reproduction quality, but can also cause the decoder to reset due to resource allocation problems. For example, if the encoder changes picture size from QCIF to CIF and this information is not correctly received, the decoder may have to reset to accommodate higher than expected number of blocks and display resolution. FEC coding can be applied in a cascade manner, where the same FEC code is first applied to all header information, and then re-applied to the more important part of the same information. Since header information is usually a relatively small part of the bit stream, FEC overhead can be minimal.

FEC techniques can also be employed in a packet-based network [53]. In this case, one or more packet(s) with parity information is (are) calculated for a group of packets. A missing packet can be reconstructed using the information of the correctly received packets and the parity packet. Interleaving schemes can be employed to achieve resilience against burst packet losses. The main drawbacks of packet-based FEC techniques are similar to those of bit-oriented

techniques: the need of similar data packet sizes, and the added complexity, bandwidth requirements and latency. FEC techniques are currently used with some success in packet-based Internet audio telephony, but are yet to prove effective for packet based video transmission.

Automatic repeat request

ARQ is effective against burst errors and packet losses. In this closed-loop error recovery method, a feedback channel is maintained from the receiver to the transmitter. This feedback channel conveys the status of received packets, providing the encoder with the possibility of either re-transmitting erroneously received packets or containing the effect of their losses.

Different ARQ techniques have already been proposed for video communication in [54, 55]. In a video communication system, lost or erroneously received video data frames are retransmitted using a feedback channel. An ARQ buffer is present at the transmitting side which contains the video data frames that have been transmitted but not yet acknowledged, as well as those frames not yet transmitted. The decoder sends an acknowledgment (ACK) or a negative acknowledgment (NACK) back to the transmitter to inform it of the status of the received data frames. The transmitter will determine if the lost data frames should be retransmitted, based on the delay requirements of the system. It is important to avoid losing complete video frames, as complete frame losses have a significant impact on video reproduction quality. This can be done, for example, with a layered coder, which ensures that the base layer is received through retransmission, and the enhancement layer is only trans-

mitted when the ARQ buffer fullness is below a certain threshold based on the delay requirements and/or the network condition. Moreover, the enhancement layer may never be retransmitted in order to minimize delays, when the network is congested. Other methods include the spatial division of the source pictures into several picture segments, which can then independently be updated in case of a loss. Several of today's video coding standards do allow for such a segmentation of source pictures.

While ARQ techniques are usually more effective than the FEC ones, they cannot be used in many video communication applications such as TV broadcasting and video-on-demand, which do not support a feedback channel. Even when a feedback channel is available, real-time constraints require that ARQ delays be kept within an application-specific limit. Therefore, ARQ techniques are generally not suited for real-time video communication over error prone networks. If a video communication system employs ARQ, the existing data protocols (e.g. TCP) are usually used.

3.3.2 Spatial error resilience coding techniques

An important aspect of error resilience video coding is to contain the effects of errors. As mentioned earlier, locating the exact position of bit errors in a compressed video bit stream is often not possible, and, depending on the extent of the predictive coding and the entropy coding methods employed, a large amount of possibly correct information must be discarded. In the next sections, we first describe techniques that contain the effects of such

errors to small spatial regions without an excessive overhead rate increase. These techniques include the placement of synchronization markers and data partitioning.

It is also possible to recover information within a corrupted video segment using error resilience entropy coding methods. We describe briefly current methods including reversible variable length coding, the error resilient entropy code and fixed length coding.

Error resilience entropy coding methods are useful for bit error prone networks. However, they provide limited error resilience performance in a packet lossy environment. Multiple description (MD) coding techniques, which allow for the recovery of data in a packet lossy network, are discussed last.

Synchronization markers

Synchronization (sync) markers are codewords that are uniquely identifiable in the error-free bit stream. Therefore, a VLC, a combination of VLCs, or any other codeword combination in the bit stream cannot reproduce the sync word. Moreover, the sync word must also be detectable in the presence of bit errors. These restrictions put a limit on the minimum length of the sync word. In H.263, for example, a 17-bit sync word equal to 00000000000000001 is employed.

In order to contain the errors to a small spatial region, sync words may be inserted at various locations, either at a uniform spatial interval in the coded frame, or at a uniform bit interval in the bit stream. In H.263 or MPEG-4, for example, sync words can be inserted at the beginning of every row of blocks

(uniform spatial interval), thus forming groups of blocks (GOBs). Using the slice structure, it is possible to insert sync markers before any coded block, regardless of the block's spatial position. It is then possible to insert a sync marker after a target number of bits for the slice has been reached (uniform bit interval)[1]. When sync words are inserted at a uniform bit interval, they will be closer (spatially) in high activity regions, since more bits are necessary to represent these parts of the picture.

In order to contain the errors within a slice, data dependencies across slice boundaries within the video frame are usually forbidden by video coding standards. To remove such dependencies, the encoder modifies the coding strategy at the beginning of a new slice. For example, the motion vector of the first block of a new slice is not predicted. Moreover, to ensure spatial synchronization at the decoder, additional fields, for important information such as the block address and quantizer value, are inserted in the bit stream after the sync word.

Some researchers have also proposed to modify sync words to provide additional error detection capabilities, instead of relying on the detection of invalid VLCs [56]. This is done by replacing some of sync word's bits by parity bits of blocks of data following the sync word. When searching for synchronization at the decoder, the parity bits for consecutive blocks of data are calculated and compared to the parity bits in the sync word. If an error is detected, the current parity bit along with the remaining bits of the sync

[1] A coded macroblock should not be divided by a sync marker. Therefore, sync markers are inserted at the next macroblock after the target number of bits in the slice has been reached.

marker are compared to the corresponding known sync word bits. If no error is detected, the remaining parity bits should match the sync word bits and synchronization is hence obtained. When the receiver is in synchronization with the bitstream, the parity bits are used to verify the integrity of the data. If an error is detected, the decoder conceals the associated blocks and looks for the next sync word.

Data partitioning

The term data partitioning is used for two fundamentally different mechanisms. In the first case, more than one representation of the same source picture is coded in order to convey video data on independent virtual channels with different characteristics, or identical characteristics, but using different error control schemes for the different representations. One typical example for such techniques is layered coding. The main reason why this mechanism is not yet widely employed in real systems lies in the lack of appropriate network infrastructure, that offers different QoS's for virtual channels between the same physical endpoints. Future developments in networking, such as the recent Resource ReSerVation Protocol (RSVP) [57], may offer such options.

In the second case, different types of video data within the same representation of the source picture are separated. For example, data partitioning in MPEG-4 separates the two main components of a video bit stream: motion vectors and DCT coefficients. In this case, the different partitions do not require a different delivery guarantee. In this section, we will discuss the second approach to data partitioning.

Without the use of data partitioning, the motion vectors and DCT coefficient information in a DCT-based video coder are usually coded together for every block. This eliminates the overhead for addressing a block when both classes of information are skipped. When using data partitioning, the motion vector and DCT coefficient information within a slice are respectively grouped together and separated by a boundary marker. The boundary marker is a uniquely decodable codeword that signals the end of the motion vector information and the beginning of the DCT coefficient information. Doing so yields several advantages. First, errors can be localized to a certain type of data, and the unaffected information can be employed for video reconstruction at the decoder. For example, when an error occurs in the DCT data but the motion vector data is not affected, the motion vector information can still be used to assist in error concealment. Second, if undetected errors occur in the video packet, the received data can be considered invalid if the boundary marker is not detected. However, if the boundary marker is detected but the block address is incorrect after detecting the sync word at the start of the next slice, the decoder can assume that the motion vector information was correct and discard the DCT coefficients information.

Data partitioning provides improved video quality under different error conditions, as reported in [58] for MPEG-4 and in [59] for H.263.

Reversible variable length codes

Reversible VLCs (RVLCs) [8] must satisfy the prefix condition for instantaneous forward decoding as well as the suffix condition for instantaneous

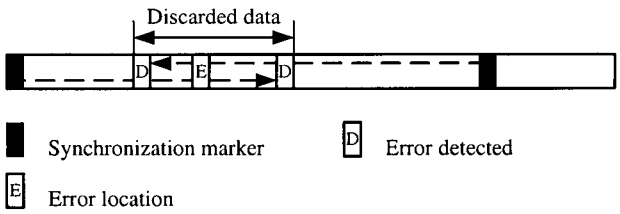


Figure 3.3: Reversible Variable Length Codes enable the recovery of data by allowing decoding in the forward and reverse directions.

backward decoding. The prefix condition guarantees that each codeword does not coincide with the prefix of longer codewords while the suffix condition guarantees that each codeword does not coincide with the suffixes of longer codewords. VLCs usually satisfy only the prefix condition. RVLCs can therefore be uniquely decoded in both directions. When the decoder detects an error while decoding in the forward direction, it can look for the next sync marker and start decoding in the reverse direction until an error is encountered. Based on the position of the detected errors in the forward and reverse directions, the decoder is able to locate the error within a smaller region in the bit stream and recover additional symbols, as illustrated in Fig. 3.3. This does not necessarily guarantee that additional blocks will be recovered, as discussed below.

One simple method to construct RVLCs is to add a fixed length prefix and suffix to a constant Hamming weight VLC code set. The decoder is able to decode in the forward and reverse directions by counting the number of 1s in the code to detect the end of the code. More complete RVLCs can be generated by increasing the length of the prefix. RVLCs can also be designed

by modifying the Golomb-Rice code, usually providing better compression efficiency [9].

RVLCs can be constructed to be symmetric [8]. However, since they allow a more flexible bit assignment, asymmetrical RVLCs provide better compression efficiency than symmetrical RVLCs, at the expense of the need to have two separate coding tables. To construct an asymmetrical RVLC, we start with an optimal non-reversible VLC, such as a Huffman code, which satisfies the prefix condition. The suffix of each codeword can be extended by adding bits, as illustrated in Fig. 3.4 and described below:

1. A reverse binary tree is set up in such a way that the ends of the codewords are placed on the root of the reverse tree from the shorter codewords to the longer codewords (see Fig. 3.4(b) and (c)).

2. When a suffix, which is also a prefix in the reverse tree, coincides with a shorter codeword, another codeword with the same bit length is assigned in the reverse tree instead. In Fig. 3.4(d), "D" is assigned in the reverse tree instead of "C" since the suffix of "C" coincides with "A".

3. When there is no other codeword with the same bit length whose suffix does not coincide with one of the shorter codewords, the minimum number of bits needed to satisfy the suffix condition are added to the end of the codeword. In Fig. 3.4(e), one bit "1" is added to the end of "C".

4. After bit length assignment is completed, new codewords are sorted by bit length, and codewords are re-assigned to the symbols according to

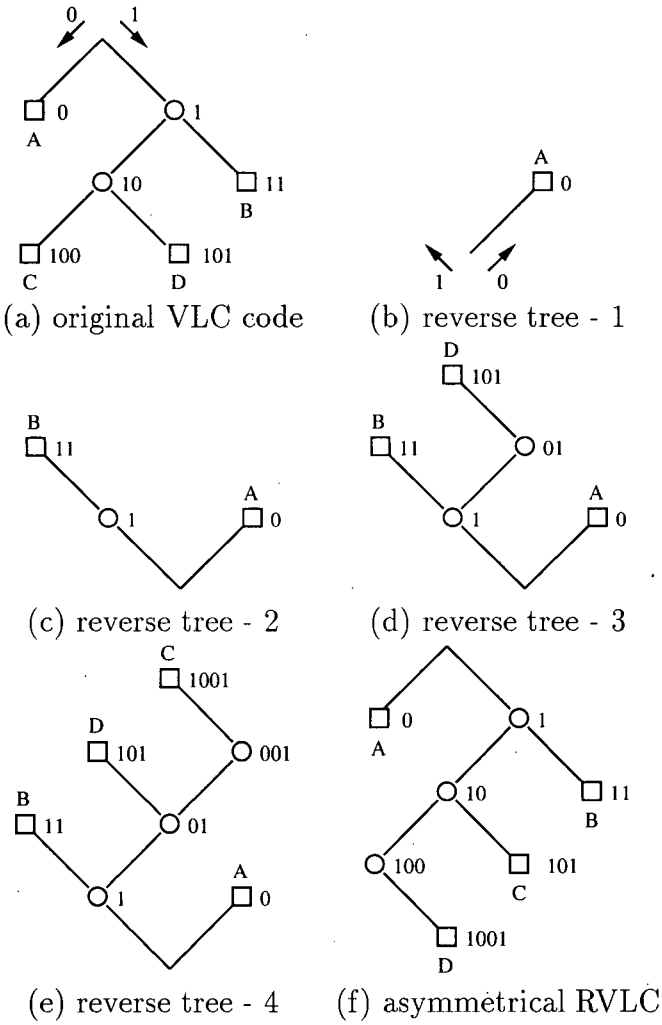


Figure 3.4: Conversion from a VLC set into an asymmetrical RVLC one.

their occurrence probabilities (Fig. 3.4(f)).

Other methods of generating RVLCs have been proposed in [8] and [9]. In general, the disadvantage of two-way decodable RVLCs is a reduction in compression efficiency. RVLCs can be designed to match the probability distribution of the coded video data, being motion vector information or run-length coded DCT coefficients, such that the compression efficiency is maximized. Also, to take full advantage of the RVLCs properties, codewords representing the same type of data should be grouped together. Thus, RVLCs are usually employed with data partitioning.

The error resilient entropy coding (EREC) method

The EREC method, originally proposed in [10], is used to convert VLCs to fixed length blocks of data. This allows the decoder to be synchronized with the bit stream at the start of each EREC frame with minimal additional redundancy.

The EREC frame structure is composed of N slots, each of length s_i bits, for a total length of $T = \sum_{i=1}^{N} s_i$ bits per EREC frame. It is assumed that the values of s_i, T, and N are known by the decoder. The slot lengths s_i can be predefined as a function of T. The number of EREC slots N can usually be fixed in advance, and thus, only T needs to be transmitted to the decoder.

The VLC blocks of data of length b_i are placed into an EREC frame using a bit reorganization structure that relies on the ability to determine the end of each variable-length block. Fig. 3.5 shows an example of the

The parameter ϕ_n is a predefined offset sequence, and it is set to 1 in the example illustrated in Fig. 3.5. A pseudo-random offset sequence ϕ_n was shown to provide better error resilience properties [10], because of its uncorrelated nature, i.e., $\phi_{n+k} - \phi_n$ is independent of n. Thus, two VLC blocks k slots apart will not be searched in the same order.

It was shown in [10] that the EREC method provides a relatively large improvement in reproduction quality when applied to DCT-based still image and video coding as compared to using VLCs with GOB sync markers. In [60], a joint source-channel coder using H.263 video coding and the EREC method was developed, yielding a significant improvement over the use of VLCs. It is important to note that the EREC method requires that some information be transmitted on highly protected channels, and that it does not guarantee image/video frame spatial synchronization. Therefore, sync markers should still be employed.

Fixed length codes

Fixed length codes (FLCs), also known as block codes, are less sensitive to error propagation than VLCs. As opposed to the VLC case, where the input symbols are of fixed length and the output codewords are of variable length, the FLC process converts variable length input codewords to fixed length block codes. An algorithm to construct optimal FLCs, in terms of minimum output bit rate, was proposed by Tunstall [11]. FLCs for video compression was proposed by Llados-Bernaus in [61].

The lower bound on the average length of a FLC is the same as that of

a VLC, which is the entropy of the source U, $H(U)$, for a discrete memoryless source. However, the lower bound of a FLC is limited by the probability of the less likely symbol, p_{min}. As a result, the coding efficiency decreases as the alphabet size increases. Modified Tunstall codes were first presented in [62], and later in [12], to address this particular drawback.

The upper bound of the average length of a FLC is different from that of a VLC. In a VLC, the upper bound is $H(U)+1$. In a FLC, the upper bound decreases as the length of the output codewords, N, increases. This puts a limit on the minimum output codeword length possible to achieve a certain level of compression efficiency.

As for RVLCs, data partitioning should be used with FLCs to improve their coding efficiency, since different data types may use different fixed block sizes. Even though FLCs allow codeword synchronization, they do not guarantee spatial synchronization, and sync codewords are therefore still necessary. For example, if a fixed length code word representing DCT information is corrupted by errors and the resulting codeword is still valid but represents a different spatial location in the video frame, i.e., a different DCT block, the following data will be reconstructed at the wrong location. Large areas of the frame will be displaced until synchronization is re-gained by detecting a sync codeword. However, FLC codeword assignment can be designed to minimize such errors, as proposed in [63].

Multiple Description Coding

Multiple description (MD) coding allows a video decoder to extract meaningful information from a subset of the bit stream. For example, an SNR scalable bit stream allows meaningful information to be retrieved when only the base layer is received. However, in order to make use of the enhancement layers, the base layer must be received correctly. This can only be assured if priority classes are supported by the network. MD coders also have the ability to produce descriptions that may be equally important (balanced descriptions) and are also mutually refinable, i.e., receiving two descriptions together and combining them is better than receiving the descriptions separately.

Fig. 3.6 shows a block diagram of a typical MD coder. An encoder produces two descriptions that are transmitted over two channels. The encoder has no knowledge of the status of the channels. The two-channel model is usually employed to represent two different network packets transmitted on the same network. At the receiving end, we have three decoders: a channel 1 decoder, a channel 2 decoder, and a combined channel 1 and 2 decoder. If information from channel 1 or channel 2 only is received, the corresponding decoder is able to reconstruct a video sequence with distortion level D_1 or D_2, respectively. If information from both channels is received correctly, then the central decoder is able to reconstruct a video sequence with distortion level D_0. The achievable rates for given distortion levels D_0, D_1, and D_2 were derived in [64] for a Gaussian source.

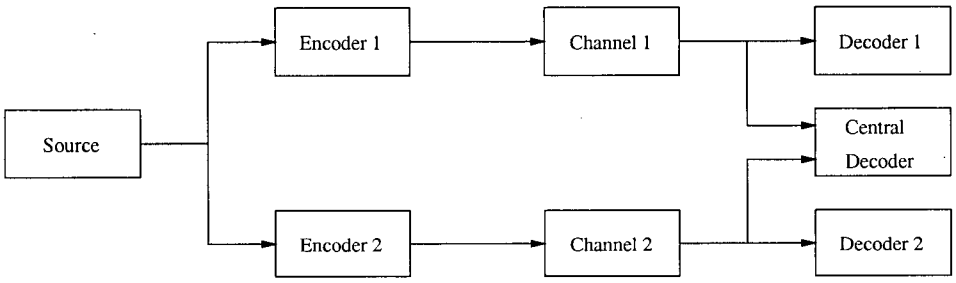


Figure 3.6: Multiple Description coder.

Multiple Description Scalar Quantizers Different methods have been proposed for the design of MD coders. One method is to design multiple description scalar quantizers [13]. The input signal x is quantized to yield an integer index $l = q(x)$, where $q(\cdot)$ represents the mapping of a uniform threshold quantizer with central bin index l. Information about l is mapped to a pair of indexes $(i, j) = a(l)$. The index i is transmitted on channel 1, while the index j is transmitted on channel 2. If information for channel 1 or 2 only is received, the distortion level D_1 or D_2 will be incurred, respectively. If information from both channels is received, the index pair (i, j) is mapped back to the central index l through $a(l)$ to reconstruct x with distortion level D_0. Fig. 3.7 presents an example of a MD quantizer mapping. If only one index is received, it is possible to estimate the index l by choosing the central index in the row/column of the received index i or j, respectively. Details of index assignments can be found in [13]. This technique has been applied to intra coding of blocks in a DCT-based image/video coding framework. Significant gains in video production quality were reported in a packet lossy environment

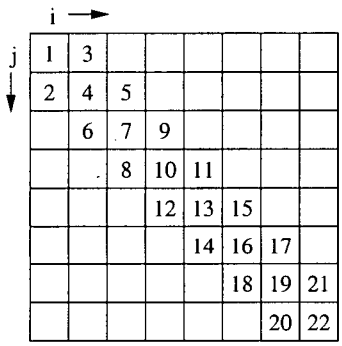


Figure 3.7: Example of a multiple description quantizer assignment for a 22 level quantizer.

[13].

Multiple Description Transforms Another method to obtain multiple descriptions is to use transforms that introduce a controlled level of correlation between the two descriptions [14]. This is realized by forcing dependencies between pairs of transformed coefficients, allowing either coefficient to be estimated from the other when one is lost. A transform based MD method codes a pair of input variables A and B by forming and coding a pair of transformed variables C and D, where

$$\begin{bmatrix} C \\ D \end{bmatrix} = \mathbf{T} \begin{bmatrix} A \\ B \end{bmatrix}. \tag{3.1}$$

Redundancy is added by controlling the correlation in the pair (C, D) of transformed coefficients through $\mathbf{T}$. The one-channel distortion is reduced because the correlation between C and D allows the information from the unreliable channel to be estimated from information received on the reliable channel.

A transform MD method can be applied to blocks of transform coefficients in a DCT-based video coder as follows. Quantized DCT coefficients of the same frequency belonging to spatially neighbouring blocks are paired as input variables (A, B). Blocks are paired spatially far apart in order to minimize correlation between paired coefficients. A correlating transform T is designed for each group of paired frequencies, depending on their visual importance. These correlating transforms are applied to each quantized transform coefficient pair, then entropy coding is applied to the correlated variables. As they are the most important, DC coefficient pairs would be coded with high redundancy (correlation).

Generalized Multiple Description Coding The MD techniques described above assume that only two channels are available. When applied to packet network communications with possibly many packets per image, using more than two descriptions would increase performance. This problem of generalized multiple description coding (GMDC) was addressed in [15], and later applied to images using correlating transforms in [65, 66].

3.3.3 Temporal error resilience coding techniques

As seen earlier, error propagation due to motion compensation can be difficult to evaluate, and such errors can propagate over a large frame area and over many frames. This section describes current methods to contain the effects of errors caused by temporal prediction.

Temporal error resilience can be achieved when it is possible to use an

earlier picture than the last transmitted one for temporal prediction. This reference picture is chosen to minimize error propagation. This can be done with or without a feedback channel, using reference picture selection or video redundancy coding, respectively. These two techniques are discussed next. Perhaps superior temporal error resilience can be achieved by 1) random intra coding of blocks, 2) intra coding of blocks based on feedback information. Such intra coding methods are described next.

Reference picture selection

Reference picture selection (RPS) dynamically replaces reference pictures in the encoder in response to an acknowledgment signal from the decoder sent through a feedback channel [67]. This method is also known in the literature as the NEWPRED method. Two modes of operation are defined depending on the acknowledgment message. In the ACK mode of operation, the decoder sends a positive acknowledgment message to the encoder every time a picture is received correctly. The encoder replaces the reference picture according to the returned ACK. If an error occurs, no ACK is sent back and the reference picture remains the last known to be correct frame. In the NACK mode of operation, the decoder send a negative acknowledgment message to the encoder only when a picture has not been received correctly and the encoder adjusts the reference picture accordingly. Once a NACK message is sent by the decoder, the decoding process is stopped until the erroneous picture is received correctly.

Both the ACK and NACK methods have their advantages and disad-

vantages. If the round trip delay is longer than the video frame interval, using the ACK method will lead to a loss in coding efficiency since the reference picture will be further than necessary. However, temporal error propagation will not occur, as a reference picture that is not available at the decoder will not be used for prediction at the encoder. In the NACK method, an advantage is that feedback delay will not lead to a loss in coding efficiency. A disadvantage, however, it that the decoding process is stopped until the subject erroneous picture is received correctly.

Both methods are also sensitive to errors in the feedback channel. In the ACK case, if an error in the feedback channel occurs, the reference picture is not replaced, leading to a decrease in coding efficiency, but no serious picture quality degradation at the decoder. In the NACK case, if an error occur in the feedback channel, the encoder will not use the correct reference picture, until a NACK is correctly received, leading to a significant picture degradation at the decoder. It has been confirmed that the NACK method is effective for low error rates while the ACK method is effective for high error rates [67]. It was also shown that when errors in the forward and back channel are correlated, the ACK method outperforms the NACK method.

Since the number of stored reference pictures is limited and those reference pictures are usually kept in a sequential order, it might even be necessary for the encoder to react by sending a full I-picture, if it observes that no correctly received reference picture is available at the decoder. In the case of point-to-point connections with low transmission delay characteristics, feed-

back channel mechanisms are a valuable addition to achieve temporal error resilience, especially if augmented by error concealment techniques. In application scenarios that involve some ten, hundred, or even more endpoints and that are based upon multicast or broadcast communication mechanisms, as available in LANs, Intranets, and the Mbone of the Internet, feedback channels are obviously not practical.

Video redundancy coding

Video redundancy coding (VRC) improves temporal error resilience using multiple prediction options without the use of a feedback channel [68]. The principle of VRC is to divide the sequence of pictures into two or more threads in such a way that all pictures are assigned to one of the threads in a round-robin fashion. Each thread is coded independently. Obviously, the frame rate within one thread is much lower than the overall frame rate: half in case of two threads, a third in case of three threads and so on. This leads to a substantial coding efficiency penalty because of the generally larger changes and the longer motion vectors, typically required to represent accurately the motion related changes between two P-pictures within a thread. In regular intervals, all threads converge into a so-called sync frame. From this sync frame, a new thread series is started.

Fig. 3.8 illustrates VRC with two threads, five pictures per thread, and three threads, three pictures per thread. If one of these threads is damaged because of a packet loss[2], the remaining threads stay intact and can be used

[2]It is here assumed that every coded picture is transmitted as a packet.

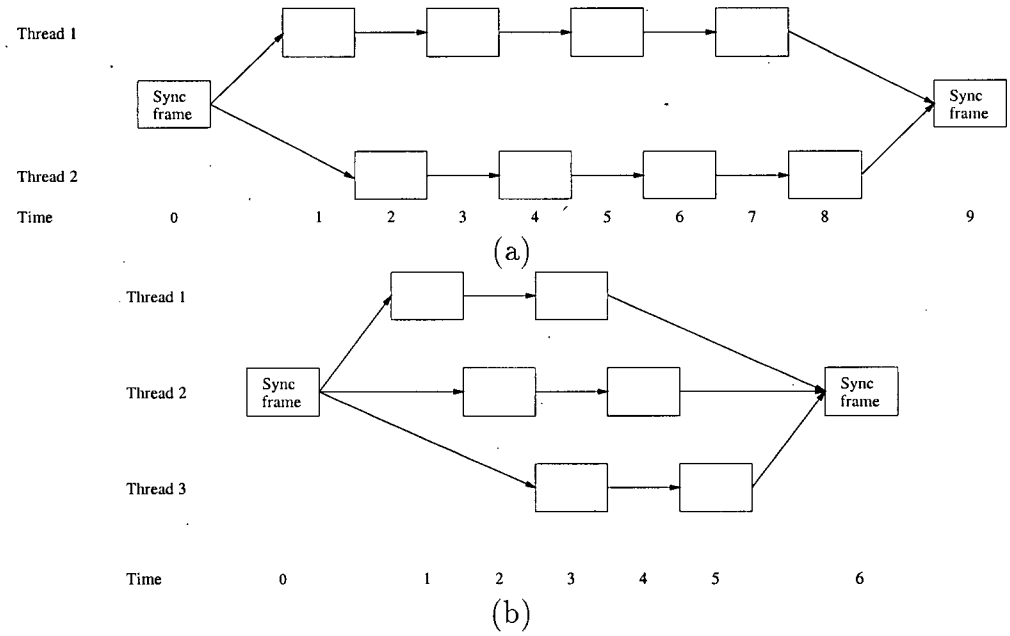


Figure 3.8: VRC example: (a) two thread, five pictures per thread, (b) three thread, three pictures per thread.

to predict the next sync frame. It is possible to continue the decoding of the damaged thread, which leads to slight picture degradation, or to stop its decoding which leads to a drop in the frame rate. If the length of the threads is kept reasonably small, both degradation forms will persist only for a very short time, until the next sync frame is reached. Sync frames are always predicted out of one of the undamaged threads. This means that the number of transmitted I-pictures can be kept small, because there is no need for complete re-synchronization. Only if all threads are damaged between two sync frames, a correct sync frame prediction is no longer possible. In this situation, annoying artifacts will be present until the next I-picture is decoded correctly, as it would have been the case without employing VRC. The choice of the number of threads and the number of frames per thread depends on the expected network characteristics. Many Shorter threads with fewer frames per threads is preferable for high packet loss rates, at a lower coding efficiency when errors do not occur, since the distance between the coded frame and it's reference frame will increase as the number of threads increases. Experimental results [68] show that VRC with three threads and three pictures per thread provides good video quality for a picture loss rate of 20%.

Random intra coding

A simple technique to avoid error propagation in the temporal direction is to increase the frequency of intra coded frames. However, intra coded frames require a substantially larger number of bits than inter coded frames, resulting in high latency and macroblock error rate in a constant bit rate, error prone

environment. Instead of intra coding a complete video frame, it is often preferable to intra code only some blocks within a frame. One simple technique is to intra code blocks in a random pattern. Random intra coding of blocks has been suggested in [69, 70]. In [69], Haskell proposes that the portion of blocks to be intra updated in a coded frame be chosen based on the life expectancy of the errors. He states that this life expectancy depends on the intra block refresh rate but is fairly independent of the probability p that a block is in error. Another method was proposed in [71], where only blocks with high activity are intra coded.

Intra coding based on feedback information

Using feedback information, the coding strategy of video blocks is modified so as to minimize the reproduction quality degradation given the motion compensation error propagation and the error concealment method employed [72]. Because the exact locations of the errors are known by the encoder, the coding strategy can be modified to prevent prediction from erroneously received blocks. Each time an error is detected during decoding, the video receiver reacts in the following manner. Damaged blocks are specified, based on the packetization structure. For example, all blocks between two sync words may be considered lost if any part of the bit stream between those sync words is corrupted by errors. Information about the lost blocks and their addresses is then sent back to the transmitter. Finally, the damaged blocks are concealed.

After receiving the feedback information, the video encoder responds by adapting its coding strategy in one of the following two methods. Using

method A, the affected picture area in the encoder reference frame is evaluated, and a tree is built to track the temporal propagation of errors resulting from the damaged blocks. The error propagation tree is constructed by keeping a table entry for each block, which includes spatio-temporal addresses of blocks used for its prediction. More sophisticated encoder error tracking methods have been developed in [73]. More specifically, the affected blocks at the encoder are weighted the number of pixels used for motion compensation. Encoding is then resumed without using the affected picture area (e.g., by not using the corrupted blocks for prediction in the inter mode or by intra coding the affected blocks). Tradeoffs between coding efficiency and error resilience can be achieved by adapting the number of affected blocks to be updated based on their effect on temporal error propagation. Using method B, the decoder error concealment is also applied to the damaged blocks at the encoder. The encoder's decoding loop is then updated.

Method A is less complex than method B, as only the propagation tree used to determine the affected picture area needs to be stored and updated at the encoder. However, method A also decreases coding efficiency. For a short round trip delay Method B is clearly more efficient.

Typically, the QoS of the link conveying feedback information is about the same as of the forward channel. Thus, a low-delay feedback channel is subject to similar transmission errors as the forward channel. As feedback data can have a substantial size due to the large number of spatial positions that have to be reported, it is not unlikely that the feedback data is lost

Standard	FEC	Sync words	RVLCs	Data Partitioning	Independent Segment Decoding	Reference Picture Selection	HD
H.263	Yes	Yes	Yes	No [a]	Yes	Yes	No[b]
MPEG-4	No	Yes	Yes	Yes	No	No [c]	Yes

[a]Data partitioning will be included in Version 3 of H.263.

[b]Although H.263 does not support this mode, the transport protocol for H.263 may support it, for example in RFC-2429.

[c]Reference picture selection will be included in Version 2 of MPEG-4.

Table 3.1: Error resilience tools supported in H.263 and MPEG-4.

or corrupted, with unpredictable results for the reconstructed picture quality. FEC can be employed to enhance the quality of the feedback channel messages.

3.4 Error Resilience Tools in Current Standards

H.263 includes error resilience tools which are defined in the baseline syntax and in four of its normative Annexes [46, 74]. The visual part of the MPEG-4 standard also provides support for error resilience [75, 76]. We here summarize the error resilience tools included in the above video coding standards and listed in Table 3.1.

3.4.1 Forward error correction

The FEC mode is the oldest of the error resilience oriented optional modes, and is available in H.261 and H.263. If this mode is enabled, the video bit stream is divided into FEC frames of 492 bits each. A (511,492) BCH forward

error correction checksum is calculated for all the bits of each frame, along with one bit that is necessary for the synchronization to the frame structure. This FEC mechanism was designed for ISDN, which is an isochronous, very low error rate network.

3.4.2 Synchronization words

For both packet lossy and bit error prone channels, the intelligent use of synchronization markers is necessary to ensure fast re-synchronization during decoding. Even in the baseline mode of operation, synchronization markers can be inserted using the optional GOB headers. Unfortunately, the size of coded GOBs cannot be chosen according to a fixed packet size, or adapted freely to a bit-interval chosen according to the error rate. More freedom in placing synchronization markers is allowed when slices are employed, which are optionally supported in H.263 and MPEG-4. As seen earlier, slices permit the insertion of synchronization markers after each coded macroblock, thus allowing for a more judicious placement of the synchronization markers within a coded picture. In addition to providing bit stream re-synchronization, sync word also allow for spatial re-synchronization within a coded video frame.

3.4.3 Reversible variable length codes

When Annex D of H.263 is used in Version 2 of the standard, the coding of motion vector information is performed using RVLCs. In MPEG-4, when the data partitioning mode is employed, DCT coefficient information is coded

using RVLCs. RVLCs were discussed in Section 3.3.2.

3.4.4 Data partitioning

The forthcoming third version of H.263 will likely include data partitioning, already supported by MPEG-4. Data partitioning was presented in Section 3.3.2. The syntax details of the partitioning, however, differ slightly in H.263 and MPEG-4. In H.263, the decoding of the motion vector data from both forward and reverse directions is possible, and in MPEG-4, the decoding of DCT information from both directions is possible, as they are respectively coded using RVLCs.

3.4.5 Independent Segment Decoding

As already described, slices are self-contained in so far that all prediction mechanisms within a coded picture are interrupted. This, however, does not prevent error propagation due to motion compensation. Error propagation can be prevented using the Independent Segment Decoding (ISD) mode, supported in H.263, which enforces the treatment of segment boundaries as if they were picture boundaries. A segment is defined as a slice, a GOB, or a number of consecutive GOBs with empty GOB headers. This mode allows the independent decoding of picture parts, if and only if, the shape of the independently decodable segments remains identical between two I-pictures. In such a case, error propagation outside the segment boundaries (due to motion compensation) can be avoided.

3.4.6 Reference Picture Selection

The Reference Picture Selection (RPS) mode, defined in Annex N of H.263, support reference picture selection and video redundancy coding. The methods can be applied either to pictures or to individual rectangular (GOB or slice) picture segments. If feedback messages are employed, they can be either multiplexed into the H.263 data stream of the opposite direction (VideoMux back channel sub-mode), or conveyed out of band (separate logical channel sub-mode). The VideoMux back channel sub-mode is only applicable for bi-directional video communication, because the back channel messages are conveyed within the video data of the opposite direction. The separate logical channel sub-mode is often seen as a more friendly way of conveying back channel data, but makes a separate data channel necessary, which may incur significant additional overhead in some environments.

3.4.7 Header Duplication

Some of the header information present at the beginning of a coded video frame must be received at the decoder to allow for the decoding of the frame. This includes information such as spatial dimension of the frame, temporal location, and the coding mode (*intra/inter*) of the frame. If some of this information is not received by the decoder, the whole frame may have to be discarded. In order to increase the probability of receiving this information, Header Duplication (HD) allows the introduction of duplicate copies of important picture header information in the video packets. This technique reduces significantly

the number of discarded video frames in error prone environments. This mode is supported in the visual part of MPEG-4 using the Header Extension Code (HEC), and a similar tool is included in the RTP payload specification for H.263, defined in RFC2429 [77].

3.5 Transport of Video over Networks

In packet lossy environments, the use of appropriate packetization techniques can improve error resilience substantially. Whenever it is possible, a video communication system will attempt to start a packet with a synchronization point of the video bit stream, allowing the independent decoding of the packet. The size of a packet can be chosen subject to network constraints such as those dictated by the maximum transfer unit (MTU) size and packetization payload/overhead tradeoff considerations (e.g., the Internet). As seen earlier, it is possible to perform the entropy decoding of a slice independently of other bit stream segments. Some video coding algorithms, such as H.263 with Annex R enabled, allow for complete independence of slices, so that spatial or temporal error propagation can be contained within a slice. The latter allows for relatively high error resilience at the cost of lower coding efficiency due to lack of spatial prediction across the slice boundaries [74]. However, the error resilience gained from using such a method often do not overweight the coding penalty.

Video bit streams are usually not transmitted as is, and additional standards describe their transport for different networks. These standards provide

additional capabilities for multiplexing of control, audio and data information and allows for receiver synchronization and sequential decoding of the received media streams. They may also provide error recovery mechanisms. In Chapter 7 and 8, we will describe the transport of H.263 bit streams over the Internet and mobile networks, respectively.

3.6 Error Concealment

Before errors can be concealed at the decoder, they must be detected. The decoder can receive information from the multiplexing layer regarding the status of a packet, or it can detect that a packet is missing from the block numbering in the video bit stream. The block address should be incremented by one for every received block. When a GOB/slice header is decoded, the block address included in the header is compared to the expected address, thereby enabling the detection of missing blocks. The decoder can choose to drop the complete packet if errors are indicated or detected. If a packet received with errors is passed to the video decoder, or if errors are not detected by the transport layer, bit errors can still be detected using syntactic or semantic violations of the video bit stream [76]. These include:

1. motion vector outside of allowable range,
2. invalid VLC table entry,
3. DCT coefficient out of range, and
4. number of DCT coefficients in a block exceeding 64.

Once an error is detected, the decoder searches for the next synchronization point. Error concealment is then performed. Error recovery via error resilient entropy coding techniques such as those described in Section 3.3.2 or semi-fixed length codes proposed in Chapter 5 may provide additional improvement in reproduction quality.

There are two basic approaches for error concealment, namely spatial and temporal interpolation. In spatial interpolation, missing pixel values are reconstructed using neighbouring spatial information, whereas in temporal interpolation, the lost data is reconstructed from data in the previous frames. In our work, we employ both spatial and temporal interpolation techniques. Spatial interpolation is used to reconstruct the missing data in intra coded frames. For reconstructing the missing data in inter coded frames, we use a simple motion compensated error concealment technique.

Many error concealment techniques have been proposed in the literature, and an excellent review is available in [78]. However, many of these techniques require substantial additional complexity that can prevent real-time video decoding. The concealment method used in this work achieves reasonable performance and requires very little additional computational complexity. Therefore, real-time video decoding can still be easily maintained.

3.6.1 Spatial error concealment

Spatial error concealment (SEC) is employed when there are no previous frame and temporal error concealment is not possible. Our method for spatial error

concealment (SEC) is based on the directional filtering method proposed in [79], [80]. First, edges in the available blocks surrounding the missing block are determined using a gradient measure. The edge for pixel $x(i,j)$ in the surrounding blocks is computed by [3]

$$\begin{aligned} g_x &= x_{i+1,j-1} - x_{i-1,j-1} + 2x_{i+1,j} - 2x_{i-1,j} + x_{i+1,j+1} - x_{i-1,j+1}, \quad \text{and} \quad (3.2) \\ g_y &= x_{i-1,j+1} - x_{i-1,j-1} + 2x_{i,j+1} - 2x_{i,j-1} + x_{i+1,j+1} - x_{i+1,j-1}. \end{aligned}$$

The magnitude and angular direction of the gradient at pixel (i,j) are

$$\begin{aligned} G &= \sqrt{g_x^2 + g_y^2}, \quad \text{and} \\ \theta &= \arctan\left(\frac{g_y}{g_x}\right). \end{aligned} \tag{3.3}$$

The angular value of the gradient is rounded to one of the eight directions equally spaced between zero and 180°. There is a counter corresponding to each of the eight directions. If the direction of the edge at the pixel (i,j) implies that it passes through the missing block, the counter corresponding to the direction of that edge is incremented by the amount of the gradient. This procedure is performed for all the pixels in the blocks to the left, down, down-right, down-left, up, up-left, and up-right of the missing block (if applicable), and for each of them, eight values corresponding to eight directions are obtained. As we assume errors will result in the concealment of a complete slice, what surrounding blocks that will be available will depend on the location of the previously received slice. The direction corresponding to the counter with maximum value is called the direction of the dominant edge. The pixel

[3] This is essentially the Sobel mask.

values in the missing block are interpolated along this direction using pixels in the available neighbouring blocks and a series of one-dimensional interpolators. The interpolated value is a weighted average of the values of pixels in the neighbouring blocks that are on the line parallel to the direction of the dominant edge. That is, the estimated value of a missing pixel is given by

$$x_i = \frac{\sum_{j \in \eta} \left(\frac{x_j}{d_{i,j}^w} \right)}{\sum_{j \in \eta} \left(\frac{1}{d_{i,j}^w} \right)}; \quad i \in \text{lost block} \tag{3.4}$$

where η represents the 9 neighbouring blocks, x_i is a pixel in the missing block, x_j is a pixel in one of the neighbouring blocks on a line with x_i which is parallel to the direction of the dominant edge, $d_{i,j}$ is the distance between the pixels x_i and x_j on the line, and w is a constant.

3.6.2 Motion compensated temporal error concealment

The Motion Compensated Temporal Error Concealment (MC-TEC) method used in this work is based on the TCON model described in the H.263 Test Model TMN-10 [81], and it is summarized next. The motion vector of the missing block is set to the median value of the motion vectors of the blocks to the left, above and above right of the missing block. If no surrounding motion vectors are available, the motion vector is set to *zero*. Then, the block from the previous frame at the spatial location specified by this motion vector is copied to the location of the missing block in the current frame.

Chapter 4

Optimal Spatial Synchronization

4.1 Introduction

A feature supported by the new generation of video coding standards is spatial error localization through the use of synchronization markers. Such markers limit the propagation of errors within frames, as discussed in Chapter 3. Bit errors can cause two types of synchronization errors. The decoder may lose bit synchronization while decoding the bit stream and/or lose spatial synchronization while reconstructing the video frame. Synchronization markers allow for bit synchronization of the decoder with the bit stream, they reset the spatial location in the decoded frame, and they prevent error propagation due to spatial prediction. However, if used too frequently, synchronization markers will introduce an unnecessary increase in bit rate.

While the placement of synchronization markers according to a uniform pattern often performs quite well, a significant improvement in performance

can be obtained using more sophisticated algorithms. Moreover, it may be difficult to determine the optimal frequency of a uniform placement of synchronization markers. In this chapter, we propose an effective method, for increasing spatial error resilience of video transmission over bit error prone networks, that is based on a rate-distortion optimized synchronization marker placement algorithm. The channel condition and the error concealment method used by the decoder are used during the encoding process to optimize the placement of synchronization markers in the compressed bit stream. More specifically, we select the number and location of synchronization markers, such that, given the channel condition and the decoder error concealment, the distortion for a given rate is minimized.

4.2 Probability of Slice Errors

In this section, we determine the probability of a slice being in error based on the channel characteristics. Let's first assume that the video is packetized using slices, and that a synchronization marker can be placed at the start of the slice. Let's also assume that a bit error occurring within a slice will result in the whole slice being discarded. Using the size of the slice, $L(s)$, and the channel average bit error rate (BER), P_e, the slice error rate (SER) is first evaluated. The probability of error of a video block, p, is then set to the SER value.

The insertion of a synchronization marker will reduce the length of the previous slice, thus reducing the SER and the probability of error of a

video block. The SER is then computed for two scenarios: 1) a slice header is inserted at the beginning of the current block, and 2) no slice header is inserted. In the first case, $L(s)$ is the number of bits of the slice header plus the number of bits of the current block, and in the second case, $L(s)$ is the number of bits since the location of the last slice header plus the number of bits of the current block. Given $L(s)$ (case 1 or 2), one can obtain for a binary symmetric channel (BSC)[1]

$$SER = p = 1 - (1 - P_e)^{L(s)}. \tag{4.1}$$

Assuming that bits are uniformly distributed among blocks, the expected slice length (in units of blocks), $E[L_{blocks}(s)]$, can be obtained as a function of the expected probability of error of a block, $E[p]$, that is,

$$E[L_{blocks}(s)] = \frac{log(1 - E[p])}{log(1 - P_e)} \times \frac{1}{L_{bits}(block)}, \tag{4.2}$$

where $L_{bits}(block)$ represents the average length in bits of a block, which is set by the bit rate constraint.

However, bits are usually not uniformly distributed among blocks, since different coding modes will yield substantially different numbers of bits. Fig. 4.1 presents the slice error rate versus the slice length, for different BERs. With a SER of above 20%, it is very difficult to obtain reasonable video reproduction quality. To estimate how many blocks can be placed in a slice, typical examples of the distribution of the size of *inter* and *intra* blocks are illustrated in Fig. 4.2 (a) and (b), respectively. The cumulative distribution

[1]The BSC case represents the worse slice error rate, for a given average BER.

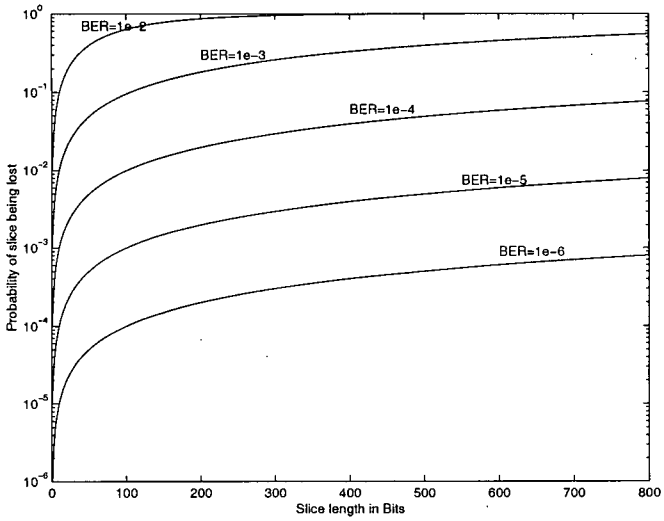
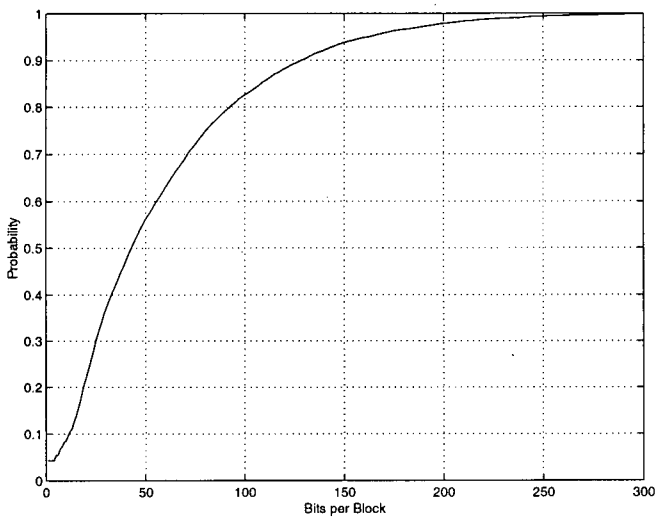
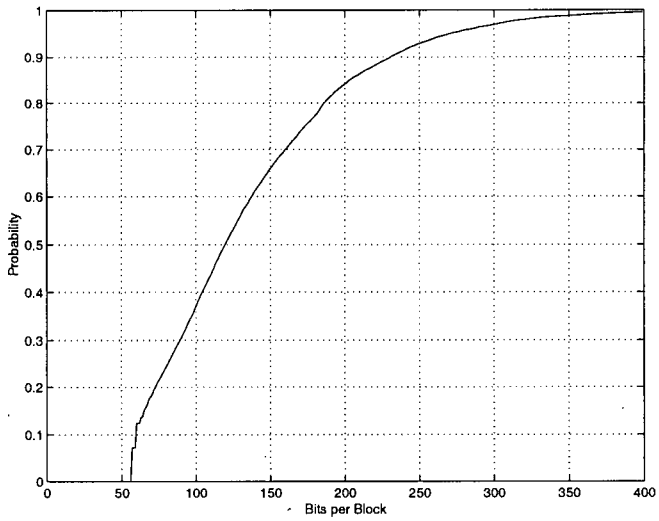


Figure 4.1: The slice error probability versus slice length for different BERs.

functions (CDF) are obtained by coding the video sequence **FOREMAN** at 48 kbps. At BERs higher than 10^{-3}, the slice size must be less than 223 bits in order to maintain a SER that is less than 20%. Under such a condition, a slice may contain a maximum of approximately 5 *inter* blocks, or approximately 2 *intra* blocks. At higher BERs, a slice containing any number of *intra* blocks will most likely result in a SER that is higher than 20%. At such high error rates, we will see later that the proposed algorithm in fact chooses many *skip* blocks, because such blocks contain only *one* bit and will less likely be corrupted by errors. However, if the same bit stream is transmitted in an error free environment, the loss in video reproduction quality will be significant.



(a)



(b)

Figure 4.2: Cumulative Distribution of the size of (a) inter and (b) intra blocks.

4.3 Computation of Distortion

In an error prone environment, the distortion of the reproduced video at the decoder can be attributed to quantization during the encoding process and to transmission errors. Video decoders usually employ error concealment, and thus channel errors will lead to error concealment distortion. For a given video block, we denote D_q as the distortion caused by quantization, and D_c as the distortion caused by error concealment. If a video block is lost due to channel errors with probability p, it will be concealed at the decoder. This probability will depend on the coding mode selected and the number of video blocks in the slice, as discussed in the previous section. The placement of synchronization markers will improve the spatial error resilience, but it will not directly affect the temporal error propagation of errors. Therefore, we do not consider the error propagation process in the computation of the overall distortion. For a given video block, we estimate the decoder distortion, resulting from coding and error concealment, by

$$D_{sync}(mode, p) = (1-p)D_q(mode) + pD_c. \tag{4.3}$$

The encoder can simulate the known decoder error concealment in order to evaluate D_c. This distortion is weighted by the probability that the block will be lost and concealed, p. If the block is received at the decoder with probability $(1-p)$, the resulting distortion caused by quantization will be D_q. D_{sync} is employed in the Lagrangian minimization to determine the location of synchronization markers, as described in the next section.

4.4 Placement of Synchronization Markers

We next describe our rate-distortion optimized algorithm for the placement of synchronization markers. In addition to the coding rate, which depends on the selected coding mode, the size of the synchronization marker as well as the transport overhead rate are incorporated in the Lagrangian minimization. For a small increase in bit rate resulting from inserting a synchronization marker, a large improvement in video reproduction quality at the decoder can be achieved. The Lagrangian cost function for the slice header localization, J_{sync}, is given by

$$J_{sync} = D_{sync}(mode, p) + \lambda_{sync} R(mode, sync), \tag{4.4}$$

where $sync \in \{0, 1\}$, with 0 meaning no synchronization marker is inserted and 1 meaning a synchronization marker is inserted.

The mode decision process is not RD optimized, but chosen according to the thresholding based method described in the H.263 Test Model [81]. This method employs the sum of absolute difference (SAD) values obtained by coding a block using different modes, and chooses the mode that yields the lowest SAD value. Some coding modes may be favored by subtracting a fixed SAD threshold before mode selection.

Independently for each coded block and the given video coding mode, J_{sync} is computed for the two scenarios: no synchronization marker is inserted ($sync = 0$) and a synchronization marker is inserted ($sync = 1$). These scenarios will result in different SER and bit rates. The SER is employed as the probability that the block will be lost and concealed, p, which is used in

the computation of the distortion, D_{sync}. The value of $sync \in \{0, 1\}$ resulting in the lowest J_{sync} for a given λ_{sync} will be selected. Figure 4.3 presents a flowchart summarizing the proposed algorithm.

The insertion of a synchronization marker increases the rate for a complete slice and must be distributed among blocks within a slice in the minimization of J_{sync}, which is performed at the block level. When the insertion of a synchronization marker is considered, the synchronization rate is divided by the number of blocks since the last synchronization marker, including the current block.

In order to obtain an optimal value of the Lagrangian parameter λ_{sync} for different BERs, the average slice length that yields the minimum Lagrangian J_{sync} for a given value of λ_{sync}, is obtained. Values of slice length (in units of coded blocks) versus λ_{sync} are presented for the sequence **FOREMAN** in Fig. 4.4 and for the sequence **NEWS** in Fig. 4.5, both coded at 48 kbps, QCIF resolution and 10 frames per second. It it interesting to note that, although the two sequences have substantially different video contents (the sequence **FOREMAN** contains a lot of motion and camera movement while the sequence **NEWS** is a typical low motion head-and-shoulder videophone sequence), they lead to a very similar relation between λ_{sync} and the average slice length. At higher BERs, λ_{sync} has little impact on the average slice length. However, at lower BERs, the chosen value of λ_{sync} will affect significantly the size of the slice. Thus, a relationship between λ_{sync} and the BER can be evaluated. Assuming that bits are uniformly distributed among coded blocks in a slice, and

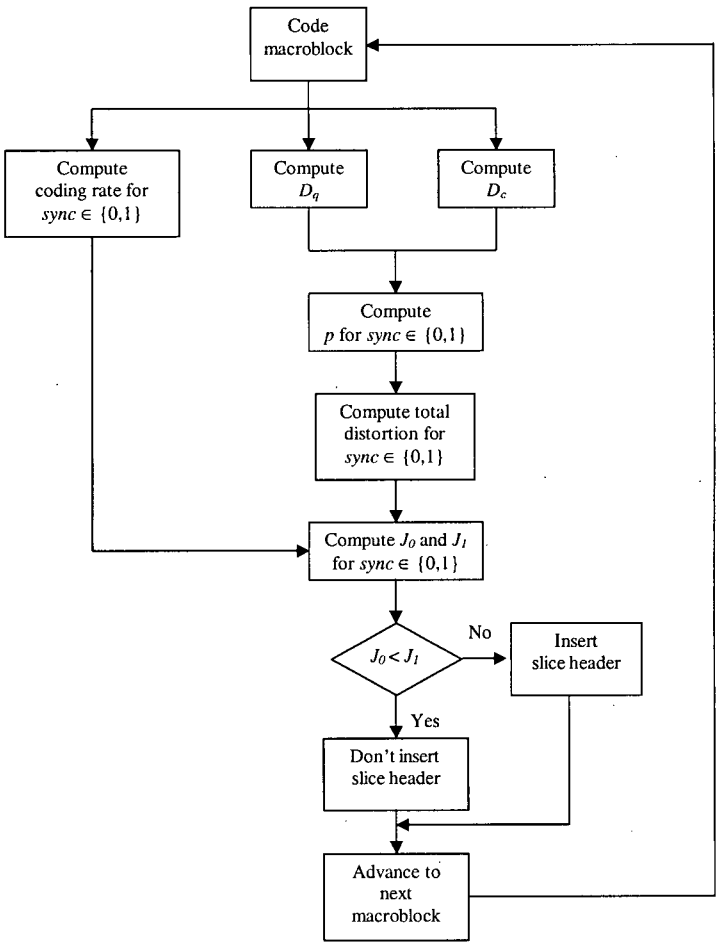


Figure 4.3: Flowchart of the proposed algorithm for the placement of synchronization markers.

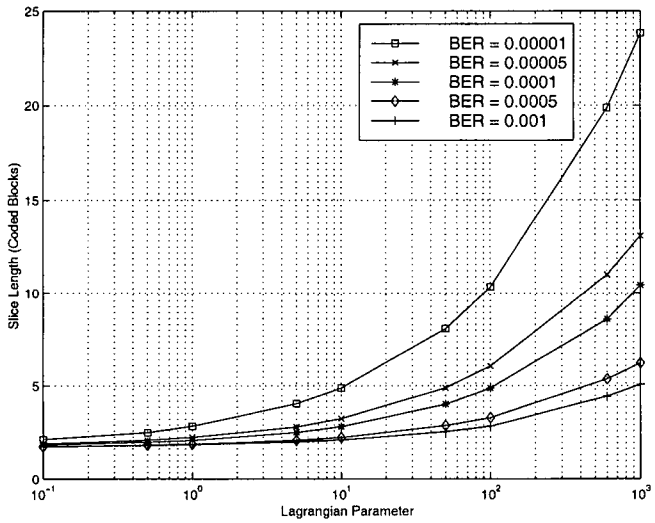


Figure 4.4: Average slice length (in units of coded blocks) versus the Lagrangian parameter λ_{sync} for different BERs for the sequence **FOREMAN**.

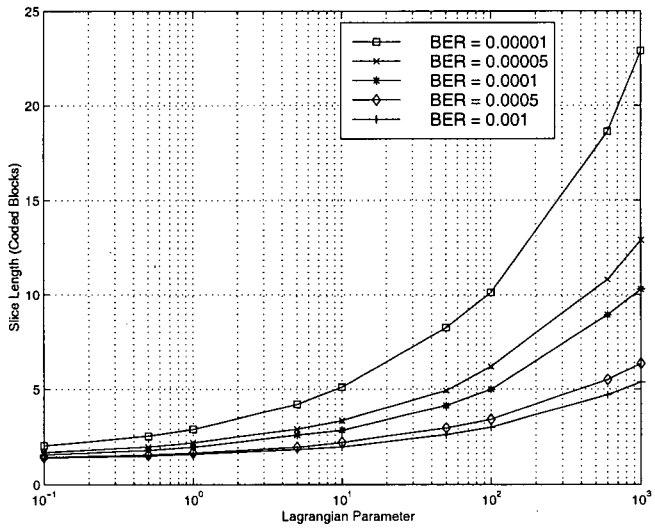


Figure 4.5: Average slice length (in units of coded blocks) versus the Lagrangian parameter λ_{sync} for different BERs for the sequence **NEWS**.

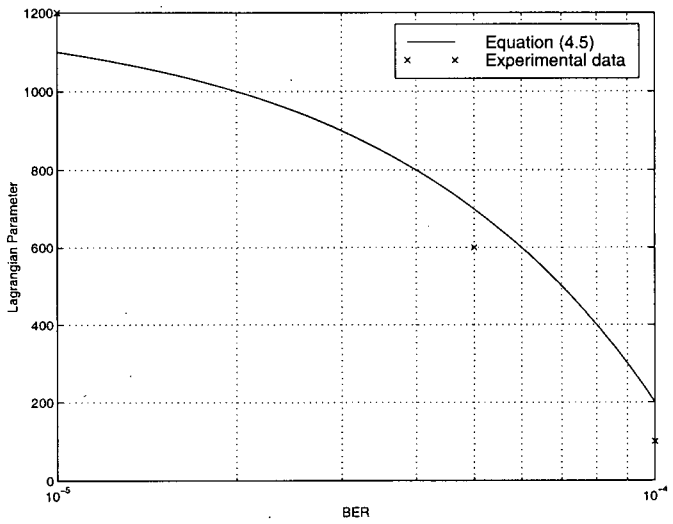


Figure 4.6: Comparison of Eq. (4.5) to experimental data.

that we wish to obtain an average block loss rate $E[p]$ of less than 3%, λ_{sync} can be approximated, via experiments using different video sequences and bit rates, by the following simple relationship as a function of P_e,

$$\lambda_{sync} = \beta \times max(-10^7 P_e + 1200, 1), \tag{4.5}$$

where $max(x, y)$ represents the larger value of x and y, and β represents the portion of the blocks within a frame that are coded. We compare the estimate obtained from Eq. (4.5) to experimental data in Figure 4.6. The experimental data is obtained as follows: given the target block loss rate, $E[p]$, the channel BER, P_e, and the bit rate allocation, the average slice length is obtained from Eq. (4.2). Given the average slice length, the Lagrangian parameter, λ_{sync}, is obtained from experimental data presented in Figures 4.4 and 4.5, where the average value from the two figures is employed. As can be observed from

Figure 4.6, Eq. (4.5) approximates well the Lagrangian parameter for a target block loss rate of 3%.

4.4.1 Complexity of the proposed method

The proposed algorithm for the placement of synchronization markers requires additional encoder complexity. First, the decoder error concealment must be performed at the encoder for each block and the incurred concealment distortion must be computed. Then, the rate and overall distortion must be computed twice based on the SER: once with no synchronization marker inserted, and once with a synchronization marker inserted. For example, in our implementation, encoding time is increased by a factor of roughly 2:1 on a Sun Sparc workstation (when a fast motion estimation algorithm [28] is employed). Of course, no additional decoder complexity in incurred.

4.5 Experimental Results: Analysis

In order to illustrate the advantages of the RD optimized synchronization marker insertion algorithm, we compare its performance to that of inserting a synchronization marker at the start of every GOB, which we refer to as the anchor model. For the transmission of the encoded video bit streams, we employ the H.223 simulation model described in Section 8.2. We discard all video packets for which the checksum indicates the presence of errors. For each error condition tested, 50 simulations are performed to obtain statistically significant results, and the luminance PSNR (Y-PSNR) is averaged over all

coded frames and simulation runs. Video source material is coded at QCIF spatial resolution (176×144 pixels) and temporal resolution of 10 frames per second. We employ a channel bit rate of 64 kbps and assume that the video data requires 75% of the bandwidth [82]. Therefore, video sequences are coded at 48 kbps using the TMN-8 rate control method described in Section 2.5.3. The encoder employs the exact channel BER for the value of P_e used in the computation of the slice error rate. The same error concealment used by the decoder is assumed at the encoder. Both the anchor bit streams and the bit streams generated by our proposed method are decoded using the same video decoder, which employs the error concealment method described in Section 3.6.

Results are presented in Fig. 4.7 for the sequence **FOREMAN** and in Fig. 4.8 for the sequence **NEWS**, both coded at 48 kbps and sent over the BSC. For a given BER, the BSC model is actually the worst channel model, yielding the highest probability of error in a slice, as compared to a channel with correlated errors (burst errors), since the length of typical bursts in a mobile channel is much smaller than that of a slice.

As much as 3 dB improvement in reproduction quality is achieved at a BER of 10^{-3}. At the lower BER, the reproduction quality obtained using the RD optimized synchronization placement algorithm is slightly lower than that obtained using the uniform GOB synchronization markers. By placing a lower limit on the maximum average block loss rate (much less than 5%), the proposed algorithm will perform better than the anchor model at such low

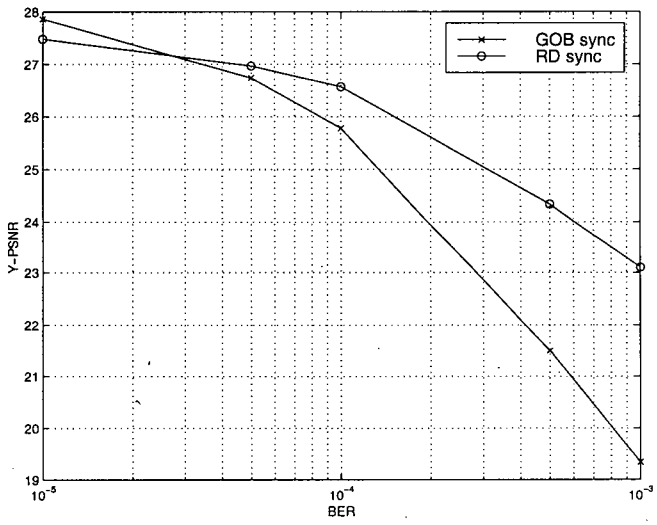


Figure 4.7: Performance of the RD synchronization marker insertion algorithm versus GOB synchronization for the sequence **FOREMAN**.

error rates.

We next evaluate the effects of varying λ_{sync} on the decoded PSNR for a wide range of BER. Results of Y-PSNR versus λ_{sync} are presented in Fig. 4.9 for the sequence **FOREMAN** and in Fig. 4.10 for the sequence **NEWS**, both coded at 48 kbps and sent over the BSC at different error rates. The optimal value of λ_{sync}, i.e., the value yielding the highest PSNR, depends on the error rate of the network, as well as the video sequence coded. A higher value is usually more suitable for lower BERs. The resulting PSNR is very similar for values of $\lambda_s ync$ below 100, and drops rapidly for $\lambda_s ync > 100$. Since a lower value for $\lambda_s ync$ would result in more synchronization markers per coded frame, unnecessary overhead would be incurred if the same bit stream is transmitted in an error free environment, with little improvement

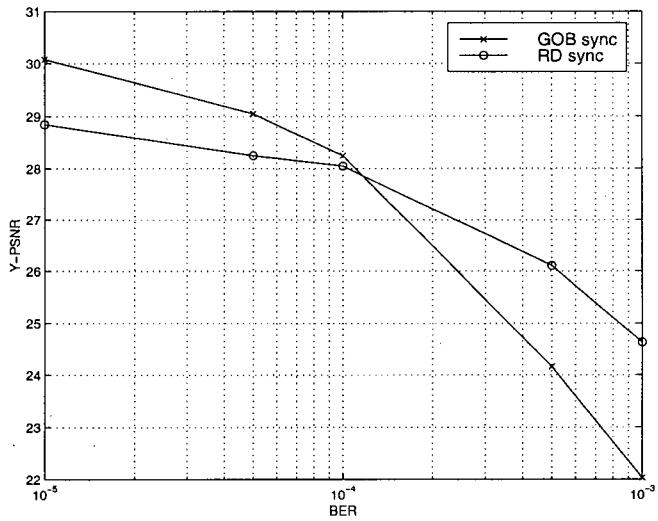


Figure 4.8: Performance of the RD synchronization marker insertion algorithm versus GOB synchronization for the sequence NEWS.

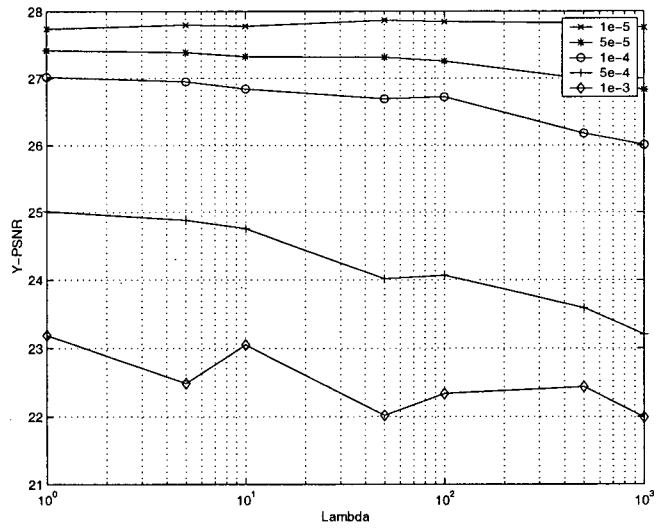


Figure 4.9: Performance of the RD synchronization marker insertion algorithm versus λ_{sync} for the sequence FOREMAN.

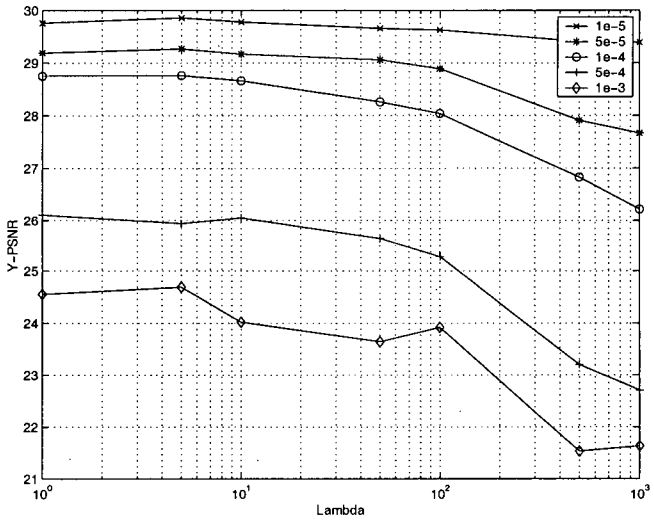


Figure 4.10: Performance of the RD synchronization marker insertion algorithm versus λ_{sync} for the sequence **NEWS**.

in an error prone environment. Therefore, a fixed value of $\lambda_{sync} = 100$ will give a good compromise between error resilience of the coded bit stream for a wide range of bit error rates and sequences. Results for the RD-optimized placement of synchronization markers with $\lambda_{sync} = 100$ compared to fixed GOB synchronization are presented in Fig. 4.11 for the sequence **FOREMAN** and in Fig. 4.12 for the sequence **NEWS** using $\lambda_{sync} = 100$ and the same coding parameters as above. With this value of λ_{sync}, the proposed algorithm always outperforms the GOB synchronization structure.

The proposed algorithm optimizes the frequency of synchronization markers, as seen above, and also optimizes the location of the markers, as shown next. We compare the results of the proposed placement of synchronization markers to those of using spatial-uniform and bit-uniform placement

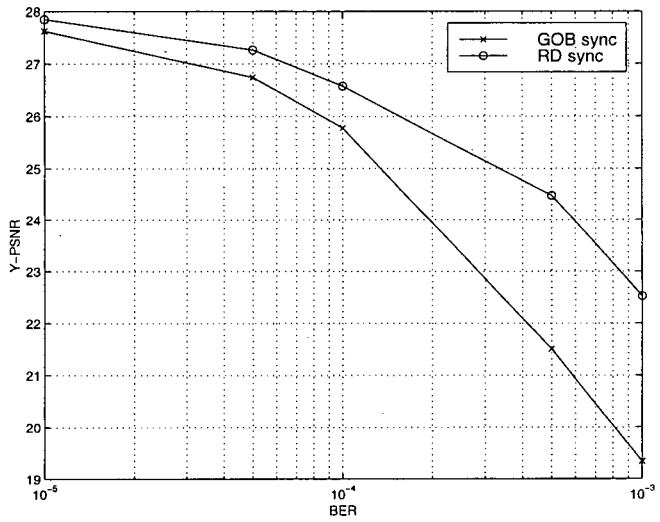


Figure 4.11: Performance of the RD synchronization marker insertion algorithm versus GOB synchronization for the sequence **FOREMAN** with $\lambda_{sync} = 100$.

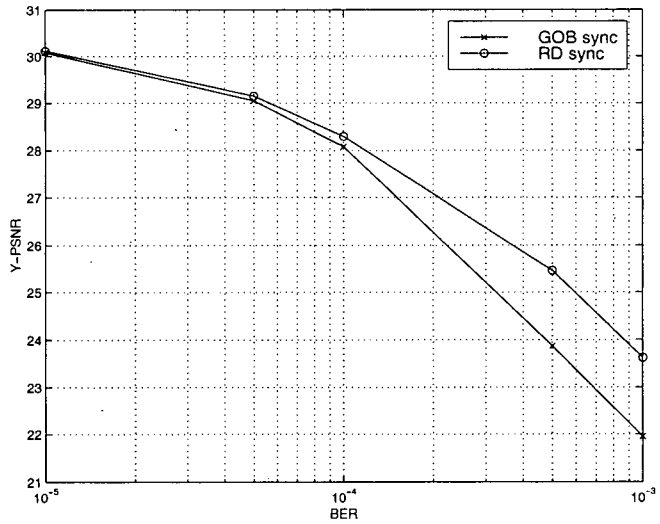


Figure 4.12: Performance of the RD synchronization marker insertion algorithm versus GOB synchronization for the sequence **NEWS** with $\lambda_{sync} = 100$.

BER	10^{-5}	5×10^{-5}	10^{-4}	5×10^{-4}	10^{-3}
FOREMAN	10	16	20	28	32
NEWS	6	10	12	18	21

Table 4.1: Number of synchronization markers per frame using the proposed algorithm for the different BER and sequences considered.

BER	10^{-5}	5×10^{-5}	10^{-4}	5×10^{-4}	10^{-3}
FOREMAN	480	300	240	171	150
NEWS	800	480	400	266	228

Table 4.2: Number of bits per slice for the bit-uniform placement of synchronization markers for the different BER and sequences considered.

of markers. Table 4.1 presents the number of synchronization markers per frame resulting from using the proposed algorithm for the different BER and sequences considered. The same number of slices is employed with a uniform spatial and uniform bit distribution in order to evaluate the performance of our synchronization marker localization algorithm. The number of bits per slice for the bit-uniform placement of synchronization markers is shown in Table 4.2, assuming a uniform distribution of 4800 bits per frame (48 kbps and 10 fps).

Experimental results are presented in Fig. 4.13 for the sequence **FOREMAN** and in Fig. 4.14 for the sequence **NEWS**. The RD optimized algorithm employs a fixed value of λ_{sync}, $\lambda_{sync} = 100$. A bit-uniform placement of synchronization markers provide approximately the same or worse performance as compared to a spatial-uniform placement. This contradicts recommenda-

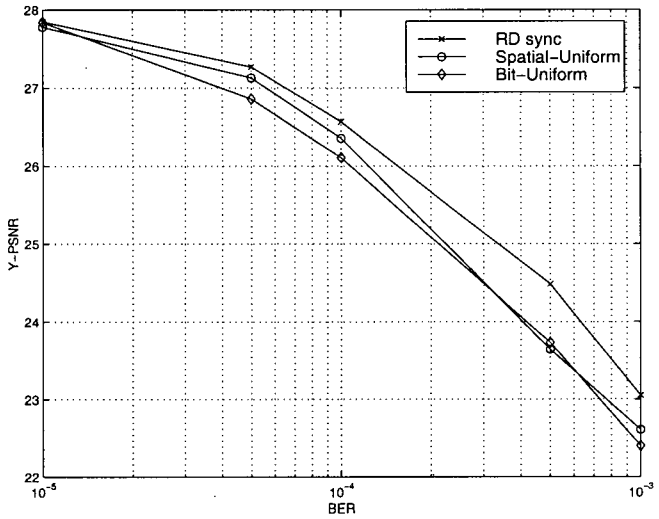


Figure 4.13: Performance of the RD synchronization marker insertion algorithm versus spatial-uniform and bit-uniform placement of synchronization markers for the sequence **FOREMAN**.

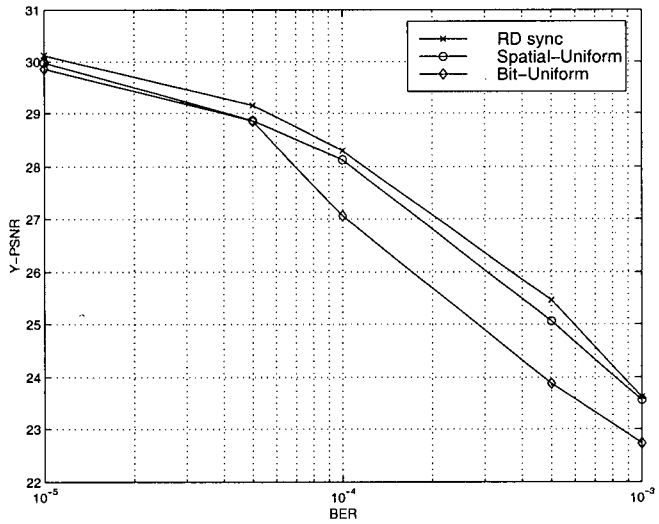


Figure 4.14: Performance of the RD synchronization marker insertion algorithm versus spatial-uniform and bit-uniform placement of synchronization markers for the sequence **NEWS**.

tions presented in [76], where it was suggested that the bit-uniform placement should provide better error resilience that a spatial-uniform placement. This can be explained by the fact that a bit-uniform placement of synchronization may leave large portion of the coded image without synchronization, when this portion does not require many bits to be coded. However, such large picture areas may be difficult to conceal, leading to poor performance when transmitted over a bit error prone network. For the same number of synchronization markers, the proposed algorithm always outperforms a uniform placement of synchronization markers. Moreover, the proposed method maintains a higher level of reproduction quality throughout most of the video sequence, as illustrated in Figs. 4.15 and 4.16, where we compare the performance of the proposed method for the first 300 frames of the sequence **FOREMAN** at a BER of 10^{-4} to that of a bit-uniform and spatial-uniform placement of synchronization markers, respectively. The same bit error pattern is employed in all cases. The bit-uniform placement of synchronization markers exhibits significant degradation in reproduction quality for some frames (around frame number 110 in Fig. 4.15), leading to annoying artifacts in the reproduced video frames. The decoded frames (number 169) for the different methods are shown in Fig. 4.17. Even though the spatial-uniform placement of synchronization markers results in the highest PSNR for this decoded frame, the subjective quality of the proposed method is obviously superior. In the RD optimized algorithm, some video blocks contain small concealment distortion, whereas in the spatial-uniform case, fewer blocks contain much worse error

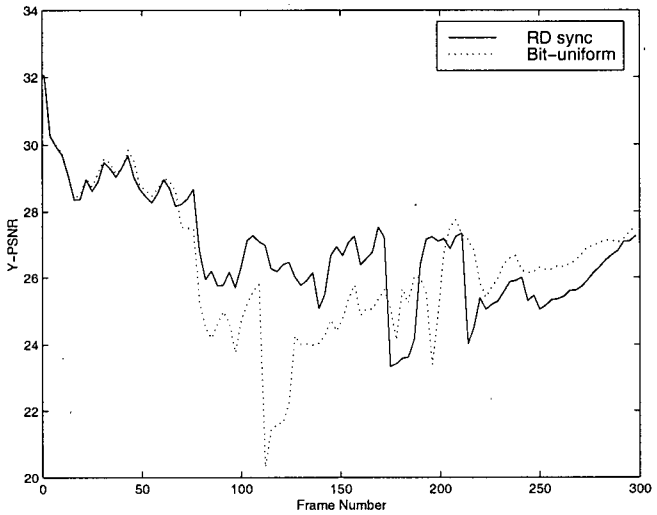


Figure 4.15: Performance of the RD synchronization marker insertion algorithm versus bit-uniform placement of synchronization markers for 300 frames of the sequence **FOREMAN**.

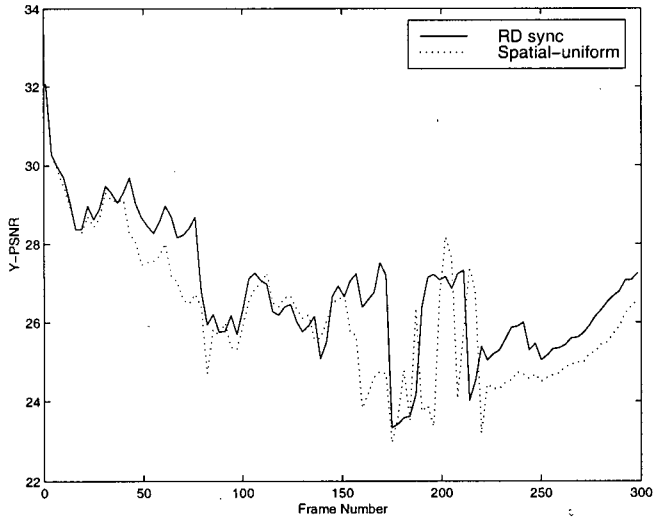


Figure 4.16: Performance of the RD synchronization marker insertion algorithm versus spatial-uniform placement of synchronization markers for 300 frames of the sequence **FOREMAN**.

concealment artifacts. These blocks are much more noticeable when observing the decoded video sequence in real-time.

4.6 Summary

In this chapter, we presented a rate-distortion optimized algorithm for the insertion of synchronization markers for a bit error prone network, that improves significantly video reproduction quality. The algorithm employs the network condition and knowledge of the decoder error concealment method, to determine the error probability of a slice and to estimate the expected distortion, respectively . We have shown that the proposed algorithm optimizes both the frequency and the location of synchronization markers in order to minimize the distortion caused by both coding and error concealment.

Chapter 5

Semi-Fixed-Length Motion Vector Coding

5.1 Introduction

When using VLCs as means of entropy coding, it is impossible to determine the exact location of a bit error. In the previous chapter, we have assumed that video segments containing errors are discarded. However, it may be possible to recover information within a corrupted video segment using error resilience entropy coding methods. Some of these methods were discussed in Chapter 3. In this chapter, we propose a semi-fixed-length entropy coding method. Residual information from a motion compensated video coding algorithm is useless when motion vectors are not received correctly. Therefore, we only present a motion vector semi-fixed-length coding method. Unless motion vector information is guaranteed to be received without errors, texture information for a

known to be incorrect video block is discarded.

In H.263, motion vectors are encoded by applying variable length codes (VLCs) to difference motion vectors, obtained by subtracting two-dimensional median predicted vectors from the actual motion vectors. The horizontal and vertical motion vector components are coded independently, using the same VLC table. In the presence of channel errors, such a variable length coding technique can easily cause a loss of synchronization, leading to the loss of all macroblocks until the next synchronization point is reached.

We propose an efficient motion vector coding method that addresses the above problem [83]. The proposed method targets low bit rate applications based on the H.263 video coding standard. The motion vectors are encoded using semi-fixed-length codes, yielding essentially the same levels of rate-distortion performance and subjective quality achieved by H.263's Huffman-based VLCs in a noiseless environment. The advantage of our method, however, is that the corresponding video coder generates bit streams which are more error resilient than those of typical H.263 compliant coders in a bit error prone environment.

Figure 5.1 illustrates the basic idea of the semi-fixed-length coding method. When semi-fixed-length codes are used, unequal error protection (UEP) can be employed, with more protection applied to the critical part of the code (VLC part), thus reducing overhead as compared to equal error protection (EEP) applied to VLCs. If the VLC part of the code is received correctly at the decoder, bit stream synchronization can be maintained. The

components of the median prediction difference vector. Using this approach, an error in a motion vector will likely propagate until the next synchronization marker is detected in the bit stream. However, by using semi-fixed-length motion vector encoding techniques [84] and protecting the important part of the motion vector bits, the probability of error propagation can be minimized, at the expense of only a small increase in total bit rate.

Experimental results indicate that a large percentage of the motion vectors selected using a full search motion estimation method belong to either Region 0 or 1 in Fig. 5.2. By minimizing the lengths of the corresponding codes, most of the coding gain achieved by H.263's VLCs can still be obtained. In [84], a simple semi-fixed-length coding method was developed that improves channel error resilience significantly. However, such a method works only for integer-pel accuracy motion vectors. Moreover, motion vectors which do not belong to the probable region are assigned codes representing the nearest candidates which belong to the probable region. Our proposed method is designed for half-pel accuracy motion vectors, as they yield a significant performance improvement [85, 47], especially at the higher bit rates (e.g., as much as 2 dB at 64 kbps). Furthermore, motion vectors belonging to the outer layers are also assigned codes, as their exclusion can often result in a considerable performance degradation.

Based on the statistical behavior of the motion vector differences, the search area centered at the predicted motion vector is divided into *five* regions. Fig. 5.2 depicts the difference motion vector regions and Table 5.1

Region	Integer-pel code, Header is in ()	Half-pel code	Probability
0	(0)	N/A	0.4387
1	(1 00)	XXX	0.3683
2	(1 01) XX	XXX	0.1295
3	(1 10) XX XXXX	XXX	0.0242
4	(1 11) XXXXX XXXXX	XX	0.0473

Table 5.1: Semi-fixed-length codes for the difference motion vectors.

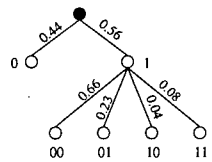


Figure 5.3: Tree representation of the headers of the semi-fixed-length codes (with their respective probabilities).

	-1.5	-1	-0.5	0	0.5	1	1.5
-2.5	0.0001	0.0001	0.0006	0.0036	0.0007	0.0002	0.0001
-2	0.0002	0.0003	0.0006	0.0035	0.0011	0.0005	0.0003
-1.5	0.0003	0.0003	0.0021	0.0138	0.0027	0.0010	0.0004
-1	0.0008	0.0016	0.0044	0.0286	0.0054	0.0019	0.0005
-0.5	0.0012	0.0021	0.0168	0.0760	0.0229	0.0026	0.0010
0	0.0028	0.0061	0.0342	0.4213	0.0376	0.0071	0.0033
0.5	0.0012	0.0026	0.0222	0.0639	0.0194	0.0024	0.0015
1	0.0006	0.0020	0.0051	0.0292	0.0050	0.0019	0.0007
1.5	0.0007	0.0011	0.0032	0.0176	0.0030	0.0004	0.0003
2	0.0002	0.0006	0.0012	0.0080	0.0009	0.0003	0.0001
2.5	0.0003	0.0005	0.0010	0.0058	0.0009	0.0003	0.0001

Table 5.2: Statistics of the motion vector field around the predicted motion vector (horizontal x component and vertical y component).

with a fixed number of bits. This also allows the decoder to maintain synchronization if the VLC part of the code is received without errors. The popular *gray* encoding method was employed to generate the specific codes for the fixed-length part of the code. That is, the codes within a region are assigned such that an error in a bit would result in the assignment of a neighbouring motion vector. This minimizes the error when a codeword is not correctly received.

Region 0 corresponds to the $(0,0)$ difference motion vector and is represented by *one* bit, as this vector is the most probable one. Region 1 represents the half-pel vectors around the $(0,0)$ integer-pel difference vector. Region 2 represents the $(1,0)$, $(0,1)$, $(-1,0)$, $(0,-1)$ integer-pel difference motion vectors, and their corresponding half-pel vectors. The remaining difference vectors are represented by Regions 3 and 4, both extending to the $(\pm 16.5, \pm 16.5)$ rectangular boundaries. In Regions 1, 2 and 3, *three* additional bits are necessary to represent the half-pel vectors. However, only *two* bits are necessary to represent the half-pel vectors in Region 4, since the sign of the half-pel part is assumed to be the same as the sign of the full-pel part.

We next illustrate the above encoding procedure via two examples. First, we encode a difference motion vector of $(3,-2.5)$, which belongs to Region 4. The header VLC would then be coded as 111. The fixed-length part would be coded using the assigned 10-bit codeword for the full-pel part, $(3,-2)$, followed by the 2-bit codeword for the half-pel part, $(0,0.5)$. The sign of the half-pel part does not need to be coded, as it can be determined directly

from the sign of the respective full-pel component. As a second example, we encode a difference motion vector of $(6, -0.5)$, which belongs to Region 3. The header VLC would then be coded as 110. The fixed-length part would be coded using the assigned 6-bit codeword for the full-pel part, $(6, 0)$, followed by the 3-bit codeword for the half-pel part, $(0, -0.5)$. In this case, only the sign of the half-pel vertical component of the motion vector can be determined from the full-pel motion vector. Therefore, 3 bits would be necessary for the half-pel code in order to determine the sign of the horizontal component.

In terms of encoding efficiency, the performance of the proposed method is very close to that of the use of VLCs. Using a training set of 11 QCIF video sequences, our method yields an average difference motion vector code length of 4.64, as compared to an average of 4.21 for the VLC method. The advantage of our coding method, however, is that we are now able to protect a small portion of the motion vector code (header shown in Fig. 5.3), increasing significantly error resilience.

The codeword assignment described above provides for the smallest average codeword length, based on the collected statistics, half-pel representation and choice of Regions. This assignment also provides for good compression performance in an error free environment. Other semi-fixed-length codes with similar performance could be employed, and better codeword assignment could be obtained for a specific bit error rate and/or type of video sequences. As an example, an alternative codeword assignment is presented in Table 5.3, where Regions 2 and 3 of Figure 5.2 are combined to form Region 2, and Region 4

Region	Integer-pel code, Header is in ()	Half-pel code	Probability
0	(00)	N/A	0.4387
1	(01)	XXX	0.3683
2	(10) XX XXXX	XXX	0.1537
3	(11) XXXXX XXXXX	XX	0.0473

Table 5.3: Alternative semi-fixed-length codes for the difference motion vectors.

becomes Region 3. This particular alternative codeword assignment yields an average difference motion vector code length of 5.07, as compared to an average of 4.64 for the code presented above. However, the new codes will provide better performance if the probability of the $(0,0)$ difference motion vector decreases significantly, as would be the case for more active video sequences.

In order to achieve unequal error protection, data partitioning must be applied to the motion vector data. This is also performed in MPEG-4 video coding [76] when error resilience is desired. In our implementation, motion vector information is blocked following each GOB synchronization marker. The motion vector codes are separated into two fields by a synchronization marker. The header information is blocked first, followed by the remaining data, and ending with another synchronization marker. Of course, this method may impose some additional complexity and processing delay, since the motion vector bits are here treated as data blocks [1].

To illustrate the higher channel error resilience of the proposed method,

[1]The motion vector bits of a macroblock are usually mixed with other bits in the same bit stream.

a rate compatible punctured convolutional (RCPC) channel coder was used [51, 86] to protect the motion vector header. This type of convolutional coder provides an efficient means of realizing variable protection rates using a single encoder/decoder pair. We design the error protection system so that it offers different overall code rates. The system employs channel-state information to decide what rate should be used. The selected rate can be specified in the bit stream syntax. The RCPC coder used here is described in [86] and is based on a rate 1/4 mother convolutional code.

5.3 Experimental Results

We compare the performance of our semi-fixed-length coding method to the VLC method for the coding of motion vectors in noiseless and noisy environments.

5.3.1 Performance of the semi-fixed-length codes in a noiseless environment

We first compare the semi-fixed-length encoding method to H.263's VLC one in a noiseless channel environment (with no header protection), while employing full-search motion estimation in both cases. Frames 0 to 149 of the following 11 QCIF sequences were encoded at 10 fps: CARPHONE, MISS AMERICA, CONTAINER SHIP, COASTGUARD, CLAIRE, FOREMAN, GRANDMA, MOTHER-DAUGHTER, SALESMAN, SUZIE and TREVOR. Exact bit rates are obtained

Sequence	8 kbps	10 kbps	12 kbps	14 kbps	16 kbps	24 kbps	Average
CARPHONE		-0.1359	-0.1344	-0.0590	-0.1000	-0.0349	-0.0928
MISS AMERICA	-0.0345	-0.0158	-0.0279	-0.0327	-0.0216	+0.0112	-0.0202
CONTAINER SHIP	+0.0717	+0.0892	+0.0659	+0.0195	+0.0493	+0.0164	+0.0502
COASTGUARD					-0.0330	-0.0420	-0.0375
CLAIRE	-0.0401	-0.0129	0.0084	+0.0703	-0.0867	+0.0261	-0.0058
FOREMAN			+0.0075	+0.0161	+0.0041	-0.0240	+0.0009
GRANDMA	-0.0079	+0.0460	+0.1218	+0.0118	-0.0157	+0.0377	+0.0323
MOTHER-DAUGHTER	-0.0165	-0.0726	-0.0137	-0.0250	-0.0309	-0.0066	-0.0275
SALESMAN	-0.0293	+0.0014	+0.0012	-0.0019	-0.0724	+0.0424	-0.0097
SUSIE	-0.0736	-0.1641	-0.0535	-0.1551	-0.1051	-0.0456	-0.0995
TREVOR		-0.0114	+0.0148	-0.0606	-0.0915	-0.0007	-0.0297
Average	-0.0186	-0.0306	-0.0009	-0.0216	-0.0457	-0.0018	-0.0210

Table 5.4: Improvement (+) or degradation (-) in PSNR (dB) for the proposed semi-fixed-length coding method over H.263's VLC method.

using TMN-8 [48] rate control method described in Section 2.5.3.

The results are shown in Table 5.4. The VLC method is only 0.02 dB, on the average, better than ours. Notice that our resulting encoder even outperforms slightly the VLC-based one in PSNR when coding slow-motion video sequences. This is due to the fact that the proposed method uses less bits for small difference motion vectors, making available more bits for quantization and coding of the residual blocks. As the semi-fixed-length coding method was specifically designed for low bit rate applications at QCIF resolution, the performance may degrade at higher resolution and/or for more active video sequences. However, since the semi-fixed-length coding method allows the use of long motion vectors, the degradation should not be significant.

5.3.2 Performance of the semi-fixed-length codes in a noisy environment

A unique advantage of the new encoder is that we now have the flexibility to protect the motion vector header using a few additional bits, yielding significantly increased channel error resilience. Using a RCPC coder, protection levels can be adjusted according to different noise sensitivity levels of the proposed semi-fixed-length codes, thus achieving UEP. UEP usually outperforms EEP, as the former applies more protection to more important bits. However, UEP cannot be easily applied to VLCs.

To maintain zero-tolerance for bit errors in the header part of the code, very high protection rates are applied to this part. Since the size of this header is relatively small, high channel error protection is usually not costly in additional bits. Bit errors in the remaining part of the code will produce localized reconstruction errors, of course, assuming the header is received without error. Since the length of this part is fixed, loss in synchronization can be avoided. However, predictive coding of motion vector differences will continue to be affected by such errors until the next synchronization marker is reached. For comparison purposes, EEP is applied to H.263's VLCs.

The Binary Symmetric Channel (BSC) is employed as the channel model in our first set of simulation experiments. A random number generator is used to produce a binary noise pattern with a bit error rate (BER) exactly equal to the target one. We have experimented with several levels of protection before designing a UEP system having a good overall source-

BSC Error Pattern	VLC	Semi-Fixed-Length Code	
		Header	Fixed part
RCPC Rate	8:11	8:16	8:10
Source Coding	73%	50%	80%
Channel Coding	27%	50%	20%

(a)

GSM Error Pattern	VLC	Semi-Fixed-Length Code	
		Header	Fixed part
RCPC Rate	8:12	8:24	8:10
Source Coding	67%	33%	80%
Channel Coding	33%	67%	20%

(b)

Table 5.5: Source coding and channel coding rates for (a) BSC and (b) GSM error patterns.

channel coding performance over a wide range of BERs. In our simulations, the video sequences **Miss America** (10 fps), **Akiyo** (10 fps) and **Carphone** (5 fps) are coded at an overall bit rate of 9.6 kbps. At this low bit rate, a large proportion of the bit stream is used for the coding of the motion vectors. For example, 30% of the **Carphone** bit stream contains motion vector data. The source-channel coding simulation for each BER is repeated more than 100 times, and the average PSNR value is recorded. Table 5.5(a) lists the source coding and channel coding rates for the semi-fixed-length motion vector codes and H.263's VLCs. From the 1/4 mother convolutional code employed for the RCPC, different puncturing matrices will result in different convolutional codes with different error correction capabilities [51]. The combination of error correction codes for the Header and Fixed parts of the semi-fixed-length codes that yields the best rate-distortion performance over a wide range of error

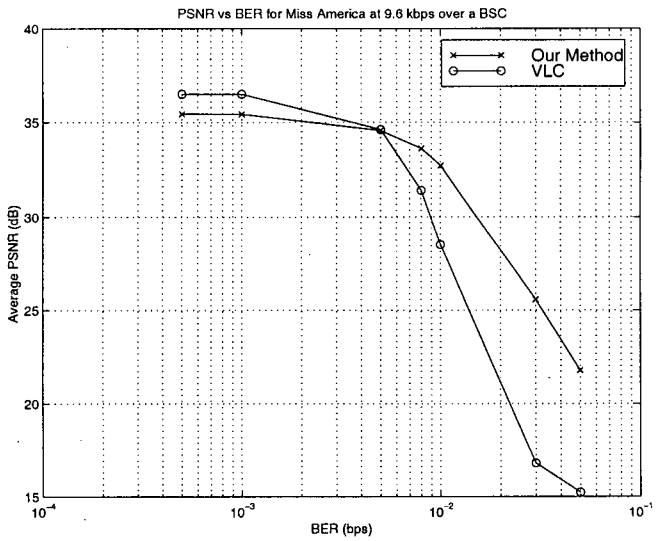


Figure 5.4: PSNR performance comparison between our coder and the VLC-based coder for a binary symmetric noisy channel model as a function of bit error rate (BER) for the video sequence **Miss America**.

conditions is selected via experimental results. For comparison purposes, the equivalent number of bits for EEP is employed for the H.263 VLC.

The total bit rate of 9.6 kbps includes both the source coding and the channel coding bits in both the semi-fixed-length coding and the VLC cases. The simulation results are shown in Figs. 5.4, 5.5, and 5.6 for the video sequences **Miss America**, **Akiyo**, and **Carphone**, respectively. As expected, when no protection is applied, both the VLC and the semi-fixed-length coding methods lead to significantly lower PSNR and subjective quality levels. Employing EEP for the VLC-based coder improves the performance at the lower BERs. However, the performance degrades significantly at the higher BERs. Clearly, UEP applied to the semi-fixed-length codes provides a

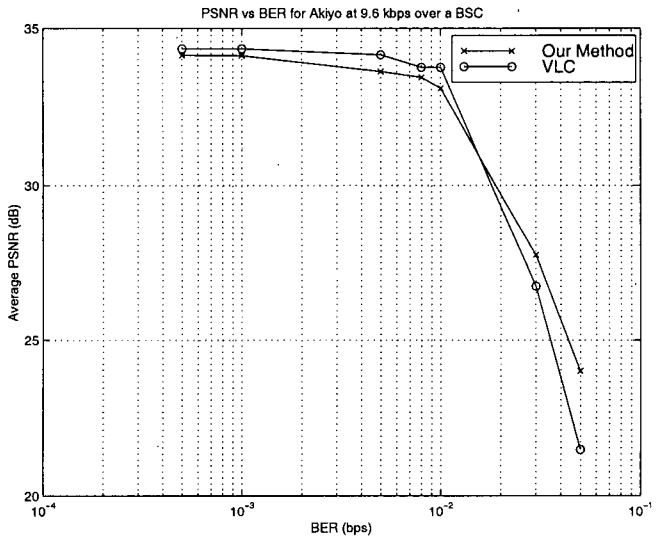


Figure 5.5: PSNR performance comparison between our coder and the VLC-based coder for a binary symmetric noisy channel model as a function of bit error rate (BER) for the video sequence AKIYO.

substantial performance advantage, by as much as 6 dB at a 5% BER.

In order to illustrate the benefits of our method in a wireless environment, a simulated GSM error pattern[2] is employed as the channel model in our second set of experiments. This pattern consists of 37.5 seconds of transparent GSM data using the GSM standard, which employs a 1/2 rate convolutional coder punctured down to a 5/6 rate, and simulated in the Tu50 (Typical Urban 50 km/h) channel mode. Also in this set of simulation experiments, the sequences MISS AMERICA (10 fps), AKIYO (10 fps) and CARPHONE (5 fps) are coded at an overall code rate of 9.6 kbps. The source-channel coding simulations for each Carrier power to Interferer power ratio, or $\frac{C}{I}$, is repeated more

[2]The error pattern was provided by Telia Research AB, and it is currently used within the ITU-T international standardization community for testing purposes.

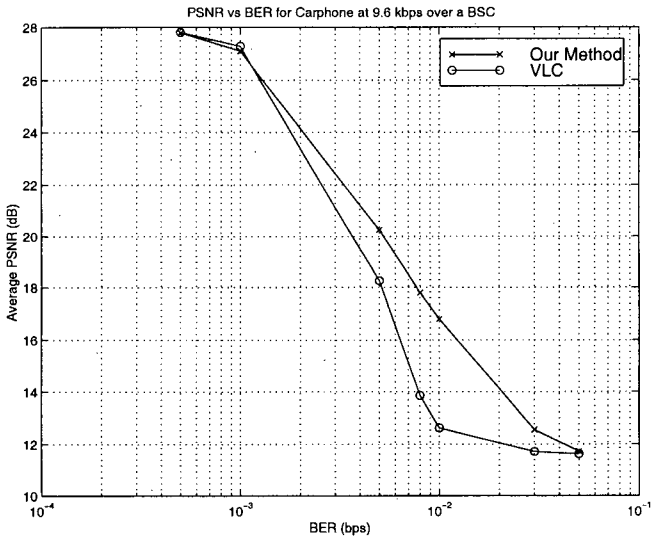


Figure 5.6: PSNR performance comparison between our coder and the VLC-based coder for a binary symmetric noisy channel model as a function of bit error rate (BER) for the video sequence **CARPHONE**.

than 100 times, and the average PSNR value is recorded. Table 5.5(b) lists the source coding and channel coding rates for the semi-fixed-length coding and VLC methods, both designed for burst channel errors. Our experiments have shown that more error protection is necessary for the header part of the semi-fixed-length coder to alleviate the effects of burst errors. Simulation results are shown in Figs. 5.7, 5.8, and 5.9 for the video sequences **MISS AMERICA**, **AKIYO**, and **CARPHONE**, respectively.. Again, UEP applied to the semi-fixed-length codes provide substantially better overall PSNR performance as compared to EEP applied to VLC, by as much as 6 dB at a $\frac{C}{I}$ ratio of 9 dBm. At high $\frac{C}{I}$ (very low BER), our method performs slightly worse than the VLC method. This can be explained by the fact that the VLC method performs

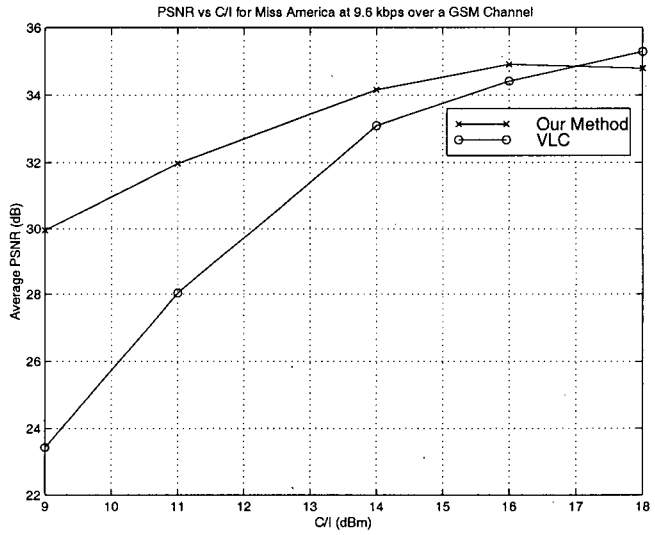


Figure 5.7: PSNR performance comparison between our coder and the VLC-based coder for a GSM channel model as a function of carrier power to interferer power ratio ($\frac{C}{I}$) for the video sequence MISS AMERICA.

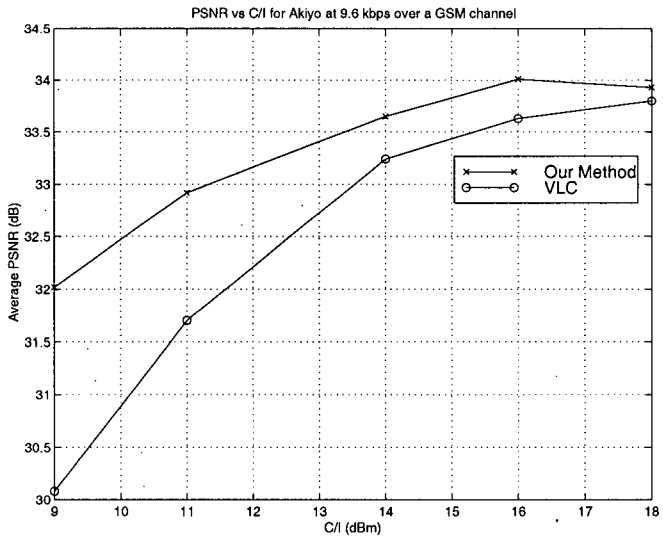


Figure 5.8: PSNR performance comparison between our coder and the VLC-based coder for a GSM channel model as a function of carrier power to interferer power ratio ($\frac{C}{I}$) for the video sequence AKIYO.

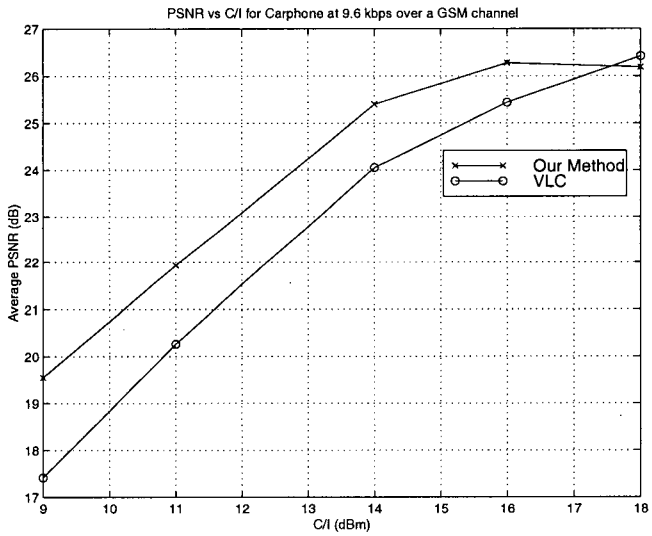


Figure 5.9: PSNR performance comparison between our coder and the VLC-based coder for a GSM channel model as a function of carrier power to interferer power ratio ($\frac{C}{I}$) for the video sequence **CARPHONE**.

slightly better than ours in a noiseless environment (which is essentially the same environment at high $\frac{C}{I}$).

Subjective quality improvements are also significant. Loss of synchronization due to channel errors usually localizes the decoding errors, often making the decoded video sequence unusable. This is mainly due to the loss of a whole sequence of macroblocks. Of course, the sequence length depends greatly on the frequency of synchronization markers. Adding a few protection bits to the header in our coder can eliminate the potentially catastrophic behavior of a VLC-based coder, where a large number of whole macroblocks in a P-picture can be lost, that is, until the next synchronization marker is found.

5.4 Summary

We have presented an efficient motion vector coding method that increases channel error resilience, with very little loss in PSNR performance and no loss in subjective quality in a noiseless environment. The proposed method produces a new motion vector code structure that is robust with respect to input video statistics by allowing half-pel motion vectors and relatively large search areas. In a noiseless environment, the proposed method yields essentially the same levels in PSNR and subjective quality as compared to H.263's VLCs. However, the semi-fixed-length coding method can be used in conjunction with an unequal error protection coder, leading to higher channel error resilience. To illustrate this, the proposed method was tested using both a BSC model and a GSM fading channel model, providing substantially better overall PSNR performance.

Chapter 6

Error Resilience: Optimal Intra Coding of Blocks

6.1 Introduction

The inter picture predictive coding employed in video compression algorithms lends itself to inevitable temporal error propagation. Coding complete video frames in the intra mode is an effective method to stop temporal error propagation. Unfortunately, intra coded frames require many more bits than inter coded frames, yielding undesirable long delays when transmitting video over a fixed bit rate channel. Therefore, higher performance levels are expected by intra coding only some of the blocks within a frame. While intra coding of blocks is still very expensive in terms of bits, we will see that it is effective in terms of error recovery. In fact, very good tradeoffs between compression efficiency and error resilience can be achieved.

Randomly choosing to code blocks in all frames in the intra mode at a certain frequency may introduce unnecessary redundancy. However, this method provides a simple and efficient method to improve error resilience. In order to only intra code blocks that cannot be appropriately concealed at the decoder, we develop a rate-distortion (RD) optimized mode selection method that employs both the error concealment distortion and the quantization distortion in the minimization criterion [87]. When a video encoder is aware of the concealment technique used by the decoder, it can perform the same error concealment at the encoder and choose the coding mode (inter/intra) that yields the best RD tradeoffs. The concealment distortion for a video block is weighted by the probability that this block is lost during transmission. The performance of this method is directly affected by the effectiveness of the error concealment method used, but the described techniques are applicable to any error concealment method. In [88], Lee proposes a multiple description technique to protect blocks that are badly concealed. The performance of the error concealment is measured in terms of incurred distortion, without taking into account the incurred rate for the added redundancy of the second description. Considering the rate in the cost function considerably improves performance.

The rest of this chapter is divided as follows. First, we evaluate the effects of random intra updating on the performance of the video coding system in error prone environments. We follow with a description of the proposed intra updating method using RD optimized mode selection. Finally, experimental results are presented using different block loss rates[1].

[1]Block loss rates can be evaluated from the packet loss rates, or they can be estimated

6.2 Random Intra Coding of Blocks

In a typical low bit rate video sequence, errors in many blocks can efficiently be concealed. Therefore, randomly intra coding all blocks in a frame would not be efficient. We propose an intra coding pattern where only blocks that contain texture information (i. e. excluding blocks that are skipped or only motion compensated) are intra coded. Therefore, most of the active regions and/or regions of interest will then be intra updated.

Randomly updating blocks using the intra coding mode to achieve error resilience has been proposed before (see Section 3.3.3). However, previous intra updating methods do not consider network conditions. In this thesis, we develop a relationship between the probability that a block is corrupted by errors, p, and the intra updating frequency, I_{freq}, based on our experimental results. To obtain this relationship, we encode different video sequences with a fixed bit rate and vary the intra block refresh rate, and record the decoded PSNR over four different block loss rates $p = 0\%$, 5%, 10% and 20%. In this thesis, decoder Y-PSNR results are presented as an example for the sequence **PARIS** encoded at 64 kbps in Fig. 6.1. Using all ITU-T video test sequences and for a wide range of bit rates (32-512 kbps), it can be observed that the optimal intra updating frequency (i.e., the intra block updating frequency resulting in the highest PSNR for a given value of p) can be approximated by

$$I_{freq} = \frac{1}{p}. \tag{6.1}$$

For example, for a probability of block loss p of 20%, each candidate coded

from bit error rates, as seen in Chapter 4.

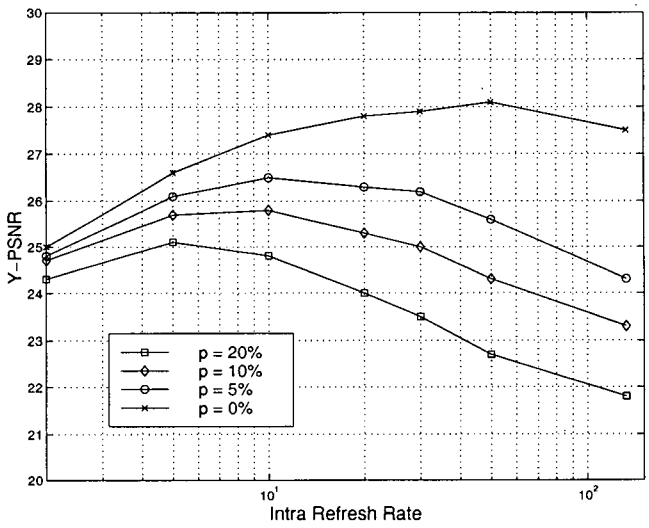


Figure 6.1: Performance of intra block refresh with block loss rate for PARIS.

block should be updated once in every 5 coded frames.

We evaluate the performance of the proposed random intra refresh pattern and frequency for a given probability of block error, p, with the error concealment method described in Section 3.6 at the decoder. Results are presented for the video sequences FOREMAN, PARIS, and CONTAINER in Figs. 6.2, 6.3 and 6.4, respectively, where CON means that concealment is used at the decoder and I-MB means that random intra macroblock updating is used at the encoder. It can be observed that intra coding alone or error concealment alone yields different performance levels, depending on the video sequence. Combining random intra coding and error concealment at the decoder improves significantly the video reproduction quality, by as much as 7 dB at a probability of block error p of 20% for the considered video sequences.

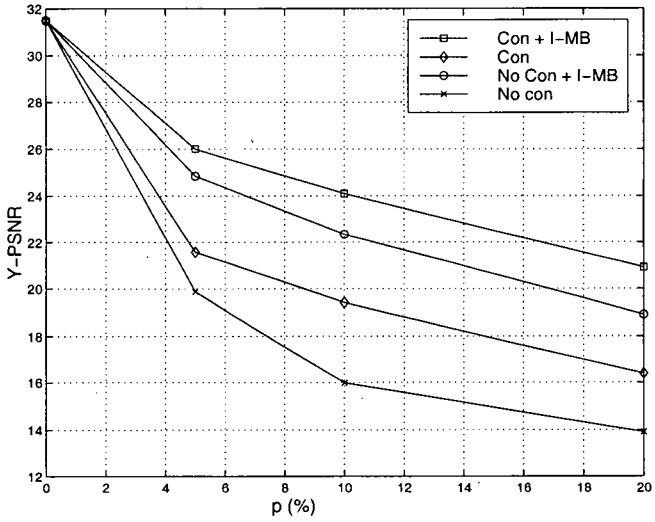


Figure 6.2: Performance of the random intra updating with/without error concealment for the video sequence FOREMAN.

6.3 RD Optimized Mode Selection in Error Prone Environments

In this section, we present an RD optimized mode decision algorithm for error prone networks. We first describe different distortion measures that take into account the temporal error propagation process. We then present the Lagrangian minimization method and determine a suitable value for the Lagrangian parameter.

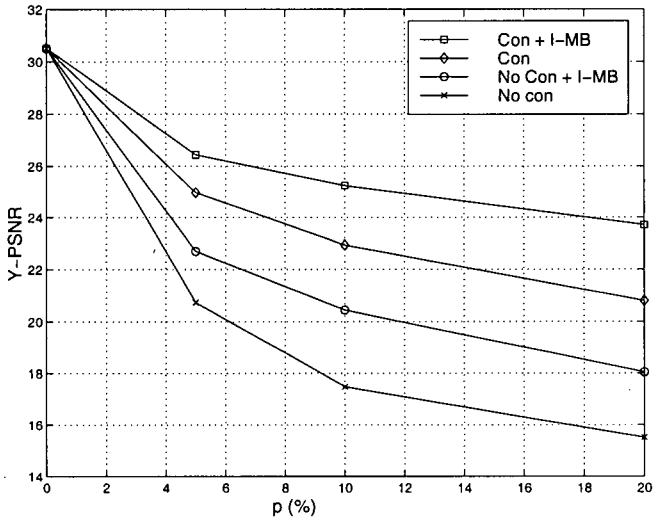


Figure 6.3: Performance of the random intra updating with/without error concealment for the video sequence **PARIS**.

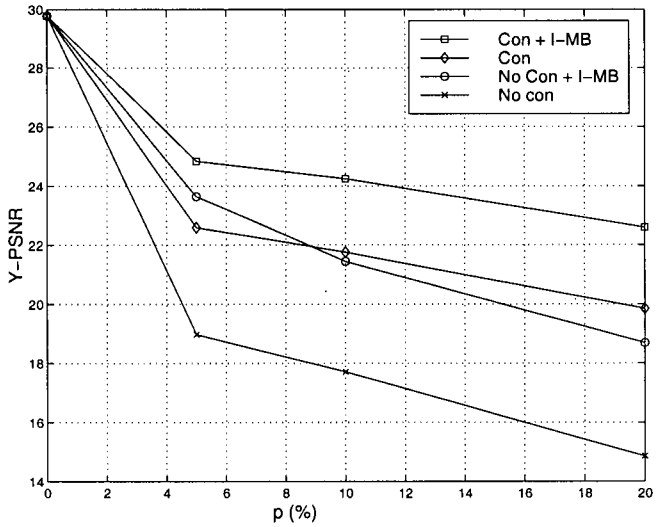


Figure 6.4: Performance of the random intra updating with/without error concealment for the video sequence **CONTAINER**.

6.3.1 Computation of distortion

We need to evaluate the distortion caused by quantization and concealment errors in a video frame, taking into consideration error propagation due to inter picture prediction. Three different methods for estimating the decoder distortion are introduced.

Since channel errors are random, it can be very difficult to evaluate the effects of error concealment on the video reproduction quality. Moreover, the human visual system is more sensitive to rapid spatial or temporal changes than to slowly varying changes. Error concealment can often produce quick variations, spatially at block boundaries where error concealment is applied, as well as temporally. Therefore, analytical measures of distortion such as SSE do not always accurately represent the subjective quality. However, this metric is tractable and easy to compute. Thus, it is employed to estimate the perceived distortion at the decoder.

We refer to the video encoder and decoder block diagrams in Fig. 2.1 and 2.2, respectively, and estimate the perceived distortion at the decoder. The input video block x_n at time n[2] is first predicted from the motion compensated block in the previous frame, y_{n-1}, resulting in an error block e_n. Assuming an error free channel, $r_n = s_n$, $\hat{y}_n = y_n$ and the only lossy stage is quantization. Thus, the error between the encoded signal and the decoded signal is $\hat{y}_n - x_n = q_n$, where q_n is the quantization error. However, if an error occurs during transmission, the received block r_n will be corrupted and concealed

[2]We abbreviate the pixel values (k, l) for a block at spatial location (i, j) and at time n, $x_{n,ij}(k, l)$, with x_n

during the decoding process. Thus $\hat{y}_n = c_n$, where c_n represents the pixel values of the block used for concealment. In this case, the error for the next block predicted from the previously concealed one, assuming this next block is received correctly, will be $\hat{y}_n - x_n = q_n + (\hat{y}_{n-1} - y_{n-1}) = q_n + (c_{n-1} - y_{n-1})$, where $(c_{n-1} - y_{n-1})$ is the concealment error. Depending on the coding mode, inter/intra, and the probability that a video block is lost and concealed, p, we can estimate the pixel values of the received block. For the intra mode, we obtain

$$\hat{y}_n \mid_{mode=intra} = (1-p)y_n + pc_n. \tag{6.2}$$

For the inter mode, temporal error propagation will occur as described above, and we can estimate the pixel values of the received block as

$$\hat{y}_n \mid_{mode=skip,inter} = (1-p)(e_n + \hat{y}_{n-1}) + pc_n. \tag{6.3}$$

A concealed video frame containing the pixel values $\hat{y}_n$ is stored at the encoder. The filtering operations such as half pel motion estimation and loop filters are applied to these concealed pixels in the same manner as the correct pixels. The video encoder still employs the correct pixel values y_n as prediction for future frames. The use of $\hat{y}_n$ at the encoder would result in high mismatch between the encoder and decoder prediction frames, and poor reproduction quality would result, especially if this same bit stream were transmitted over a channel with lower error rates than assumed by the encoder. We then estimate the decoder distortion at spatial location (i, j), using the SSE as a distortion measure

$$D_1 = SSE_1 = \sum_{l,k=1}^{N} (o_{n,ij}(k,l) - \hat{y}_{n,ij}(k,l))^2, \tag{6.4}$$

where $o_{n,ij}(k,l)$ represents the original pixel values of the block, and N is the size of the block. As we will see later, this method offers poor performance at low bit rates, because the difference between the estimated pixel values, $\hat{y}_n$, and the error free reconstructed pixel values, y_n, can be relatively small when quantization is coarse. This can be explained by the fact that at low bit rates, the concealed pixel values, c_n, can be very close to the reconstructed pixel values, y_n, and that weighted incorporation of c_n does not accurately represent the actual subjective quality.

Better performance may be possible by weighting the concealment distortion itself instead of averaging pixel values. This is possible because, when computing the total coding distortion, we can consider quantization errors, and errors due to prediction from a concealed block, separately. For a given video block, we denote D_q as the distortion caused by quantization, and $D_{conceal}$ as the distortion caused by error concealment. These distortions are, respectively,

$$D_q = SSE_q = \sum_{k,l=1}^{N} (o_{n,ij}(k,l) - y_{n,ij}(k,l))^2, \quad \text{and} \tag{6.5}$$

$$D_{conceal} = SSE_{conceal} = \sum_{k,l=1}^{N} (y_{n,ij}(k,l) - c_{n,ij}(k,l))^2. \tag{6.6}$$

For the block under consideration, two distortions are computed: the coding distortion D_{coding} and the concealment distortion $D_{conceal}$. Then, D_{coding} is weighted by the probability $(1-p)$ that this block is received, and $D_{conceal}$ is weighted by the probability p that the same block is lost and concealed at the decoder. $D_{conceal}$ is the distortion incurred by concealing the block being considered using the concealment method assumed by the encoder and is constant for all coding modes considered. For each coded block, $D_{conceal}$ is

stored and used for the computation of D_{coding} in subsequent frames. D_{coding} is the coding distortion and includes the incurred distortion from predicting from already concealed blocks. D_{coding} depends on the coding mode considered. For the intra mode, the distortion D_{coding} is the same as the error free coding distortion. For the *skip* and *inter* modes, D_{coding} is the quantization distortion plus the concealment distortion of the block(s) from which it is predicted. More precisely, for the considered block in Frame n at position (i,j), we obtain

$$\begin{aligned} D_{coding}(n,(i,j),mode) &= D_q(n,(i,j),mode) + \\ & \quad p(n-1,(i+v_i,j+v_j))D_{conceal}(n-1,(i+v_i,j+v_j)). \end{aligned} \tag{6.7}$$

The distortion $D_{conceal}(n-1,(i+v_i,j+v_j))$ represents the concealment distortion of the block in the previous frame $(n-1)$, at the current spatial location (i,j) displaced by the motion vector $v=(v_i,v_j)$. The value of p may be constant, but it can also vary. For example, p may vary over time due to network feedback. It can also vary for different spatial locations if the size of the slice varies (measured in number of bits per slice) and is employed to determine the value of p (see Section 4.2). In this case, the value of p can be constant if the block size and/or the slice size remain constant, or it will vary if the block size and/or the slice size vary. Therefore, we denote p as a function of the frame number n and spatial location (i,j). The motion vector v is set to $(0,0)$ for the *skip* mode case, and it is set to the motion vector of the considered block for the *inter* mode case. For the intra mode, only the quantization distortion contributes to the coding distortion. The overall distortion, D_2, is

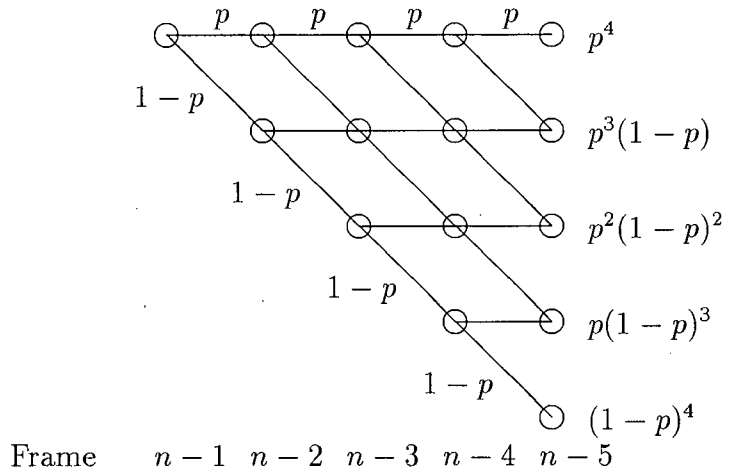


Figure 6.5: Error propagation tree due to error concealment.

then obtained:

$$D_2(n,(i,j),mode) = (1-p(n,(i,j))D_{coding}(n,(i,j),mode) + p(n,(i,j))D_{conceal}(n,(i,j)). \quad (6.8)$$

The accuracy of this distortion measure can be improved by considering error propagation beyond the previously coded frame. Fig. 6.5 represents a tree of the probability that a coded block at time n be predicted from a corrupted block in the previous frame considering that errors will propagate due to the predictive process of motion compensation. More precisely, for the considered block in Frame n at position (x,y), we can obtain

$$D_3(n,(i,j),mode) = (1-p(n,(i,j))D_q(n,(x,y),mode) + \quad (6.9)$$

$$\sum_{k=1}^{N} p(n-k,(i+v_i,j+v_j)) \times \quad (6.10)$$

$$D_{conceal}(n-k,(i+v_i,j+v_j)) + \quad (6.11)$$

$$p(n,(i,j))D_{conceal}(n,(i,j)),$$

where D_q represents the quantization distortion. The distortion $D_{conceal}(n-k,(i+v_i,j+v_j))$ represents the concealment distortion of the block in the previous frame $(n-k)$, k time intervals before the current frame, at the current spatial location (i,i) displaced by the motion vector $\mathbf{v}=(v_i,v_j)$. The motion vector $\mathbf{v}$ is set to $(0,0)$ for the *skip* mode case, and is set to the motion vector of the considered block for the *inter* mode case. The maximum number of frames considered, N, is set to the number of coded frames since the last intra coded block at the current spatial location.

In order to obtain $p(n-1,(i+v_i,j+v_j))$ and $D_{conceal}(n-1,(i+v_i,j+v_j))$[3], we use the stored values of $p(n-1,(i,j))$ and $D_{conceal}(n-1,(i,j))$ computed for the previous frame. The $D_{conceal}(n-1,(i,j))$ value is the constant (with respect to the mode) concealment distortion values at $(v_i,v_j)=(0,0)$ in frame $(n-1)$. We assume a maximum motion vector range of $(\pm 16,\pm 16)$, and a block size of 16×16. We compute $D_{conceal}(n-1,(i+v_i,j+v_j))$ by weighting the distortions of the surrounding blocks in the previous frame that overlap with the motion compensated block, as illustrated in Fig. 6.6. For each of the surrounding blocks numbered 1 to 9, a weight $w_n, n=1,...,9$, which is proportional to the overlap area, is computed. These weights are presented in Table 6.3.1. We then obtain

$$p(n-1,(i+v_i,j+v_j))D_{conceal}(n-1,(i+v_i,j+v_j))= \qquad (6.12)$$

[3]The same procedure is employed to compute the recursive concealment distortion, $D^r_{conceal}$.

$$\sum_{l=1}^{9} w(l)p(n-1,(i,j)_l)D_{conceal}(n-1,(i,j)_l),$$

where $(i,j)_l$ represents the block l overlapping the considered block at position (i,j) as shown in Fig. 6.6. The values of $w(l)$ are set to *zero* if the conditions in Table 6.3.1 are not met, i. e., if the block used for prediction does not overlap with the block represented by the weight $w(l)$.

6.3.2 Lagrangian minimization

Using one of the distortion measures described in the previous section, we present the Lagrangian minimization procedure for error prone networks and obtain a suitable value for the Lagrangian multiplier. Three coding modes are considered: *skip, inter, and intra.* Independently for every block at the spatial location (i,j) and in Frame n, we choose the mode that minimizes the Lagrangian given by

$$J_{mode}(n,(i,j)) = D_{mode}(n,(i,j),mode) + \lambda_{mode}R(n,(i,j),mode), \qquad (6.13)$$

that is, we choose the coding mode that yields the best RD tradeoffs for the block. The distortion D_{mode} is calculated as described above, and R is the incurred rate of coding the block. Using

$$\lambda_{mode} = c \times \left(\frac{Q}{2}\right)^2 \qquad (6.14)$$

has been shown to provide good RD tradeoffs [42], where Q is the quantization step size of the block. The quantization parameter (QP) in H.263 is defined by $QP = \frac{Q}{2}$. In order to find the optimal value of c, we follow the same

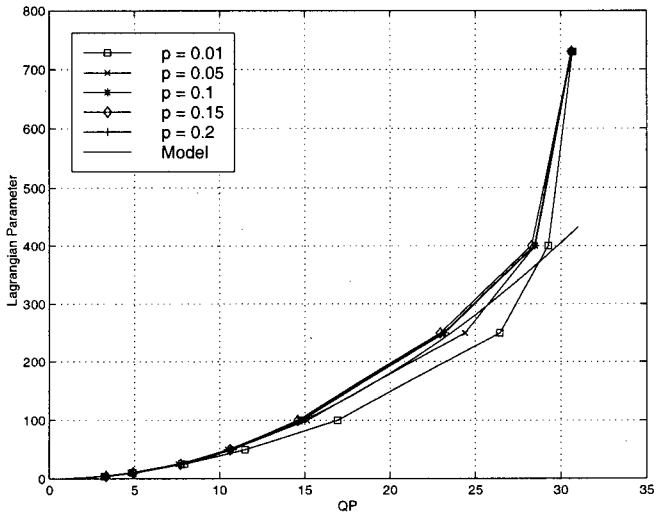


Figure 6.7: The Lagrangian parameter λ_{mode} versus the H.263 quantizer parameter QP for different probability of block loss rates p.

procedure described in [42]. That is, for a fixed value of λ_{mode}, the quantizer parameter value QP that minimizes Eq. 6.13 is recorded. This procedure is repeated over a range of λ_{mode} such that all QP values are covered. For the modes considered, we obtain $c(skip, inter, intra) = 0.45$ in an error free environment. In order to obtain an appropriate value of $lambda_{mode}$ for an error prone environment, we repeat this for a range of p. We employ D_3 as the distortion measure D_{mode}. We obtain

$$\lambda_{mode} = 0.45 \times \left(\frac{Q}{2}\right)^2 + \Delta\lambda(p, Q), \tag{6.15}$$

where $\Delta\lambda(p, Q)$ depends on the probability of block error as well as the quantizer step size. Experimental results for the video sequence **FOREMAN** are presented in Fig. 6.7. Similar results are obtained with other video sequences. It can be observed that the dependencies between λ_{mode} and p are negligible.

Therefore, the effects of p on λ_{mode} are not considered in the mode selection process, and $\Delta\lambda(p, Q)$ of Eq. (6.15) is set to *zero*.

Using this above minimization, good RD tradeoffs can be achieved subject to the probability of error and concealment constraints. The error concealment method will directly affect the mode decision. A better error concealment method than the one employed here will yield better RD performance given the same probability of error rate. Note that, by minimizing Eq. (6.13), blocks that are compensated from well-concealed ones in the previous frame will most probably not be coded in the intra mode.

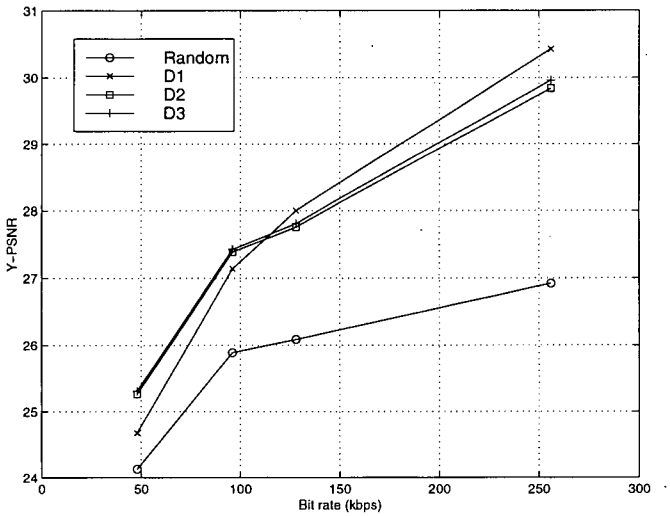
6.3.3 Complexity of the proposed method

The proposed mode selection algorithm requires additional encoder complexity. First, the decoder error concealment must be performed at the encoder for each block and the incurred concealment distortion must be computed. Then, for the each of the coding modes considered (*skip, inter, and intra*), the blocks must be predicted, transformed, and quantized. The corresponding distortion (considering concealment) and coding rate must then be computed for the Lagrangian minimization. For example, in our implementation, encoding time is increased by a factor of roughly 2.5:1 on a Sun Sparc workstation (when a fast motion estimation algorithm [28] is employed). Of course, no additional decoder complexity in incurred.

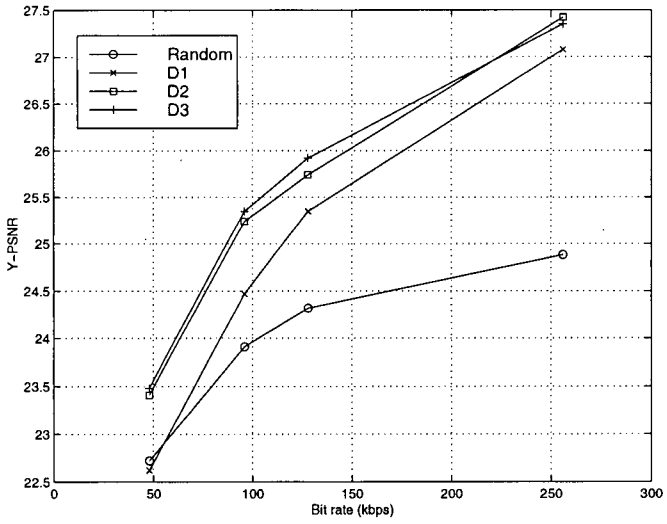
6.4 Performance of the Proposed Method

In this section, we present results of the proposed RD optimized mode decision algorithm and evaluate the performance of the different distortion measures. We compare the results to those of the random intra updating method. GOB synchronization and decoder error concealment is employed in all cases.

Results are presented in Fig. 6.8 for the video sequence FOREMAN, and in Fig. 6.9 for the video sequence NEWS, for a block probability of error p of (a) 10% and (b) 20%. As expected, using block distortions (D_3) rather than pixel values (D_1) provides superior reproduction quality at the decoder. Over 1.5 dB improvement is achieved over the random intra updating method in all considered cases. The performance improvement increases as the available bit rate increases. This is because the cost of intra coding reduces as the bit rate is increased, and better error resilience can be achieved. The performance of the use of distortion D_1 approaches that of the use of D_3 at higher bit rates, but the use D_3 always outperforms any other distortion measure. Therefore, we employ this distortion measure for RD optimized mode selection in the rest of this thesis. The use of distortion D_2 provides reasonable performance at reduced complexity. In this case, only the last computed concealment distortion needs to be stored and the estimated pixel values $\hat{y}_n$ need not be computed. This distortion measure may be more appropriate for low complexity implementations.

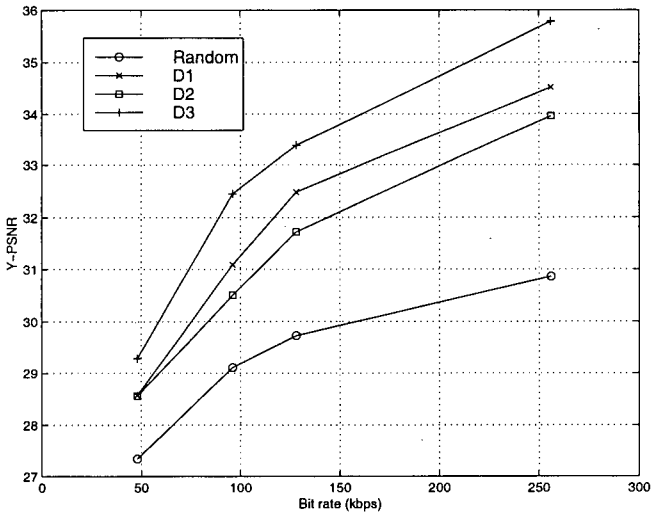


(a)

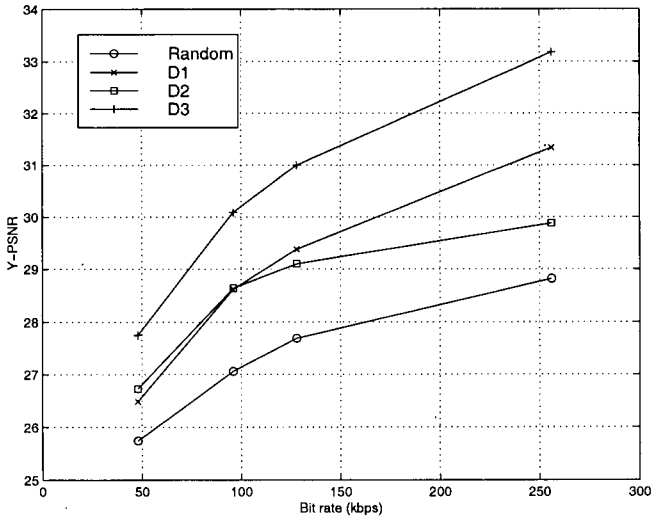


(b)

Figure 6.8: Performance of the proposed mode decision with different distortion measures for the video sequence FOREMAN and for a block loss rate of (a) 10% and (b) 20%.



(a)



(b)

Figure 6.9: Performance of the proposed mode decision with different distortion measures for the video sequence **NEWS** and for a block loss rate of (a) 10% and (b) 20%.

6.5 Summary

In this chapter, we presented a rate-distortion optimized mode decision for error prone environments. The proposed method takes into account the probability that a block is lost during transmission and concealed at the decoder. It was shown that the proposed method performs substantially better than a random intra updating method, and that the performance highly depends on the adopted distortion measure. We presented three different distortion measures, and we have shown that a measure that weighs the block distortions, rather than concealed pixel values, provides better performance.

Chapter 7

Performance of the Proposed Algorithms: Internet

7.1 Introduction

The Internet does not guarantee any QoS, which results in high packet loss rates, variable delays, etc. In this chapter, we present different video packetization methods suitable for real-time interactive video communication applications and evaluate their performance. The use of bit-oriented error resilience methods (e.g., semi-fixed-length coding, placement of synchronization markers) for packet-based network is not applicable. Therefore, we only incorporate the RD optimized intra updating method of Chapter 6 with the proposed packetization methods and present simulations results in the last part of this chapter [89].

7.2 Transport of Video over the Internet

Video communication over the Internet can be divided into two broad categories: interactive two-way communication and non-interactive store-forward communication. Interactive communication requires low end-to-end delay. Internet-based Videotelephony and Videoconferencing are good examples of such applications. Non-interactive communication applications can sustain reasonable end-to-end delay in the order of up to a few seconds, as long as continuous playout of the video stream is possible. Typical examples include Internet video streaming, and most forms of surveillance applications. In this section, we first introduce the transport of video for different Internet applications, and we then propose different packetization methods suitable for real-time applications.

7.2.1 Non-interactive applications

From a transport point-of-view, non-interactive video communication is a relatively simple problem. In a point-to-point scenario, a possible protocol hierarchy could consist of TCP/IP [90, 91] for the transport of video information and RTP [92] for providing timing information. Such a RTP/TCP/IP protocol combination would allow for an extremely low packet loss rate if the playout-delay is long enough to give TCP time for almost all re-transmissions. In a multipoint or broadcasting scenario, either reliable multicast protocols or a simpler method similar to the one presented next for interactive applications could be employed in order to guarantee reasonable performance.

7.2.2 Interactive and real-time applications

Interactive communication requires low end-to-end delay, and thus cannot rely on end-to-end re-transmission protocol algorithms. The introduced re-transmission delay is, in case of an IP packet loss, at least three times that of the one-way transmission delay, since the packet loss has to be signaled (by the arrival of a packet with a higher than expected sequence number, or by timeout), the re-transmission has to be requested and the re-transmission itself has to take place. For many off-campus Internet connections involving more than a few routers, an end-to-end transmission delay of 100 ms or more can be assumed on the routing layer, leading to 300 ms or more on the application layer after a single re-transmission. Adding an assumed typical video coding/decoding delay of 200 ms, the end-to-end delay from a user's point of view results in half a second. This is clearly too high for useful interactive two-way communication.

Clearly, a transport protocol not based on re-transmission has to be employed. The majority of today's Internet-based video communication systems employ the user datagram protocol (UDP) [90]. As a datagram protocol, UDP's major functionality is the application addressing. It does not perform any improvement in the transmission quality of service, with the exception of an (optional) CRC check to ensure the integrity of the payload data. In a typical environment, UDP packets arrive at the receiver often with a substantial packet loss rate, due to the lack of re-transmission. Recent research has shown that loss rates of 20% or more are common for many public Internet connec-

tions [93, 94, 95]. On top of UDP, the real-time transport protocol (RTP) is employed to provide some real-time application-layer information such as a playout time-stamp and data type. RTP consists of a payload and a control part called RTCP. The data part of RTP is a thin protocol providing support for applications with real-time properties such as continuous media (e.g., audio and video), including timing reconstruction, loss detection, security and content identification. RTCP provides support for real-time conferencing of groups of any size within an intranet. RTP packets may carry in their payload either the media data directly, as it is done for most types of coded audio, or a payload specific additional header may be necessary. The payload header for H.263 video coding is defined in RFC2429 [77]. The video conferencing standard for an IP-based network is defined in the ITU-T Recommendation H.323 [96]. In this standard, video is transported separately on its own logical channel using RTP on top of UDP and IP. In this thesis, such an IP/UDP/RTP protocol hierarchy is assumed for the transport of video over this Internet. Most of the key functionalities enabled by this payload header are used by the packetization methods proposed next.

7.3 Video Packetization for the Internet

A typical Internet "video packet" consists of header information for IP, UDP, RTP, the RTP payload header, and the payload data itself. The size of those headers is quite substantial. As a minimum, we need 20 bytes for IP, 8 bytes for UDP, 12 bytes for RTP, and a variable number of bytes for the RTP

payload header. Given a minimum amount of 40 bytes of header information for each video packet, there is a need to produce video packets as large as possible to minimize packetization overhead. However, two upper bounds on the video packet size have to be considered. First, transmitting more than one picture in a single packet is not recommended due to delay constraints. Second, the typical Maximum Transfer Unit (MTU) size of IP packets has to be considered. While all the mentioned Internet protocols do allow for packets of up to 64 Kbytes, the MTU size is usually assumed to be much smaller - around 1500 bytes. The reason for this number lies in the history of packet networks, especially Ethernet-type local area networks. On an Ethernet, every IP packet exceeding 1500 bytes has to be split into at least two Ethernet packets, thus effectively doubling the IP packet loss rate for a given Ethernet packet loss rate. Although the Ethernet is no longer relevant for long-distance Internet connections, many router implementations still seem to use split/recombine algorithms for packet sizes larger than the MTU size [95].

Given a maximum video packet size of approximately 1450 bytes, or 11600 bits per packet, one coded picture could easily fit into one packet for most current applications. For example, at 10 frames per second, the bit rate when using the full maximum payload size of a packet would be 116 kbps on a bit-oriented channel, which is more than enough for good quality QCIF or even CIF video sequences. At higher frame rates, the bit rate would be correspondingly higher, so that, at 30 fps, 348 kbps would be the upper limit using a single picture per packet and an MTU-size of 1450 bytes.

All the above numbers suggest a general rule for minimum overhead packetization that can be expressed in one picture per video packet. However, when applying this rule, a single packet loss means the loss of a whole coded picture. From an error resilience point of view it would be desirable to divide a coded picture into a large number of packets to keep the spatial area affected by a packet loss as small as possible. In the next section, we present two standard-compliant packetization methods which yield optimal tradeoffs between error resilience and overhead reduction.

7.3.1 Packetization of H.263 using RFC2429

Even if bit errors do not occur in the Internet, synchronization markers are still necessary in case of packet losses in order to maintain spatial synchronization at the decoder. As shown in Chapter 4, uniform spatial synchronization outperforms uniform bit synchronization. In this section, we present two packetization methods using uniform spatial synchronization suitable for the Internet.

Error concealment is most effective when information surrounding the missing video packet is still available. Therefore, we limit the maximum size of a video segment to one GOB. The most straightforward packetization method is to put each GOB into one video packet, leading to an overhead of 40 bytes per GOB. This overhead is significant but may be justified by the error resilience gained, and it can be lowered by using IP/UDP/RTP header compression on the critical, bandwidth limited link between the terminal and the first

router.

We propose an alternative packetization method that lowers the packetization overhead, but at the same time still provides high error resilience. We would like to packetize more than one GOB into a single packet, while keeping a maximum video packet size of one picture. A solution is a simple interleaving method that consists of packing all even GOBs into one packet, and all odd GOBs into another one. This leads to two packets per picture and eases concealment of all macroblocks of a picture if only one of those two packets is lost. The constant packetization overhead (consisting of the IP/UDP/RTP headers, 40 bytes per packet in total) is 80 bytes per coded picture, at any spatial resolution. This is substantially lower than in the case of one GOB per packet, where the overhead is 320 bytes per picture at QCIF resolution.

Note that the 2-byte minimum RFC2429 header has no impact on the overhead, because the use of this header allows for the deletion of the 16 bit Picture Start Code or GOB Start Code that precedes each picture/GOB in H.263. This codeword, used in bit-oriented environments for synchronization, is represented by a single bit in the RFC2429 header and is thus deleted from the video bit stream during the packetization. Other parameters of the GOB header that allow for spatial synchronization at the decoder remain present.

When using the above packetization method three different situations can occur at the receiver, depending on the packet loss situation:

- Both packets are received. In this case the de-packetizer will re-arrange the GOBs into their original order and feed them to the decoder.

- Only one packet is received. Immediately after detection (performed by RTP due to either a timeout or the detection of a wrong sequence number) the received video data is fed to the decoder. The decoder detects missing information using the GOB numbering information and applies error concealment.

- Both packets are lost. No data is available for decoding. The video decoder can learn by out-of-band signaling about this condition, which can be easily detected by RTP mechanisms using the sequence numbers. The decoder can request a full intra frame, employ reference picture selection, or perform temporal interpolation. In our case, the decoder ignores the above information and relies on our intra block updating method for error recovery.

7.4 Simulation Results

In this section, we first evaluate the performance of the two packetization methods proposed above, namely the one GOB - one packet and the one picture - two packets methods, for different packet loss rates (PLRs). Second, we evaluate the performance of the intra updating algorithm presented in Chapter 6 in an Internet environment.

We assume that packet losses occur randomly. There are contradictory reports on the burstyness of packet losses. However, even when packet losses are bursty, such bursts do not seem to have a long enough duration to lead to

Available Bit Rate	Packetization Bit Rate	Video Bite Rate	Y-PSNR
64 kbps	0.0 kbps	64.0 kbps	31.54
64 kbps	6.4 kbps	57.6 kbps	31.07 (1)
64 kbps	28.8 kbps	35.2 kbps	29.04 (2)
128 kbps	0.0 kbps	128.0 kbps	34.68
128 kbps	6.4 kbps	121.6 kbps	34.53 (1)
128 kbps	28.8 kbps	99.2 kbps	32.56 (2)

Table 7.1: Example of possible PSNR degradation due to packetization methods: (1) one picture - two packets and (2) one GOB - one packet, for the video sequence FOREMAN.

bursty errors in video traffic, since pictures arrive only every 1/10th of a second when encoding at 10 frames per second. Therefore, it is reasonable to assume a random error pattern model. For each PLR considered, 25 simulations are performed and the average luminance PSNR (Y-PSNR) is computed. The video source material is coded at a QCIF spatial resolution and a temporal resolution of 10 frames per seconds.

7.4.1 Performance of the packetization methods

As seen above, Internet video packetization requires a significant bit rate overhead, but it can be justified by the improvement in error resilience. Table 7.1 presents the effective PSNR degradation due to the reduced bit rate available after packetization for the video sequence FOREMAN. Figs. 7.1 and 7.2 show the PSNR of the different packetization methods versus PLR for a channel bit rate of 50 and 150 kbps for the sequences FOREMAN and NEWS, respectively. A random intra updating of 20% is employed in all cases. Note that the bit

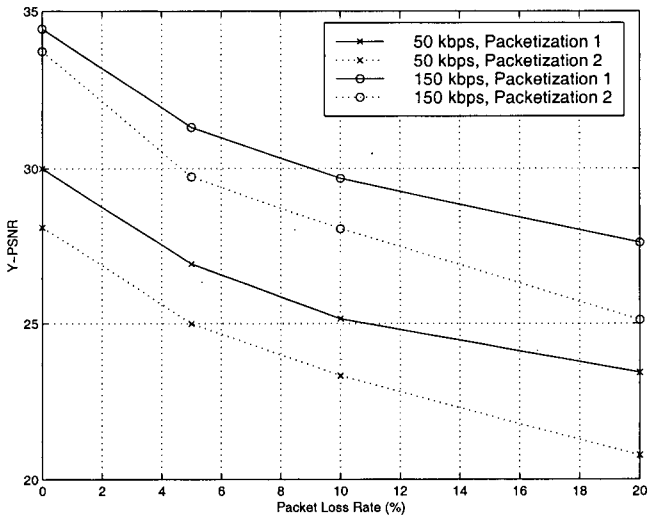


Figure 7.1: Relative performance of the two packetization methods for the sequence FOREMAN at different bit rates and PLRs.

rates include the packetization overhead. The one GOB - one packet method requires an overhead of 28.8 kbps at QCIF resolution and 10 fps, which limits the lower video bit rate reasonably achievable for this packetization method. Thus, for low bit rates (e.g. Modem connections), this method cannot be used. But even for higher bit rates, the one picture - two packets method performs consistently better than the one GOB - one packet method. Therefore, the one picture - two packets method is employed in the simulation results presented next.

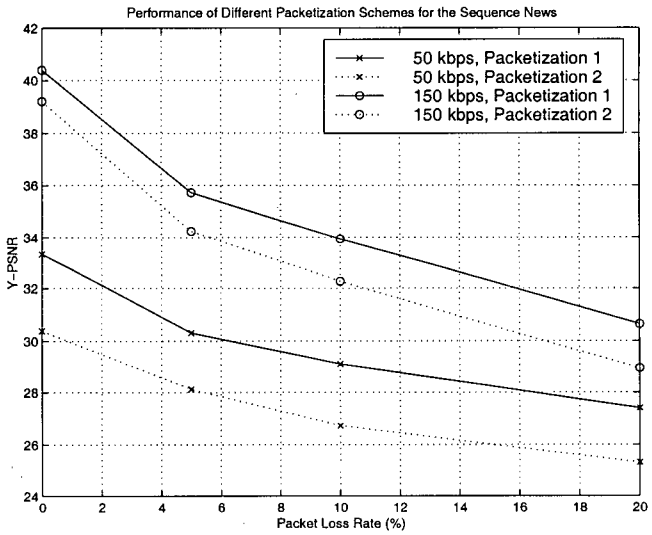


Figure 7.2: Relative performance of the two packetization methods for the sequence NEWS at different bit rates and PLRs.

7.4.2 Performance of the RD optimized intra updating algorithm

In this section, we present results for our RD optimized intra updating algorithm described in Chapter 6, where the same decoder error concealment method described in Section 3.6 is used at the encoder and decoder. The value p can be obtained from the receiver report mechanism of RTCP [92]. We compare the RD optimized intra updating mechanism versus the random updating that is proposed in Section 6.2 and adopted by the ITU-T [97] for Internet Test Model simulations. A random refresh pattern of 20% is used in all simulations. Packet Loss Rates (PLR) of 5, 10 and 20% are considered, which are typical of the public Internet [93]. We present results in Figs. 7.3 and 7.4

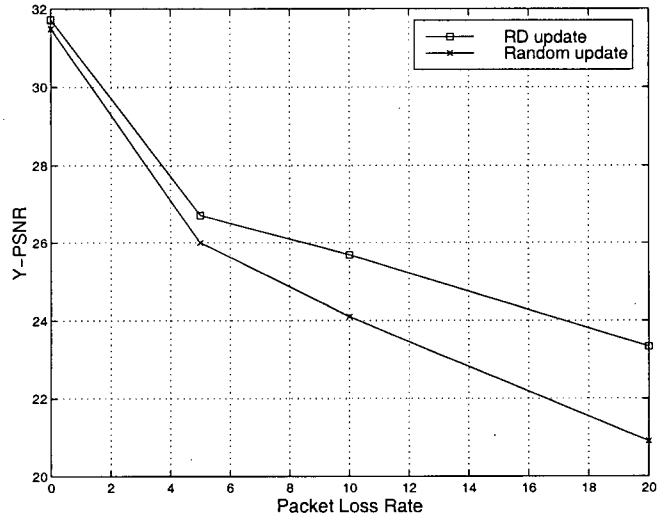


Figure 7.3: Performance of the random and RD optimized intra updating methods, (with error concealment) for the video sequence**FOREMAN**.

for the video sequences **FOREMAN** and **COASTGUARD**, respectively, coded at a channel bit rate of 64 kbps. We also present results for different bit rates in Figs. 7.5 and 7.6 for the video sequences **FOREMAN** and **NEWS**, respectively. Here, we consider three bit rates: 20 kbps and 50 kbps to simulate modem and ISDN dialup connections to an ISP, and 150 kbps to simulate high-bandwidth connections to the Internet backbone, or for LAN connections. All bit rates include network and video packetization overhead. By splitting the video frames into GOBs, additional coding penalties are incurred from both the size of those headers themselves and from predictive coding limitations (e.g., motion vector prediction). This has to be taken into account when comparing the PSNR values to those obtained by a coder optimized for a lossless environment.

At 20% PLR, as much as 2.4 dB gain is achieved by the proposed RD

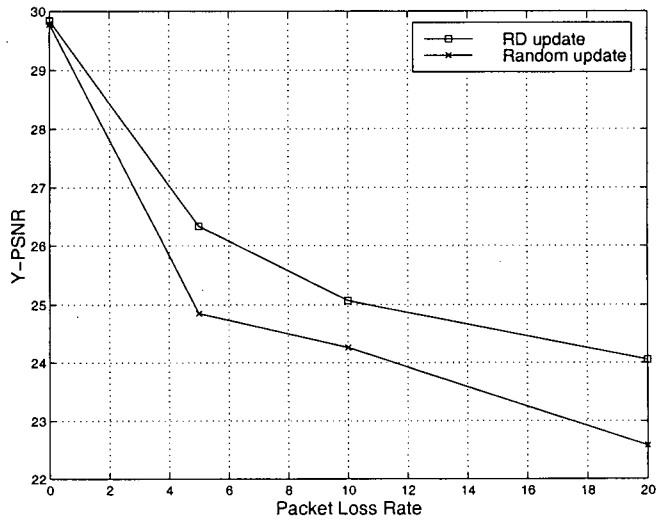


Figure 7.4: Performance of the random and RD optimized intra updating methods, (with error concealment) for the video sequence **COASTGUARD**.

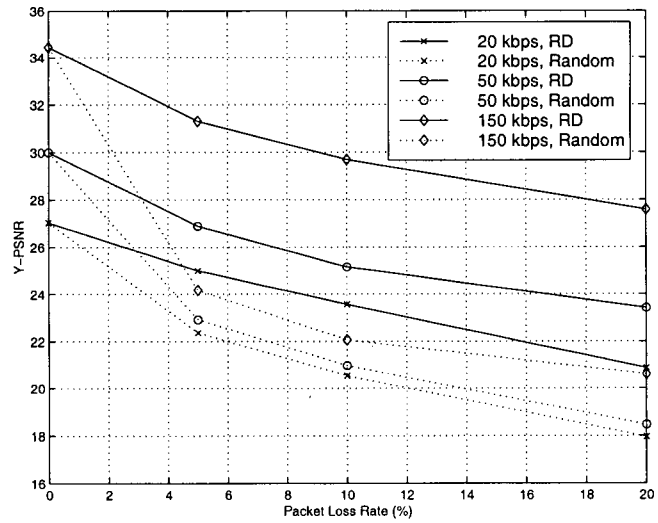


Figure 7.5: Performance of RD-optimized mode decision for the sequence **FOREMAN** at different bit rates and PLRs.

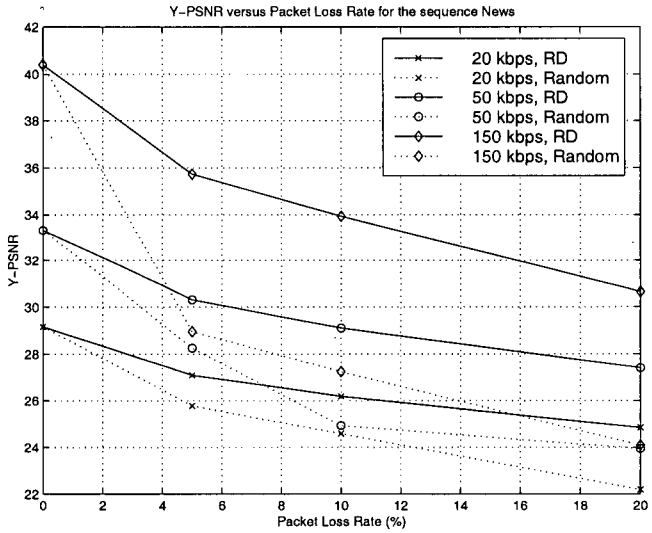


Figure 7.6: Performance of RD-optimized mode decision for the sequence **NEWS** at different bit rates and PLRs.

updating mechanism for bit rates of 150 kbps. The gain achieved by the RD intra updating method is due to the consideration of the error concealment technique at the encoder and the incorporation of the error rates in the video coding mode selection. Fig. 7.7 shows the Y-PSNR of the first 50 decoded frames of the sequence **FOREMAN** coded at 64 kbps, at a PLR of 20% for random intra updating and the RD method. Clearly, our method consistently outperforms the random intra updating one in terms of PSNR. The subjective quality is also significantly improved, as seen in Fig. 7.8, where the decoded frame number 40 of the sequence **FOREMAN** is shown.

In the random updating case, every coded block is updated once every five frames. Using our method, only blocks significantly affected by packet losses and/or where error concealment fails will be more often intra coded.

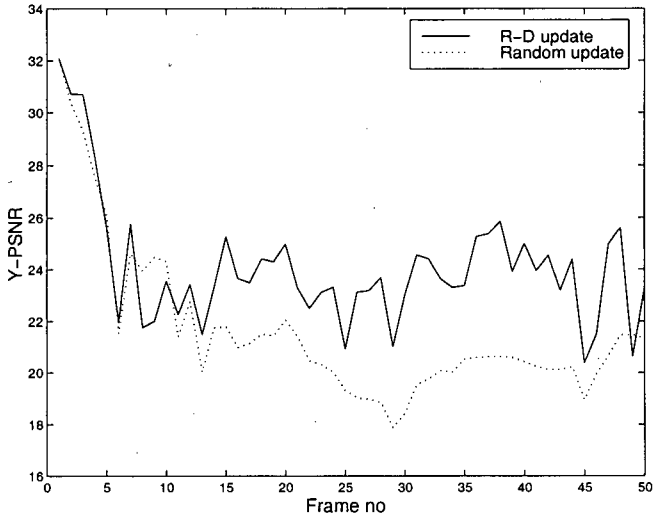


Figure 7.7: Performance of the random and RD optimized intra updating methods (with error concealment) at a PLR of 20% for the first 50 frames of FOREMAN.

These corresponding regions are usually of interest in a typical videophone or video conferencing application. Low activity regions and static backgrounds are usually well concealed, and need not be coded accurately. This can be observed in Fig. 7.8, where the background in the random updating case is of slightly better quality, but the facial expression in the RD optimized updating case is much better reproduced. In particular, the blocks representing the mouth of the person in the video sequence are correctly received in the RD intra updating case. In the intra updating method, this region is not intra updated and badly concealed, resulting in an incorrect lip position. At a 20% packet loss rate, our method still provides reasonable video reproduction quality, whereas the usefulness in the other reproduced video sequence is questionable.

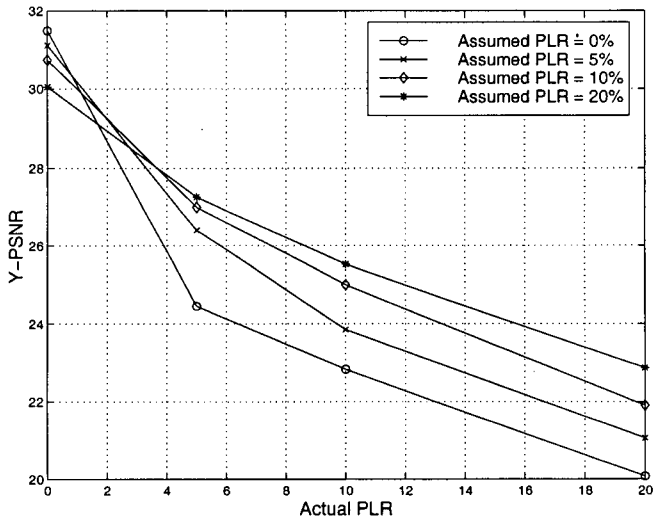


Figure 7.9: Performance of the random intra updating method for several assumed PLRs versus the actual PLR for the video sequence FOREMAN.

The proposed method is, of course, somewhat dependent on the end-to-end PLR. If a specific PLR is assumed by the encoder, that is different from the actual PLR, the resulting video quality will be degraded. In order to evaluate the effects of PLR mismatch, we encoded the video sequence FOREMAN at an assumed PLR of 0, 5, 10 and 20% and a fixed bit rate of 64 kbps and simulated the transmission of each resulting video bit stream over a network with different actual PLRs. Results are presented in Fig. 7.9 for the random intra updating method and in Fig. 7.10 for the RD intra updating method, where the case of an assumed PLR of 0% is equivalent to the error-free RD optimization case. In the random intra updating method, a mismatch clearly affects the resulting video reproduction quality. However, the performance of our method is not affected significantly by the PLR mismatch, when PLRs of

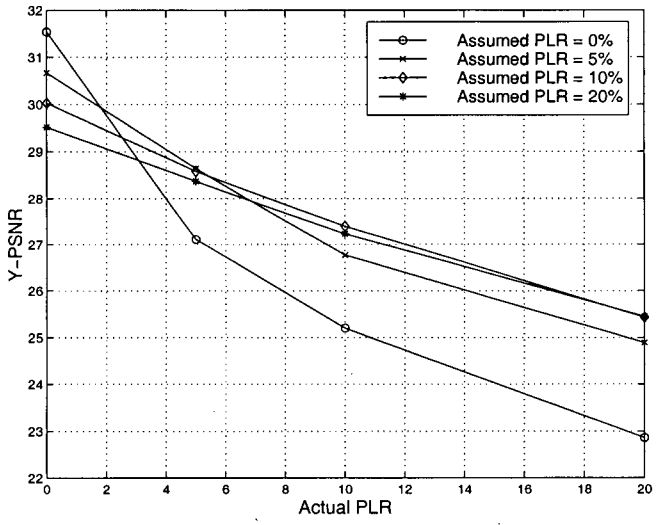


Figure 7.10: Performance of the proposed method for several assumed PLRs versus the actual PLR for the video sequence **FOREMAN**.

5%, 10% and 20% are assumed. However, in the error free transmission case (actual PLR of 0%), the PSNR loss resulting from our method ranges from 0.8 to 2.0 dB. Finally, the corresponding degradation in video quality is visible as coding artifacts, which are often less disturbing than concealment artifacts. It appears that, by assuming a PLR of 5%, the proposed method performs quite well for the whole range of actual PLRs.

7.5 Summary

In this chapter, we presented results for a standard-compliant video communication system for an Internet environment consisting of a RTP/UDP/IP transport protocol stack. The system consists of video encoder with RD optimized

intra coding, decoder with error concealment, and a packetization method. The system was designed to allow for uni-directional, real-time communication, as it does not rely on any feedback mechanisms other than information about the packet loss rate. This information can be obtained from the network through RTCP receiver reports or can be assumed a priori. Nevertheless, the performance of our method is not significantly affected when the assumed packet loss rate differs from the actual one. In this situation, the proposed method clearly outperforms the random intra updating method. The combination of the proposed algorithms for Internet video communication yields good quality of the reconstructed video pictures even at high packet loss rates such as 20%.

Chapter 8

Performance of the Proposed Algorithms: Mobile Networks

8.1 Introduction

Transport of video over mobile networks can be characterized by a very low bit rate channel, highly time-variant bit error prone conditions, and the need of a low delay and low overhead transport protocol. In this chapter, we combine the spatial synchronization, semi-fixed-length coding of the motion vectors, and RD optimized intra updating, and evaluate the performance of the resulting system over bit error prone mobile networks [98]. We also evaluate the effect of encoder/channel and encoder/decoder mismatch in the joint optimization, for the algorithms that rely on such information. More precisely, we evaluate the effects on the video reproduction quality when the error rates assumed by the video encoder are different than the actual channel error rates and

when the error concealment method assumed by the video encoder is different than the error concealment method employed by the video decoder. Next, we introduce the transport protocol employed for video communications over mobile networks.

8.2 Video Transport over Mobile Networks

In a mobile environment, a terminal using the H.324 ITU Recommendation [99] is usually employed. Within this terminal standard, the H.223 ITU Recommendation [100] is used for multiplexing. Control, audio and video information is multiplexed into one bit stream for transmission over a mobile channel. A Cyclic Redundancy Code (CRC) is employed for error detection in audio and video payloads. A media packet in H.223 is called an Adaptation Layer Protocol Data Unit (AL-PDU). An AL-PDU consists of an optional control field, the media payload, and checksum information. Audio data usually employs H.223 AL-2 while video usually employs H.223 AL-3. AL-2 and AL-3 payloads are appended with a 8-bit and 16-bit CRC, respectively. The CRC is used at the receiving end to verify the integrity of the data. Depending on the error handling capabilities of the video decoder, erroneously received video payloads may be processed or discarded. For mobile applications, we consider one AL-3 PDU as a video packet.

An AL-PDU is conveyed as one or more Multiplex Service Data Units (MUX-SDUs). MUX-SDUs from different AL-PDUs are multiplexed according to a Multiplexing Table. MUX-SDUs may be segmentable. MUX-SDUs form

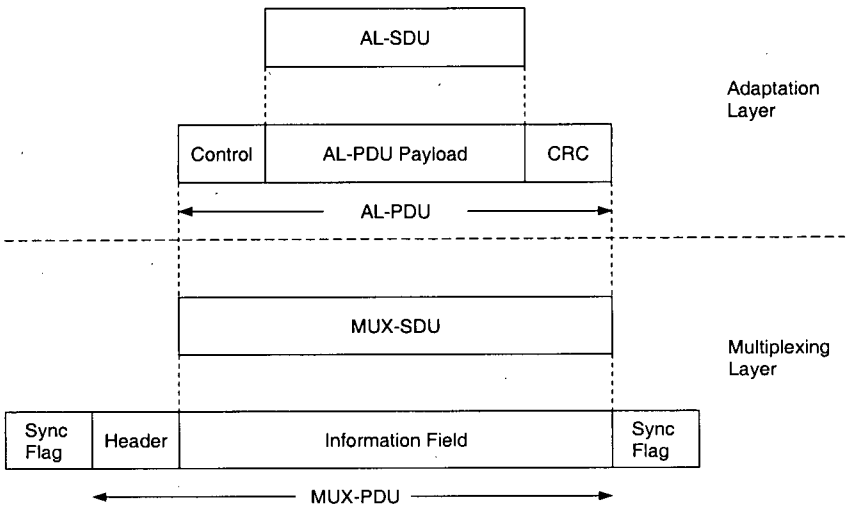


Figure 8.1: Example of the packetization of one complete non-segmentable MUX-SDU into one MUX-PDU.

the information field of the MUX-PDU. Fig. 8.1 shows an example of the packetization of one complete non-segmentable MUX-SDU into one MUX-PDU. As a minimum overhead for video only communications, H.223 requires 2-bytes for the AL-3 CRC, a 3-byte header for the MUX-PDU (with H.223 Annex B) and a 2 byte MUX sync flag, for a total of 7 bytes. Therefore, a small AL-PDU may be usually employed, leading to much smaller video packets as compared to the Internet case. With H.223 Annex B, the Maximum Payload Length (MPL) of MUX-SDUs is 254 bytes. A MUX-SDU may be packetized into one or more MUX-PDUs. However, if a MUX error occurs for a given MUX-PDU, all other segments of a segmented MUX-SDU will be lost due to either a false error detection in the AL-PDU or a loss of synchronization in the video decoder. Therefore, it is preferable not to split AL-PDUs to reduce the

video packet loss rate. It is preferable to employ the scan order Slice Structure (SS) of H.263 for mobile applications, which allows the use of video packets that are as small as one image block in size. As an upper bound, the size of the slices must conform to the MPL restriction.

In [101], an H.223 Annex B AL-3 multiplexing simulator is provided. The simulator receives video packets, simulates audio traffic, applies errors to the multiplexed bit stream according to an error pattern stored in a file, and outputs the packets with errors along with the checksum results. The possible loss of synchronization at the H.223 multiplex layer is also simulated, which will result in additional packet losses. Burst errors will most likely not corrupt two consecutive video AL-PDUs, since audio AL-PDUs are inserted between them. This simulator is employed for all simulation results presented in this chapter.

8.3 Joint Optimization of Spatial and Temporal Synchronization

The placement of synchronization markers (Chapter 4) and the RD optimized intra coding (Chapter 6) can be optimized independently. However, the decision to insert a synchronization marker at the current macroblock location will affect the mode decision of this same macroblock. For example, the probability that a macroblock is in error employed by the mode decision method will be affected by the location of the synchronization markers. Better per-

formance can be achieved by jointly optimizing the synchronization marker location and mode decision [43]. This is done by minimizing the overall Lagrangian $J = J_{mode} + J_{sync}$ for all different sync $\in \{0, 1\}$ and coding mode $\in \{skip, inter, intra\}$ cases.

8.4 Simulation Results

For the transmission of the encoded video bit streams, we employ the H.223 simulation model described in Section 8.2. We drop all video packets for which the checksum indicates the presence of errors, except when semi-fixed-length coding of motion vectors is employed. In this case, corrupted data is passed to the video decoder, and the recovered motion vectors are employed during error concealment. For each error condition tested, 25 simulations are performed to obtain statistically significant results, and the luminance PSNR (Y-PSNR) is averaged over all coded frames and simulation runs. Video source material is coded at QCIF spatial resolution (176 $\times$ 144 pixels) and temporal resolution of 10 frames per second. We employ a channel bit rate of 64 kbps and assume that the video data occupies 75% of the bandwidth [82]. Therefore, all video sequences are coded at 48 kbps using rate control. We use our proposed H.263 Test Model for a mobile environment [82] to generate anchors, which are used for comparison purposes. These conditions include a 20% random intra updating, and the insertion of a sync marker at the start of every GOB.

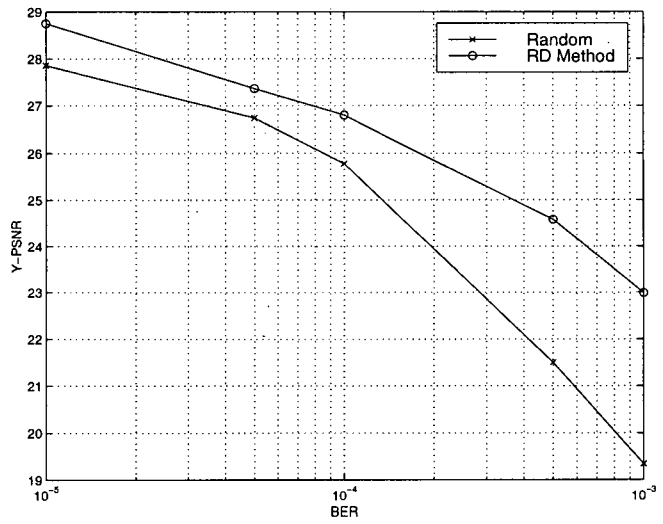


Figure 8.2: Performance of the RD optimized spatial and temporal synchronization methods for the sequence **FOREMAN**.

8.4.1 Performance over a binary symmetric channel

We first evaluate the performance of the proposed methods using the simple binary symmetric channel (BSC). For a given average bit error rate (BER), this channel model is actually the worst model, yielding the highest probability of error in a slice, as compared to a channel with correlated errors (burst errors). Results for the RD optimized spatial and temporal synchronization methods are presented in Figs. 8.2 and 8.3 for the video sequences **FOREMAN** and **NEWS**, respectively. BERs of $10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}$, and 10^{-3} are simulated and results of the RD optimized spatial and temporal synchronization methods are compared to those of the anchor model. The encoder employs the exact channel BER for the value of P_e used in the computation of the slice error rate (SER). The SER value is employed for the RD optimization

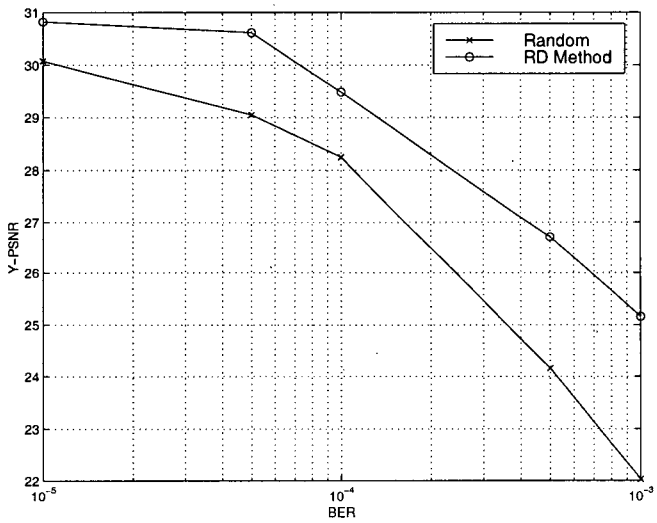


Figure 8.3: Performance of the RD optimized spatial and temporal synchronization methods for the sequence **NEWS**.

of the slice location as well as the RD optimization of the intra coding. The same error concealment used by the decoder is assumed at the encoder. Both the anchor bit streams and the bit streams generated by the proposed method are reconstructed using the same video decoder, with the error concealment method described in Section 3.6. The RD optimized spatial and temporal synchronization methods improve significantly the video reproduction quality, especially in poor network conditions. Over a 3.5 dB improvement in quality is achieved at a BER of 10^{-3} for the two considered sequences.

We also evaluate the additional performance improvement of using the semi-fixed-length motion vector coding and present results in Figs. 8.4 and 8.5 for the video sequences **FOREMAN** and **NEWS**, respectively. The motion vector data is protected in the same manner as described in Chapter 5, and the

overall bit rate includes the protection overhead. However, in the VLC coding case, motion vector data is not decoded and error concealment is applied using the predicted motion vectors. Using the semi-fixed-length coding of motion vectors, an additional 0.5 dB and 1.2 dB improvement is achieved at a BER of 10^{-3} for the video sequence **FOREMAN** and **NEWS**, respectively. For highly active sequences (e.g., **FOREMAN**), a slight degradation in PSNR occurs at low BERs. This is due to the fact that an error in motion data may lead to the decoding of a motion vector that is far less accurate than the prediction motion vector, and this error propagates through motion vector prediction until the next synchronization marker is reached. At the higher BERs, the decoded motion vector becomes more accurate than the predicted motion vector, since many video blocks used for prediction may be corrupted and many more synchronization markers are inserted, reducing error propagation.

Next, we evaluate the effects of a mismatch in BER between the assumed value at the encoder, P_e, and the actual channel BER, for the RD optimized spatial and temporal synchronization methods. To do so, we encode video sequences at different assumed P_e and transmit them over a BSC with different values of BERs. Results are presented in Fig. 8.6 for the video sequence **FOREMAN** and in Fig. 8.7 for the video sequence **NEWS**. On the average, the best performance is achieved when P_e used at the encoder matches exactly the network condition, which confirms the accuracy of the proposed model. The impact of the mismatch is worse at higher error rates. At a BER of 10^{-5}, the mismatch incurs a distortion penalty of at most 0.5 dB. How-

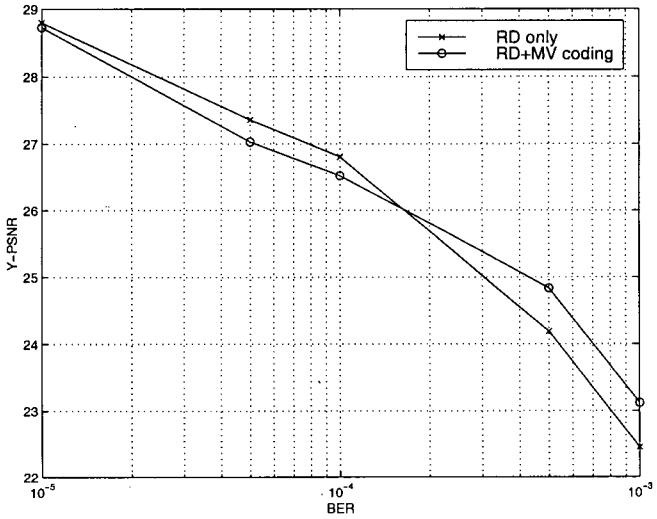


Figure 8.4: Additional performance improvement using semi-fixed-length coding of the motion vectors for the sequence FOREMAN.

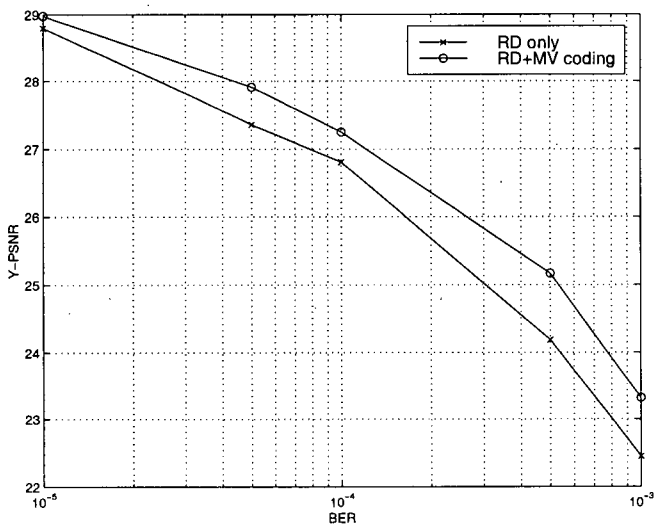


Figure 8.5: Additional performance improvement using semi-fixed-length coding of the motion vectors for the sequence NEWS.

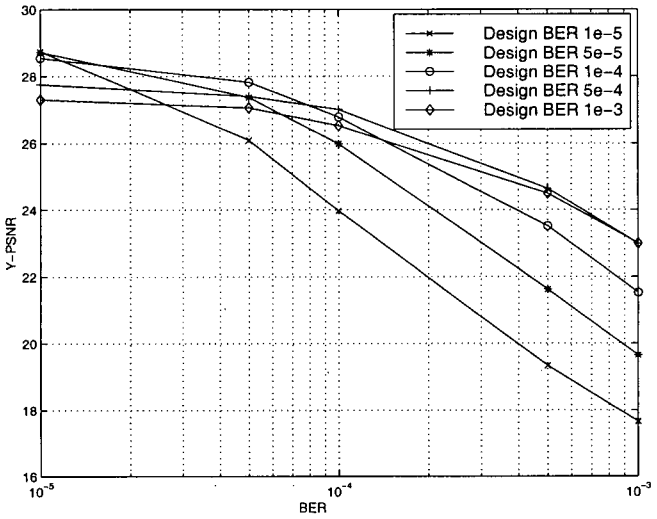


Figure 8.6: Performance of the proposed method with encoder/channel mismatch for the video sequence **FOREMAN**.

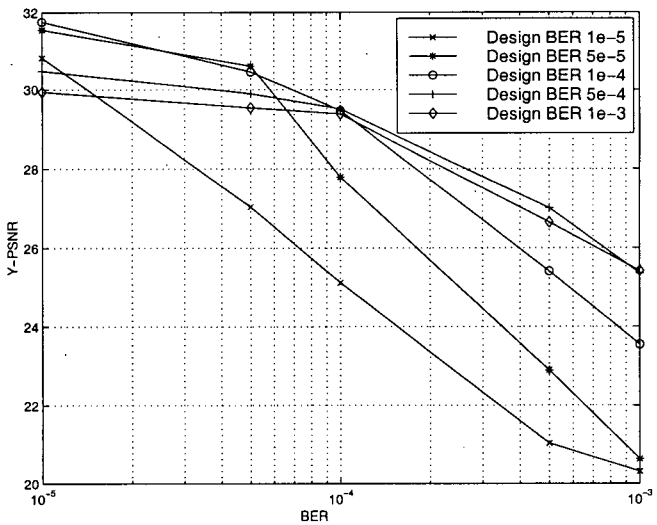


Figure 8.7: Performance of the proposed method with encoder/channel mismatch for the video sequence **NEWS**.

ever, at a BER of 10^{-3}, the mismatch can degrade performance by as much as 5 dB. Assuming a higher P_e is always better than assuming a lower P_e. In some cases, assuming a higher P_e than the actual network BER yields a slight increase in performance, especially at lower network BERs. This happens because the Lagrangian parameters λ_{mode} and λ_{sync} were obtained using a large set of video sequences and bit rates, and may not be optimal for a specific video sequence and bit rate. To compensate for the above effects and the possible feedback delays of the network, it is preferable to always assume worse network conditions at the encoder.

Finally, we evaluate the effects of a mismatch between the error concealment method assumed by the encoder and the one used by the decoder for the RD optimized spatial and temporal synchronization methods. We evaluate two error concealment (EC) methods: EC1, where the missing block is simply repeated from the previous frame, and EC2, where the median motion vector is employed to copy the motion compensated block from the previous frame. We use the correct channel BER for the value of P_e at the encoder. Results are presented for the sequence **FOREMAN** in Fig. 8.8. The resulting PSNR variation due to the error concealment mismatch is consistent across the range of tested BERs, except at very low BERs, where the error concealment method used at the encoder or decoder has almost no effect on the reproduction quality. The worst case happens when EC2 is assumed at the encoder and EC1 is employed at the decoder. However, the mismatch effects on the reproduction quality are relatively small, more specifically less than 1 dB. For

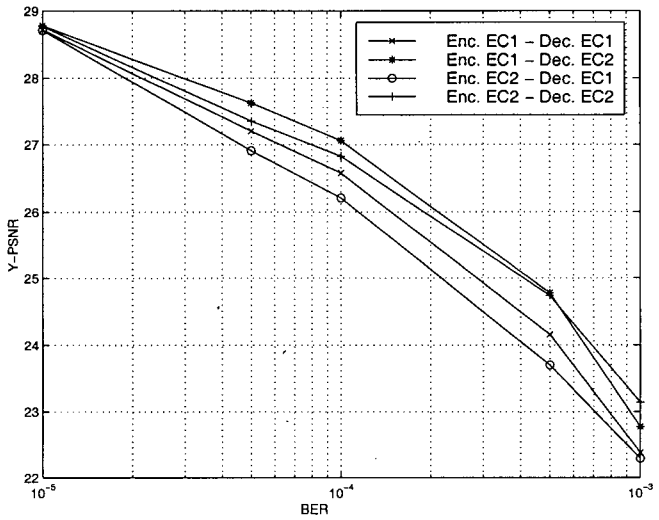


Figure 8.8: Performance of the for the RD optimized spatial and temporal synchronization methods with encoder/decoder error concealment mismatch for the video sequence **FOREMAN**.

higher BERs, it is actually better to assume a worse error concealment method at the encoder than the one used by the decoder. In such a case, the only loss occurs at a BER of 10^{-3} and it is less than 0.5 dB.

8.4.2 Performance over a WCDMA network

It is a difficult task to model a mobile network if one wants to account for all coding and channel parameters such as channel coding, interleaving, modulation, path loss, Rayleigh fading, mobile motion, etc [102]. From a user's perspective, mobile networks can be (simplistically) characterized by burst bit errors. Instead of trying to accurately model such a channel, we employ bit error patterns that characterize a specific mobile network.

Parameter	Value
User bit rate	64 kbps
Carrier frequency	1.9 Ghz
Info bit rate	65600 bps
	(incl. 16 bit CRC on 640 bit blocks)
Target BER	10^{-3} and 10^{-4}
Doppler frequency	5.3, 70, 211 Hz
	(corresponding to vehicular speed of 3, 40 and 120 km/h)
Number of processed frames	18000 (3 min)
Interleaving	10 ms
Coding	Convolutional, rate 1/3,
	constraint length 9, code length: 656 bits, 8 tail bits
Channel type	Vehicular A
Diversity	2 antenna branches, 4 RAKE fingers in each branch
Chip rate	4.096 Mcps
Spreading PDCH	16

Table 8.1: Parameters for the generation of WCDMA error patterns.

WCDMA (with convolution codes) error patterns provided in [103] for a user bit rate of 64 kbps are used in this second part of our simulations. Other error patterns are available from the ITU-T Study Group 16 (a. k. a. Multimedia Terminals and Systems Expert Group), in particular WCDMA with Turbo coded at 384, 128, 64 and 32 kbps in [104]. Turbo codes are usually better in a video communication system, since the average length in bits of a burst error in the received signal is greater than in the case of convolutional codes, thus affecting a smaller portion of slices for a given average BER. The WCDMA error patterns employed in our simulations are generated using the parameters described in Table 8.1 [103]. The channel itself is modeled as Raleigh fading according to the popular Jakes' model [105]. A summary of the characteristics

Error file	Doppler frequency (Hz)	Average bit error rate	Average burst length (bits)
wcdma-1	5.3	1.35×10^{-3}	16
wcdma-2	70	1.26×10^{-3}	17
wcdma-3	211	9.73×10^{-4}	15
wcdma-4	5.3	8.17×10^{-5}	11
wcdma-5	70	1.21×10^{-4}	13
wcdma-6	211	9.37×10^{-5}	11

Table 8.2: Characteristics of the WCDMA error patterns.

of these error patterns is provided in Table 8.2, where the average burst length is calculated under the assumption that at least 8 error free bits exist between two consecutive bursts. The different Doppler frequencies have little effect on the video reproduction quality, since the average length of the bursts is always much smaller than the length of a slice. Therefore, we only employ the error pattern for a vehicular speed of 120 km/h, corresponding to a Doppler frequency of 211 Hz (error patterns wcdma-3 and wcdma-6). We compare the results of our proposed method to those of the anchors in Table 8.3. Anchors are generated as in the BSC case, i. e., with 20% random intra coding and one sync marker per GOB. For all sequences considered, our method always outperforms the anchor model. Gains in Y-PSNR ranges from 0.09 dB to 2.06 dB. Moreover, our coder improves substantially error free quality in most cases, allowing the same bit stream to be transmitted on an error free channel with a better quality than the anchors. This is particularly important in the case of mobile video communications where parameters can be optimized for a worse case scenario (BER in deep fades, for example), and still provide

Sequence	Error Pattern	Y-PSNR (dB) Anchors (error free)	Y-PSNR (dB) Our Method (error free)	ΔPSNR
FOREMAN	wcdma-3	26.11 (28.07)	26.63 (27.39)	+0.53 (-0.68)
	wcdma-6	27.63 (28.07)	28.19 (28.48)	+0.44 (+0.41)
HALL	wcdma-3	28.94 (31.06)	30.01 (32.87)	+1.07 (+1.81)
	wcdma-6	30.57 (31.06)	32.63 (33.86)	+2.06 (+2.80)
CONTAINER	wcdma-3	29.27 (31.09)	29.36 (30.63)	+0.09 (-0.46)
	wcdma-6	30.70 (31.09)	32.10 (33.38)	+1.40 (+2.29)
NEWS	wcdma-3	25.62 (30.24)	25.99 (29.94)	+0.37 (-0.30)
	wcdma-6	28.68 (30.24)	29.91 (32.95)	+1.23 (+2.71)
SILENT	wcdma-3	28.38 (30.75)	28.89 (30.78)	+0.51 (+0.03)
	wcdma-6	30.23 (30.75)	31.27 (31.58)	+1.04 (+0.83)

Table 8.3: Average Y-PSNR for the different sequences over a WCDMA network.

excellent video reproduction quality during normal network conditions.

We show a visual example in Fig. 8.9 where the decoded frame number 101 of the video sequence FOREMAN is shown for the different methods. A video communication system without any error resilience mechanism is obviously not acceptable, as illustrated in Fig. 8.9 (b). The GOB synchronization combined with random intra updating and decoder error concealment provides an acceptable quality level for bit error prone environments (Fig. 8.9(c)). However, the proposed coder provides excellent reproduction quality, as illustrated in Fig. 8.9 (d). The visual quality is comparable to the error free quality, shown in 8.9 (a).

8.5 Summary

In this chapter, we presented results for the combination of the proposed algorithms in a video communication system for mobile networks, that consists of a video encoder with RD optimized placement of synchronization markers, semi-fixed-length coding of motion vectors, and RD optimized intra updating, with a corresponding decoder with error concealment. The only non-standard compliant method is the semi-fixed-length coding of motion vectors. The system was designed to allow real-time interactive communication, as it does not rely on any feedback mechanisms other than information about the bit error rates. This information can be obtained from the network or can be assumed a priori. Nevertheless, the performance of our method is not significantly affected when the assumed bit error rate differs from the actual one. Assuming a worse bit error rate can improve the performance with little degradation when the actual bit error rate is lower. We also evaluated the mismatch effect between the error concealment method employed by the encoder and the one employed by the decoder. In this situation, the effects of mismatch are minimal, and assuming a worse error concealment method provides slightly better results. Finally, we presented results for the binary symmetric channel as well as a WCDMA network. The combination of the proposed algorithms is shown to yield good video reconstruction quality even at high bit error rates such as 10^{-3}.

Chapter 9

Conclusions

9.1 Thesis Contributions

This work presented error resilience video coding methods that substantially improve the delivery of video on packet lossy and bit error prone networks. The resulting video communication systems exploit the video input statistics, channel condition knowledge, and information about the video decoder error handling capabilities to optimize the encoded bit stream for its transmission over error prone networks.

Low bit rate video coding was introduced and we briefly summarized the H.263 video coding Recommendation. This very successful standard provides high compression efficiency and includes numerous video coding tools that allow a high level of flexibility. This video coding framework was employed throughout this thesis. Error resilience video coding contributions in the literature were then reviewed.

We first proposed a method to improve spatial synchronization by optimizing the placement of synchronization markers. A new distortion measure that takes into account source coding and error concealment distortions was introduced. Based on the new distortion measure and the error rate of the network, we proposed a rate-distortion optimized synchronization marker placement algorithm which can effectively determine the frequency and location of synchronization markers.

Next, we presented a semi-fixed length coding method for the efficient entropy coding of motion vectors. With the use of unequal error protection techniques, the proposed method provides additional bit stream and spatial synchronization. Results for different video sequences and error patterns were presented, showing the superior performance of semi-fixed length codes over VLCs.

A new rate-distortion optimized video coding mode selection algorithm based on network conditions and decoder error concealment was then introduced. Different distortion measures that account for error propagation in a motion compensated video coding algorithm were evaluated. An iterative method for the computation of distortion was presented. This method accounts for temporal error propagation due to motion compensation and the filtering effects of different coding tools (e.g., half-pel motion compensation and deblocking filter). The proposed method provides temporal synchronization by choosing the optimal frequency and location of intra coded blocks, given a decoder error concealment method. As demonstrated by experimen-

tal results, the proposed method improves significantly the video reproduction quality when video blocks are lost due to channel errors.

Finally, we presented experimental results of the proposed error resilience methods for two different scenarios: the Internet and mobile networks. In the Internet case, we developed an efficient packetization method suitable for low delay video communications. We evaluated the performance of the packetization method and the RD optimized mode selection algorithms for different packet loss rates. In the mobile case, we introduced the transport protocol and evaluated the performance of the RD optimized synchronization marker placement, semi-fixed-length coding of motion vectors, and optimal intra updating under random bit error conditions. We also presented results for a specific wide-band code division multiple access (WCDMA) simulated mobile network. The effects of mismatch between the parameters used by the video encoder and the actual channel conditions and decoder error handling capabilities are then evaluated. The proposed method can provide acceptable video quality when as much as 20% of the video data is lost.

Most of the error resilience video coding methods presented in this thesis are also H.263 compliant, and they can have immediate applications. Therefore, we believe that the work presented in this thesis can immediately benefit industry.

9.2 Future Research Directions

Error resilient video coding is a relatively new research area, and many developments are expected in the coming years. We expect that more contributions will include a rate-distortion optimized framework such as the one presented in this thesis. However, more accurate theoretical models for error propagation in a complex motion compensated and transform video coding framework are needed, and they present a challenging research topic.

On the transport side, advances in Internet real-time transport protocols, such as Resource Reservation Protocol (RSVP), Scalable Reservation Protocol (SRP), and Differentiated Services (DIFFSERV), will allow different levels of Quality of Service (QoS) and other important services improving transport of real-time multimedia information. This will allow for the development of scalable error resilience algorithm and other coding methods that take advantage of different classes of service. Scalable video coding and error resilience algorithms along with joint source-channel coding can be used in such environments to provide a substantial performance improvement. Moreover, with recent advances in wireless communications, multimedia communications over such networks will become possible, and wireless Internet access will surely become widespread. In addition to the challenge of multimedia communications over the Internet, error resilience algorithms have to cope with fading and burst errors present in wireless networks. Finally, we list specific research directions envisioned at present:

1. Error resilience methods for transmission over networks with guaranteed

QoS channels:

(a) Un-balanced multiple description coding.

(b) Entropy coding techniques amenable to unequal error protection.

2. Better statistical models for error propagation and decoder distortion, which include the filtering effects of sub-pel motion estimation, loop filters, etc.

3. Transport-aware video packetization methods for video streaming applications.

4. Joint rate control and joint error resilience between multimedia streams, including video, audio and data.

Bibliography

[1] K.-H. Tzou, H. G. Musmann, and K. Aizawa, "Special Issue on Very Low Bit Rate Video Coding," *IEEE Trans. on Circuits and Syst. Video Technol.*, vol. 4, pp. 213–367, June 1994.

[2] W. Li, Y.-Q. Zhang, and M. L. Liou, "Special Issue on Advances in Image and Video Compression," *Proc. of the IEEE*, vol. 83, pp. 135–340, Feb. 1995.

[3] H. Li, A. Lundmark, and R. Forchheimer, "Image Sequence Coding at Very Low Bitrates: A Review," *IEEE Trans. on Image Processing*, vol. 3, pp. 568–609, September 1994.

[4] B. Girod, K. Younes, R. Bernstein, P. Eisert, N. Färber, F. Hartung, U. Horn, E. Steinbach, and T. W. K. Stuhl muller, "Recent advances in video compression," in *IEEE International Symposium on Circuits and Systems*, Feb. 1996.

[5] B. Girod, "Advances in digital image communication," in *Proceedings 2nd Erlangen Symposium*, (Erlangen, Germany), Apr. 1997.

[6] T. Cover and J. Thomas, *Elements of Information Theory.* New York: John Wiley and Sons, Inc, 1991.

[7] B. Girod and N. Färber, "Feedback-based error control for mobile video transmission," *Proc. IEEE, Special Issue on Video for Mobile Multimedia*, 1999. to appear.

[8] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable lenght codes," *IEEE Trans. on Communications*, vol. 43, pp. 158–162, Feb. 1995.

[9] J. Wen and J. Villasenor, "A class of reversible variable length codes for robust image and video coding," in *International Conference on Image Processing*, vol. 2, (Santa Barbara, CA, USA), pp. 65–68, Oct. 1997.

[10] D. Redmill and N. Kingsbury, "The EREC: An error resilient technique for coding variable-length blocks of data," *IEEE Trans. on Image Processing*, vol. 5, pp. 565–574, Apr. 1996.

[11] B. P. Tunstall, *Synthesis of noiseless compression codes.* Ph. D. Thesis, Georgia Institute of Technology, 1968.

[12] R. Llados-Bernaus and R. Stevenson, "Fixed length entropy coding for robust video compression," in *International Conference on Image Processing*, (Santa Barbara, CA, USA), Oct. 1997.

[13] V. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Trans. on Information Theory*, vol. 39, pp. 821–834, May 1993.

[14] Y. Wang, M. Orchard, A. Reibman, and V. Vaishampayan, "Redundancy rate distortion analysis of multiple description image coding using pairwise correlating transforms," in *International Conference on Image Processing*, (Santa Barbara, CA, USA), pp. 608–611, Oct. 1997.

[15] V. Goyal and J. Kovacević, "Optimal multiple description transform coding of Gaussian vectors," in *IEEE Data Compression Conference*, (Snowbird, UT, USA), pp. 388–397, Mar. 1998.

[16] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston: Kluwer Academic Publishers, 1992.

[17] N. Nasrabadi and R. King, "Transform coding using vector quantization," in *IEEE Int. Conf on Information Science and Systems*, (The John Hopkins University, Baltimore, Maryland, USA), pp. 23–25, Mar. 1983.

[18] B. Mandelbrot, *The fractal geometry of nature.* Freeman, 1982.

[19] G. Lu, "Fractal image compression," *Signal Processing: Image Communication*, vol. 5, pp. 327–343, 1993.

[20] J. W. Woods and S. D. O'Neil, "Subband coding of images," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 1278–1288, Oct. 1986.

[21] H. Gharavi and A. Tabatabai, "Sub-band coding of monochrome and color images," *IEEE Trans. on Circuits and Systems*, vol. 35, pp. 207–214, Feb. 1988.

[22] J. W. Woods, ed., *Subband Image Coding.* Norwell, MA: Kluwer Academic Publishers, 1991.

[23] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. on Image Processing*, vol. 1, pp. 205–220, Apr. 1992.

[24] M. Kunt, "Second-generation image-coding techniques," *Proc. of the IEEE*, vol. 73, pp. 549–574, Apr. 1985.

[25] W. Pennebaker and J. Mitchell, *JPEG Still Image Compression Standard.* New York: Van Nostrand Reinhold, 1993.

[26] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. on Communications*, vol. COM-29, pp. 1799–1808, Dec. 1981.

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

[28] M. Gallant, G. Côté, and F. Kossentini, "A computation constrained block-based motion estimation algorithm for low bit rate video coding," *IEEE Transactions on Image Processing.* To appear.

[29] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Trans. on Communications*, vol. 41, pp. 604–612, Apr. 1993.

[30] K. P. Rao and P. Yip, *Discrete Cosine Transfroms:Algorithms, Advantages, Applications.* New York: Academic Press, 1990.

[31] N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception," *Proceedings of the IEEE*, vol. 81, pp. 1385–1422, Oct. 1993.

[32] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, 1952.

[33] G. Langdon, "An introduction to arithmetic coding," *IBM J. Res. Dev.*, vol. 28, pp. 135–149, Mar. 1984.

[34] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, pp. 520–540, 1987.

[35] M. Miyahara, "Quality assesment for visual service," *IEEE Communications Magazine*, vol. 26, pp. 51–60, Oct. 1988.

[36] P. Barten, "The squared root integral (SQRI): a new metric to describe the effect of various display parameters on perceived image quality," *Proc. SPIE*, vol. 1077, pp. 73–82, 1989.

[37] M. Miyahara, K. Kotani, and V. R. Akgazi, "Objective picture quality scale (PQS) for image coding," *IEEE Trans. Comm.*, vol. 46, pp. 1215–1226, Sept. 1998.

[38] Testing Ad Hoc Group, "Jpeg2000 testing results," *ISO/IEC JTC1J/SC29/WG1/N705*, Nov. 1997.

[39] H. Everett, "Generalized lagrange multiplier method for solving problems of optimum allocation of resources," *Operation Research*, vol. 1, pp. 399–417, 1963.

[40] T. Berger, *Rate Distortion Theory.* New Jersey: Prentice-Hall, Inc., 1971.

[41] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," in *IRE National Convention Record, Part 4*, pp. 142–163, 1959. Also in *Information and Decision Processes*, R. E. Machol, Ed. New York, NY: McGraw-Hill, 1960, pp. 93-126.

[42] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Proc. Magazine*, vol. 15, pp. 74–90, Nov. 1998.

[43] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Proc. Magazine*, vol. 15, pp. 23–50, Nov. 1998.

[44] T. Wiegand, M. Lightstone, D. Mukherjee, T. Campbell, and S. Mitra, "Rate-Distortion Optimized Mode Selection for Very Low Bit Rate Video Coding and the Emerging H.263 Standard," *IEEE Trans. on Circuits and Syst. Video Technol.*, vol. 6, pp. 182–190, Apr. 1996.

[45] A. Schuster and A. Katsaggelos, "Fast and efficient mode and quantizer selection in the rate-distortion sense for H.263," in *SPIE Proc. Visual Communications and Image Processing*, vol. 2727, pp. 784–795, 1996.

[46] ITU T Rec. H.263, Version 2, "Video Coding for Low Bitrate Communication," Jan. 1998.

[47] G. Côté, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Trans. on Circuits and Syst. Video Technol.*, vol. 8, pp. 849–866, Nov. 1998.

[48] ITU Telecom. Standardization Sector of ITU, "Video Codec Test Model Near-Term, Version 8 (TMN8), Release 0," *H.263 Ad Hoc Group*, June 1997.

[49] J. Ribas-Corbera and S. Lei, "Rate control in DCT video coding for low-delay communications," *IEEE Trans. on Circuits and Syst. Video Technol.*, vol. 9, pp. 172–185, Feb. 1999.

[50] S. Lin, D. Costello, and M. Miller, "Automatic-repeat-request error-control schemes," *IEEE Communications Magazine*, vol. 22, pp. 5–17, Dec. 1984.

[51] S. Wicker, *Error Control Systems for Digital Communication and Storage*. Toronto: Prentice Hall Canada Inc., 1995.

[52] A. Andreadis, G. Benelli, A. Garzelli, and S. Susini, "FEC coding for H.263 compatible video transmission," in *International Conference on Image Processing*, (Santa Barbara, CA), pp. 579–581, Oct. 1997.

[53] J. Rosenberg and H. Schulzrinne, "An RTP payload format for generic forward error correction," *Internet Draft*, Feb. 1999. Available from http://info.internet.isi.edu:80/in-drafts/files/draft-ietf-avt-fec-05.txt.

[54] M. Khansari, A. Jalali, E. Dubois, and P. Mermelstein, "Low bit-rate video transmission over fading channels for wireless microcellular systems," *IEEE Trans. on Circuits and Syst. Video Technol.*, vol. 6, pp. 1–11, Feb. 1996.

[55] H. Lui and M. El Zarki, "Performance of H.263 video transmission over wireless channels using hyrbid ARQ," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 1775–1786, Dec. 1997.

[56] W. Lee, M. Pickering, M. Frater, and J. Arnold, "Error resilience in video and multiplexing layers for very low bit-rate video coding systems," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 1764–1774, Dec. 1997.

[57] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP) – version 1 functional specification," *RFC 2205*, Sept. 1997. Available from ftp://ftp.isi.edu/in-notes/rfc2205.txt.

[58] R. Talluri, I. Moccagatta, Y. Nag, and G. Cheung, "Error concealment by data partitioning," *Signal Processing: Image Communications Magazine*, vol. 14, pp. 505–518, May 1999.

[59] D. Park, J. Park, J. Kim, and Y. Kim, "Error-resilient video coding in H.263+ against error-prone mobile channels," in *SPIE Proc. Visual Communications and Image Processing*, vol. 3653, (San Jose, CA, USA), pp. 200–207, Jan. 1999.

[60] K. Ngan and C. Yap, "Combined source-channel video coding," in *Signal Recovery Techniques for Image and Video Compression and Transmission*, Boston: Kluwer Academic Publishers, 1998.

[61] R. Llados-Bernaus and R. Stevenson, "Fixed-length entropy coding for robust video compression," *IEEE Trans. on Circuits and Syst. Video Technol.*, vol. 8, pp. 745–755, Oct. 1998.

[62] T. Algra, "Fast and efficient variable-to-fixed-length coding algorithm," *Electronics Letters*, vol. 28, pp. 1399–1401, July 1992.

[63] R. Llados-Bernaus and R. Stevenson, "Codeword asigment for fixed-length entropy coded video streams," in *IEEE Data Compression Conference*, (Snowbird, UT, USA), Mar. 1998.

[64] A. E. Gamal and T. Cover, "Achievable rates for multiple descriptions," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 851–857, Nov. 1982.

[65] Y. Wang, M. Orchard, and A. Reibman, "Optimal pairwise correlating transforms for multiple description coding," in *International Conference on Image Processing*, (Chicago, Illinois, USA), Oct. 1998.

[66] V. Goyal, J. Kovacević, R. Arean, and M. Vetterli, "Multiple description transform cosing of images," in *International Conference on Image Processing*, (Chicago, Illinois, USA), Oct. 1998.

[67] S. Fukunaga, T. Nakai, and H. Inoue, "Error resilient video coding by dynamic replacing of reference pictures," in *IEEE Global Telecommunications Conference*, vol. 3, (New York, NY, USA), pp. 1503–1508, Nov. 1996.

[68] S. Wenger, "Video redundancy coding in H.263+," in *Proc. AVSPN*, (Aberdeen, UK), Sept. 1997.

[69] P. Haskell and D. Messerschmitt, "Resynchronization of motion compensated video affected by ATM cell loss," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, vol. 3, (San Francisco, CA, USA), pp. 545–548, Mar. 1992.

[70] N. Naka, S. Adachi, M. Saigusa, and T. Ohya, "Improved error resilience in mobile audio-visual communications," in *IEEE International Conference on Universal Personal Communications*, vol. 1, (Tokyo, JAPAN), pp. 702–706, Nov. 1995.

[71] J. Liao and J. Villasenor, "Adaptive intra update for video coding over noisy channels," in *International Conference on Image Processing*, vol. 3, (Lausanne, Switzerland), pp. 763–766, Sept. 1996.

[72] M. Wada, "Selective recovery of video packet loss using error concealment," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 807–814, June 1989.

[73] E. Steinbach and N. Färber and B. Girod, "Standard compatible extension of H.263 for robust video transmission in mobile environment," *IEEE Trans. on Circuits and Syst. Video Technol.*, vol. 7, pp. 872–881, Dec. 1997.

[74] S. Wenger, G. Knorr, J. Ott, and F. Kossentini, "Error resilience support in H.263+," *IEEE Trans. on Circuits and Syst. Video Technol.*, vol. 8, pp. 867–877, Nov. 1998.

[75] ISO/IEC 14496-2, "Coding of audio-visual objects: visual, final draft international standard," *ISO/IEC JTC1/SC29/WG11 N2502*, Oct. 1998.

[76] R. Talluri, "Error resilient video coding in the MPEG-4 standard," *IEEE Comm. Magazine*, vol. 26, pp. 112–119, June 1998.

[77] C. Bormann, L. Cline, G. Deisher, T. Gardos, C. Maciocco. D. Newell, J. Ott, G. Sullivan, S. Wenger and C. Zhu, "RTP payload format for the 1998 version of ITU-T Rec. H.263 video (H.263+)," *RFC 2429*, May 1998. Available from ftp://ftp.isi.edu/in-notes/rfc2429.txt.

[78] Y. Wang and Q. Zhu, "Error control and concealment for video communication: A review," *Proceeding of IEEE*, vol. 86, pp. 974–997, May 1998.

[79] H. Sun and W. Kwok, "Concealment of damaged block transform coded images using projection onto convex sets," *IEEE Transactions image processing*, vol. 4, pp. 470–477, April 1995.

[80] W. Kwok and H. Sun, "Multi-directional interpolation for spatial error concealment," *IEEE Transactions on consumer electronics*, vol. 39, pp. 455–460, August 1993.

[81] ITU Telecom. Standardization Sector of ITU, "Video Codec Test model near-term, Version 10 (TMN10)," *ITU-T Study Group 16, Video Experts Group, Document Q15-D-65*, Apr. 1998. Available from ftp://standards.pictel.com/video-site/9804_Tam/q15d65.doc.

[82] S. Wenger and G. Côté, "Video test model description for H.324/M based communication," in *ITU-T Study Group 16, Video Experts Group, Document Q15-F-46*, (Seoul, Korea), Nov. 1998. Available from ftp://standards.pictel.com/video-site/9811_Seo/q15f46.doc.

[83] G. Côté, M. Gallant, and F. Kossentini, "Semi-fixed length motion vector coding for H.263-based low bit rate video compression," *IEEE Trans. on Image Processing*, vol. 8, pp. 1451–1455, Oct. 1999.

[84] Y. Lee, F. Kossentini, R. Ward, and M. Smith, "Towards MPEG4: An improved H.263-based video coder," *Special Issue of Signal Processing: Image Communication on MPEG-4*, vol. 10, pp. 143–148, July 1997.

[85] B. Girod, E. Steinbach, and N. Färber, "Comparison of the H.263 and H.261 video compression standards," in *Standards and Common Interfaces for Video Information Systems, SPIE*, vol. CR60, (Philadelphia, Pennsylvania), pp. 475–482, Oct. 1995.

[86] J. Hagenauer, "Rate-compatible punctured convolutional codes and their applications," *IEEE Trans. on Communications*, vol. 36, pp. 389–400, Apr. 1988.

[87] G. Côté and F. Kossentini, "Optimal intra coding of blocks for robust video communication over the internet," *Signal Processing: Image Communication, Special Issue on Real-time Video over the Internet*, vol. 15, pp. 25–34, Sept. 1999.

[88] W. Lee, M. Frater, M. Pickering, and J. Arnold, "A diversity-based scheme for reducing error propagation in video," in *International Conference on Image Processing*, vol. 3, (Santa Barbara, CA), pp. 582–585, Oct. 1997.

[89] S. Wenger and G. Côté, "Using RFC2429 and H.263+ at low to medium bit-rates for low-latency applications," in *Packet Video '99*, (New York, NY, USA), Apr. 1999.

[90] J. Postel, "User datagram protocol," *RFC 768*, Aug. 1980. Available from ftp://ftp.isi.edu/in-notes/rfc768.txt.

[91] J. Postel, "Internet protocol," *RFC 760*, Jan. 1980. Available from ftp://ftp.isi.edu/in-notes/rfc760.txt.

[92] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," *RFC 1889*, Jan. 1996. Available from ftp://ftp.isi.edu/in-notes/rfc1889.txt.

[93] J. M. Boyce and R. D. Gaglianello, "Packet loss effects on MPEG video sent over the public internet," in *ACM MULTIMEDIA 98*, (Bristol, UK), Sept. 1998.

[94] M. Handley, "An examination of mbone performance," *UCL/ISI Research Report*, Jan. 1997.

[95] J. Ott and S. Wenger, "Application of H.263+ video coding modes in lossy packet network environments," *Journal for Visual Communication*, 1998. Accepted for publication.

[96] ITU T Rec. H.323, "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service," 1996.

[97] G. Côté, F. Kossentini, and S. Wenger, "Mobile anchor sequences," in *ITU-T Study Group 16, Video Experts Group, Document Q15-G-31*, (Monterey, CA, USA), Feb. 1999. Available from ftp://standards.pictel.com/video-site/9902_Mon/q15g31.doc.

[98] G. Côté, S. Shirani, and F. Kossentini, "Optimal mode selection and synchronization for robust video communications over error prone networks," *IEEE Journal on Selected Areas in Communications*, Sept. 1999. Accepted for publication.

[99] ITU T Rec. H.324M, "Multimedia Terminal for Low Bitrate Visual Telephone Services over the GSTN," Feb. 1998.

[100] ITU T Rec. H.223, "Multiplexing Protocol for Low Bitrate Multimedia Communication," 1995.

[101] G. Sullivan, "A simple video packet mux simulator program for video streams in H.324/M using AL3 mux of H.223 Annex B," in *ITU-T Study Group 15, Video Coding Experts Group, Document Q15-F-16*, (Seoul, Korea), Nov. 1998.

[102] B. Sklar, "Rayleigh fading channels in mobile digital communication systems, part I: Characterization," *IEEE Communications Magazine*, vol. 35, pp. 90–100, 1997.

[103] Ericsson, Sweden, "WCDMA Error Patterns at 64 kb/s," in *ITU-T Study Group 16, Multimedia Terminals and Systems Expert Group, Document Q11-F-05*, (Cannes, France), June 1998. Available from: ftp://standards.pictel.com/q11-site/LBCmobile/error_pattern/wcdma-64kb-error-files.zip.

[104] Nokia, Finland, "WCDMA Error Patterns," in *ITU-T Study Group 16, Multimedia Terminals and Systems Ex-*

pert Group, Document Q11-I-11, (Monterey, CA, USA), Feb. 1999. Available from: ftp://standards.pictel.com/q11-site/LBCmobile/error_pattern/wcdma_turbo.zip.

[105] W. Jakes, *Microwave mobile communications.* New York: Wiley, 1974.